Usability Optimization and Testing Of Social Music Service

JESPER AHLBERG
ANDREAS ANDRÉN

Degree project in Communication Systems
Second level, 30.0 HEC
Stockholm, Sweden
Usability optimization and testing of social music service

Master of Science Thesis

Jesper Ahlberg
jesahl@kth.se

Andreas Andrén
aandre@kth.se

2011-07-18

Supervisor and examiner: Professor Mark T. Smith
Royal Institute of Technology (KTH), Stockholm
Information and Communication Technology Programme
Abstract

Music playback in venues is very often controlled by the small group of people administering the business or locale, and not by the audience of listeners themselves. The people listening rarely have any ability to affect the choice of music within the public place or business they are currently situated in.

This master thesis is built around a developed social music service, that lets the listeners control the playback of music together. The project is called: Blicko, and it enables its users to collaboratively build up a playlist of music, and then vote on the order of the upcoming tracks to be played. It can be thought of as a tool which has similar functions to a jukebox. The service enables the visitors or attendees at any type of public place or venue (i.e. cafes, bars, lobby, restaurants etc.) to be a part of deciding and controlling which music is being played.

The thesis is dedicated to examine the best way of refining the service, in terms of optimizing the User Experience (UX), by application of different practices and methods within the field of Human-Computer Interaction (HCI). This also involves managing the complications that rises when developing a multi-platform service for a wide range of devices.

Sammanfattning

Uppspelning av musik i lokaler är ofta kontrollerat av den lilla grupp människor som administrar verksamheten eller platsen, och inte själva publiken som lyssnar. De som lyssnar har sällan någon möjlighet att påverka musikvalet inom det offentliga utrymmet eller verksamheten som de för tillfället befinner sig i.

Detta examensarbete är byggt kring en utvecklad social musiktjänst, som låter lyssnarna kontrollera uppspelningen av musik tillsammans. Projektet är kallat: Blicko, och det tillåter sina användare att samarbetemässigt bygga upp en spellista med musik, och sedan rösta på ordningen i vilken de kommande låtarna ska spelas upp. Det kan jämföras med ett koncept som påminner om funtionerna hos en jukebox. Tjänsten möjliggör att besökarna av en offentlig lokal (i.e. kaféer, barer, lobbys, restauranger etc.) kan vara med och påverka beslutet av vilken musik som ska spelas upp.

Examensarbetet är dedikerat åt att utforska det bästa sättet att förfina tjänsten, i den särskilda aspekten att förbättra användarupplevelsen. Detta genom att applicera metoder inom området: Människa-Dator Interaktion (MDI) och hantera User Experience (UX) relaterade faktorer. Detta innefattar även hantering av komplikationer som uppstår vid utveckling av en tjänst till flera olika plattformar och enheter.
Acknowledgements

We would first and foremost like to express gratitude to our academic supervisor and examiner: Professor Mark T. Smith, for his feedback and support throughout this master thesis project. The positive and constructive input straightened this thesis on its course and guided us in our process.

I; Jesper, would like to give thanks to my girlfriend Michaela as well as my family and friends, for their support over the last couple of months. My gratitude is mainly from a personal perspective but also for not hesitating to volunteer as guinea-pigs during our testing.

I; Andreas, would like to thank my family for their support during the writing of this thesis, especially my brother and sister who took part in the testing. In addition, I would like to say thanks to all of my friends that in some way either directly, or indirectly, have helped or supported me during this time.

We have also been very pleased to be able to take advantage of the facilities at Wireless@KTH’s lab. It has proven to be a very productive environment (especially for writing), and we have had close access to all the tools and expertise necessary for our thesis.

Finally we would like express our deepest appreciation towards our fellow project member: Theodor Zettersten. Without whom Blicko most likely would not exist today.
# Table of Contents

Abstract ......................................................................................................................... i
Sammanfattning .......................................................................................................... i
Acknowledgements .................................................................................................... ii
Table of Contents ...................................................................................................... iii
Table of Figures .......................................................................................................... vi
Table of Tables .......................................................................................................... viii
Abbreviations and Acronyms .................................................................................... ix
Terminology ................................................................................................................ x

1. Introduction ............................................................................................................. 1
   1.1 Background ........................................................................................................ 1
   1.2 Problem definition ............................................................................................. 1
   1.3 Scope .................................................................................................................. 1
   1.4 Expected Results ............................................................................................... 2

2. Architecture ............................................................................................................ 3
   2.1 Introduction ....................................................................................................... 3
   2.2 Overview .......................................................................................................... 3
   2.3 Subprojects ....................................................................................................... 4
      2.3.1 Global ......................................................................................................... 5
      2.3.2 Tuner Application ..................................................................................... 5
      2.3.3 Stations ..................................................................................................... 7
         2.3.3.1 Big ...................................................................................................... 9
         2.3.3.2 Small .................................................................................................. 10
         2.3.3.3 Upcoming ......................................................................................... 11
         2.3.3.4 Search (section) .............................................................................. 13
         2.3.3.5 Feed ................................................................................................ 14
         2.3.3.6 Users ................................................................................................. 15
   2.4 Libraries ........................................................................................................... 16

3. Concept .................................................................................................................. 17
   3.1 Introduction ....................................................................................................... 17
   3.2 Similar services and restrictions ....................................................................... 17
      3.2.1 MixApp ................................................................................................... 17
      3.2.2 Grooveshark and Tubeify ..................................................................... 18
      3.2.3 Restrictions and dependencies ................................................................ 18
   3.3 Voting algorithms ............................................................................................. 19
5.2.2.2 Approach ........................................................................................................ 54
5.2.2.3 Participants ........................................................................................................ 55
5.3 Results ...................................................................................................................... 55
5.3.1 Task oriented test .................................................................................................. 55
  5.3.1.1 Demographic background ............................................................................. 56
  5.3.1.2 Task success rate .......................................................................................... 57
  5.3.1.3 Task completion time .................................................................................... 57
  5.3.1.4 Post-task rating ............................................................................................... 59
  5.3.1.5 Task efficiency ............................................................................................... 60
  5.3.1.6 System Usability Scale .................................................................................. 62
  5.3.1.7 Observations .................................................................................................. 64
5.3.2 Collaborative user test ......................................................................................... 64
  5.3.2.1 Demographic background ............................................................................. 64
  5.3.2.2 System Usability Scale .................................................................................. 65
  5.3.2.3 Test session data ........................................................................................... 67
  5.3.2.4 Observations .................................................................................................. 69
6. Analysis ...................................................................................................................... 70
  6.1 Introduction ............................................................................................................. 70
  6.2 Usability optimization ............................................................................................ 70
  6.3 Usability testing ..................................................................................................... 71
  6.4 Conclusions and future ........................................................................................ 77
7. References .................................................................................................................. 79
8. Appendixes ............................................................................................................... 85
  8.1 Division of work ..................................................................................................... 85
  8.2 Digital appendixes .................................................................................................. 85
    8.2.1 Web references .................................................................................................. 85
    8.2.2 Usability testing screen recordings ................................................................. 85
    8.2.3 Questionnaire forms, data and excel spreadsheets .................................... 85
    8.2.4 Images and screenshots ................................................................................. 86
    8.2.5 Test session material ....................................................................................... 86
    8.2.6 Style sheets ..................................................................................................... 86
  8.3 Testing appendixes ................................................................................................ 86
    8.3.1 Post test implementations .............................................................................. 86
    8.3.2 Questionnaire data .......................................................................................... 86
      8.3.2.1 Demographic Background ........................................................................ 87
      8.3.2.2 Task Test Protocol ..................................................................................... 88
      8.3.2.3 Evaluation .................................................................................................. 93
      8.3.2.4 Collaborative .............................................................................................. 94
    8.3.3 Collaborative test user events .......................................................................... 96
# Table of Figures

- Figure 1: Overview of the architecture of Blicko………………………………………3
- Figure 2: Block diagram over a Tuner Application and its layers……………………6
- Figure 3: Connect box with login options when connecting to a Station………………8
- Figure 4: Block diagram of a Station and its layers……………………………………9
- Figure 5: Dialog box in Big layout…………………………………………………10
- Figure 6: Navigation page in Small layout…………………………………………11
- Figure 7: View of “Upcoming” within a Blicko Station………………………………12
- Figure 8: View of “Upcoming” in Small layout………………………………………13
- Figure 9: View of “Search” in Big layout………………………………………………13
- Figure 10: View of “Search” in Small layout…………………………………………14
- Figure 11: View of “Feed” in Big layout Station……………………………………15
- Figure 12: View of “Users” in Small layout…………………………………………16
- Figure 13: Wilson score interval……………………………………………………..21
- Figure 14: Five base colors of Blicko’s interface………………………………………25
- Figure 15: The typefaces (fonts) of Blicko’s interface………………………………26
- Figure 16: Metaphors (icons) for search function and section…………………………27
- Figure 17: Metaphors (icons) for “Blick” and “Block” functions…………………………27
- Figure 18: First mock-up of “Upcoming” interface…………………………………29
- Figure 19: Wireframe adaptation of mock-up…………………………………………29
- Figure 20: Screen resolution comparison by visualization……………………………31
- Figure 21: Screenshot of aggregated data over one hour usage scenario in Station……33
- Figure 22: Best practice of tabular navigation………………………………………36
- Figure 23: Blicko’s tabular menu in Big layout………………………………………36
- Figure 24: Example of page with scrolling issue in Android………………………38
- Figure 25: Animated loaders for constant feedback…………………………………..40
- Figure 26: Onbeforeunload dialog……………………………………………………43
- Figure 27: Formula for number of usability problems found in a usability test………..51
- Figure 28: Graph of usability problems found per number of test users…………..52
- Figure 29: Test site as introduction for participants in collaborative user test……….54
- Figure 30: Laplace law of Succession………………………………………………….57
- Figure 31: Chart of average task completion time per task for Big…………………58
- Figure 32: Chart of average task completion time per task for Small………………58
- Figure 33: Chart of average task rating per task for Big……………………………59
- Figure 34: Chart of average task rating per task for Small…………………………60
- Figure 35: Chart of average and optimal number of clicks per task for Big………..61
Figure 36: Chart of average and optimal number of clicks per task for Small.............61
Figure 37: Chart of participant efficiency for Big..............................................62
Figure 38: Chart of participant efficiency for Small...........................................62
Figure 39: Chart of participant SUS-score in task-oriented test for Big......................63
Figure 40: Chart of participant SUS-score in task-oriented test for Small....................63
Figure 41: Chart of average SUS-score for task-oriented test ................................64
Figure 42: Chart of participant SUS-score in collaborative test for Big.......................66
Figure 43: Chart of participant SUS-score in collaborative test for Small.....................66
Figure 44: Chart of average SUS-score for collaborative test.................................67
Figure 45: Pie chart of events which occurred in the collaborative test session............68
Table of Tables

Table 1: The three different types of services that Global host.............................5
Table 2: The differences of interface components between Small and Big layouts.........8
Table 3: The information of a track-item in “Upcoming”......................................12
Table 4: Design metrics for responsiveness of UI..................................................32
Table 5: Showing approximate data loads in example usage scenario.......................32
Table 6: Tasks for task oriented test session.........................................................48
Table 7: Demographic background data from the task-oriented test..........................56
Table 8: Internet habits data from the task-oriented test........................................57
Table 9: Demographic background data from the collaborative test..........................65
Table 10: Internet habits data from the collaborative test........................................65
Table 11: Example of particular behavior from the collaborative test..........................68
Table 12: Comments from participants and response..............................................76
Abbreviations and Acronyms

3G - Third Generation Mobile Telecommunications
AJAX - Asynchronous JavaScript and XML
API - Application Programming Interface
CSS - Cascading Style Sheet
DOM - Document Object Model [1]
HCI - Human-Computer Interaction
HD - High Definition
HTML - HyperText Markup Language
HTTP - Hypertext Transfer Protocol
iOS - iPhone OS [2]
IP - Internet Protocol
JPEG - Joint Photographic Experts Group
JS - JavaScript
MDI - Människa-Dator Interaktion (Swedish translation of HCI)
MP3 - MPEG-1 or MPEG-2 Audio layer III
OS - Operating System
PNG - Portable Network Graphics
px - Pixel
RGB - Red Green Blue
SUS - System Usability Scale [3]
TaC - Terms and Conditions
TCP - Transmission Control Protocol
ToS - Terms of Service
UI - User Interface
URL - Uniform Resource Locator
UX - User Experience
WiFi - Wireless Fidelity
WLAN - Wireless Local Area Network

e.g. - for example
i.e. - in other words
etc. - and so forth
Terminology

This thesis regards a quite comprehensively designed and developed social music service, including all of its components. These (internal) components or concepts are frequently referred to throughout the report and can be distinguished from their corresponding (regular) terms by their beginning with a capital letter and or surrounding quotation marks. For example: “Small” is the denomination of a layout (see below), and not a phrase regarding size as: “small” would be.

Blicko - Name of the social music service of which this thesis is based.
Global - Central server and public web page (see chapter: 2.3.1).
Tuner - Music playback application (software) (see chapter: 2.3.2).
Station - Main concept of which users connect to (see chapter: 2.3.3).
“Stations” - View containing the list of online Stations.
“Upcoming” - Main view of playlist and current song (see chapter: 2.3.3.3).
“Search” - View containing search function and results (see chapter: 2.3.3.4).
“Feed” - View containing all events in current Station (see chapter: 2.3.3.5).
“Users” - View of all the users currently connected (see chapter: 2.3.3.6).
User - Person that is connected to a Station by means of any device.
Spectators - Type of user without rights to interact with a Station.
Owner - The creator of a Station (automatically Administrator as well).
Administrator - User with administrative rights or permissions at a Station.
Session - Denotation of the connection a User has to a Station.
Venue - Public place with music playback, such as: cafes, restaurants, bars, cafeterias etc.
Big - Client side layout of a Station that is accessed by all devices with a “big” resolution (see chapter: 2.3.3.1).
Small - Client side layout of a Station that is accessed by all devices with a “small” resolution (see chapter: 2.3.3.2).
1. Introduction

1.1 Background

Disagreement is a common situation when several people come together in a place where there is music currently playing. When the responsibility of choosing what music to play is shared between more than one person, compromises have to be done and the tracks being played rarely reflects the combined musical taste of the group. A problem arises and a lot of people are quite selective when it comes to what music they should be exposed to. This in turn can have major influence over the group's or person's feelings about (for instance) a public place or business.

Music playback in venues, at offices or at other events, is very often controlled by the small group of people administrating the business or locale, and not by the primary audience: the visitors or customers themselves. The music playing in a venue can substantially affect people's (perhaps paying customers) thoughts and feeling about the place. It can even be the deciding factor when a group of people are settling on where they should go for the evening, especially in urban areas where the selection of choices is many. The kind of music that is being played through a venue’s speaker system will also have a significant influence on its whole atmosphere, so why not leave this responsibility up to the actual customers or visitors? A couple of decades ago there was a machine that handled this issue by letting the visitors of a locale decide what music that was being played, namely: the jukebox.

1.2 Problem definition

The general issue that Blicko sets out to deal with is the one that arises when several minds, with individual taste in music, set out to agree on what that should be played. This problem is described in the Background section above, and is the underlying reason that the service came to be. However, this thesis is focused on a more narrow issue that evolves when trying to develop this service with and for the technology and people of today. It is about adapting the service in the best way that is possible to the different devices and people that are about to use it. Resulting in a service that is intuitive to use and access on a range of devices with different situations and features in mind. Herein lays the challenge of creating a user interface which works both on computers as well as on mobile devices, such as smartphones etc. The service aims to be easily understandable even for the inexperienced user, which should be able to recognize components and familiar functions from other services on different platforms.

1.3 Scope

Although the subject for this master thesis is built around a quite complex software development project with many different components and technologies used, the main focus will be slightly shifted away from the architectural solutions and techniques. The main objective will be related to concepts and methods related to user experience, and testing of the same. One could say that the scope is shallow to the service, in that it for example, addresses (often visible) HCI factors within UI elements, rather than algorithm efficiency of
The scope of this thesis will mainly be on the development, testing and optimization of Blicko’s different UI’s. The development will be focused on making all the incorporated interfaces as intuitive as possible for the end-user, by means of methods related to HCI and usability optimization. The priority is shifted towards the vast majority of users who will use Blicko’s two main site layouts: Small (see chapter: 2.3.3.2) and Big (see chapter: 2.3.3.1). Therefore peripheral views of the interfaces, such as administrator settings and developers pages etc., will not be taken into account during this thesis. Also the Global webpage (see chapter: 2.3.1) is presented and displayed but not actively reviewed by any methods in neither the Usability Optimization, nor the Usability Testing.

The sole purpose of the usability testing (see chapter: 5) will be to evaluate the UI of Blicko, in search for indications of usability related problems. The test itself will be conducted as efficient as possible with a minimum amount of resources, whilst still being manageable by only two facilitators. The test results will be presented with conclusions on what changes that ought to be made to the interfaces in order for it to be optimized. These proposed modifications will be analyzed in chapter: 6 and presented in Appendix: 8.3.1, but only implemented to some extent (due to lack of time).

The section about the concept of Blicko (see chapter: 3) will be slightly different from the orientation of the rest of this report. It is provided to extend some interesting factors about the idea as a whole, and is to be seen in a greater perspective than the revising research that follows it.

1.4 Expected Results

The results of the work based on this thesis can be divided into two categories: firstly the results related to the usability optimization of Blicko according to applied concepts and methods (see chapter: 4), and secondly the results of usability testing sessions (see chapter: 5).

The expectation is that when methods from the field of usability optimization are applied to Blicko, an iterative and agile development [5] will be adopted to incorporate the changes accordingly. This will mainly affect the UI of Blicko (Station interface in particular) and refine the users interaction with the service. The results should therefore be more intuitive and “user-friendly” components and functions, which manipulates in familiar fashion and acts as the users expect it to.

The results from the usability testing will firstly come in form of many different metrics, such as time to complete task and number of clicks etc. These metrics will in turn hopefully indicate flaws stumbled upon by our testers in the test sessions. Any comments and notes will be evaluated and compared with the relevant metrics, in order to support any changes to the UI or functions. These changes will be implemented according to their severity and difficulty to realize. Time will tell whether these implementations will fit into the span of this thesis, however, the suggested changes will be presented in one way or another (see Appendix: 8.3.1).
2. Architecture

2.1 Introduction

This chapter provides an overview on the different components that constitutes Blicko, and can in turn be divided as subprojects. Throughout development the architecture has been modified to fit different needs and several optimizations have been done in terms of performance. The different components of Blicko are also presented under the “Terminology” chapter, where short descriptions of each concept are stated.

Blicko has been mentioned as a “social music service” that lets the users, i.e. visitors of a venue, collaboratively control (or influence) the playback. This chapter will present the different concepts that enable this function.

2.2 Overview

![Figure 1 - Overview of the architecture of Blicko, and its subprojects *]
The figure on the previous page illustrates an overview on how the different components of Blicko relate to each other. The different parts in this image will be further presented in chapter: “2.3 Subprojects” below.

If describing the picture one could say that the upper half represent the back end of Blicko, while the lower half shows the components that are apparent to the end user (i.e. front end). The clients are illustrated by the devices that the users might access a Blicko Station with, simply by pointing their browser to a specific address (URL). All communication goes through a server called: “Global” (see chapter: 2.3.1), that pairs the client with the certain Station. This connection between the client and a Station is hereafter referred to as a session. The termination of a session would occur either if a client actively disconnects (“leaves”) or if it time-outs, i.e. no user activity is registered for an hour. Another type of pairing that Global handles is the connection between Tuner Applications (see chapter: 2.3.2) and Stations (see chapter: 2.3.3). These two components have denominations that are conscious metaphors referring to radio stations and radios (“tuners”), which is a very accurate comparison when it comes to how they both relate to each other.

To provide an abstraction that exemplifies the illustration in fig. 1 this paragraph will be describing a hypothetical real life usage scenario. Disclaiming that it is for illustration purpose only and that it merely presents one usage example where in reality the users often has several optional ways to achieve an action. Scenario: the hypothetical user: “John Doe” walks into a bar called “Paddy’s Pub” (illustrated by orange area in fig. 1). Music is playing and he is aware that the venue uses Blicko as a music service. John picks up his smartphone and writes in the following URL in the address field of its default browser: “www.blicko.com”. The browser updates and shows a list of online Blicko Stations. He recognizes the Station name: “Paddy’s” further down in the lists and clicks the item to connect. A new page loads that asks John if he would like to sign in with his Facebook credentials or anonymously (see fig. 3). He chooses Facebook and connects to the Blicko Station: “Paddy’s”. Inside he is faced with the “Upcoming” screen (see chapter: 2.3.3.3) that lets him know what is in the queue to be played. He decides to submit a song by accessing “Search” (see chapter: 2.3.3.4) and querying: “Killing For Love”. He presses the “José Gonzales” track in the results list and choose to “Submit” in the dialog that emerges. John then proceeds back to “Upcoming” where he notices that his submitted track has position seven in the playlist. He then votes for a couple of songs (already submitted by others) that he also prefers and would like to hear being played soon. End of scenario.

2.3 Subprojects

Blicko is a project that consists of several elements that is easily represented as own “subprojects”. These projects are in turn called: “Global”, “Tuner Application” and “Stations”, and will be further explained and presented in this section. If described shortly (also see fig. 1 for reference):

- **Global** - A centralized server that handles all communication between Stations, clients and Tuner Applications.
- **Tuner Application** - A software program that connects to a station and play back music.
- **Station** - Is a virtual representation that users can connect to and participate in controlling the playback of music (often inside a venue).
2.3.1 Global

Global (see fig. 1) can be defined as the centralized server that handles the communication (TCP/IP traffic) between Stations (see chapter: 2.3.3), its connected Tuner Applications (see chapter: 2.3.2), and its users or clients. By means of three processes (see table 1) that handle the traffic of different requests and sessions (for example) Global acts as the connecting node between parties, and can be seen as the back end of Blicko. It keeps tracks of each Stations current state and records all events affecting it. Because of this, when a session is lost between a Station and Global, it could easily be rebuilt by implementing or rebuilding the same events from a certain state, in order for a Station to recover. At the moment all the components of Global (i.e. “StationService”, “PublicWeb” and “DataBase” (see table 1)) are hosted in Microsofts cloud service [6] named: Windows Azure and Microsoft SQL Azure [7]. The representations of a Stations is kept within Global in such way that in case of expansion, more (virtual) servers or resources could be added or allocated in able to hold more instances of Stations. The three different services shown in table 1 are independent of each other and could or should be (actually: “is”, because of cloud service [6]) hosted on different servers.

<table>
<thead>
<tr>
<th>Hosting</th>
<th>Name</th>
<th>Function</th>
<th>Located</th>
</tr>
</thead>
<tbody>
<tr>
<td>Stations</td>
<td>“StationService”</td>
<td>Hosts the representation of a Station and enables sessions towards connected clients.</td>
<td>Windows Azure Cloud</td>
</tr>
<tr>
<td>Webpage</td>
<td>“PublicWeb”</td>
<td>Public webpage accessible through the domain: <a href="http://www.blicko.com">www.blicko.com</a></td>
<td>Windows Azure Cloud</td>
</tr>
<tr>
<td>Database</td>
<td>“DataBase”</td>
<td>SQL database with tables for users and Stations etc.</td>
<td>Microsoft SQL Azure Cloud</td>
</tr>
</tbody>
</table>

*Table 1.* The three different types of services that Global hosts.

Global is also providing the window to the world by means of a web server that accommodates a public webpage. This webpage introduce each new visitor to the concept of Blicko and provide the navigation to progress further. It is a portal for sub directories where the Tuner Application (see chapter: 2.3.2) and Stations view (see chapter: 2.3.3) can be found. The domain acquired for the Global page is: www.blicko.com, which was perhaps the biggest factor when settling on the name itself. The acclaimed usability expert: Jakob Nielsen describes how the name, domain and trademark should correspond with each other, and how the top-level domain should be: “.com” if the site has a worldwide appeal. [8]

2.3.2 Tuner Application

The “Tuner” is responsible for the actual music playback, and comes in form of an application (.exe). The executable needs to be installed on the computer (with Windows operating system (OS)), that is connected to the sound system. The configuration of the application is all done through a web interface located on: Global. After that any Tuner Application can connect to any Blicko Station by a click of a button, given that the Owner of a Station has set the permissions to allow it. Each venue that would like to use Blicko for music playback, will (as of this version (see chapter: 2.4 and: 6.4 for future intentions)) have to install the Tuner Application. When the administrators of a Station have configured which music libraries it should make available, the users are welcome to connect and submit
songs into the upcoming playlist. The single reason for the Blicko Tuner being an executable is that it is required to in order for it to take advantage of “libspotify” (see chapter: 2.4). [9] This makes the application currently confined to Windows OS with the .NET-framework [10] available. The different layers and components that interact with the Tuner Applications are illustrated in fig. 2.

When the music libraries Blicko takes advantage of supports it (see chapter: 2.4), the move over to a web-based “tuner” will be imminent. Which in turn enables all devices with compatible browsers [11] to “tune-in” to a Station at anytime from anywhere (given the Station’s administrators allows it). Each Station will have different levels of privacy settings when it comes to who they allow to listen or participate, but also regarding the ability to tune in with a Tuner Application. The settings administrators have a choice between are:

- **Open** - meaning that anyone can access and connect to the Station. Either with Facebook credentials, anonymously or as a passive Spectator (see fig. 3).
- **Restricted** - meaning a password (set by administrators) has to be provided to connect.
In the first case of “Open” permissions at a Station, the resemblance between Blicko and a conventional radio station is quite strong, where anyone with a tuner can listen. The restricted settings are more applicable when the administrators want to keep the access limited to a physical area or group, such as at a venue or an office. It enables several people to listen in on the same (local) station via different Tuner applications. This could for example be practical when employees in the same office still want to listen to the same music, and participate in choosing of the same, but some prefer listening through their headphones (i.e. the music is played at their individual workstation which the headphones are connected to and not by a central speaker system). Another example is when the facilitators of a venue want the same music to be synchronized between rooms (although there are currently a slight difference in sync between applications, it could be minimized beyond discernibility). If this setup is scaled even further it can be favorably applied to whole branches, i.e. chains of coffee shops could play the same music at the same time, controlled either collaboratively together or by centralized management.

2.3.3 Stations

A Blicko Station is perhaps the most fundamental concept within the architecture of the service. This is where the users connect to in order to participate and control the playback, and it is to this the Tune Application can be connected (or “tuned in to”). It is a metaphor related to a radio station and it functions in a similar way, but here the listeners can participate actively. A Station can be thought of as a virtual space but if Blicko is used in a locale where one computer handles the playback of music, the Station could represent the physical space as well. All the customers (or attendees and visitors of the venue) are possible contributors and all they have to do is to connect by means of any device with a web-browser (see example scenario in chapter: 2.2). Anyone can create a Blicko Station and become its “owner”. This proprietor then possesses the right to distribute administrator permissions on the current Station to other users, which gives them certain authority when it comes to settings and control. The administrators of a Station have the power to manipulate certain mechanisms or functions. For example they are the one that can change the policies when it comes to which users that can: connect, tune-in and submit songs etc. The administrators also have a possibility to override the playback at any time, i.e. forcing the Station to skip a track, among other direct options.
In order for a user to connect to a Station he or she either has to manually select it from the list in the “Stations”-view (located at: http://www.blicko.com/stations), or by pointing the browser directly to the address (URL) of the specific Station. This specific address is customized for each Station and has the format: http://www.blicko.com/”station-name”. When entering the user is given different options on how to log in, depending on the Station’s settings (see fig. 3). The alternatives may vary, and the Stations could also be password protected, but by default the user can connect by means of his or her Facebook account. [12] If activated the other options would be to sign in as: “Anonymous” or as a “Spectator” without possibility to affect the Station (see fig. 3)

**Table 2** - The physical differences in UI components between Big and Small layout.

<table>
<thead>
<tr>
<th>Interface component</th>
<th>In Big</th>
<th>In Small</th>
</tr>
</thead>
<tbody>
<tr>
<td>Dialog box</td>
<td>Pop-up dialog</td>
<td>Whole dedicated page</td>
</tr>
<tr>
<td></td>
<td>(see fig. 5)</td>
<td></td>
</tr>
<tr>
<td>Navigation</td>
<td>Tabular menu</td>
<td>Top menu and grid navigation screen (see fig. 6)</td>
</tr>
<tr>
<td></td>
<td>(see fig. 20)</td>
<td></td>
</tr>
<tr>
<td>Margins and padding</td>
<td>More (except for clickable items) (see fig. 7)</td>
<td>Less (except for clickable items) (see fig. 8)</td>
</tr>
<tr>
<td>Fonts</td>
<td>Smaller size</td>
<td>Bigger size</td>
</tr>
<tr>
<td>Album cover image</td>
<td>Bigger size</td>
<td>Smaller size</td>
</tr>
</tbody>
</table>

In the following subproject chapters of this thesis the two different layouts of the Station interface will be presented, called: Big (see chapter: 2.3.3.1) and Small (see chapter: 2.3.3.2). As well as the four different pages (or views) that are available at each station (see fig. 4).
One of the two different layouts will be used depending on which device the users visit Blicko with. Where “Small” is designed for use with smartphones and devices with relatively low screen resolution (and perhaps touch input) and “Big” is designed for anything “bigger” than that (see table 2 for differences).

![Figure 4 - Block diagram of a Station * and its layers with layouts and views.](image)

A Blicko Station has not always existed as it is presented here or as it exists today. Many of the design choices are explained in the chapter on “Usability optimization”, and its implementation is discussed in chapter: 4.3. Subsequently the report will not always reflect a chronologically (from beginning to end) correct image of a Stations UI (for example). Especially relevant, before introducing the layouts: Big and Small (below), is the chapter on: 4.3.1 (“Separation of interfaces”), that describes how and why the two different types of interfaces (layouts) came to be.

### 2.3.3.1 Big

Big is a congenially named interface layout that the users can access a Blicko Station through (see fig. 4). It is an ordinary web interface that automatically sets in if the user uses a device with a higher resolution than 480*320 px. This filtering of users is done by means of JavaScript (JS) that automatically redirects a user according features of the user agent [13] being used.

The layout is active on all the pages within a Stations interface, and it induces certain properties regarding UI element sizes and amount of data to download etc. Some functions (e.g. administration settings and instant options) are only available when accessing a Blicko Station from Big, and not from its minor counterpart: Small. This layout is also closely tied together with Global, by for example: shared design themes, certain UI components and by clear navigation options in between the two. Whenever a song item is accessed (i.e. clicked) in Big a type of dialog box is displayed on top of the UI with information about the track and buttons with available actions (see fig. 5). Although, even Big have some size restriction when it comes to interface elements. Some of the components have set “max-width” and
“max-height” written as Cascading Style Sheets (CSS) properties that limits their size. This is for example used to control the amount of negative (dark) space when viewed on a widescreen HD television set etc. (see chapter: 4.2.2.1).

Figure 5 - This image displays the overlaying dialog box that appears in Big when clicking a tracks row in “Upcoming”.

2.3.3.2 Small

Whenever a user visits a Blicko Station from a device with a smaller screen resolution than 480*320 px, the Small interface layout sets in (see for example fig. 6). The filtration of user agents sets in and every user will either experience the Station, and all its underlying pages, through the Small or Big layouts. The differences in between the two are controlled by means of JavaScript (JS) and two separated files with CSS-files (see Appendix: 8.2.). Small were created out of necessity, for several reasons (see chapter: 4.3) related to the differences in devices that people use to access web pages with today. This layout is tailored to fit a typical smartphone, such as an Apple iPhone [14] or an Android [15] device. Since the interface is web based it can (as previously stated) be accessed through any browser, however, tailored applications for Small in iPhone OS (iOS) [2] and Android are planned to be developed. This could be done by implementing so called “web views” inside the application that fundamentally is a small browser window within another UI.

The most prominent difference from Big is that Small uses a different navigation system (see fig. 6), and have some UI components and functions disabled. The navigation page can be accessed from anywhere within the layout by clicking a button on the top row. Explanations and further exploration of these differences is presented in chapter: 4.3. Another difference is that the dialog boxes from Big (see fig. 5) are turned into separate pages, due to size restrictions.
2.3.3.3 Upcoming

The main view of a stations interface, which all connected users will be presented with upon connecting, is: “Upcoming”. It consists of two smaller parts. One that presents the track that is currently playing, and one that lists all the tracks currently in queue (see fig. 7). The upper part is displaying the following information of the track that is currently being played:

- **Album cover** - Cover art shown to the left.
- **Source logo** - Indicates which library that provides the current track (e.g. Spotify)
- **Track title** - Shown on top in large white typeface (e.g. “Teardrop”)
- **Vote count** - Final vote count reached before track started to play (e.g. “1”)
- **Artist name** - Name of the band or artist in bold (e.g. “José Gonzales”)
- **Album name** - Name of album title (e.g. “In Our Nature”)
- **Submitter name** - The user that submitted the track to the playlist (e.g. “Andreas Andrén”)
- **Progress bar** - Shows current progress by gradually filling graph with timestamps.

Note that the “Upcoming” section of a Station was previously known as “Player” which can be apparent within some of the figures.
Figure 7 - This image is a screenshot of a Station’s main view: “Upcoming” in the layout: Big. It is shown in Google Chrome [16] browser at screen resolution: 1366*768 px.

The tracks traversing the list in “Upcoming” are the representation of a classic playlist, sorted by descending order, according to vote count (see chapter: 2.3). The position is also indicated by an index to the left. This descending list is showing items in rows with three columns each that holds the following information:

<table>
<thead>
<tr>
<th>Column</th>
<th>Content</th>
<th>For example:</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Index number</td>
<td>“1” (if next in queue)</td>
</tr>
<tr>
<td>2</td>
<td>Track title</td>
<td>“Killing For Love”</td>
</tr>
<tr>
<td>2</td>
<td>Artist name</td>
<td>“José Gonzales”</td>
</tr>
<tr>
<td>2</td>
<td>Album name</td>
<td>“In Our Nature”</td>
</tr>
<tr>
<td>3</td>
<td>Vote box (with vote-count)</td>
<td>“2”</td>
</tr>
</tbody>
</table>

Table 3 - The table shows the information of a track-item in “Upcoming”.

The vote-box in Big allows for direct manipulation by clicking the small arrows, while the vote-box in Small only shows the vote-count and the current user’s individual vote. It is by clicking the rows in the list that connected users can influence the order or position of the tracks, then posting a vote on it (i.e. “Blick” or “Block”, see fig. 5). If the “Upcoming” view is empty of track items, the space will instead be occupied by instructions on how to submit songs to (re-)populate the playlist.
2.3.3.4 Search (section)

The view: “Search” is the section of a Station that contains the search function (described in chapter: 4.2.3.5). If a user is connected as anything but a spectator (see fig. 3) “Search” is where the participant would go to query the connected libraries for songs to submit to upcoming. Similar to the other sections it consists of sub-views, with the input field on top and the list of search results populating a list below it (see fig. 9).

The results returned after a query is sorted according to relevance (i.e. similarity with search string) and popularity, and if more libraries than one are connected the search string will be
posted towards them all simultaneously. If more libraries are made available, the different results will be presented in individual lists, as well as together on the first results page. There are some settings that could be necessary for the administrators of a Station, to set up in order for search to function properly. For example with Spotify's API, a region setting is needed to avoid returning search result items that not are available for streaming in the current region. In Big the items in the result list shows a small button on hover (e.g. moving the mouse cursor over the item), that enables for a simpler submission by just one click. If a certain track is already present in “Upcoming”, the same button turns into a “Submitted” label (see fig. 9), that indicates its inability to be submitted again. When a user has submitted a certain track from the search results the track will appear in the upcoming playlist. The action will also trigger an event that can be seen in the “Feed” view (see chapter: 2.3.3.4).

![Image of search results]

*Figure 10* - The image above shows how “Search”, looks on a smartphone.

The content and design for “Search” in Small, shown above, is quite similar to the Big version (see fig. 9), but with certain adaptations for the input type and screen size etc.

2.3.3.5 Feed

“Feed” is a quite special view within the Blicko UI. It is provided in order for the users to fully see each interaction that takes place within a Station. Every manipulation triggers an event that is presented in form of a list in “Feed”. An event can for example be: a submission of a song, an upvote (“blick”) on a track, a user connecting or if the playback is paused by the administrators etc. This descending list is sorted chronologically after the timestamp of the event, and also shows which user that activated the action itself (see fig. 11).
The “Feed” can be thought of as a window to everything that is happening inside a Station, at any given moment. The list occupies itself and any interactions made by the current user are marked by displaying his or her username in orange. The view is coherently designed in Small and in Big and has some resemblance to the “News Feed” presented on a user’s Facebook-profile for example. Some examples of what rows display in “Feed” could be:

- “Andreas Andrén - submitted track: José Gonzales - Crosses (2:42) 14:37”
- “Jesper Ahlberg - blicked: José Gonzales - Teardrop (3:21) 14:36”
- “Player - started track: José Gonzales - Heartbeats (2:43) 14:33”

Each event-row has a unique icon displayed in the left column that indicates which type of event that occurred (without having to read the whole line).

2.3.3.6 Users

The section of a Station that goes under the alias: “Users” is dedicated to present all current connections (i.e. sessions) to the certain Station. Each users name is retrieved from their Facebook account, as well as their profile picture. In the right column, a timestamp shows how much time has progressed since the user logged on. This view is subject to upcoming changes and additions of further information, without being prioritized because of its peripheral functions. However, expansion of the “Users” section could provide valuable ways of transitioning into, for example, profile pages on Global etc.
2.4 Libraries

Each Blicko Station that is paired with a Tuner Application needs to be connected with a music library, in order for the connected clients to be able to search for, and to submit, tracks. The available song pool to choose from is set up by the administrators of the Station. Currently there is only one source of songs which is supported by Blicko, namely: Spotify.

Blicko is able to take advantage of Spotify’s vast music library because of their API: “libspotify”. [9] It is a library in C (programming language) [17] for third party developers, that, by means of a so called application key, enables streaming from Spotify’s service. To access these features each Station need to provide their Spotify Premium credentials in the Tuner Application configuration page. By taking advantage of libspotify Blicko also needs to openly show that their so called “Spotify Core” service is being used. This is done by branding according to stated guidelines. [9] However, the simple search queries do not take the unnecessarily complex detour via libspotify. Simple queries go through the Spotify API called: Metadata API [18], which is used to explore the music catalogue.

A criterion for Blicko is that it is independently functional without being tied to any single specific music library. Therefore the intention is to expand the service by implementing more libraries into Blicko. Then letting each Stations owner decide which of the several libraries that should be enabled to search and submit track from. There are currently two factors that would be prioritized when looking for additional libraries. The first being: an open library without external accounts, or paid subscription, required. Secondly a library with the ability stream directly into the browser would be very favorable, since no (platform specific) application would be needed. The future objective is to detach the Tuner Application (see chapter: 2.3.2) component of Blicko from any specific platform or hardware too, by enabling it to run as a (standalone) web application. Playback of sound in the browser has been possible for a while (with for example: Adobe Flash [19]), but with the new <audio>-tag implementation directly in HTML5, the functionality will more natively be supported by almost all browsers. [20] [11] These desirable factors and future possibilities are further mentioned in chapter: 6.4.
3. Concept

3.1 Introduction

Blicko is a service that (as stated earlier) mimics the functions of a jukebox; in that it lets the listeners decide which music that should be played at any given moment. The listeners in for example a locale can decide to become “users” by simply connecting to the Station currently playing the music. They do this by mean of an ordinary web browser and can access a station from various types of devices, such as: smartphones, laptops, gaming consoles and tablets etc. Collaboration when constructing playlists is nothing new, but there are two more factors that make the concept of Blicko unique. The first one being: that all interaction affects the Station in real-time, and the second factor is the implementation of a voting mechanism to control the order of playback.

3.2 Similar services and restrictions

The Blicko service incorporates a mix of different mechanism to enable its use. The functions are not independently innovative, but when combined they form a unique concept. Throughout development several similar concepts have emerged, although often only related to some discrete aspects of Blicko at a time.

In this chapter services that are similar to Blicko will be presented by first shortly describing their functions, and in the second paragraph of each section, their differences in comparison to Blicko. The last section (see chapter: 3.2.3) will present the different restrictions and limiting factors, both by law and third party agreements, that emerge when playing music publicly at venues etc.

3.2.1 MixApp

MixApp [21] is a web based service that allows it users to submit video clips, from YouTube [22], to a collective playlist. The playlist can then be played back simultaneously while all participants can communicate with each other in the chat interface. It can therefore be seen as an online video player that several people can experience at the same time. The users submit videos to build up a playlist that are played back in real-time, and displayed inside the browsers of each connected client.

“MixApp lets you and your friends listen to your music together in a private chat room.” [21]

The similarity between MixApp and Blicko are many and most obvious is the real-time playlist collaboration feature. However, there are two main differences between them two. Firstly the voting mechanism within Blicko does not have a representation within MixApp. It simply plays back the items in the order they were submitted. Secondly, and perhaps most evident, is MixApp’s conscious orientation away from listeners at the same physical location. Blicko has an expressed priority in enabling users to contribute to the playback of music in a given space or locale. On the contrary the functions (e.g. chat feature, no mobile adapted
interface) of MixApp are indicating that the service is best suited for computers and prioritizes collaboration with distance between users.

### 3.2.2 Grooveshark and Tubeify

Grooveshark [23] and Tubeify [24] are two, quite similar, online music services that both enable their users to playback music from their vast library directly in the browser (by means of HTML5 [11]). They both look and act as a music player desktop application one would have installed in an operative system (OS). However, all interaction is done through a web interface that takes advantage of different online libraries of music (not locally stored files, such as: mp3’s). Grooveshark is the most commonly known of the two and it has been involved in quite some controversy because of the openness of their music library. Grooveshark has a very vast and rapidly growing library of songs available for streaming. This is because they allow anyone to upload any song from their personal collection (i.e. local mp3 files) and thereby make it available for all users to reach. They denounce their responsibilities toward record labels and artists by completely redirecting all liability toward the uploaders them self. They pose as an innocent middleman that simply enable users to upload their personal music files to their servers, and then gives access to them from any web browser. Tubeify is a similar service but instead of having their own library, they rely on YouTube as a content provider. Their browser interface resembles a desktop music application and they do not require a login to make use of their service (neither does Grooveshark).

Both these services can be seen as web-based representations of a desktop music application that can be reached from anywhere with a browser. However, this is where the similarities between the two services and Blicko end. The mentioned services have no support for collaboration at all, with the exception of manual (i.e. sharing of links) transactions of playlists. There is also no social elements tied to (for example) physical venues, but only shallow integration of social network features (i.e. Facebook [25] and Twitter [26]). Blicko differs in its real-time playback of a common playlist (created together by the listeners) and its (intended) focus towards use in public places.

### 3.2.3 Restrictions and dependencies

Music services are always fighting on several fronts when it comes to copyright issues and licenses. The case with Blicko is sensitive because it is intended to be used in public place. To be able to do this there are two factors that needs to be considered.

Firstly; in order for it to be legal to play back music in a public locale, the proprietor needs to make sure the playback does not infringe on any (national) laws. These laws are often enforced by (national) associations or interest organizations that administer licenses and rights to music and lyrics on behalf of artists (i.e. STIM [27] and SAMI [28] (in Sweden)). These enforcements do only restrict and affect Blicko in an indirect way. This because liability always lies with the proprietor that needs to pay the enforced fees and licenses as soon as he or she plays music “publicly”, regardless of its source (i.e. Blicko, Spotify or radio etc.).

Secondly; all music providers (i.e. libraries) have individual Terms of Service (ToS) or Terms and Conditions (TaC), that need to be obliged in order to (legitimately) take advantage of their material. Many libraries, or the simpler so called: Application Programming Interfaces
(API’s), also demand that their trademarks are thoroughly presented according to certain branding guidelines. [9] Each service provider state their own ToS but they often include a paragraph that grants the user access for “personal non-commercial use” [29] only. This vague exclusion of public use is rarely complied with and most likely only exists to align with the national laws (see previous paragraph). The third party music providers themselves have nothing to gain by excluding public playback, it could (on the contrary) even be a beneficial way to further spread the word about their service.

If for example a cafe decides to use Spotify (through Blicko or not) as the source of music, the proprietor needs to know two things. He or she will have to (as always) pay the fees or licenses to conform to the national laws and to avoid issues of copyright infringement related to public playback. The owner is also, when using Spotify publicly, defying the Terms of Service he or she has accepted by downloading their software (or by simply accessing the service). This invalidation of the agreement is towards Spotify, which rightfully could pursue with a judicial (civil law) process, but has no real incentive to do so. Also if a country ensue laws against direct streaming of music, the provider themselves might be in trouble for enabling access to copyright music and could need authorization to stream. This however, does not affect Blicko since it merely acts as a middle man or channel of which the user can experience, for example: Spotify’s, music through.

There are currently some other dependencies that a Station’s owner needs to be aware of to fully make use of Blicko. However, this is stated with emphasis on: “currently” cause solutions are being developed and implemented as of writing. In order to run the Tuner Application (that plays the actual music, see chapter: 2.3.2) the owner needs access to a Windows PC (with .NET framework [10]). Consequentially the owner also needs a Spotify Premium account tied to his or her credentials, in order to stream music from libspotify [9] with the Tuner Application.

3.3 Voting algorithms

After submitting a song into a Blicko Stations upcoming queue (see chapter: 2.3.3.3), a whole new type of mechanism sets in play. The order of which songs are played is entirely up to the connected users themselves. Each song also has a value that represents its popularity, i.e. vote-count (see table 3). When this value changes a song can either get promoted up the list, meaning it is more likely to reach the top sooner, or demoted down (meaning the opposite). This depends on whether the value increased enough to surpass another track. If two (or more) items share the same vote-count the oldest one (i.e. the one that was submitted into upcoming first) will be prioritized and placed on top of the other(s) in the queue. The voting algorithms are therefore the powerful tool that allows users to affect the order in which songs are played. A specific song’s order in upcoming should perfectly reflect its popularity, at any given moment. However, there are other distorting factors that come into play, which should be minimized by applying the best algorithm for the situation. The terminology for a positive vote within Blicko is a: “Blick”, and respectively a negative vote becomes a: “Block” (see fig. 5).

3.3.1 Differential

A basic algorithm for representing individual popularity with elements (i.e. tracks (see table 3)) in a group (i.e. list (see chapter: 2.3.3.3)) is the “Differential algorithm”. The voter can
inflict either a positive vote or a negative vote for each item, according to his or her preference. An item’s vote-count at any given moment is calculated by the difference between the total number of positive votes and the total number of negative votes. For example: if an item has a total of 7 positive votes and 4 negative, the resulting (differential) vote-count would be: 3 (= 7 - 4).

One distinct factor of this algorithm is the ability to post negative votes, and that only one vote per item is allowed. If users are allowed to change or retract their vote, their new value should override their old one. This gives each user the power to influence each item’s vote-count by the values -1, 0 or +1.

The differential algorithm can be seen on different web sites with different purposes. Their intent is to provide some sort of rating on the content associated with it, whether it is for promoting social news posts (as in for example Reddit [30]) or grading buyers and sellers reliability in web-auctions (as with reputation rating at eBay [31]) for example. The rating is often weighed with a variable depending on the time or age of the item. This solves some of the issues that are related to differential voting, but it still has some inadequacies that need to be taken in concern for it to balance orderly. One of them is the scale of the voters and the items available to vote on. With relatively small groups of people affecting a small list of articles (with few votes distributed) the system can be abused to promote the best interest of single individuals and their articles (see chapter: 5). One solution to this can be to hide the total score of the item until it reaches a certain elevation, this to: “mitigate the bandwagon effect” as Reddit describes it. [30] Another drawback with differential score is that if only the end result is presented, there is no indication on the scale of each side. If the item has acquired a large number of positive votes, as well as a large number of negative votes, the differential will still only show a relatively small number (e.g. 577 - 573 = 4). The amount of votes indicates a type of controversy when getting bigger, a dimension which is completely invisible if only the end result is showing.

3.3.2 Positive only

If the ability to post negative votes is retracted from differential voting, the result would be a “Positive only” algorithm. In this type of promoting algorithm each user can only manipulate an items score or vote by a single indication of approval, i.e. each user can choose to “like” an article or not. Factors influencing a “Positive only”-decision could be for example: the accessibility of the interaction and the effects propagation on social networks etc.

This type of promoting has become widely spread around the Internet and most people are familiar with it because of Facebook’s “like” function. Their (Facebook’s) like button can be implemented anywhere on the web (by means of their official API) and when pressed by a user their profile page will indicate that they “Liked” the item (e.g. post or page) and it will also be presented via their friends news feed. [32]

“Digg.com” [33] was among the first services to implement this type of promoting on their social news website, which lets the user submit and vote on articles (e.g. blog posts, pictures or videos etc.). However, they offer a way to “bury” (i.e. dislike or negative-vote) posts as well, but in such fashion that it is significantly harder to perform in terms of availability. By doing this the algorithms impact can actually be indirectly controlled (balanced) through the
different functions amount of accessibility in the interface. Recently the “Positive only”
promoting mechanism were announced to be adopted by Google, and directly implemented
into search results etc. [34]

3.3.3 Wilson score

One of the more complex score algorithms which can be used to score items in relation to
each other is the so called: Wilson score. [35] It is an approximation of certain items
popularity in relation to number of measurements (votes). The Wilson score is a formula for
a binomial confidence interval that is an enhancement over normal approximation interval
(see fig. 13). Its biggest advantage, over for example the differential algorithm, is that it
reflects properties well even though there is a small number of voters or items. Wilsons
score is a good representation of popularity in between items because it balances the
proportion of positive votes with the uncertainty that comes with a small number of
observations. [35] In this thesis this algorithm will be discussed merely as a possible option
for voting algorithm (perhaps available as an option to the administrators of a Station),
mostly as base for comparison between the other solutions.

\[
\hat{p} + \frac{1}{2n} \frac{z_{1-\alpha/2}^2}{1-\alpha/2} \pm \frac{z_{1-\alpha/2}^2}{2n} \sqrt{\frac{\hat{p}(1-\hat{p})}{n} + \frac{z_{1-\alpha/2}^2}{4n^2}}
\]

Figure 13 - The Wilson score formula for a binomial confidence interval. [35]

In the formula above: “\( \hat{p} \)” stands for the proportion (fraction) of positive and negative
votes. The best representation of vote-count is to use the lower bound of the Wilson score
confidence interval, done by ignoring the plus between the two last terms in the numerator.
The other variable: “\( Z_{1-\alpha/2} \)”, represents the \( (1-\alpha/2) \) quantile (i.e. 95% confidence interval) of
the standard normal distribution, where “\( n \)”is the total number of votes.

However, the biggest drawback with Wilson score is in its inability to be presented in a
distinct way towards the users. A submitted vote would not necessarily represent an abstract
or consistent number, which could be communicated efficiently through the UI. This
phenomenon is demonstrating the importance of, to the user, predictable interaction.
Functions will be ignored if the effect is unclear or its impact invisible to the end user. If it
does not seem like their actions make any impact (or differing impact depending on
situation and invisible factors), then the users will to interact or participate will decrease.

3.4 Potential business models

The idea of Blicko grew out of an idea of personal interest, where a problem or issue had
been identified. In this way it very much resembles a business idea that this chapter will
delve further into by evaluating different possible business models. Because of different
current restrictions the investigation of business models will be for research purposes only.

3.4.1 Licenses

Perhaps the most apparent way of distributing a software based service is by making use of
some sort of licensing fee per user or application. Where, in order for a user to be able to
gain (fully) access to the service, they have to purchase the permission to use (in our case) the Blicko Tuner Application (see chapter: 2.3.2). Licenses can then be restricted in different ways when it comes to time-period or number of active users per license.

The deployment of software with licenses has had very different effects and often results in parallel distribution of illegal copies. The security of an application can be exploited, with for example a “keygen” (serial key generator) [36] etc., and it is very hard to control its restrictions. Without control over distribution it is impossible to sustain license based models since the service depends on it (e.g. if a software is easily attainable for free (although illegally) fewer will actually pay for the license). Therefore a subsequent shift in software distribution and its associated business models has been made lately. Firstly there has been a significant move away from providing: “software products” towards instead providing: “services”, which are often packaged as subscriptions. [37] In turn this change has brought on a new type of subscription based services that constrain the distribution by having the users pay a fee regularly (e.g. monthly). Commonly known examples of this type of subscription based services are for example: Spotify (music) [38] and Netflix (movies) [39]. This type of distribution is made possible by tying the usage of a certain application to a global user account that can be managed by the provider globally.

If applied by Blicko the subscription based service could be designed in regard to both time frame and number of users. The account-holders (i.e. owners of a Station) would need to pay a certain fee to be able to use the service. Favorably this could be combined with a free model that would benefit the spreading of the service. For example Blicko could be entirely open and free for anyone to take part of, up until the Blicko Station reaches five simultaneous users connected. In order for more users to connect, the Stations owner would need to register as a subscriber (and pay the given fee). This model could then be scaled even further where larger groups of users would result in larger fees etc. Licenses could also be tailored as agreements to users (i.e. customers) that have special or significantly larger needs. For example if a branch of coffee shops would like to employ Blicko throughout their franchise a partner contract could be set up according to the specific situation.

3.4.2 Ad and information based

A service, especially a “social” one, is nothing without its users. In order for a service to grow and function properly, its user base needs to be active and preferably expanding. The first growth phase of any business is of course crucial, but even more so when the business model itself is depending on its amount of adaptation among users. One relevant option when it comes to business models for web or software based services, such as Blicko, is an advertisement (ad) based solution. The amount of activity within the service can, by means of advertising, generate enough revenue to further fuel development.

Advertising can be implemented with different depths into a service. One common, superficial, solution is to add banners and links to the global website of a service by affiliate deals and programs etc. The service gains profit by users clicking through the ads, and the more relevant the advertisement is, the more likely visitors will click it. The same method could be brought in even further into the service. In Blicko’s case, where a Station plays back music or sound (perhaps even publicly), the ads could be a part of the upcoming view (see chapter: 2.3.3.3) inside a Station. They could ascend the queue as any other item, but with a different appearance. This type of integration is quite similar to the context sensitive
advertising of Google AdSense [40], which in turn could be used by Blicko. The implementation could even go as deep as letting the advertisement items reach the top of the playlist, where they would be audibly announced through the speakers. This type of integration would be most appropriate when the ads are relevant on a local level.

When a service reaches a certain user base size, its aggregation of events and various usage data grows valuable to other parties. The information within the service and how its users utilize it can be taken advantage of to further improve accuracy of advertisement and more. Even external parties (e.g. representatives from the record industry or media) could find value in the usage metrics (e.g. most played track or artist).
4. Usability Optimization

4.1 Introduction

The field of HCI and Usability are legitimately closely tied together and always extremely relevant to any development involving computers and users. People’s cognitive abilities and intuitions have to be taken advantage of, and the feedback retrieved from their combination of senses has to be considered to maximize any experience.

The recent year’s evolution of portable devices such as smartphones and tablets has likely increased the magnitude of so called “Mobile”-HCI substantially. The different mobile devices and platforms have brought into functions and services from desktop computers and Internet into portable devices that can be used anywhere at any time. However, all these different groundbreaking devices require vast prioritizing when it comes to usability because of their varied formats and functions. One example of this is the recent change in screen sizes with mobile devices, which has nearly doubled in the last ten years (from 1.5” to 3” in average). [41] Another example of recent change in mobile devices is the different input types such as: touch or voice.

This segregation of devices with different formats, concurrent with the spreading of services and application over platform borders, further increases the importance of HCI. Therefore the usability optimization of Blicko has to be performed considering a vast number of factors, related to different platforms, but also different users and situations. In this chapter of “Usability Optimization” these factors will be presented and evaluated (heuristically). The application of methods and concepts will be compared and related to references and in the “Implementation” section (see chapter: 4.3) some of the changes will be implemented and further evaluated. A disclaimer that should be kept in mind is that none of these chapters involve any explicit user testing (as in chapter: 5), but are instead the result of research and comparison with references (implementation of best-practices etc.) and thorough evaluation. All of this assessment and implementation will be performed by the developers and often include direct comparison of Blicko versions and methods such as trial and error etc. (see chapter: 4.3).

4.2 Methods and concepts

This chapter on usability optimization is presenting various factors within the UI (as well as inner functions) of Blicko that can be improved in regards to the users experience. The methods and concepts used are related to HCI and applied by heuristically evaluating each component of the interface, in search for imperfections or issues. Any improvement made to any element of the UI (for instance) is evaluated in all aspects regarding usability and then implemented (see chapter: 4.3) accordingly. Each component should interact the same way (or as similar as possible) regardless of what type of device (i.e. which layout that is being used) the user is using Blicko with. There is an aspiration to avoid compromises as long as possible, and since there is no clear understanding on the allocation of the user base no assumptions can be made.
4.2.1 Design

The layout and design of Blicko, and mainly its Station interface, have been developed according to several factors that will be presented in this chapter. Overall the aesthetic design has been developed towards being as minimal as possible, especially in the Small layout, without afflicting usability negatively.

4.2.1.1 Colors

The user interface of Blicko has been based on five base colors, two of which are the standard black (#000000) and white (#FFFFFF). The remaining colors (see fig. 14) were chosen by considering factors such as: contrast (legibility), compatibility (Web-safe colors [42]) and associations (emotional and recognition based).

“Finally, when selecting colors for a web site, the designers should take into consideration the web-safe color palette. Two hundred and sixteen web-safe colors appear the same across different browsers and platforms.” [43:100]

![Figure 14 - The five base colors of Blicko’s interface. Starting with (from the left): dark grey (#373737), orange (#FF9933), grey (#666666), white (#FFFFFF) and black (#000000).](image)

The background consists of the two grey hues, depending on interface-element availability and focus. The grey theme is inspired by the Spotify desktop client, as well as other software for music playback (e.g. Winamp [44] and iTunes [45]). This conscious choice was made to familiarize the user, and minimize the initial amount of new impressions. A very frequent element in Blicko’s UI is: lists that consist of rows scaling downwards. When presenting this type of information in long lists there is an imminent risk of items floating together (visually). This phenomenon is prevented by means of two methods in every section (i.e. “Upcoming”, “Search”, “Users” and “Feed”) of a Station, namely: alternating row colors and demarcating borders. Alternating row background color is also known as: “Zebra Stripping”, and proven to increase readability of lists and tables (especially with multiple columns and spacing) dramatically. [46] The background color is changed to a slightly darker grey hue on every other row by means of a JS when populating the lists. The top and bottom of each items row is also banded with a dotted border (in a lighter grey), in order to make a distinction between tracks or events etc.

The majority of the text-elements in the UI are either solid white or black, depending on the background (which is inverted), to create higher contrast and consequently increase legibility.

“Optimal legibility requires black text on a white background (so-called positive text). White text on a black background (negative text) is almost as good.” [8:125]

In browsers that supports it, the CSS property: “text-shadow” were used to outline the text even more. Grey text is also present in the interface to display information of lower
hierarchy (less significance), and text that is clickable (links) is often indicated by its orange color (either by default or on hover).

The accent color: orange, combines aesthetically with the grey tones and will possibly be the color most associated with the trademark: Blicko. It constitutes the majority of the logo and is also often displayed in different hues. Erik G. Nilsson describes this as a branding mechanism which originates from the overall graphical profile of the company, which reappears throughout the UI’s. [47] Peoples associations with orange are quite binary in that they are either rather positive or negative (i.e. rarely in between). The color is supposed to affect us by for example: stimulating activity and encouraging socialization. [48] Two factors which are highly relevant to the “social music service” of Blicko.

Another UX-optimizing method thoroughly applied in the interface is: gradients. The seamless transfer between two shades of a color smoothens the interface and adds a bit of depth and dimension. [49] This factor is taken advantage of in interaction-elements such as buttons and text fields etc., and can indicate feedback when manipulated by for example inverting the gradient on hover. The gradients are mostly applied by setting the CSS property: “background-image”, as the URL to a “.png”-file that are 1px wide and repeated horizontally (in benefit of page size).

“Subtle gradients and "soft edges" (created by subtle drop shadows or embossing) lighten the visual impact of the screen on the user, making it actually easier to use.” [49]

4.2.1.2 Fonts

The typefaces used in the design of Blicko were kept down to a number of three, which corresponds with Nielsen’s recommendations. [50] This in order to minimize the noise created by mixing several different typefaces in the same UI. The sans-serif font: Helvetica, is used throughout the vast majority of the text (including: headings, track-titles and artist etc.), and a more uncommon font: Agency FB, is used in logos and links with images. The fonts were coherently designed for both layouts: Small (see chapter: 2.3.3.2) and Big (see chapter: 2.3.3.1). Since the benchmarking point were to keep the typeface as readable as possible while being as small as possible, most of the font assessment were done in Small (and then subsequently reviewed in Big).

![Agency FB](image1.png) ![Helvetica](image2.png)

*Figure 15 - The two fonts used in Blicko: Agency FB (bold) and Helvetica.*

When selecting a typeface for the website one of the biggest factors where legibility when text is displayed in a small size. Since Blicko is developed to work on a wide range of devices, including for example smartphones, the resolution and size of the screen can be very restricted. This factor should not compromise the user experience and therefore we mainly focused on sans-serif fonts, which are easier to read on digital screens. [43] Both Helvetica and Agency FB met the expectations when applied to the Blicko UI in small sizes. This after comparison with for example: Verdana and Arial, which proved to be wider and therefor insufficiently narrow. The declaration of typeface in Blicko, by means of the CSS
property: “font-family”, were set as follows: “Helvetica, Arial, Tahoma, sans-serif”. This list is used by the user’s or client’s browser to prioritize which font is used, starting from the left and using the first available. Agency FB was chosen because of its relatively clear and condensed appearances, especially in bold type, which makes it ideal for even small graphics (e.g. icons and logos).

A complementary serif font was also used in the Global site of Blicko. A current trend within web design state that the use of serif fonts will not necessarily impact legibility negatively, because of the constantly growing screen resolutions and enabling technologies (such as ClearType). [51] Since the text size on the Global page is larger than the others, and the content is kept in fairly short snippets, a serif font suits it well. The primary font used in this occasion is: Cambria, but since it is mostly distributed via Microsoft’s OS’s, a secondary equivalent (standard) font was declared as: Georgia.

4.2.1.3 Metaphors

Blicko uses a variety of different metaphors (i.e. visual cues such as icons) to make the user feel more familiar with the UI. They play a vital role in making the UI more user friendly since they require neither a specific language nor reading capabilities to be understood. Many of the metaphors have real world relations and associations which then transcend into the interface and aid the user in their navigation.

“A metaphor needs to fit both in terms of the conceptual interface and in terms of the visual interface.” [43:96]

Metaphors can of course easily be misinterpreted and it is therefore very important that they are only used when they truly fit in the conceptual and visual interface. The visuals are chosen so that the user quickly can understand what the purpose of a function is. For example the “Search” view has a magnifying glass as an icon which is very common in the computer world and is almost always representing “Search” (see fig. 16). In equivalence a real magnifying glass is used to observe minuscule details, which makes it a good metaphor for “Search”. The “Search” view is one out of four navigation pages which all use metaphors as a mean of identification.

![Figure 16](image16.png)

Figure 16 - One use of metaphors are for the search function. The “Search” view has both a magnifying glass and text while the search button itself only has a magnifying glass.

Blicko’s vote function (see chapter: 3.3) is another example of where the use of metaphors is enhancing usability. The labels “Blick” and “Block”, together with the metaphors (see fig. 17), indicates their function. The colors of the arrows helps to emphasize their impact on vote count by representing positive (green) and negative (red) actions.

![Figure 17](image17.png)

Figure 17 - Blick and Block has an up arrow and down arrow to indicate the increase and decrease of votes which results after a click.
4.2.1.4 Content

The textual content of Blicko have been designed according to the “Web Usability”-fact that users do not read web pages (in the true meaning of the word), they scan them [52]. Therefore all text, such as: instructions, titles and general information, should be kept as short and concise as it is possible in order to further improve its readability (especially important in the “Small” layout). Jakob Nielsen calls this method: “writing for scannability”, and describes how the author of a text that is to be presented on the web, should intend for half the textual amount in relation to if it were to be used in printed media. [8]

“We’re thinking “great literature” (or at least “product brochure”), while the user’s reality is much closer to “billboard going by at 60 miles an hour”.” [52:21]

Inside a Blicko Station most of the content is presented as items in list (mostly tracks). Inside “Upcoming” tracks are displayed descending according to vote count, and inside the “Search-view” query-results are listed similarly. Items are presented with their relevant data, and the whole (rectangular) area of the interface they fill is clickable, not merely the text. All the titles and text are as concise they can be and descriptions have been taken out where the content is obvious in of itself. An example of this is that there are almost no labels visible in the interface at all, e.g. “Artist:”, “Album:”, “Track:” etc. The differences in visual appearance between the information-pieces are a part of its hierarchical formatting, and also because it is similar to how other music playback software group information.

The Global (see chapter: 2.3.1) website have also been designed so that the first instructional (and most apparent) text on how to: get, setup and use Blicko is divided into a list and a title, much similar to what Nielsen calls: “Scannable Layout”. [8]

4.2.1.5 Prototyping

A method thoroughly used in development of this project is the concept of: prototyping (and mock-ups), especially when it comes to the user interface aspect. Closely related to agile development methods; prototyping demands an iterative approach and regular re-evaluation of (in this case) the UI. Initially a mock-up of the interface was sketched using Adobe Photoshop [53] (and on paper before that), with different components in separate layers (see fig. 18). This was before any actual functionality existed at all, in order to get estimation on how it would look and which elements it would consist of. Basic design choices, such as fonts and colors, could thereafter easily be manipulated and the composition could be experimented with by moving and switching components. This type of mock-up relates to so called: Horizontal Prototypes, which represents a “dumb” or “shallow” interface with no or little functionality. [54] The process of using these prototypes can also be categorized as: “evolutionary”, since they are used iteratively throughout the development. The interface evolves through several generations (or versions) and for each step further functionality is added. [54]
Figure 18 - First mock-up of main screen interface done in Photoshop [53] as a flat image, with UI components in different layers.

In the process of prototyping the visual mock-up image were translated into HTML and CSS (see fig. 19). The different sections were made into rows and columns of a <table> element with the corresponding styles attached, which in turn often is referred to as a Wireframe representation. [55] This basic foundation of a website interface is where the functionality is added, which eventually will end up as the final release of Blicko’s client version.

Figure 19 - Wireframe version of mock-up in fig. 18, constructed with HTML and CSS.
The prototypes were only used internally to test different aspects of the design in the development process, although they could have been used in early prototype-testing with users as well. This was neglected because we were eager to enable functionality, and thereafter the iterations came rather frequently. However, at several stages within the development of Blicko an idea of a new component has surfaced, that were firstly prototyped in Photoshop in order see how it would look inside a context of other components.

4.2.2 Optimization and flexibility

Since Blicko is to be used on a very wide range of devices, with diverging screen resolutions, the dynamic scaling of the UI is a crucial factor when it comes to usability. The interface is supposed to be equally intuitive on a mobile screen, as on a HD-monitor.

4.2.2.1 Dynamic scaling issue

Dynamic scaling is a factor related to the size of content in relation to the screen resolution of the device that displays it. Since resolutions vary, the content has to adapt and be constantly flexible. Not only does this produce difficulties because the typical smartphone screen pixel count today is a mere sixth to that of a monitor \((1280*720)/(320*480) = 6\), but because the aspect ratio is almost completely reversed (see fig. 18). The width in relation to the height of a computer screen, has inverted proportions on most mobile devices (i.e. the height is superior to the width). Most of the content on Blicko is presented as lists of items within both Small and Big layout, such as tracks in playlists or results from search queries. This enables for easy vertical scaling by adding scrolling functions, but it also generates issues related to the input types of different devices (see chapter: 4.2.2.3). The consequence of larger resolutions vertically, is simply an addition of more list-items that can be viewed instantaneously.

The crucial and much more severe factor is the horizontal scaling. To manage this concern all the widths in the UI have been defined as relative (percentile) values, in order for it to adapt to the resolution of the device being used.

“The main principle for resolution independent design is to never use a fixed pixel-width for any tables, frames or other design elements.” [8:29]

No horizontal scrolling should ever be needed in order to prevent the users from losing context. [47] The issue is based within the traditional comprehension of a webpage. It is extremely rare to see pages that extend the width of a screen and forces its visitors to (manually) scroll to the left or right. This apprehension of webpages has traversed with its user over to the use of a mobile platform as well, and is therefore also critical in the Small layout.

The issue with horizontal scrolling poses for substantial difficulties within the UI of a Blicko Station because no single row of the interface can at any time stretch further than the width of a relatively narrow mobile interface for example. Although the UI still needs to be aesthetically pleasing on a wide aspect-ratio device such as a widescreen laptop or monitor, without filling it with too much of negative blank space and empty areas (see fig. 20). The Full HD and HD resolutions also have an aspect ratio of 16:9, while the HVGA display (which is common on smartphones such as iPhone [14] etc.) has 2:3 (standing) proportions.
A possible compromise to the difficulty with reversed aspect ratio and narrow width of mobile screens is to enable landscape positioning of devices. The interface would then be rotated to fit the display lying on its side which would result in a UI with more width than height (the same as with monitors). However, this alignment compromise was swiftly dismissed when tested. The resulting height was not adequate to our vertical- and list-oriented composition of the UI. In landscape mode the user could only see a couple of items at a time without scrolling, which negatively affected the overview and “scannability” (see chapter: 4.2.1.4) to such extent that it became nearly unusable. Therefore Blicko’s main orientation on mobile devices is standing, due to the vertically concentrated and arranged content, and landscape mode has no priority on smaller resolutions.

4.2.2.2 Response time

One factor that always has to be taken into account when developing a web based service (of any type), especially when it is available on a wide range of platforms, is: response time. [56] Jakob Nielsen often writes about the importance of response times, and has concluded that there are three basic thresholds:

- “0.1 second is about the limit for having the user feel that the system is reacting instantaneously, meaning that no special feedback is necessary except to display the result.” [57]

- “1.0 second is about the limit for the user’s flow of thought to stay uninterrupted, even though the user will notice the delay.” [57]

- “10 seconds is about the limit for keeping the user’s attention focused on the dialogue, … without wanting to perform other tasks while waiting.” [57]

Response time is an important factor that is closely proportional to satisfaction. A lot of people lose interest and focus when the response time is too long. There are few things as frustrating as waiting for something to load on a computer, for example when: starting up a
program, booting the computer or opening a web page. If the web page does not load quickly the overall satisfaction will go down drastically. By prioritizing response time Blicko can minimize the stress for the users. The far most effective method for doing this is to avoid as many requests between the client and the Station (in Global) as possible. In Blicko this is done by loading the entire Station with its views once directly upon entering it (see table 5). The rest of the manipulations, which affect the UI in some way (i.e. casting a vote on a track item), is done by means of: Asynchronous JavaScript and XML (AJAX). The manipulations triggers certain events that periodically gets synchronized with the Station (at Global), and then propagates to all connected clients. The responsiveness of a system is defined by the time it takes between manipulation and feedback. When refining this into design metrics for how responsive the Blicko UI is (or should be (see chapter: 4.3.1)), Nielsen’s thresholds (see list above) were used in two different ways or types (see table 4).

<table>
<thead>
<tr>
<th>Design Metric</th>
<th>Ideal response time</th>
<th>Example manipulation triggering it</th>
</tr>
</thead>
<tbody>
<tr>
<td>Load response</td>
<td>&lt; 10 seconds</td>
<td>Connecting to a Station (with loading of views)</td>
</tr>
<tr>
<td>UI response</td>
<td>&lt; 1 second</td>
<td>Any internal manipulation (i.e. clicking link, inputing text etc.)</td>
</tr>
</tbody>
</table>

*Table 4* - Design metric for responsiveness of UI, when loading (“load response”) a Station or simply manipulating an internal element (“UI response”).

Some approximate results of measuring the response time of certain actions within a Blicko Stations UI can be seen below in table 5. Compared with the thresholds stated by Nielsen and our defined design metrics (see table 4), all our values were lower than the critical 10 second limit (i.e. “Load response”). This with approximations of average download speed for households as: 26.5 Mbit/s [58] (i.e. visitors with Big layout), and for 3G connections as: 0.85 Mbit/s [59] (i.e. visitors with Small layout) in Sweden. Although with numbers like these (close to 10 seconds) it is crucial to indicate to the user that the system is working (or not), and therefore Blicko uses different techniques to give feedback of the current state (see chapter: 4.2.3.6).

<table>
<thead>
<tr>
<th>Type</th>
<th>Triggered</th>
<th>Data - Big</th>
<th>Time - Big *</th>
<th>Data - Small</th>
<th>Time - Small **</th>
</tr>
</thead>
<tbody>
<tr>
<td>Initial load</td>
<td>When entering a Station (with nothing cached)</td>
<td>~1.54 MB</td>
<td>465 ms</td>
<td>~0.75 MB</td>
<td>7100 ms</td>
</tr>
<tr>
<td>Refresh load</td>
<td>When refreshing browser manually inside Station</td>
<td>~0.50 MB</td>
<td>151 ms</td>
<td>~0.45 MB</td>
<td>4240 ms</td>
</tr>
<tr>
<td>Incremental updates</td>
<td>Every five seconds</td>
<td>11 B</td>
<td>0.003 ms</td>
<td>11 B</td>
<td>0.104 ms</td>
</tr>
<tr>
<td>Album cover</td>
<td>After each song</td>
<td>~100 kB</td>
<td>30 ms</td>
<td>~7 kB</td>
<td>66 ms</td>
</tr>
<tr>
<td>Aggregated in 1 hour</td>
<td>N.A.</td>
<td>~2.12 MB (see fig. 21)</td>
<td>640 ms</td>
<td>~0.71 MB</td>
<td>6700 ms</td>
</tr>
</tbody>
</table>

*Table 5* - Showing approximate data loads. Measured with Google Chrome 11.0 and with * = 26.5 Mbit/s (average household download rate in Sweden) [58] and where ** = 0.85 Mbit/s (average 3G download speed in Sweden) [59] as average speed.
The table above is showing some approximate, depending on use, browser and cache settings, measures of data load when accessing a Blicko Station (in current state). The benchmarking were done in Google Chrome [16] with cleared cache initially (first row), and with a user agent [13] extension that acted as an HTC Hero [60] device for the Small layout testing. Then the Station was manually refreshed (second row) through the browser, and then the incremental updates started to drop in (third row). This was done for an hour and then the final amount of data were noticed (see fifth row in table 5 and fig. 2f). After loading a Station the album cover (see fourth row in table 5) stood for the majority of the data, as can be seen in the example calculations below.

An important disclaimer about the content i table 5 is that the last row of aggregated data should only be seen as a solitary example of use, where the Station was left idle in “Upcoming”. This since the data normally depends on how many interactions a user performs within the time period. However, it is interesting to note that the size of the incremental updates on the Station’s state is 11 B every five seconds. This totals out to about 8 kB over an hour (11/5*3600), but on top of this is an additional download of album cover image (once for each song played). The 174 x 174 px image in jpeg format is ~100 kB (in Big and ~7 kB in Small (65 x 65 px)) in size and changes every four minutes or so (average song length (approximated)), which enumerates to about 1.5 MB per hour. In total the calculated amount of data would be approximately: 0.5 MB (refresh load) + 1.5 MB (album cover) + 0.08 MB (incremental updates) = 2.08 MB which is very close to the: 2.12 MB measured (see fig. 2f). The same calculation can be done for the tests with Small layout: 0.45 MB (refresh load) + 0.10 MB (album cover) + 0.08 (incremental updates) = 0.63 which is also (proportionally comparable) close to the: 0.71 MB measured.

The bottleneck for Blicko in terms of response time, is when a user loads the main page and when the user connects (and therefore loads) the Station view (see table 5). By keeping images small, making effective scripts and minimized data or content the loading time can be reduced to as little as possible. Even though the average Internet connection is better today there is a parallel (and radical) shift occurring, where more and more people use mobile devices to access the web. This reformation is crucial to Blicko because there lays a priority to keep a Station as accessible as possible, especially with mobile users at for example venues. Sometimes the venues themselves offer access to a WiFi network with (relatively) fast Internet access, which would minimize the issue with response time.
However, it is perhaps more common that users connect their smartphones to the Internet via 3G networks. Even if a venue would be full of guests (even hundreds) the amount of traffic would only peak if many users connected simultaneously since the whole Station is loaded at once upon entering. The exception being traffic like the incremental updates (every five seconds) and album covers (every four minutes (or so)), which is marginal because of their diminutive size (11 Bytes and ~7 kB). Unfortunately no thorough stress evaluation data can be provided on larger scale usage scenarios (due to uncertainties with too many factors such as available bandwidth and amount of clients and events etc.), but please see chapter on “Collaborative user test”: 5.2.2 for references on more than 10 simultaneously connected clients.

One reason for introducing Small (see chapter: 2.3.3.2 and “Separation of interfaces” chapter: 4.3.1) was to keep the response time down for mobile users. This is done by using a more simple design with smaller and fewer images. Another way Blicko tries to reduce response time for users are by minimizing the amount of HTTP requests. In Blicko this is done by loading the entire Station page in one go with all the subpages to a Station page included. This also gives a better response time when navigating inside a station.

4.2.2.3 Input methods
The evolution of smartphones brought a whole new type of input method with them, namely: input by touch. The touch screen interfaces of today's generation of mobile devices do not only require a quite different motorical manipulation in comparison to the classic mouse and keyboard, but it also leaves a different impression inside the system. The most obvious dissimilarity being a loss of position input during motions. This meaning that the “cursor” input is either being clicked or not (binary), and the actual movement in between clicks is undetectable to the interface or underlying system. When interacting with a mouse as input device the movement is frequently recorded because the system always has its current position within the interface. Therefore such interface effects as “hover” loses their function when manipulating the UI with a touch screen.

Another issue that emerges is related to functions that a mouse has dedicated buttons for. Two quite ordinary functions like this are: the right click context menu and scrolling with the middle wheel. The right click function of Windows OS’s has been emulated with touch screen devices by registering a long press (pushing the finger down for about a second) on for example a list item. This function does not really affect Blicko’s interface since we have no intention to incorporate any other manipulation to items than clicking.

The scrolling however is a far more advance issue, which is further explained in chapter: 4.2.3.4. The key lies within the ability of touch screen devices to recognize the duration of a click. A fast press followed by a stroke, upwards or downwards on the screen, scrolls the UI in the according direction. A slightly longer press and release will instead be interpreted as an ordinary click.

4.2.3 Navigation

“Navigation interfaces need to help users answer the three fundamental questions of navigation:
- Where am I,
- Where have I been,
- Where can I go. “ [8:188]
4.2.3.1 Persistent navigation

The quote above by Jakob Nielsen, associates to a concept of persistent navigation, in which the user always feel confident of his or her position within the UI. Having a coherent navigational frame enables the users to imagine their current orientation, as well as available choices of traversal. It keeps the visitors from having to guess and use trial and error to find what they want. Blicko’s interface has been developed with this concept of persistent navigation as a requirement, and several choices cohere with its features. Of which the most substantial component is the tabular menu (see chapter: 4.2.3.2). The menu will account for the majority of navigation activity within a Station, and it is absolutely crucial that it accurately indicates user orientation. However, persistent navigation is also a factor when it comes to: the use of metaphors (icons) (see chapter: 4.2.1.3), dialog boxes, fonts (see chapter: 4.2.1.2) and colors (see chapter: 4.2.1.1) etc. The same manipulation of a similar interface element needs to consistently give similar (i.e. familiar) results throughout the interaction experience. Inside a Blicko Station; all dialog boxes has the same size, their content looks uniform and their available manipulations are the same. The style sheets (CSS) are shared between components, and even sites. This does not only facilitate a consistent aesthetic theme, but reduces the amount of data transferred.

Nielsen describes how no instructions should be necessary for the user to manage navigation within an interface, it should implicitly indicate on where to begin and what the available options of traversal are. [52] Any link of navigation within Blicko’s interface has very similar indications of feedback when manipulated. When hovering the item or link the background often turns into the characteristic orange hue (or a gradient of the same), and when clicked its outline changes (inverts). The tabular menu (see fig. 23) always stays in the bottom of the interface, with all of its tabs constantly accessible and showing. However, depending on which tab the user is currently browsing, the look of the remaining tabs might change to indicate availability (i.e. “implicit” recommended direction of traversal). This enables the users to perceive where they came from and how to get back, but also the sorting of the tabs (from left to right) indicates a prioritized hierarchy of information on each tab. All navigation options within a Blicko Station stay consistently in the same position when moving in between pages, and they (i.e. buttons and links) often change appearance upon availability.

4.2.3.2 Tabular menu

The menu within Blicko’s UI has had many shapes in its different iterations of the site, although they have all been somewhat similar to that of a tabular menu. The main sections of a “Station” are accessed through buttons on the bottom of the Big interface and the buttons appearance has indicated current orientation. The selected tab is directly attached to its content without any borders, indicating the user’s current position within the Station. Whereas the other tabs has a different color and distinctly marked boundary to the content above. One could say that the tabs shift focus depending on which part of the interface the user is currently visiting, much like its “real world” equivalent in tags or tabs on archive folders, files or index cards (see fig. 22). This is also one of the four reasons promoting tabular navigation according to Steve Krug (i.e. “suggests physical space”). The other three positive factors with tabs, that he mentions, are: “self-evident”, “hard to miss” and “slick”. [52:80-81]
Best practice while implementing tabular navigation is to keep high contrast between foreground and background tabs. [Reproduced with permission [61]]

Erik G. Nilsson describes tabs as: “Finger friendly menu choices” because of their often large clickable areas, which is perfect when it comes to user interface design for mobile applications [46:6]. Since a Blicko Station site have to be easy to use with for example a smartphone, it is vital that the navigation is adapted to fit the same. Although, the tabular menu later had to be removed from the Small layout (see chapter: 4.3.1), it is still an appropriate factor when a Station is accessed by a tablet etc. The tabular menu is positioned near the bottom of the interface to be easily accessible with fingers (i.e. thumbs) on a touch screen device, without the need to change from a one-handed grip. One reason why tabular navigation is more difficult on smaller screens (such as mobile devices) is that they often require a quite wide interface. Each tab with textual label can only be as small as the width of the tab-title, or as small as possible for it to be conveniently clickable.

With several tabs, five in this case (see fig. 23), always showing in the interface (to maintain persistence) the width easily gets overflowed on most devices. A compromise made to deal with this issue was possible because of metaphors (see chapter: 4.2.1.3). Descriptive icons can be added to the tabs and a dynamic width-function that automatically re-size them depending on user focus. If the user goes from one tab to another, the text-label only shows on the active tab, and all the other tabs (i.e. unfocused) are narrowed down to only show their metaphors. This way the main tab (currently focused on by the visitor) occupies most of the width, but the content of the other tabs is presented by means of their icons.

The tabular menu is only accessible through the Big layout and is therefore not present in small. Instead a grid navigation page is shown (see fig. 6), that has the same corresponding icons to each page. The icons that are representing the content of each tab, for example magnifying glass for “Search” and person (or avatar) for “Users”.
4.2.3.3 Back button issue

While surfing the web one of the most important buttons is the back button of the browser itself. It is used to reverse the browser back to the page last visited.

“Hyperlinks were by far the preferred method of traversal, accounting for 52% of all document requests. Second, accounting for about 41%, was the “Back” command.” [62:4]

Using the back button while browsing is one of the most basic commands a user learns and uses. The functionality has been around for as long as Internet browsers has existed [63] and can be considered as deeply rooted interaction behavior. The way that Blicko is designed raises some problems for the users when it comes to using the traditional function of the back button.

When a user enters a Station with a laptop all of the subsections of Station are loaded directly. This means that when the user presses the back button of the browser, they will leave the Station completely. This regardless of where inside a Station the user currently is situated. The issue that immediately arises is that whenever a user wants to go back to a previous view, and does so by instinctively clicking the back button of the browser. Then the result will be that the user actually leaves the Station. An example of this could be when the user first enters into “Upcoming” but then decides to continue into “Search”, in order to find some songs to submit. After submitting a few songs the user wants to go back to the “Upcoming” view and see how his or her songs are doing in the playlist. This is the situation where the problem with the back button can appear. The user recognizes “Upcoming” as the last page visited and expects the back button to take him or her there. However, a use of the back button would instead bring up the Stations view, which was the last actual page visited. “Upcoming” is simply a view within an enclosing page that can change depending on the internal navigation via tabular menu etc. The same phenomena appear when using Blicko on smartphones where a physical “return” or “back” button has the same function. The issue can become even greater because the different layouts of a Station in Small and Big. Because of the somewhat different designs and navigation systems it is harder for the user to know, at any given moment, that he or she is browsing “internally” between views when in Small. It is not as apparent that the entire Station with all of its subsections is loaded in one go, upon entering. Although, it will not make any noticeable difference until the user wants to go back.

This issue is common amongst sites that incorporate internal navigation and it can be addressed in two ways. The first involves changing the history field of a browser that the back button calls on when pressed. It could then act as a refresh button instead of a back button while visiting a Blicko Station. However, this method is vastly discouraged because it traps the visitor without the users consent or knowledge. The other way to handle the issue is by means of a pop-up dialog that informs you that you are about to leave the site (also see chapter: 4.3.3). [64]

4.2.3.4 Scrolling issue

While using Blicko the user will encounter lists in almost all views of the UI. Therefore scrolling inherently is an important feature of manipulation for Blicko. When visiting a Station within the Big layout, used in an ordinary browser, scrolling works exactly as
expected. The issue emerges when using a smartphone to access via the Small interface (see chapter: 2.3.3.2).

When surfing on regular web pages with a smartphone, scrolling works as it should, by quickly pressing down and swiping upwards or downwards. Although, when trying to scroll within a page that has several frames of content that overflows (see fig. 24), a problem arises. What happens is that instead of getting a scrollable frame, the content in the frame gets cut and scrolling gets disabled. This affects all sites that have a fixed position UI element within a page, especially when the element is placed at the bottom of the page since it will not able to scroll below it.

![Figure 24](image)

*Figure 24* - Example of a page where scrolling could be a problem for a smartphone running Android. The frames overflow which is indicated by the scroll bars.

This is a widely known issue that is most apparent within the Android built in browser. [65] The issue lies in how the Android browser handles the CSS attribute overflow, which controls the scroll bar when the content overfills the frame. The iOS has solved this by incorporating a double-touch scroll, to scroll inner frames. However, since this functionality is far from intuitive and not widely known of, the issue remains the same. There are existing workarounds addressing this problem. One of which is to use a JS solution called: iScroll. [66] iScroll was initiated for the single purpose to scroll content inside a fixed size page element.

### 4.2.3.5 Search (function)

One of the main functions of a Blicko Station is the ability to search the song pool set up by the administrators, in order to submit tracks to the “Upcoming”. The search window is
allocated in its dedicated tab, accessible via the tabular menu. It is labeled by means of an magnifying glass icon, and the search field itself should look very familiar to any web user (see fig. 9). It consists of a text field for the input query that occupies 70% of page width, and the button to submit the query constitutes the rest. The search field was widened as much as possible, according to the fact that it actually influences the user’s expectations of the probable size of the query string. [67]

The text size in the text field is set to fairly large, in order to further increase usability on smaller screens and to accommodate users with visual disabilities. One can also post the typed search string by pressing the return-key, as in common fashion. When first entering the search-tab the user is informed by text on which music libraries that the host is connected to, and the text-marker is automatically positioned within the input field.

The search function is designed in a minimal approach and has been stripped down to the bare minimum of functions, which is analogous to Nielsen’s view. [68] There are no options available for the search (i.e. radio buttons for track, album or artist), which align with Steve Krug’s point of view on whether to provide detailed scope options for search functions, or not. [52] Any search string will be matched against the connected libraries and compared with track-, artist- and album-titles at the same time. It is a common convention to handle searches like this and any results will be returned as a number of track items (i.e. no single item for artist or album). Since the query to the music library provider (via its API) often is without scope, there is no need to provide alternatives since the extra dimension will be lost anyway. Jakob Nielsen expresses the following on the topic about scoped search:

“Scoped search lets users limit the search to result from specific areas of the site (the search scope). In general, this is dangerous. Users often overlook the scope, or they think they are in a different site area than the one they are actually searching” [69]

Directly after submitting the query, either by pressing the button or return key, a loading indicator is showing in the main content area of the search page. The moving (.gif) animation shows that the query is being processed and indicates that the input was indeed submitted. As soon as results are returned from the connected libraries, they fill up the results page as track items. The items will be sorted after the default sort from each provider, for example Spotify track will be displayed in descending order according to popularity (exactly as they would in the Spotify Desktop Client).

4.2.3.6 Constant user feedback

One of the drawbacks of choosing to develop an application for a range of platforms (in contrary to just one) is the compromise when it comes to responsiveness of the software. An application specifically made for iOS (for example) will always be faster and more exact, when it comes to interaction, than a solution developed to run on both smartphone platforms and in browsers. This since there is better resource control and lesser layers between the UI and the actual hardware doing the computations. Blicko’s Station is built as a web application (involving: HTML, JavaScript, AJAX and CSS etc.) and is far from tailored to specific devices or platforms. This makes the concept of “Constant user feedback” critical and hard to manage, because of the increased layers between hardware resources and actual application. The concept involves factors that help the user, instantaneously and continuously, be aware of the systems status. An example of this is that the user should never doubt if interactions with the Station are available or not. The
Station’s status should always be transparent so that the users know when he or she can interact with it. When the Blicko Station UI is busy with, for example, posting a search query and retrieving its results, the main part of the interface is occupied with a loader to give feedback that something is indeed happening “behind the scenes”. This is done in any situation where the response time succeeds our pre-defined design metric interval for “load time” show in table 4.

![Loader example](image)

**Figure 25** - Three animated loaders that gives the user feedback whether interactions is available or not. The different animations are used in different places within the Blicko UI, for example depending on current background color.

Another example of animated user feedback is the progress bar. Not only does it show how long a song has been played, but also indirectly indicates to the user that the interface is working. Movement indicates that there is no deadlock and that the UI should be ready for user interaction soon. When the user then decides to manipulate an element of interaction, the feedback should be given instantly. An example of this is the hover effect Blicko’s Client has on items and buttons. When the users move the mouse cursor over an item or button its appearance changes in order to indicate that it is clickable and that the UI is responding. For example: when the cursor moves over a track item in the upcoming playlist its background turns to an orange gradient and the mouse-pointer changes. Then when the element is pressed (i.e. clicked) other subtle changes are swiftly done to its looks, but only whilst it is being manipulated. This method of constant user feedback raises some issues in a Stations UI because not all interaction will be done via a mouse and keyboard on a computer. The very relevant alternative being touch input from smartphones or tablets (see chapter: 4.2.2.3). When touch input is used instead of a mouse, one dimension of interaction is lost because there obviously is no indication of “hovering”. Therefore the next level of feedback on UI elements (which is the same as it is with a keyboard and mouse) is during its manipulation itself.

### 4.3 Implementation

The heuristic examination of each component and function, in regard to their usability, within (primarily) a Stations interface resulted in many corrections and adjustments. Some of these factors were of greater significance than others. This section is dedicated to the implementation three of these adjustments that were felt to make the most difference. The corrections have been done iteratively with design, implementation and evaluation with each change. These iterations will however not be described in depth, but the examination processes will be presented tied to each example. The examples of issues addressed below are presented according to issues and features mentioned in the previous chapter (“Methods and concepts”). Please note that chapter: 2 on the architecture of Blicko (primarily chapter: 2.3.3) are presenting the state of the Station UI (etc.) after the changes mentioned in this section were implemented. It provides for good comparison but contradicts the chronological order implied (in order of report).
4.3.1 Separation of interfaces (layouts)

During implementation of changes to Blicko’s interface and function several serious factors lead to an inevitable separation of the interface of a Station. The separation has been touched upon previously and the two resulting “offspring” layouts are called: Small (see chapter: 2.3.3.2) and Big (see chapter: 2.3.3.1). Where Big is the old interface (best suited for computer screens or tablets) without any adaptations, and Small is the layout customized to fit smaller devices such as smartphones. The separation of these interfaces came from aggregation of several critical issues that will be further explained in this chapter. The problems would not allow for compromises as long as only one UI were being used and the differences between the resulting layouts can be seen in table 2 (see chapter: 2.3.3).

The most crucial problem with having one unified layout proved to be the scrolling issue (see chapter: 4.2.3.4). This issue brought severe responsiveness problems with it, that were nowhere near our defined boundaries for “UI response” (see table 4). The issue made it impossible to use the vertical scroll on Android [15] based devices A quick implementation of iScroll [66] resulted in very varying scroll-behaviors when tested with a HTC Hero [60] (Android) and an iPhone [14] (iOS). The scrolling worked but it was at the expense of responsiveness in Android and this to such a degree that it was doing more harm than good. In Android the list would scroll but for each stop it would take several seconds before the (then implemented) tabular menu popped back up at the bottom. The menu disappeared when scrolling and almost seemed to stick to its position in the list temporarily. This continued until movement ceased for about one second (estimated), which then caused the menu to snap back after a total of approximately 2-3 seconds. The same procedure where simultaneously conducted on an iPhone where the experience were much more pleasant. In comparison one could say that it was unbearable jerky (i.e. greater delay than: “UI response”-metric defined in table 4) on the HTC while seamlessly fluent on the iPhone (i.e. delay < 0.1 seconds (estimated)).

The parting of the interface was inevitable and when the Small layout had been created, with its own navigation (see fig. 6). The solution to the issue was surprisingly simple. By removing the tabular navigation from the bottom of the Small interface, workarounds were no longer needed. This since the page could now be scrolled, as any other page that grows in height, “downwards” when the lists inside each view overflow the screen. Because there were no longer any fixed UI elements at the bottom the lists could be scrolled as expected in any platform or device (with a browser).

The separation of Small and Big has proven beneficial when it comes to many factors. The responsiveness of a Station’s interface could be refined individually for the layouts. The width of the tabular menu (see chapter: 4.2.3.2) was no longer a concern since a new navigation was brought in (see fig. 6). Interface elements that only renders on hovering (i.e. mouse-over) could be removed completely from Small, since they would be useless with touch input. The amount of data transfer could be minimized exclusively for mobile devices, by for example scaling down images and components that would be unnecessarily large for a small resolution screen (see table 2). The effect of all these optimizations would not only decrease response time, but also be advantageous because many people use mobiles on contracts that have data restrictions (see chapter: 4.2.2.2). The separation of interfaces also enabled a customization to be excluded from the Big layout that was related to the back button issue (see chapter: 4.2.3.3) that is further explained in chapter: 4.3.3. The division of
interfaces does although create a bit of a gap in between computers and smartphones. A perfect example of devices that fall in between this gap is the tablet devices with relatively (in comparison to smartphones) big screens and touch input. As the filtering is configured now a tablet would automatically be brought into the Big layout, and the interface would fit it really well except for the (inevitably) missing “on hover” effects etc. However, a link has also been added in the footer that makes it possible to instantly switch between the Big and Small layouts.

All the physical differences of Small in relation to Big (see table 2) were implemented either by JavaScript’s that never instantiated the specific component at all in Small, or by means of three CSS files. The files are called: “stations.css”, “stations_small.css” and “stations_big.css” (see Appendix: 8.2.6). The first one (mentioned) contains all the styles affecting elements that Big and Small shares, and the two following are exclusively designed for the layout indicated by their name. This division minimizes duplication of CSS-class declarations and redundancy etc.

4.3.2 Touch events

After comparison of the UI’s responsiveness on different mobile platforms, there was an substantial different noticed between different versions of the Android OS [15] and its built in browser. The comparison was conducted by us developers by simultaneously accessing a Station and then manipulating the interface by pressing a link or button element. The responsiveness is defined by the time that elapses between the click and the UI presenting (i.e. giving feedback) the resulting dialog page (in this case). The interaction should fall under the interval of our defined ideal “UI response” shown in table 4 of our design metrics of responsiveness.

The comparison was made between an Android 2.1 device (HTC Hero [60]) and an iOS device (iPhone 3G [14]). Later we repeated the simple test (of clicking a link) with a HTC Desire HD [70] and the differences where quite obvious. The HTC Hero showed significant delay (> 1 second) in the UI (between manipulation and feedback) which were greater than our defined design metric interval for ideal response time within a Station (see “UI response” in table 4). The other two devices also showed a noticeable lag, at almost 1 second of delay (estimated), before the dialog page popped up. This lag issue made it feel like each page was loaded instead of simply fetched from the cache. Considering the basic type of manipulation that was done this responsiveness is considered intolerable according to our previously defined threshold: “UI response” (see table 4). A small difference that end user perhaps would not have noticed (depending on device), but inevitable not to consider when knowing how responsive it should or could feel.

After a fair amount of troubleshooting and comparison of different methods on different platforms, it was noted that the regular JavaScript click events was the root of the unresponsiveness-symptoms. The benchmarking was done by using a HTC Hero and an iPhone (same as above) to simultaneously access a Station and clicking around. Then different methods related to click input on touch devices were implemented and compared with both devices (by trial and error). The methods were subjectively compared to the standard case with a regular click-event activated. This method for detecting clicks with JS actually consists of two smaller events: “mousedown” followed by “mouseup”. [71] These
actions would of course be triggered on touch input from a smartphone device as well (see chapter: 4.2.2.3).

The solution to all our problems were found in a more suitable event (that were specifically designed for touch inputs) in the Document Object Model (DOM) events called: “Touch events”. [1] These events are fully supported in the Android and iOS’s and after implementing them on some navigational links in the UI, the improvement in responsiveness was remarkable. The difference from our standard case were blatantly obvious, and the feedback after manipulating were felt to be below our threshold of 0.1 seconds for “UI response” (see table 4). Together with some reduction of external JS’s in Small this implementation of so called: “touch events” contributed to a significant improvement when it comes to the total responsiveness of the interface (see chapter: 4.2.2.2).

4.3.3 On before unload

The critical concern with the back button described in chapter: 4.2.3.3, had two possible solutions, one of which being a JS event called: “onbeforeunload”. [72] The event triggers upon clicking the back button and activates a dialog box that query the user whether he or she really wants “to navigate away from this page” (see fig. 26).

![Figure 26](https://via.placeholder.com/150)

*Figure 26 - Event “onbeforeunload” dialog on trigger of back button.*

This solution, which was implemented into the Blicko Small UI, is common in internal navigation systems that changes views instead of loading whole pages. It is a way to make sure that the user really intended to leave the page, not only traverse back as the intuitive (and traditional) function of the button might be. If this event is not active a press of a browser’s back button inside a Station would throw the user out into the Station’s View (without warning). The implementation of “onbeforeunload” was decided to only affect the Small layout of the UI, because the problem was much more frequently apparent in Small. This, in turn, is surely related to the fact that Android smartphones have a dedicated standard physical button for backward traversal. [73] This function is persistent in every component of the platforms UI, which makes it a more widely implemented feature (*i.e.* not only in the browser). Its deep integration coheres with the user’s habits, which would interfere with usability if not prevented of use. The effect would be devastating since leaving the Station would mean that more data needs to be transfer in order for the user to re-connect, negatively affecting responsiveness etc. In the Big layout the back button problem still existed but too much lesser extent, and the cost of wrongfully traversing back becomes negligible. The navigation by the tabular menu (see chapter: 4.2.3.2) stays persistent and it always shows all available options of traversal. This type of navigation is common within websites and it therefore further diminishes the need for implementation of “onbeforeunload” within Big, because fewer users resort to pressing the back button in their browser.
5. Usability Testing

5.1 Introduction

Usability testing plays a key role when it comes to developing and making a product successful. Steve Krug simply puts the meaning of usability like this:

“After all, usability really just means making sure that something works well.” [52:133]

Usability testing would then imply that you test if something works well or not. To make Blicko as good and as user friendly as possible, usability testing will be a crucial part of the development.

Even though the designer and architect of the system believe that the service or product is extremely user friendly and cannot be misunderstood, the reality can be very different. The designer and architect cannot see it with fresh eyes after creating it and gets a very biased view of the product. Therefore usability testing is crucial if you want to find out how good the product really is. [52] The goal with usability testing Blicko is to find out what needs to be improved. Blicko should be able to be used with ease by experienced as well as inexperienced users. As Tullis and Albert says in their book, “Measuring the User Experience” [74], no one has ever complained that something is too easy to use. [74]

“When a product or service is truly usable, the user can do what he or she wants to do the way he or she expects to be able to do it, without hindrance, hesitation, or questions” [75:4]

To see how users actually interact with the product is also a part of usability testing. The intended use as the designers thought of might be different from how the users interact with the service. This is a very important aspect for Blicko since it is a new concept that has not been tried before.

Usability testing has limitations and it is by no means an exact measure of the usability or how well the product functions. Even the best usability tests in the world will not with 100% accuracy find all the usability problems. There are a many reasons for this and a few of them, which are listed below, is taken from the book “Handbook of usability testing”:

- “Testing is always an artificial situation” [75:26]

Since testing represents a depiction of the actual situation and not the situation itself the results will not be absolutely accurate. [75:26]

- “Test results do not prove that a product works” [75:26]

Even though the results are statistically significant it does not prove that the product functions. It is just an indication or measure of the probability that the results were not due to chance. [75:26]
Even if the participants are chosen with aspiration to represent the whole target audience, it is impossible since every person is unique. These shortcomings can be minimized as much as possible to get the best results. Even though usability testing has some limitations making a usability test will always be better than not testing at all.

5.2 Methods and concepts

This chapter on usability testing is demonstrating how the usability test for this thesis is: planned, executed and presented. This according to certain methods of thorough evaluation through testing that is relevant to web services such as Blicko. Many of the concepts described in this section were introduced by Tullis and Albert’s book: Measuring the User Experience.

The first step in making a usability test is to decide if you are going to take a formative or a summative approach. The difference between a formative and a summative approach is that a formative approach helps designers to develop the product while a summative approach measures if certain target goals were achieved. A formative approach would be used during the design and development process of the product. That way you can continually get feedback on what is good and bad in your design and keep making improvements. The goal of a formative approach is to help making improvements during the development. The summative approach evaluates the finished product and tells you if you reached the target goals and how well you made it in comparison to earlier versions or other similar products. So it is more about measuring the final product than the formative approach. In this thesis a formative approach will be conducted because Blicko is in the development process and one of the goals of this thesis is to figure out what can be done to improve the design.

When you plan a usability test it is important to know the users, what their goals are and also to know the product being tested. This is crucial because depending on who the users are, what they want to accomplish and what kind of product it is, the users will interact with it differently so to figure out what matters for the user, two main aspects of the user experience is measured: performance and satisfaction.

Tullis and Albert define performance as what the user actually does when interacting with the product. This would for example include if a certain task or tasks could be completed, how long time it takes to complete a task, effort in performing a task (for example number of mouse clicks) and learnability (how long time it takes to become proficient in a task).

Satisfaction is defined, by Tullis and Albert, as what the user feels about the interaction with the product. You could perhaps say that it is the overall feeling connected to the product. For example if the colors felt “right” for the product, how the design of the product was experienced, if the product was easy to use or difficult and so on. These things are hard to put real numbers on but it is important to measure the feeling the users get when using the product so that improvements of it can be done.
In this thesis there were a few major things we wanted to figure out, for example, is Blicko easy to use? Can anybody use it or is for example age or experience an important factor? Are there any certain usability issues that make it difficult for the user? Does the concept with voting on and submitting tracks work? What do people think about Blicko in general?

Therefore, in this thesis there was a total of two usability tests, one task oriented test (see chapter: 5.2.1) and one collaborative test (see chapter: 5.2.2). In the task oriented test performance and satisfaction was measured. It was done by giving the user a set of tasks to complete on the service and then gathering metrics while the user performed the tasks. To gather more data from the participants a set of questionnaires was also presented during the task oriented test session. The task oriented test was be done with two different groups where one group tests the Big version of Blicko and one group the Small version.

The collaborative test focused more on seeing how the concept of Blicko works with many users at the same time. So instead of looking at how the users perform certain tasks the focus lied in how the user uses Blicko. The expectation was to have at least ten persons using Blicko at the same time. Data about what the users did on Blicko was gathered to be analyzed. Metrics regarding satisfaction was also gathered through questionnaires that the users were asked to fill out.

5.2.1 Task oriented test

This section contains what in the task oriented test was measured, why it was measured, how it was conducted and how the participants were chosen. The goal with the task oriented test is to measure performance and satisfaction (see chapter: 5.2). These metrics are further broken down in this section to something that is easily collected during the test to give indication on the performance and satisfaction of Blicko. Navigation and information architecture is also being tested to see if relevant information is available for the users in Blicko. A task could for example be to see who the user is connected as, or if there are other users connected to Blicko at the same time.

5.2.1.1 Performance

This chapter takes up how and why performance is being measured in the task oriented test.

The tasks chosen for this (“task oriented”) test are sets of actions which a user is most likely to encounter. This makes these tasks very interesting for testing. The expectation in this test is to find out if the tasks are easy enough to finish them without assistance, how difficult they are and if they require a lot of effort. These questions all fall into the category of performance explained in chapter 5.2.

To measure performance in a task oriented test, different metrics has to be gathered. These metrics can be gathered in different ways depending on what you want to find out. The things measured for performance in our test are the following:

- **Success vs. failure (ratio)** - If the user was able to complete the task it counts as success and if the user gives up on a task because the user is unable to finish it it counts as a failure.

- **Time (seconds)** - Time to finish a task from after the task itself has been read and understood by the user.
• **Mouse clicks (number)** - The number of mouse clicks it takes for the user to finish a certain task. If the user gives up on a task the amount of clicks done so far is still recorded.

The most common thing to measure in a task oriented test is success versus failure rate of the tasks that the user tries to accomplish. It is a good way to measure the overall effectiveness of the task itself. [74] All the tasks chosen for this test have a goal the user has to answer or perform. In this thesis a success of a task will be defined as the completion of the same, which means the user reaches the end of the task (i.e. last click or outspoken answer). This definition is in total regardless of time or number of mouse clicks (which instead will be used as metrics). A failure of a task is defined as when the participant is unable to finish the task, i.e. the participant gives up trying to accomplish the task. The results from this can help to find out if there are any major flaws with the tasks themselves. If a user isn’t able to complete the task then it means something needs to be fixed. It is important here to have tasks which are not open ended so it is clear when the task is completed and when it is not.

The second data being collected during this task oriented session is task completion time, also called: “time-on-task” which simply is the time measured from the start of the task to the end of the task. The start of the task is defined as after the facilitator has read the task to the participant and the participant understands the task. The end of the task is defined as when the user either completes a task or gives up on a task. Task completion time can give a good measurement of the effort involved in completing a set of tasks. It is most important for services where users tend to do the same operation repeatedly. [74] Which corresponds very well with Blicko since there are not that many functions available to utilize (i.e. repetition of tasks is more common than use of new functions). Task completion time will be measured on all the tasks even where failure to complete the task occurs. Then when (or: if) the tester decides to give up on a task, that he or she cannot complete, the timer is stopped. [74]

To measure the effort involved in completing a task the number of mouse clicks can be recorded. The more mouse clicks the more effort is required by the user. [74] It can further be used to match the number of mouse clicks against the optimal number of mouse clicks to complete a task or a set of tasks. This would give some indication to if the user follows the intended path as thought out by the designers.

One way to test the overall navigation and information architecture is to do what Tullis and Albert call in their book, Measuring the User Experience, is a “scavenger hunt”. [74:51] This is a task where the user is told to find information which is supposed to be readily available for the user. One way to measure this is to look at how many steps it took the user to complete the task. Measuring this would give a good view on how the information is displayed to the user and what needs to be clearer for the user. It is very important to figure out which information should be readily available to the user. One example of a task which tests the navigation and information architecture is task 7 (see table 6).

The preferred click stream for accomplishing the tasks will be figured out so that there is a template on how to conduct a task in an optimal way (see fig. 30 and fig. 31). This is the way the designers have thought the task should be solved but it is not the only way the task can
be done. By doing this, metrics can be gathered which shows both how well the navigation works but also the effort taken to complete the task.

The tasks chosen for the task oriented test is including actions that uses the most common functions or sections that the user will encounter inside a Blicko Station. There are not many functions to use in Blicko and the tasks chosen makes up for most of the things you can do in Blicko. Since these tasks are the ones that most users are expected to do they also become interesting for testing. All the different tasks chosen for this test can be found in table 6. The only task which stands out a little from the rest of the tasks is task 4. The goal with task 4 is to see if the participants are able to find any song by the artist “Yes” this means that the participant has to scan through the labels in the list to find the correct song. Searching for “Yes” both gives artists, songs and albums with the name “Yes” and “Yes” was therefore chosen for this task. This makes it a little trickier for the participant to find the correct song to submit.

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Connect to the host: “Jeppe” and see which song is currently being played.</td>
</tr>
<tr>
<td>2</td>
<td>Upvote or downvote three of your preference in the playlist.</td>
</tr>
<tr>
<td>3</td>
<td>Submit three songs from a band of your preference and see what positions it gets.</td>
</tr>
<tr>
<td>4</td>
<td>Submit any song by the band &quot;Yes&quot; and see what position it gets.</td>
</tr>
<tr>
<td>5</td>
<td>Try to find out all the songs that you have submitted to the upcoming playlist.</td>
</tr>
<tr>
<td>6</td>
<td>Which songs have you voted up and which songs have you voted down?</td>
</tr>
<tr>
<td>7</td>
<td>Try to figure out who you are connected as.</td>
</tr>
<tr>
<td>8</td>
<td>Are there any other users connected to the same station as you are?</td>
</tr>
<tr>
<td>9</td>
<td>Try to find out which action or event the user Jesper Ahlberg did last.</td>
</tr>
<tr>
<td>10</td>
<td>Try to logout or leave the station you are currently connected to.</td>
</tr>
</tbody>
</table>

Table 6 - The table shows chosen tasks for the task oriented test session. All tasks are basic actions a user might want to do while using Blicko.

In the test a click or keyboard click on “search”, to send the search query, is not counted as a click. Writing the search query does not count as clicks either. This is done to prevent search queries to have a different amount of clicks depending on what the user is searching on. Which artist the participant searches for should not affect the amount of clicks being measured since it is not related to usability.

During the test session of Blicko the user was asked to not “think aloud”. Instead the user was asked to hold any thoughts and comments until the task was completed. This was done so that the task success time will not be affected by the user explaining and thinking aloud, which most likely will affect the task completion time. [74]

5.2.1.2 Satisfaction

This chapter takes up how and why satisfaction is measured in the task oriented test.
For the task oriented test several questionnaires are given to the participants to fill out. The questionnaires are a way to gather data about satisfaction (see chapter: 5.2) and background information. This data will then be analyzed to understand how Blicko can be improved.

The first questionnaire the participants of the task oriented test will encounter is the Demographic Background Questionnaire (see Appendix: 8.3.2.1). The background questionnaire consists of several questions regarding things like: occupation, age, owning a smartphone [4], Internet experience and so on. These questions are chosen because they might be related to how well a user interacts with Blicko. For example if a participant has a smartphone it might turn out that it is easier for that participant to use Blicko. This is done to help understand the behavior and performance of the participants during the test. [75]

For example it could turn out that users need at least five years of experience on the Internet in order to understand the interface fully. This information could then be used to improve the interface so that less experience is required in order to understand it.

For users participating in the task oriented test the second questionnaire they will encounter is the Task Test Protocol (see Appendix: 8.3.2.2) which is a task-based questionnaire that is filled out after each task. The facilitator will be the one filling out the questionnaire. The first page in the questionnaire contains several questions like device used, browser used and tester number. After the first page each task itself has one page of each with things like task-on-time, success vs. failure of task, facilitator comments, comments from the participant and the difficulty of the task itself from the participant’s point of view, also called post-task rating. All ten tasks have the same questions about them so that the same metrics will be gathered for all the tasks.

After finishing or failing a task the user is asked about how difficult they felt the task was. This post-task rating is done to give a feedback from the participant about the difficulty of each task. The method used in this thesis is derived from a comparison of different methods for post-task ratings made in 2006 by Tedesco and Tullis. [74] The comparison was made between five different methods for eliciting self-reported ratings after each task. [74] The goal of the comparison was to find out if these rating techniques are sensitive to differences in perceived difficulty of the task. The first comparison was made with a very large sample and the conclusion from that comparison was that they were all effective in distinguishing the between the tasks. Since this was from a large sample the next question was if the same was true for a small sample. It turned out that it was not, and one of the five methods was better at small sample sizes. [74] Since the sample size of the task oriented session in this thesis will be small (see chapter: 5.2.1.4) this method suits this test very well. Therefore the method which stood out as the better for small sample sizes is used in the task test protocol as a post-task rating because of its ability with small sample sizes. The method in question is the following:

“Overall, this task was: Very difficult - - - - Very easy” (p. 133) [74:133]

The method is used by simply asking or letting the user read the question which the user then answer by choosing a number between 1 and 5 where 1 represents very difficult and 5 very easy.

The last questionnaire for the task oriented test session is the Evaluation Questionnaire (see Appendix: 8.3.2.3). This is done in the end of each test session to gather information on
what the user thought about the product. The Evaluation Questionnaire consists of two parts. The first part is the System Usability Scale (SUS) [76] section and the second part is comments about Blicko from the participant.

“SUS” is a ten-item, 5-point “Likert scale” [3] for evaluating ease of use. It was developed by John Brooke at Digital Equipment Corporation in 1986. It works by letting the participants choose their level of agreement to ten statements. The level of agreement is chosen from a 5-point Likert scale ranging from “strongly disagree” to “strongly agree”, and is numbered from 1 to 5. Half of the statements are negatively worded and the other half is positively worded. After filling out the SUS, the rating from the ten statements is combined to give the final result. It is calculated by summing up the ten statements scores. [74]

For the post-session questionnaire the SUS is used. SUS is used because it appears to yield more consistent ratings at relatively small sample sizes. This has been concluded from a study made by Tullis and Stetson (2004). The study compared a variety of post-session questionnaires for measuring user reactions to websites. The study showed that SUS was more consistent than the other post-session questionnaires, which were in the study, when it comes to relatively small sample size like those used in lab-based usability tests. [74]

After the SUS section a number of open-ended questions appears. These questions so that the participant can write what he or she thinks about Blicko. There are a total of four open-ended questions: “What did you like best about Blicko?”, “What did you like least about Blicko?”, “Can you think of features or functions missing in Blicko?” and “Would you like to make any other comments about Blicko?”. The answers to these questions from the participants can then later be translated into metrics by counting the number of instances. [74]

5.2.1.3 Approach

This chapter presents how the task oriented test was conducted and which methods that were used. Since Blicko essentially consists of two different sites called Big and Small, there was two task oriented tests, one for Big and one for Small. During the task testing of Small the built-in-browser in the mobile devices was be used and for Big an ordinary laptop with Google Chrome [16] as browser.

For the test a basic setup for Blicko was be organized so it could mimic a real usage situation. The basic setup will consist of a Blicko Station called: “Jeppe”, which has already been prepared with 5-10 songs submitted by the user “Jesper Ahlberg”.

The task oriented test was conducted in one-on-one sessions where the user got tasks to solve by a facilitator. For each session the facilitator brought a computer for the facilitator and also a test device for the participant to use. For the Small test the device chosen was a HTC Hero [60] with Android 2.1 [15] with the default browser and for Big a laptop with Windows 7 [77] with Google Chrome [16] as browser. Before the start of each test session an orientation script was read to the participant. This script contains a small briefing on what will be done during the test, what Blicko is and why the test is being done. It is also important to inform the participants that it is not them who are being tested but the product. [75] If the participant is screwing things up then it is because of the product lack of usability and not because of the participant. This is why you test, to see what might be wrong with the product.
After the introduction the user filled out a background questionnaire (see Appendix: 8.3.2.1). The background questionnaire was done through the computer brought by the facilitator. Once the questionnaire was filled out and submitted the user was guided to the test page and the computer had a screen recording activated to record the users actions. The test page was the client page of Blicko and was accessed by the device brought by the facilitator. Once arrived to the test page the facilitator filled out information regarding things like browser used and device used in the Task Test Protocol (see Appendix: 8.3.2.2). Then the first out of ten tasks was orally read to the participant. Once the task has been read to the user a timer on the facilitator’s phone was started by the facilitator. When the participant had finished or decided to give up on the task the timer was stopped and the time noted. The facilitator also noted if the participant completed the task or gave up on the task. When the participant performed the task the facilitator also kept track of how many keystrokes or clicks the user did while doing the task. After finishing a task the participant is directed back to the “Upcoming” view (if not already there). This is done so that the starting point is consistent for each task. “Upcoming” is also the view which the user is expected to spend most time in.

When the user have finished a task he or she will be asked about the perceived difficulty of the task and if the participant has any comments. This and the other metrics such as number of clicks are filled out in the Task Test Protocol (see Appendix: 8.3.2.2) by the facilitator. After answering the questions the user will be told to start the next task which will be read aloud from the facilitator. Each task is done only once. When the test session was completed the screen recording was stopped and the participant was be asked to fill out the Evaluation Questionnaire (see Appendix: 8.3.2.3) about the test session. The questionnaire contained questions about the overall feeling of the tasks and other comments regarding Blicko.

Since the test was not demanding when it comes to location or hardware we (as facilitators) were able to go to the participants instead of making the participants come to us. This made it more convenient for the participants since many had a busy schedule.

5.2.1.4 Participants
The participants for the usability test will be chosen to fit Blicko’s target audience as good as possible. This becomes fairly easy since the target audience of Blicko would be anyone who likes music and enjoys going to places where they play music. However, the number of participants that takes part in a test is a difficult problem. Logically the more participants in the test, the more usability problems found. But many participants is difficult to manage and according to some usability experts, like Jakob Nielsen and Steve Krug, the amount of usability problems increases marginally after just a couple of users. [52], [78]. As an illustration of this phenomenon Jakob Nielsen and Tom Landauer conducted a research where they modeled an equation that shows the number of usability problems: “X” (from a defined set), found in certain usability tests with: “n” users.

$$N(1-(1-L)^n) = X$$

**Figure 27** - In this formula: “n” is the number of participants, “N” is the total number of usability problems in the design and: “L” is the proportion of usability problems discovered while testing a single user. [78]
In relation to this thesis, the graph below (see fig. 28) and formula (see fig. 27) are to be seen (merely) as a slight justification to why a small number of testers can sometimes be “sufficient”. They are best described as a relevant example for comparison purposes. During Nielsen’s and Landauer’s research they discovered that “L” (proportion of usability problems discovered while testing a single user) had an average value of: 31%. [78] The number 31% is derived from 11 certain usability tests conducted by Nielsen and Landauer and should only be considered as a rough estimate (i.e. guideline). [79] Applying this number to the formula results in the following curve:

The curve in fig. 28 clearly shows that the first three users are very likely to find most (> 50%) of the usability problems. As expected the incline in percentage of new problems found decreases when the numbers of users increase (i.e. the usability issues had already been recognized by previous testers). When Nielsen took into account the factor of cost (money and time) per number of testers, he found that after five participants the result became marginally better in relation to the resources spent. Therefore; instead of spending the entire budget on 15 users for one test, it is better to spend the budget on five users and three tests. [78] This way you can improve the design from the first test and then try the new design and see if it has improved.

“You want to run multiple tests because the real goal of usability engineering is to improve the design and not just to document its weaknesses.” [78]

This is further supported by Steven Krug’s book [52], where he shows that it is better to use few people in several tests rather than many people in one test. As you fix the problems from the first round of testing, the users will be able to discover new problems since they will not get hung up on the already discovered issues anymore. [52]

Managing a small number of users is also much more convenient than a large number of users. In one day you can both conduct the test and do a debriefing. Then you can quickly
start focusing on making improvements and figure out how to fix any usability problems that got discovered during the test. This method fits well with agile development and iterative development processes, since each step indicates a refinement based on test results.

Albert and Tullis argues that the magic number five [78] might not be so magic. They have many arguments to why five users are not enough but in the end they conclude that five users seem to work but only under certain conditions. The first condition is that the scope of the evaluation is fairly limited. By fairly limited they mean that the evaluation is limited to about 5-10 tasks and about 20-30 web pages. The second condition is that the user audience is well defined and represented. [74]

The number of participants for the task oriented test was decided to be four persons per test based on the facts above and also because there are two tests which make the total number of participants 8. Our goal was to get participants which were as diverse as possible since Blicko targets people who enjoy music. Given that we wanted a wide diversity of participants we picked 4 males and 4 females where two of them were above 50, two under 20 and four in between in the range 20-30. They also had different backgrounds where some were students and some were working. All of these participants are acquaintance of us. None of the participants will have seen or tried Blicko before the test session. This was important to be able to see how fast completely new users will understand the concept and if the interface is made easily enough for them to understand it.

5.2.2 Collaborative user test

The collaborative user test is constructed to give an idea of how well everything works when Blicko is used with many people as Blicko is meant to be used. The focus here is not necessary the usability but more the voting algorithm and the overall behavior of the users when there are more than 2-3 persons using it. The things that is analyzed for this test is the following:

- **Stress test** - Does more users affect the performance of Blicko?
- **Voting algorithm** - How did the voting work?
- **Socially linked behaviors** - Was there any patterns to the behavior of the participants?
- **Satisfaction degree** - What did the participants think of Blicko?

The behavior which is expected might very well be different from how people use Blicko while they are in a bigger group. Participants will not be asked to perform certain tasks but instead told to just use Blicko as they see fits.

For this test, data about how the users interact with Blicko was collected. On the server which hosts Blicko each action done by a connected user can be extracted. This can then be used to record what the users actually do while connected to Blicko. The data is extracted in form of lines where each line represents one action in Blicko. Each line contains a time stamp, event type, user name (Facebook name), track artist (if the action doesn't involve music it is set to NULL), track title (NULL if no data) and vote. The different event types users were able to do during the test were: UserConnect, UserDisconnect, Vote, Submit and Withdrawal. The data can then be analyzed to find patterns regarding the behavior of users.
in Blicko. Interesting parts will be on how people vote and when they submit songs. The data extracted during this session can be found in Appendix: 8.3.3.

5.2.2.1 Satisfaction
Satisfaction will be measured during the collaborative test in the form of questionnaires that the participants will be asked to fill out. The collaborative test session is using a form called Collaborative Questionnaire (see Appendix: 8.3.2.4) which is a combination of the Demographic Background Questionnaire (see Appendix: 8.3.2.1) and the Evaluation Questionnaire (see Appendix: 8.3.2.3). This is done so that the participants of the collaborative test only have to fill out one questionnaire. The questionnaires here have the same goal as in the task oriented test, namely to measure satisfaction (see chapter: 5.2).

5.2.2.2 Approach
The location for this test was at the IT-section pub called Kistan. Here people gather every Tuesday and Thursday to enjoy drinks and socialize. This was a perfect venue to test Blicko. To set this test up the IT-section was contacted to get allowance to test Blicko at the pub. The IT-section was more than helpful and there was no problem to do the test at Kistan during a pub night. At Kistan equipment such as speakers and a projector were available. A Facebook event and an email were sent out to inform other students about the test session with things such as time and date. This was also an opportunity to attract more participants.

Before the collaborative test a special test site was set up (see fig. 29). The site was made to introduce the participants to Blicko. The test site linked to an about page about Blicko, to the station Kistan and to an evaluation questionnaire (see chapter: 5.2.2.1). The URL to the test page [http://www.blicko.com/Test](http://www.blicko.com/Test) was written on flyers posted in Kistan and also written in the email and Facebook event.

![Figure 29](image_url) - The test site which was made as an introduction for the participants to access in the collaborative user test.
For the test two laptops from the facilitators were used. One laptop was set up to the speakers at the venue to act as a station and play the music. The other computer was set up to a projector so that when people arrived to the venue they could see a Blicko Station on the wall and see what was going on.

The setup for the test was decided to start at around 18:00 on a Thursday at the student pub Kistan. In the information sent out it said that the test started at 19:00 and would end at 22:00. On the test site (see fig. 29), in the email and in the Facebook event there was also a request for the participants to answer a questionnaire.

A feedback button was also added to the interface of Blicko for this test-session. The button would open a form that promoted any type of input on the user’s experience. Since the participants do not have access to a facilitator at all time for comments the feedback button was introduced.

5.2.2.3 Participants
The collaborative testing was going to require more users using Blicko at the same time to simulate a real usage situation. For this test the participants were not chosen, instead the place for the test was chosen and people who arrived to this venue was informed about Blicko and was able to participate if they wanted. The place picked was the student pub Kistan which is located in the Kista KTH campus. Naturally this brought a lot of students and mostly students in a computer related major. This was a very narrow participant audience but also created a very unique debugging opportunity because of the students majors. The expectations was to get at least ten participants to try Blicko at the same time.

5.3 Results
This section gathers all the data from the task oriented test and the collaborative test. The data is displayed in charts and tables in the form of figures to simplify viewing. All the raw data can be found in the Appendix: 8.3.3. The standard deviations shown in this section was calculated using the “STDEV” function in Excel, and are shown as: “error bars”. [80] They represent the variation (or “dispersion”) from the average. The spreadsheets containing the calculations and graphs can be found in Appendix: 8.2.3.

Due to hardware failure with the HTC Hero [60] intended for use in the Small tests, other devices that were available at the time had to be used instead. Participant 2 and 3 used the default browser in a HTC Desire HD [70] with Android 2.3 [15] and participant 6 and 8 used Safari on an iPhone 4 [14]. Participant 7 had to use a MacBook Pro [81] instead of the laptop intended because of unavailability.

5.3.1 Task oriented test
This section shows the data gathered from the task oriented test. The task oriented test went very well and a lot of feedback from the participants was gathered. Overall the feedback from the participants seemed very positive. All of the participants succeeded in completing the tasks set out for them, which would be considered a very good result. Screen recording was only possible on one of the eight tests made partly because it was only possible on a computer which were not always accessible during the test and also because four of the tests
were on smartphones. The actual recording (i.e. a way to retrieve it) can be found in Appendix: 8.2.2.

5.3.1.1 Demographic background

The demographic background data is shown in table 7. Half of the participants tested on Big and the other half on Small. Which participant who tested on which site can be found in table 7 under “Test layout”. In this table two columns were removed. One of the columns removed was a question regarding what you usually do on the Internet. This question is moved to its own table which can be found in table 8. The other column removed regarded how often a user used Internet. For this question all the participants answered “Daily” and that column were therefore removed.

<table>
<thead>
<tr>
<th>#</th>
<th>Gender</th>
<th>Age</th>
<th>Occupation</th>
<th>Years on the Internet</th>
<th>Smartphone[4]</th>
<th>How often do you use Internet with a phone</th>
<th>Completed education</th>
<th>Browser used for test</th>
<th>Test layout</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Female</td>
<td>23</td>
<td>Full-time employee</td>
<td>12</td>
<td>Yes, Android</td>
<td>Daily</td>
<td>High-School</td>
<td>Chrome</td>
<td>Big</td>
</tr>
<tr>
<td>2</td>
<td>Male</td>
<td>20</td>
<td>Student</td>
<td>10</td>
<td>Yes, Android</td>
<td>Daily</td>
<td>High-School</td>
<td>Other</td>
<td>Small</td>
</tr>
<tr>
<td>3</td>
<td>Female</td>
<td>17</td>
<td>Student</td>
<td>9</td>
<td>No</td>
<td>Never</td>
<td>High-School</td>
<td>Other</td>
<td>Small</td>
</tr>
<tr>
<td>4</td>
<td>Male</td>
<td>25</td>
<td>Full-time employee</td>
<td>14</td>
<td>Yes, iPhone</td>
<td>Daily</td>
<td>High-School</td>
<td>Other</td>
<td>Big</td>
</tr>
<tr>
<td>5</td>
<td>Female</td>
<td>27</td>
<td>Full-time employee</td>
<td>10</td>
<td>Yes, Android</td>
<td>Daily</td>
<td>University</td>
<td>Chrome</td>
<td>Big</td>
</tr>
<tr>
<td>6</td>
<td>Female</td>
<td>51</td>
<td>Full-time employee</td>
<td>15</td>
<td>Yes, iPhone</td>
<td>Daily</td>
<td>High-School</td>
<td>Other</td>
<td>Small</td>
</tr>
<tr>
<td>7</td>
<td>Male</td>
<td>56</td>
<td>Full-time employee</td>
<td>13</td>
<td>No</td>
<td>Never</td>
<td>High-School</td>
<td>Safari Mobile</td>
<td>Big</td>
</tr>
<tr>
<td>8</td>
<td>Male</td>
<td>19</td>
<td>Part-time employee</td>
<td>13</td>
<td>No</td>
<td>Never</td>
<td>High-School</td>
<td>Other</td>
<td>Small</td>
</tr>
</tbody>
</table>

*Table 7* - The table shows the demographic background data collected during the task oriented test session.
<table>
<thead>
<tr>
<th>What do you usually do on the Internet?</th>
<th>Big: User #</th>
<th>Small: User #</th>
<th>Total:</th>
</tr>
</thead>
<tbody>
<tr>
<td>Use social networking websites (e.g. Facebook, Twitter, LinkedIn)</td>
<td>1, 4, 5, 7</td>
<td>2, 3, 6, 8</td>
<td>8</td>
</tr>
<tr>
<td>Use social news websites (e.g. Reddit, Digg, Delicious)</td>
<td>4</td>
<td>8</td>
<td>2</td>
</tr>
<tr>
<td>Read blogs (e.g. Engadget, Slashdot, Feber)</td>
<td>1, 4, 7</td>
<td>2, 3, 6, 8</td>
<td>7</td>
</tr>
<tr>
<td>Visit news websites (e.g. DN, SvD, Aftonbladet)</td>
<td>4, 5, 7</td>
<td>2, 3, 6, 8</td>
<td>7</td>
</tr>
<tr>
<td>Watch videos (e.g. YouTube, Vimeo, TED)</td>
<td>1, 4, 5, 7</td>
<td>2, 3, 6, 8</td>
<td>8</td>
</tr>
<tr>
<td>Listen to music (e.g. Spotify, Last.fm, Grooveshark)</td>
<td>1, 4, 5, 7</td>
<td>2, 3, 8</td>
<td>7</td>
</tr>
<tr>
<td>Play games (eg. World of Warcraft, Farmville, Counter-Strike)</td>
<td>4</td>
<td>2, 8</td>
<td>3</td>
</tr>
<tr>
<td>Communicate (e.g. Email, Instant Messaging, IRC)</td>
<td>1, 4, 5, 7</td>
<td>2, 3, 6, 8</td>
<td>8</td>
</tr>
<tr>
<td>Retrieve information (e.g. Wikipedia, Google, Eniro)</td>
<td>1, 4, 5, 7</td>
<td>2, 3, 6, 8</td>
<td>8</td>
</tr>
</tbody>
</table>

Table 8: The table shows data about what people use the Internet for. The data was collected during the task oriented test session in the background questionnaire.

5.3.1.2 Task success rate

The completion rate from the task oriented testing was indicating good results. Everyone who participated in the test succeeded in completing the tasks. In a test where there is either 0% or 100% success rate it becomes hard to identify different groups since everyone or no one succeeded. In this test there was a 100% success rate. All of the participants in both the test for Big and Small were able to finish all of the tasks. A task is defined as finished if the user is able to follow the instructions of the task regardless of time and number of clicks. A failure is defined as: the user giving up on trying completing the task. These results are very good and it shows that it seems to be relatively easy for the users to perform the tasks. What this means in the real world is not that 100% of all the users would be able to complete the tasks in this test. To calculate how likely it is for the rest of the population to succeed the LaPlace method is used. [82] LaPlace uses the formula shown in: fig. 30.

\[
\frac{(x+1)}{(n+2)}
\]

Figure 30: Laplace law of Succession a formula to calculate the likelihood of something happening again after a number of successes where x is the number of observed successes and n is the number of observed trials. [82]

Using the rule of succession on the task success data from Small and Big gives the result 83.33% on both. This result is calculated using a calculator [82] provided by Albert and Tullis in the book Measuring the User Experience. [74] In the calculator confidence level and “Likely Population Completion Rate” is left unchanged (should be 95% and Unknown) and when calculated the results show up in the “Best Estimation” box. Instructions for how to use the calculator can be found on the same page, and the calculation made in this section the instructions followed are under the section “When All Users Pass or Fail”. [82]

5.3.1.3 Task completion time

Since all the tasks were successfully completed, the metric of task completion time grows in importance. Task completion time was measured by starting a timer at the beginning of a
task, and then stopped when the task was finished (see chapter: 5.2.1.3). This was done for all the participants on all the tasks. The results from time on task are shown in the two diagrams below (see fig. 31 and fig. 32). The first diagram shows average task completion time for Big (see fig. 31) and the second one for Small (see fig. 32). Since it is the average time is calculated by taking the task completion time of all the users in that particular task the standard deviation is shown to illustrate the variability of the data from which the bars are based on. It also shows the reliability of the average task completion time. The error bars represents the standard deviation (from average value) and was calculated by using the “STDEV” function in Excel. [80] The data and the Excel spreadsheets can be found in Appendix: 8.2.3.

**Figure 31** - The chart shows the average task completion time of all the tasks in the task oriented test for Big. Error bars represent the standard deviation.

**Figure 32** - The chart shows the average task completion time of all the tasks in the task oriented test for Small. Error bars represent the standard deviation.
5.3.1.4 Post-task rating

After each task the participant were asked about the perceived difficulty of the task itself. The rating is based on a number scale 1-5 where 1 represents “very difficult” and 5 “very easy”. The average post-task score for Big and Small are shown below in two diagrams (see fig. 33 and fig. 34). Since the bars only represents the average post-task rating a 95% confidence is calculated and shown in the form of error bars to show the variability of the data. The error bars are located on the bars themselves and was calculated by using the confidence function [80] in Excel. The data is assumed to follow a normal distribution. The data and the Excel spreadsheets can be found in Appendix: 8.2.3.

![Big: Post-task rating](image)

**Figure 33** - The chart shows the average post-task rating that the participant were asked to fill out after each task while testing Big and represents the participants perceived difficulty. The rating goes from 1 to 5 where 1 represents “very difficult” and 5 “very easy”. Error bars represent the standard deviation.
Figure 34 - The chart shows the average post-task rating that the participant was asked to fill out after each task while testing Small and represents the participants perceived difficulty. The rating goes from 1 to 5 where 1 represents “very difficult” and 5 “very easy”. Error bars represent the standard deviation.

5.3.1.5 Task efficiency

Efficiency can be measured by several factors. The first data taken account for in this thesis is the number of clicks the user had to perform in order to finish the task. [74] This data is shown in two diagrams where the average number of clicks to finish each task is presented with the optimal number of clicks next to it, one for Big (see fig. 35) and one for Small (see fig. 36). The optimal number of clicks means the minimum amount of clicks the participant had to do in order to complete the task. This data can then be used to compare the average click per task to see which tasks were more difficult and which ones were easy. All click streams have been measured from the starting-point of the “Upcoming” view, and each task ends with returning to this view as well. The standard deviation is also calculated to show the variability of the data from which the average click bars are calculated. The data and the Excel spreadsheets can be found in Appendix: 8.2.3.
Figure 35 - The chart shows the average number of clicks and the optimal number of clicks on each task in the testing of Big. Error bars represent the standard deviation.

Figure 36 - The chart shows the average number of clicks and the optimal number of clicks on each task in the testing of Small. Error bars represent the standard deviation.

The second way efficiency is measured for this test is by calculating number of tasks completed per minute for each participant. [74] This would show the efficiency of each participant on each session and further analysis of this data can reveal interesting facts. The data for the participants efficiencies in Big and in Small are shown in the two diagrams below (see fig. 37 and fig. 38). The standard deviation that show the variance of the data is not calculated for these graphs. This is due to the fact that the tasks have different
complexity which would make the variance insignificant. The variance would show the difference in the task completion time which is not interesting in this section. But since every user did the same tasks the “task per minute” average for each user becomes interesting.

![Big: Participant efficiency](image)

*Figure 37* - The chart shows the number of tasks per minute for each participant in the task oriented test for Big.

![Small: Participant efficiency](image)

*Figure 38* - The chart shows the number of tasks per minute for each participant in the task oriented test for Big.

5.3.1.6 System Usability Scale

To calculate the total SUS-score from the evaluation questionnaire a special method is used. For the positively worded statements the score contribution is the scale position minus 1 and for the negatively worded statements it is 5 minus the scale position. The sum is then multiplied with 2,5 to obtain the SUS score. SUS scores ranges from 0 to 100 where 100 is a perfect score. [74] Each participant’s individual SUS-score is shown for Big (*fig. 39*) and Small (*fig. 40*). The total score for each user is then summed up with the other users score and then divided by the number of users to get the average score (see *fig. 41*). The standard
deviation is calculated and shown in Fig. 41 to show the variability of the SUS-score from the participants. In Fig. 41 the error bars represent their corresponding standard deviation values. The data and the Excel spreadsheets can be found in Appendix: 8.2.3.

**Figure 39** - The chart shows the SUS-score for the users testing Big in the task-oriented test.

**Figure 40** - The chart shows the SUS-score for the users testing Small in the task-oriented test.
Figure 41 - Chart showing the average SUS-score on Small as well as Big. Small got a score of 86.25 and Big got a score of 76.25. Error bars represent the standard deviation.

5.3.1.7 Observations
During the test observations made by the facilitator were noted and can be found in the Appendix: 8.3.2.2. The participant was also asked to make comments about what they thought of the task itself; positive as well as negative things (see Appendix: 8.3.2.2.).

In the end of the evaluation questionnaire the participants were asked to answer a few questions regarding what they thought of Blicko (see Appendix: 8.3.2.3). Their comments varied a lot in granularity depending on the person. Some participants described very detailed things they liked or did not like while some had comments which were broader.

5.3.2 Collaborative user test
This section shows the data gathered from the collaborative user test. Overall the test went very well and there was a lot of positive feedback. Some flaws within functions of Blicko’s interface were discovered, and is presented below but further analyzed in chapter: 6. In total there was 22 participants who connected to Blicko through Facebook and there was about 30 persons at the pub. More than 10 persons were simultaneously connected to Blicko through Facebook at its peak (see Appendix: 8.3.3).

5.3.2.1 Demographic background
The total number of participants that answered the questionnaire for the collaborative user testing was: eight. The background data from the collaborative questionnaire is shown in table 9. Three of the participants who answered the questionnaire tested on Big and the five on Small. Which participant who tested on which site can be found in table 9 under the column: “Test layout”.

In the demographic background section there was also a question regarding what you usually do on the Internet. The answers for this question are shown in table 10. In this table two columns were extracted because all the participants submitted the same answer.
The two extracted questions were:

- How often do you use the Internet? - All answers: “Daily”.
- How often do you use Internet with your phone? - All answers: “Daily”.

![Table 9](image)

*Table 9* - The table shows the demographic background data collected during the collaborative test session.

![Table 10](image)

*Table 10* - The table shows data about what people use the Internet for. The data was collected during the collaborative test session in the background questionnaire.

### 5.3.2.2 System Usability Scale

In chapter: 5.3.1.6 it is described how to calculate the SUS-score. The same method is used to calculate the SUS-score for the participants in the collaborative test session. A small
difference in this chapter is that the number of participants in Small is now five instead of four and for Big it is three instead of four. These scores are shown in the two diagrams below, one for Big (see fig. 42) and one for Small (see fig. 43). Combining the SUS-score from the different participants gives the average score for Big and Small (see fig. 44). The standard deviation is calculated to show the variability of the SUS-score from the participants. In fig. 44 the error bars represents the standard deviation. The data and the Excel spreadsheets can be found in Appendix: 8.2.3.

**Figure 42** - The chart shows the SUS-score from each participant who used a laptop or a tablet during the collaborative testing.

**Figure 43** - The chart shows the SUS-score from each participant who used a smartphone during the collaborative testing.
5.3.2.3 Test session data

During the collaborative test session a lot of data (see Appendix: 8.3.3) was recorded on the server which hosted Blicko. The data recorded were of events happening on the server. Events can for example be: “submit a track”, “vote on a track”, “user connecting to the station” and so on. For each event the time and name of the event, as well as the user that triggered it, is recorded. But for events like submit a song some more data, like the name of the artist and the song, is recorded. Due to some problems with anonymous users, all of the events which regard anonymous users were removed. The reason for this is that one user found a bug in Blicko which made it possible for the user to vote on songs an infinite amount of times. Once this bug was found the ability to join anonymously was removed, and the Station was reset. “Upcoming” was cleared and the issue had been taken care of by the official starting time of: 19:00. The pie chart below (see fig. 45) is generated from the data in Appendix: 8.3.3.
Figure 45 - A pie chart showing the events which occurred during the collaborative test session. The pink area represents total number of vote events. The numbers represents how many times that event occurred.

In total there was about 500 events done by the 22 users who were connected to Blicko through Facebook (see fig. 45). The time span for the events was about five hours.

A few users showed an unusual pattern where they would submit a song and then proceeded to down vote a lot of other (competing) songs. An example of this behavior has been extracted from Appendix: 8.3.3 and is shown in Table 11.

<table>
<thead>
<tr>
<th>Time</th>
<th>Event</th>
<th>User #</th>
<th>Track Artist</th>
<th>Track Title</th>
<th>Vote</th>
</tr>
</thead>
<tbody>
<tr>
<td>2011-03-29</td>
<td>Submit</td>
<td>19</td>
<td>El Ten Eleven</td>
<td>My Only Swerving</td>
<td>-</td>
</tr>
<tr>
<td>2011-03-29</td>
<td>Vote</td>
<td>19</td>
<td>Butterfly Boucher, David Bowie</td>
<td>Changes</td>
<td>1</td>
</tr>
<tr>
<td>2011-03-29</td>
<td>Vote</td>
<td>19</td>
<td>Studio Allstars</td>
<td>Californication - (Tribute to Red Hot Chili Peppers)</td>
<td>-1</td>
</tr>
<tr>
<td>2011-03-29</td>
<td>Vote</td>
<td>19</td>
<td>Butterfly Boucher, David Bowie</td>
<td>Changes</td>
<td>-1</td>
</tr>
<tr>
<td>2011-03-29</td>
<td>Vote</td>
<td>19</td>
<td>Moby</td>
<td>James Bond Theme (Moby's Re-Version)</td>
<td>-1</td>
</tr>
<tr>
<td>2011-03-29</td>
<td>Vote</td>
<td>19</td>
<td>Daniel Adams-Ray</td>
<td>Lilla lady</td>
<td>-1</td>
</tr>
<tr>
<td>2011-03-29</td>
<td>Vote</td>
<td>19</td>
<td>Kings Of Leon</td>
<td>Sex On Fire</td>
<td>-1</td>
</tr>
</tbody>
</table>

Table 11 - The table shows an example of how some users behaved during the collaborative test session. The data is extracted from Appendix: 8.3.3
5.3.2.4 Observations

During the test a bug was found which made it possible to abuse the voting system. Some users exploited this and made it so a few of the songs got enough votes to be played even though no one had voted on them. The bug was that a user could enter a Blicko Station as Anonymous, vote on a song and then disconnect and enter as Anonymous again. That way a user can vote on a song multiple times. How it was supposed to work was that once you disconnect from Blicko all the votes from that user gets removed. In the version tested this did not work and it made it possible to exploit it for one personal gain. Once the bug was found the log in option as: “Anonymous” was disabled. This meant that the only way to enter Blicko was through Facebook, and if a logged in user left the Station his or hers votes would be withdrawn.

A lot of positive feedback was given orally during the test session and some participants asked a lot of questions to the moderators about the service. Comments from the questionnaire that the participants were asked to fill out can be found in Appendix: 8.3.3.4.

Another observation during the test session was that when participants got a total score that was negative on songs which they had submitted the feeling became less positive. Some participants complained a little that all the songs they submitted where down voted and just stayed there with a negative score.
6. Analysis

6.1 Introduction

This analysis will concern different issues and factors that were recognized during the Usability Optimization (see chapter: 4) and Usability Testing (see chapter: 5). The data and metrics collected, as well as comments and reflections etc., will be brought up from previous parts of the study and put in a larger perspective.

6.2 Usability optimization

In the Usability Optimization section of this thesis (see chapter: 4) different components of the UI in Blicko got reviewed thoroughly according to certain aspects of HCI concepts and factors influencing usability. Some of the consecutive implementations are presented in chapter: 4.3, where the reasons and practical issues of design are also mentioned. Since then the UI has undergone further improvements, but most essentially a thorough scrutinizing during the Usability testing (see chapter: 5). During which many of the addressed or imagined issues emerged in somewhat amplified nature because of the meticulous examination in the task-testing (see chapter: 5.3.1), and because of the “stress testing” in the collaborative test session (see chapter: 5.3.2). It was satisfying to see that, except for in details, no crucial flaws were found in the UI or its functions that were not already anticipated or foreseen.

The iterative type of development of a Stations interface has proven to be very effective. The phenomena mentioned in chapter: 4.2.1.5 (Prototyping), has enforced a type of agile re-evaluation (by us developers) that brought out many flaws along the way. When addressing a small detail within an interface component, other issues can appear that perhaps are more deeply rooted that need immediate attention and correction. When several compromises add up a bigger change might have to be implemented. One example of how this affected Blicko is the splitting of the layouts for smartphone in relations to other platforms (see chapter: 4.3). The separation of Big and Small (see chapter: 4.3.1) were the biggest reconstruction of the UI, but it also brought many advantages with it. Without having two separate interfaces (i.e. layouts) the “onbeforeunload” function (see chapter: 4.3.3) would have had to been implemented into the single layout, and therefore affecting all platforms that access a Blicko Station. The same goes for the implementation of “Touch events” (see chapter: 4.3.2) as well as other issues related to responsiveness etc.

The design choices that were made early on when it comes to the visual aspects of Blicko were often grounded in HCI factors, such as: legibility and responsiveness etc., but were also inevitably influenced by personal taste. It was very satisfactory to receive appreciation during the user testing about the aesthetics of a Blicko Station and the service’s UI elements. Much of the feedback received in the test sessions justified decisions that were implemented because of factors already thought of in the usability optimization. One example of this is the Touch event implementation (see chapter: 4.3.2).
6.3 Usability testing

When examining task success data one of the most important values comes from identifying different groups of participants, who perform differently or encounter different sets of problems. These different groups can for example be age group, previous experience, education etc. [74] The factors can then be evaluated to make improvements to the service and help to identify usability problems. In the task oriented test, all of the participants succeeded in completing all of the tasks, which makes it difficult to extract any valuable information from the task success rate (100%). These results indicate that the main functions in Blicko are indeed intuitive and relatively easy figure out. The testers had no preconceived notion of Blicko at all, but our observations, as well as collected comments from the questionnaires, demonstrate that the UI has a high level of usability. All participants used Internet on a daily basis and by looking at table 8 it is clear that all participants use Internet for various things. The most important result was perhaps that 7 out of 8 used the Internet to: “Listen to music”. This indicates the common adaption of music streaming services that are highly relevant to Blicko for apparent reasons.

The task completion time varied a bit between individual participants (see chapter: 5.3.1.3), which is reflected in the high values in the charts on some of the tasks in both Big and Small (see fig. 31 and fig. 32). This is because of the variation in task completion time among the participants. Looking closer at the data (see Appendix: 8.3.2.2) it is revealed that two testers stood out as having slower times than the rest, which in turn created a bigger variation (higher average) in the task completion time. The two participants who took the longest time to complete all of the tasks were also the two oldest. Whilst it was not unexpected, it is also an indicator of room for improvement. For example by adjusting font size or contrast to better suit the visually impaired.

Looking at the graphs (see fig. 31 and fig. 32) it is obvious that task 3 (see table 6) was the one which took the longest time to complete in Small. In the Big layout on the other hand, it was the second most time consuming task. It was expected that task 3 would take a long time in comparison to the other tasks since the task was the most complicated one. The task was to submit three songs and see which position it gets. Here some users would send in three songs from the same artist while others searched for three different artists so the possible variation on the task completion time was very big.

A more surprising result was noticed with task 4 in Big (see fig. 31) which on average took longer time than task 3, even though it was expected to be significantly faster (as confirmed in Small (see fig. 32)). The task was to submit a song by the artist “Yes”. When looking closer at the individual task completion data (see Appendix: 8.3.2.2) in the Big layout, it becomes clear that there were two users who brought up the average time significantly. These two testers also commented stating their difficulty with identifying which text that represented the album name, versus the artist name. Another participant was also observed trying to sort the list by clicking on the name of the song and on the name of the artist. The participant noticed the mistake and eventually managed to submit the correct song. It coheres with the comments posted on the task, but should perhaps be cautiously considered. Task 4 is special in that it is a quite “synthetic” task that perhaps not accurately represents a real usage scenario. It is not expected (i.e. common) that users search for an artist without having any knowledge of song titles or album titles related to that very same artist (i.e. user only post queries of artists they know). Once a user recognizes a single track
item in the results list, he or she can easily identify the track, album and artist title of all the other items because they have the same (corresponding) information in the same position. For example if a user searches for “Michael Jackson” and recognizes a (single) song title in the search result, he or she can thereafter understand (decipher) which label is which for the rest of the results. This phenomenon is confirmed by the fact that no track, artist or album title confusion was apparent in task 3 (which contains very similar actions to task 4). When the participants submitted songs of their choice, which they easily recognized in the results, the problem with identifying tracks was nonexistent (see Appendix: 8.3.2.2). Some user might therefore have recognized the band “Yes” and did not have any problems sorting out which label was the artist and which label was the song. A better test would maybe have been to use a band which is less famous to get more consistent data.

Looking at individual tasks the completion time generally followed the optimal number of clicks, the more clicks required to complete the task the longer it took to complete it, except for a few cases mentioned in the previous paragraphs. The simpler tasks took shorter time and when broken down into single actions, we noticed that each task that contained a previously performed action went significantly faster than the previous. It was for example observed that after visiting the “Search” view once, the time it took to navigate there thereafter were marginal. This shows some learnability but to really get data on learnability another test is required.

The post-task ratings from chapter 5.3.1.4 shows how difficult the participants thought the tasks were. Looking at Big and Small separately there are only one task from each that really stood out as being more difficult than the others. For Big it was task 4 (submit any song by the band “Yes” and see what position it gets) and for Small it was task 9 (try to find out which action or event the user Jesper Ahlberg did last). Task 4 in Big was also the one which participants took longest time to complete (see fig. 31), and it was not expected that the task would take the longest time to complete. The task was to find a song by the artist “Yes” and submit it. The data from the post-task ratings further supports that this task was not obvious in how to complete it since the participants thought the task was more difficult than the others. The reason for this seems to be due to the fact that the participant now had to read all the labels in the search list which contains the name of the song, artist and album. Some participants thought these labels were difficult to distinguish.

For Small, task 9 (see table 6) stood out as the most difficult one according to the participants. Task 9 asked the participants to find out what the last action of the user “Jesper Ahlberg” was. Why this task was considered the most difficult can be due to many factors. Looking at the comments made by the facilitator and the users (see Appendix: 8.3.2.2) it is obvious that the icon symbolizing “Feed” might not be intuitive enough. Many participants had problems interpreting the icon and some simply arrived to feed by trial and error. Most participants also tried to click on the user name in the “User” section to find out the actions of “Jesper Ahlberg”. Once they arrived to the right tab the quickly figured out where the information they wanted were.

Overall the post-task rating is very good and all of the ratings except one are above three (see fig. 33 and fig. 34), which would indicate that the tasks are easy rather than difficult. The differences in the score of the tasks in Big and Small can be due to the design differences but the differences are so small and the number of participants so few that each participants score becomes very influential when it comes to showing the average.
Looking at the graphs with the average click per task and optimal number of clicks (see fig. 35 and fig. 36) it is easily seen that the tasks that required most clicks were also the ones that the participants clicked most on. This is expected since the more clicks you are required to make the more mistakes could occur. The tasks which required very few clicks (0-2 clicks) were the ones which also had least clicks from the participants. These graphs also show that task 9 required some effort even though it should require relatively few clicks. Looking at the standard deviation ("error bars") in fig. 35 and fig. 36, all except three tasks has a lower bound which includes the optimal number of clicks. This means that most of the future users have a very good chance of clicking the optimal way or close to the optimal way the first time they try it. We were unable to find any significant patterns that would indicate why some users clicked more than others. This is probably due to the small sample size and using a bigger sample size would probably yield more visible patterns. A hypothesis which is somewhat base on loose observations during the testing is that the younger users were more willing to try around. This without feeling committed to each click, being more confident that there were always ways to correct mistakes.

Analyzing the graphs which shows participant efficiency (see fig. 37 and fig. 38) it becomes clear that two users stood out as slowest when it came to completing tasks. Participant 7 were able to finish about 1,3 tasks per minute and participant 6 about 0,9 tasks per minute. Comparing these numbers to the average (calculated by adding up the task/minute times and dividing it by the number of participants) task/minute for Big and Small separately shows that both participant 7 and 6 were below half of the average task/minute. The average for Big was about 2,68 task/minute and for Small about 1,92 task/minute. Taking a closer look on these two participants’ reveals that they were also the two oldest in the test (see table 7). Even though they have experience on the Internet and use it frequently it can be difficult to totally grasp a new concept. Both the participants completed all of the tasks but required more time in total. But since the number of participants was so small it is difficult to say if the age has a big or a small impact, it might just be a coincident.

In the test for Small, participant number 3 and 8 did not own a smartphone, but still managed to handle the test on a smartphone very well. This is a very positive result since it inherits more difficult aspects. The learnability of operating the smartphone device itself has to be considered to some degree in these cases. What the test therefore shows is that even though users might not own a smartphone, the interface is easy enough for inexperienced smartphone users to handle.

The SUS-score for the task oriented test gave a good result. According to Tullis and Albert an average score above 80 can be considered pretty good, while one below 60 relatively poor. [74] In total it is possible to get from 0 to 100 where 100 represents a perfect score (i.e. percental scale). The standard deviation from the average SUS-score from Big (see fig. 41) gives the boundaries from 63,52 to 88,98 which would, according to Tullis and Albert, be considered relatively good. For Small (see fig. 41) the results were even better and the boundaries ranges from 73,83 to 98,66. Analyzing each individual results show that a few users might actually have misinterpreted some of the questions. One example is a participant who answered with a high score on all of the positive worded questions and a low score on all of the negatively worded questions, except for one question where the participant scored with the highest number. This could of course be an active choice but considering how the participant answered on the other questions, it is more likely that this question was
misinterpreted. The reason for this can be many but one of the biggest reasons might be the language barrier. All of the questions were given in English and the native language of the users that participated in the task oriented test all has Swedish as native language. The two participants who scored lowest on the SUS-score (see fig. 39 and fig. 40) for the task oriented test were both over 50 years old. These two participants also had the longest task completion time which coheres with the SUS-score they gave.

The SUS-score from the collaborative user testing also gave good results. Looking at fig. 44 it shows the standard deviation (shows the variability of the data from which the average score is calculated from (see chapter: 5.3.2.2)) for the average SUS-score in Big and Small during the collaborative user testing. Looking at the standard deviation for Big it got a lower boundary at 76,00 and an upper boundary at 100,00. A lower limit at 76,00 and an upper limit at 100,00 would indicate that this is a good result for Big. For Small the boundaries goes from 61,92 to 86,08 (see fig. 44). These results can be considered relatively good and the lower boundary does not go below 60 (which would have been considered relatively poor). [74] Overall the SUS-score on both the test sessions were good. There were only one exception where a participant had a score below 60, namely tester number 7 in collaborative which resulted in: 57,5 (see fig. 43). One interesting comment one of the participants made after the task oriented test was that the font size for the search results in Small might be too small. This was after the participant in question had done the test in Big and happened to see the version for the smartphones. The comment was from one of the participants with an age over 50, and it indicates that it might be hard to read the text if you are visually impaired. The font size has thereafter been increased slightly in the Small layout.

Looking at the events data collected during the collaborative test (see Appendix: 8.3.3) there are some interesting behavior which can be observed. The sum of the votes by the users was 23, where negative votes counts as -1 and positive votes as +1. This result is negatively influenced by the fact that the data does not take account for if a user removes his or her vote from a song, nor does it count the vote a song is given upon submission. The value of 23 however, still remains good since it shows that there are more up votes than down votes. In total there was 320 votes (see fig. 45).

One really interesting behavior that the data reveals was that some users submit a song and then directly after submitting down votes several songs and not up voting anything (see table 11). This behavior is an inherent flaw related to the differential voting algorithm (see chapter: 3.3.1). Since the user has the ability to down vote songs, the optimal way for a user to promote their own song is simply by down-voting all other songs. They can be seen as “competitors” in “Upcoming” so why not minimize their threat to your own advantage. This is not a beneficial behavior when it comes to collaboration and voting in a bigger picture. The behavior aimed for is that a user should submit songs he or she thinks everyone else will like, and gain votes according to its actual (regarding context and situation) popularity. Exploitation by down-voting the “competing” songs in upcoming distorts the result of the system. This phenomenon becomes more apparent the smaller the number of users is.

Apart from the users who just want their own submission to reach the top the overall behavior by the participants were very good. The pie chart (see fig. 45) shows that voting is the most popular action with a total of 320 instances and submitting songs the second most popular with 116 instances. This is a behavior to strive for since it means that users are more
active in voting on the playlist in “Upcoming”, rather than submitting songs. If it was the reverse that more users submit songs than voting, the playlist in “Upcoming” would soon be so overflowed that the users would have trouble finding the songs in the playlist that they like. The possibility of users spamming songs is a potential problem, but with active users who vote on songs the spam problem decreases. The goal is to have users vote on songs in the playlist in “Upcoming” and only submit songs if they feel they really have a good contribution according to the current context.

The overall feeling from the comments made by the participants is very positive. The comments made from the task oriented test (see Appendix: 8.3.2.3) and the collaborative test (see Appendix: 8.3.2.4), are although quite different from each other. When faced with the question “What did you like best about Blicko”, the task oriented participants seemed to focus on how easy Blicko was to use, which perfectly coheres with Blicko’s intentions. Here follows a relevant comment from a participant in the task oriented test which tested on Small:

“Very easy to use. Nice idea (the whole idea, rating system etc) and very well executed. Nice layout and interface. The amount of music (spotify). Facebook integration to make it more personal and a way to identify yourself and others.”

(see Appendix: 8.3.2.3)

When answering the question: “What did you like least about Blicko”, most of the comments seemed to address details. For example people mentioned that they had “trouble understanding the Feed-button” and that some buttons or symbols (i.e. metaphors) were not intuitive enough. This has given us input that we need to make these UI elements more intuitive or apparent. Some comments also gave some ideas to how the service can develop with more features like being able to comment songs, add friends and so on. Positive comments from the collaborative test session were, in comparison on the contrary, mostly aimed towards the concept (i.e. wider perspective). Many users thought the concept was good and they enjoyed being able to interact with the music being played. On the comments of what they liked least about Blicko the answers differed a lot. Some of them focusing on specific things like the sorting algorithm or that the upcoming playlist will not fit a small screen. A summary of the most significant user feedback can be found in table 12.
During the task oriented test one of the things many participants mentioned was that it was very easy to understand it once you had experience it. The first time they tried something it was a little bit tricky but after they did it once it was very intuitive how to use it. This is a sign that the learnability of Blicko might be very high. Even though there was only one round of task oriented testing the comments from the participants indicates that it has a high level of learnability. To truly see how good the learnability is another round of testing is needed with the same participants.

During the task oriented test a few comments about the back button in Small were made. Because it was the first time the user tried Blicko the first thought on how to go to the last page visited, was to use the back button. But once the user had understood the concept of being able to navigate through Small only using the built in buttons, the problem with the back button disappeared. The back button in Big did not pose to be a problem at all and no user mentioned or experienced it. Therefore the design of Big tends to be intuitive enough for users not to use the back button to go to the last tab visited (i.e. the tabular menu (see chapter: 4.2.3.2) distinctly offers a way of traversal).

<table>
<thead>
<tr>
<th>Test</th>
<th>Test layout</th>
<th>User #</th>
<th>Feedback</th>
<th>Response</th>
</tr>
</thead>
<tbody>
<tr>
<td>Task oriented test</td>
<td>Big</td>
<td>4</td>
<td>“Difficult to see if the song actually gets submitted. Maybe some indication that the song is submitted when you press submit. Difficult to see what band name. Should be able to sort by artist and song etc.”</td>
<td>Implement some kind of indicator to show that a song was successfully submitted. Sorting of search results might be implemented later.</td>
</tr>
<tr>
<td>Task oriented test</td>
<td>Small</td>
<td>2</td>
<td>“I couldn’t really use my &quot;back&quot; button on my phone, but once I got used to the built in buttons it really wasn’t a problem anymore. Not really any other flaws.”</td>
<td>Has since implemented “on before unload” (see chapter: 4.3.3) to cope with this problem.</td>
</tr>
<tr>
<td>Task oriented test</td>
<td>Big</td>
<td>1</td>
<td>“Feed-icon (non-intuitive).”</td>
<td>Going to find a better symbol for the feed-icon</td>
</tr>
<tr>
<td>Task oriented test</td>
<td>Small</td>
<td>2</td>
<td>“Make it even more personal, being able to write comments on songs and votes, add friends (don’t know if facebook friends are integrated in some way), join groups etc.”</td>
<td>This is something which we hope to implement in the future but for now we are focusing on getting the base stable and user friendly</td>
</tr>
<tr>
<td>Collaborative test</td>
<td>Big</td>
<td>4</td>
<td>“that the playlist had a fixed height and that the tabs were below the playlist, a user with a low resolution wouldn’t see the tabs and a user with a high resolution wouldn’t be able to use all the space on the screen.”</td>
<td>The height of the playlist has been lowered but still needs a minimum (and maximum) height for it to have any function. If the resolution is too small on the device the user should go to the mobile version. A mobile button has now been implemented to access Small</td>
</tr>
</tbody>
</table>

Table 12 - Some of the user feedback received and response to the problems. All the comments can be found in Appendix: 8.3.2.
6.4 Conclusions and future

Looking back at all the results from the Usability Optimization (see chapter: 4), it was very satisfying to see so many concepts and features confirmed efficient during the succeeding Usability Testing (see chapter: 5). The optimization section brought forward many issues of which many where related to the difficulties of combining web interfaces for different devices. When the compromise were inevitable and the layouts of Big and Small were separated (see chapter: 4.3.1), the whole process regained momentum again. This conclusion indicates that, by for example means of iterative development, we successfully identified and managed issues as efficient as possible given the situation we were in. Another conclusion we have reached is that it is indeed possible to create an immensely responsive interface for mobile platforms without native applications. By applying “bridging” tools and technologies (such as iScroll [66] or Touch events (see chapter: 4.3.2)) an almost as “rapid” UI is achievable based in web standards. Because of this the different features of the device that is used, such as input type or screen resolution, does not necessarily have to have a negative impact on the overall user experience.

After looking at how users vote in Blicko another voting system has been implemented. The new system implemented is a “Positive only” system (see chapter: 3.3.2) which makes it impossible to exploit the system by down voting “competing” songs. The best way a user can promote their own songs with the new algorithm is simply by submitting songs he or she imagines being popular in the current context. This system fixes the problem with a few users down voting all the other songs in order to promote their own.

We have learned a great deal about the advantages of testing during the process of this thesis. We strongly agree with this quote by Steve Krug from his book “Don’t make me think”:

“Testing always works, and even the worst test with the wrong user will show you important things you can do to improve your site.” [52:134]

It is fascinating to see this in action and as you let people test your product you get so much knowledge of how to improve it. The testing really did work and a lot of usability issues were found. Every comment and every action the user takes gives some kind of feedback, which can in turn be used to improve the product. Even though the number of testers in this thesis was few and there were only two tests a lot of great feedback was gathered. Because of this, a lot of suggestions for improvements have been made after the test and they can be found in Appendix: 8.3.1.

Testing in the IT-pub Kistan turned out to be a really good test and experience. Most of the participants in the IT-pub were in a computer science major which puts the service of Blicko in an extra scrutinizing situation. Some of the students really explored Blicko thoroughly which became apparent when two users found a loophole which they quickly exploited. The effect was quite innocent but extremely valuable to us as developers. We had foreseen a similar scenario but when it occurred we quickly got aware of its severity, and took care of the issue in a matter of minutes. Stress testing of web services in situations like this is therefore highly recommended, since a more thorough examination is hard to find. We hope to be able to keep working closely with the student unions of KTH in order to further test and evaluate Blicko.
In the future we hope to have native smartphone applications for the major devices on the market, such as iPhone and Android. By developing an application we get more control over the responsiveness and can also utilize the built in buttons on the phones. The idea is that the application would be a wrapper application to the Blicko Small page. The layout would be rendered as an ordinary webpage by means of platform specific WebView mechanisms that mimics the functions of a browser from within an application. We also hope to implement a system for users to locate Blicko Station according to their proximity. This would be possible by comparing the set location data of a Station and comparing them to the position of the user, by means of so called: geolocation. The nearby venues that use Blicko could be displayed as markers on a map over the local area etc.

We have taken the opportunity of submitting Blicko as a submission in the “Demos and experience” category of the conference: MobileHCI. The conference covers many topics regarding the social music service of Blicko and is congenially taking place in Stockholm in August. By then we strive for having an open to the public (by allowing access to www.blicko.com) alpha version ready where users can interact fully with all parts of the service. There will how ever be no announced release since we first want to experience how the service behaves under stress or load. The priority after that is to perhaps gain a small user base, and as it eventually grows the most sustainable of the business models (see chapter: 3.4) would initially be: “Ad and information based”. Since it does not conflict with the interests of any connected libraries, and is relatively easy to implement (both to include and exclude (if necessary)). It is also in our intentions to expand Blicko further by adding more libraries that a Station can use. The more music providers the more content can in turn be provided by us, perhaps even without account restrictions or paid models etc. Candidates for such expansion of libraries could for example be: YouTube (even video perhaps) and Grooveshark.

The concluded thought of the entire thesis is that it was very successful. We feel that we have prevailed in addressing the issues mentioned in the Problem definition (see chapter: 1.2), and reached our Expected results (see chapter: 1.4). We have successfully developed and designed a very intuitive interface and service, which then the usability testing verified to be true.
7. References

The sources below are formatted according to the IEEE citation standard. The references are numbered in order of (first) appearance.


[34] “Google - +1 Button.” Internet: www.google.com/+1/button, [Apr. 11, 2011].


8. Appendixes

8.1 Division of work

Since this thesis is a joint report written by two authors, some areas of responsibility has been divided between the two of us. However, the majority of the thesis is shared in all aspects possible. We have been constantly synchronized in each other's subject areas and exchanging feedback throughout the whole process. Both were taking part in all of the research, writing and conducting of tests etc., which eventually has resulted in this thesis and report.

The first author: Jesper Ahlberg concentrated on the Usability optimization (see chapter: 4), while the second author: Andreas Andrén had the Usability testing (see chapter: 5) as his corresponding area of responsibility. This including the chapter's related Appendixes and images etc. However, all parts of the text have been reviewed by both of us and the remaining sections can be seen as a result of collaboration between us.

The amount of hours put in has been identical and all work has been condened together, most often on location at the EIT ICT (Wireless@KTH) Labs. We have also spent some time implementing changes to the UI and Blicko in general, together with our associate project member: Theodor Zettersten (see Acknowledgements).

8.2 Digital appendixes

All digital resources collected during the process of writing this report and some associated material can be retrieved in two different ways. Either by requesting them directly from either author (contact information on first page), or by retrieving them at the following URL: http://goo.gl/B1bqz

8.2.1 Web references

Saved HTML files of web-resource as they were on date indicated i list of references (see chapter: 7).

8.2.2 Usability testing screen recordings

Video files of screen recordings during task based test sessions conducted on computer (see chapter: 5.2.1)

8.2.3 Questionnaire forms, data and excel spreadsheets

The actual forms from the questionnaires as well as all the data collected from participants in: “Demographic background”, “Evaluation”, “Task test protocol” and “Collaborative” questionnaires (see chapter: 5.2.1.2 and 5.2.2.1). This appendix also contains five Excel spreadsheets containing the data and the graphs for chapter 5.3.
8.2.4 Images and screenshots
Various interface images and resources saved during the process of developing of Blicko, writing the report, and the permissions to reproduce images from other sources.

8.2.5 Test session material
Email announcement of test at TMEIT Pub (2011-03-29) and Facebook event, as well as related images.

8.2.6 Style sheets
The four CSS files used within Blicko: “global”, “stations”, “stations_small” and “stations_big”.

8.3 Testing appendixes

8.3.1 Post test implementations
- Changing the buttons in Small so that when you are in navigation, submit, vote, feed, user, search etc the back button will show up in the left corner. This is a change from before when the navigation button was always seen in the top left corner and once you entered the navigation menu the logout button was shown instead of the navigation button. The logout button was not even seen by the users and sometimes the users pressed the top left corner twice which resulted in them logging out.
- Font size of artist in search results and upcoming items increased (due to “Yes”-task result).
- Switched buttons in top navigation small (bigger icons and different colors).
- Changed the search field from type=text to type=search so that the software keyboard on smartphones get a search keyboard rather than a normal keyboard.
- Added “Submitted” button which shows up after you press the “submit” button in Big after you have searched on songs. Gives you feedback that the song has been submitted.
- Added withdraw button on songs that you have submitted. You can now press on a song in upcoming and then see a button named “withdraw” which removes songs.
- After collaborative test added onHover withdraw button in upcoming after suggestion from participant of the test.
- Changed the “sign out” button on the facebook login in the ConnectBox to a red X.
- Change feed icon to something which reflects better what it is.
- Change player icon to upcoming icon.
- Removed block (down vote) since some users tend to down vote everything they see in order to get their own song to the top. Changed the voting concept to “Positive only”.
- Added captcha to keep bots and spammers from joining through anonymous.
- Addition of Station settings regarding number of tracks in upcoming allowed per user.
- Addition of password protections when connecting to Station.
- Potential addition of expiration time on tracks that has been stuck in “upcoming” for a long time without progressing upwards in queue, cleaning of “stale” tracks.
- Added time-out of sessions that is inactive (1 hour for now).

8.3.2 Questionnaire data
## Demographic Background

<table>
<thead>
<tr>
<th>Age Group</th>
<th>Count</th>
<th>Percentage</th>
</tr>
</thead>
<tbody>
<tr>
<td>8-10 years</td>
<td>13</td>
<td>12%</td>
</tr>
<tr>
<td>11-12 years</td>
<td>15</td>
<td>14%</td>
</tr>
<tr>
<td>13-15 years</td>
<td>17</td>
<td>16%</td>
</tr>
<tr>
<td>16-17 years</td>
<td>20</td>
<td>19%</td>
</tr>
<tr>
<td>18-20 years</td>
<td>23</td>
<td>22%</td>
</tr>
<tr>
<td>21-25 years</td>
<td>28</td>
<td>27%</td>
</tr>
</tbody>
</table>

### Gender Distribution

<table>
<thead>
<tr>
<th>Gender</th>
<th>Count</th>
<th>Percentage</th>
</tr>
</thead>
<tbody>
<tr>
<td>Male</td>
<td>40</td>
<td>38%</td>
</tr>
<tr>
<td>Female</td>
<td>45</td>
<td>44%</td>
</tr>
<tr>
<td>Unknown</td>
<td>10</td>
<td>9%</td>
</tr>
</tbody>
</table>

### Ethnicity

<table>
<thead>
<tr>
<th>Ethnicity</th>
<th>Count</th>
<th>Percentage</th>
</tr>
</thead>
<tbody>
<tr>
<td>Caucasian</td>
<td>35</td>
<td>34%</td>
</tr>
<tr>
<td>African American</td>
<td>40</td>
<td>39%</td>
</tr>
<tr>
<td>Asian</td>
<td>15</td>
<td>14%</td>
</tr>
<tr>
<td>Hispanic</td>
<td>5</td>
<td>5%</td>
</tr>
<tr>
<td>Native American</td>
<td>5</td>
<td>5%</td>
</tr>
</tbody>
</table>

### Education Level

<table>
<thead>
<tr>
<th>Education</th>
<th>Count</th>
<th>Percentage</th>
</tr>
</thead>
<tbody>
<tr>
<td>High School</td>
<td>30</td>
<td>29%</td>
</tr>
<tr>
<td>College</td>
<td>45</td>
<td>44%</td>
</tr>
<tr>
<td>Graduate</td>
<td>20</td>
<td>20%</td>
</tr>
<tr>
<td>Unknown</td>
<td>5</td>
<td>5%</td>
</tr>
</tbody>
</table>

### Employment Status

<table>
<thead>
<tr>
<th>Status</th>
<th>Count</th>
<th>Percentage</th>
</tr>
</thead>
<tbody>
<tr>
<td>Full-time</td>
<td>30</td>
<td>30%</td>
</tr>
<tr>
<td>Part-time</td>
<td>40</td>
<td>40%</td>
</tr>
<tr>
<td>Unemployed</td>
<td>15</td>
<td>15%</td>
</tr>
<tr>
<td>Unknown</td>
<td>5</td>
<td>5%</td>
</tr>
</tbody>
</table>

### Marital Status

<table>
<thead>
<tr>
<th>Status</th>
<th>Count</th>
<th>Percentage</th>
</tr>
</thead>
<tbody>
<tr>
<td>Single</td>
<td>40</td>
<td>39%</td>
</tr>
<tr>
<td>Married</td>
<td>30</td>
<td>30%</td>
</tr>
<tr>
<td>Divorced</td>
<td>15</td>
<td>14%</td>
</tr>
<tr>
<td>Unknown</td>
<td>5</td>
<td>5%</td>
</tr>
</tbody>
</table>

### Income Level

<table>
<thead>
<tr>
<th>Income</th>
<th>Count</th>
<th>Percentage</th>
</tr>
</thead>
<tbody>
<tr>
<td>Low</td>
<td>20</td>
<td>20%</td>
</tr>
<tr>
<td>Medium</td>
<td>40</td>
<td>40%</td>
</tr>
<tr>
<td>High</td>
<td>20</td>
<td>20%</td>
</tr>
<tr>
<td>Unknown</td>
<td>10</td>
<td>10%</td>
</tr>
</tbody>
</table>

### Religion

<table>
<thead>
<tr>
<th>Religion</th>
<th>Count</th>
<th>Percentage</th>
</tr>
</thead>
<tbody>
<tr>
<td>Christian</td>
<td>45</td>
<td>44%</td>
</tr>
<tr>
<td>Jewish</td>
<td>15</td>
<td>15%</td>
</tr>
<tr>
<td>Muslim</td>
<td>10</td>
<td>10%</td>
</tr>
<tr>
<td>Hindu</td>
<td>5</td>
<td>5%</td>
</tr>
<tr>
<td>Unknown</td>
<td>5</td>
<td>5%</td>
</tr>
</tbody>
</table>

### Political Affiliation

<table>
<thead>
<tr>
<th>Affiliation</th>
<th>Count</th>
<th>Percentage</th>
</tr>
</thead>
<tbody>
<tr>
<td>Republican</td>
<td>30</td>
<td>30%</td>
</tr>
<tr>
<td>Democrat</td>
<td>40</td>
<td>40%</td>
</tr>
<tr>
<td>Independent</td>
<td>15</td>
<td>15%</td>
</tr>
<tr>
<td>Other</td>
<td>5</td>
<td>5%</td>
</tr>
<tr>
<td>Unknown</td>
<td>5</td>
<td>5%</td>
</tr>
</tbody>
</table>

### Smoking Status

<table>
<thead>
<tr>
<th>Status</th>
<th>Count</th>
<th>Percentage</th>
</tr>
</thead>
<tbody>
<tr>
<td>Smoker</td>
<td>20</td>
<td>20%</td>
</tr>
<tr>
<td>Non-smoker</td>
<td>40</td>
<td>40%</td>
</tr>
<tr>
<td>Unknown</td>
<td>10</td>
<td>10%</td>
</tr>
</tbody>
</table>

### Drinking Status

<table>
<thead>
<tr>
<th>Status</th>
<th>Count</th>
<th>Percentage</th>
</tr>
</thead>
<tbody>
<tr>
<td>Drinker</td>
<td>30</td>
<td>30%</td>
</tr>
<tr>
<td>Non-drinker</td>
<td>40</td>
<td>40%</td>
</tr>
<tr>
<td>Unknown</td>
<td>10</td>
<td>10%</td>
</tr>
</tbody>
</table>

### Exercise Level

<table>
<thead>
<tr>
<th>Level</th>
<th>Count</th>
<th>Percentage</th>
</tr>
</thead>
<tbody>
<tr>
<td>Low</td>
<td>20</td>
<td>20%</td>
</tr>
<tr>
<td>Medium</td>
<td>40</td>
<td>40%</td>
</tr>
<tr>
<td>High</td>
<td>20</td>
<td>20%</td>
</tr>
<tr>
<td>Unknown</td>
<td>10</td>
<td>10%</td>
</tr>
</tbody>
</table>

### Diet Type

<table>
<thead>
<tr>
<th>Type</th>
<th>Count</th>
<th>Percentage</th>
</tr>
</thead>
<tbody>
<tr>
<td>Vegetarian</td>
<td>30</td>
<td>30%</td>
</tr>
<tr>
<td>Omnivore</td>
<td>40</td>
<td>40%</td>
</tr>
<tr>
<td>Vegan</td>
<td>15</td>
<td>15%</td>
</tr>
<tr>
<td>Unknown</td>
<td>5</td>
<td>5%</td>
</tr>
</tbody>
</table>

### Social Media Use

<table>
<thead>
<tr>
<th>Platform</th>
<th>Count</th>
<th>Percentage</th>
</tr>
</thead>
<tbody>
<tr>
<td>Facebook</td>
<td>40</td>
<td>40%</td>
</tr>
<tr>
<td>Instagram</td>
<td>30</td>
<td>30%</td>
</tr>
<tr>
<td>Twitter</td>
<td>20</td>
<td>20%</td>
</tr>
<tr>
<td>Unknown</td>
<td>10</td>
<td>10%</td>
</tr>
</tbody>
</table>

### Technology Use

<table>
<thead>
<tr>
<th>Device</th>
<th>Count</th>
<th>Percentage</th>
</tr>
</thead>
<tbody>
<tr>
<td>Laptop</td>
<td>40</td>
<td>40%</td>
</tr>
<tr>
<td>Tablet</td>
<td>30</td>
<td>30%</td>
</tr>
<tr>
<td>Smartphone</td>
<td>20</td>
<td>20%</td>
</tr>
<tr>
<td>Unknown</td>
<td>10</td>
<td>10%</td>
</tr>
</tbody>
</table>

### Travel Habits

<table>
<thead>
<tr>
<th>Destination</th>
<th>Count</th>
<th>Percentage</th>
</tr>
</thead>
<tbody>
<tr>
<td>Domestic</td>
<td>20</td>
<td>20%</td>
</tr>
<tr>
<td>International</td>
<td>40</td>
<td>40%</td>
</tr>
<tr>
<td>Unknown</td>
<td>10</td>
<td>10%</td>
</tr>
</tbody>
</table>

### Career Goals

<table>
<thead>
<tr>
<th>Goal</th>
<th>Count</th>
<th>Percentage</th>
</tr>
</thead>
<tbody>
<tr>
<td>Healthcare</td>
<td>30</td>
<td>30%</td>
</tr>
<tr>
<td>Business</td>
<td>40</td>
<td>40%</td>
</tr>
<tr>
<td>Education</td>
<td>20</td>
<td>20%</td>
</tr>
<tr>
<td>Unknown</td>
<td>10</td>
<td>10%</td>
</tr>
</tbody>
</table>

### Other Comments

- Comments on work experience, education, hobbies, and interests are not provided in the survey data.
### Task Test Protocol

<table>
<thead>
<tr>
<th>Step</th>
<th>Task</th>
<th>Group 1</th>
<th>Group 2</th>
<th>Group 3</th>
<th>Group 4</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.4</td>
<td>88</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>3</td>
<td>2</td>
<td>0</td>
<td>5</td>
<td>32</td>
<td>30</td>
</tr>
<tr>
<td>2</td>
<td>4</td>
<td>0</td>
<td>0</td>
<td>4</td>
<td>4</td>
</tr>
<tr>
<td>6</td>
<td>15</td>
<td>0</td>
<td>5</td>
<td>30</td>
<td>30</td>
</tr>
<tr>
<td>5</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>6</td>
<td>0</td>
</tr>
<tr>
<td>4</td>
<td>0</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>4</td>
<td>0</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>4</td>
<td>0</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>3</td>
<td>0</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>15</td>
<td>0</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
</tbody>
</table>

*Notes:*
- Group 1: Task 1
- Group 2: Task 2
- Group 3: Task 3
- Group 4: Task 4

*Time:*
- 17:29:04
- 17:29:22
- 17:37:26
- 20:14:27

*Comments:*
- Group 1: Successfully completed all tasks.
- Group 2: Encountered some issues with the equipment.
- Group 3: Completed tasks with minimal errors.
- Group 4: Required additional support for completing tasks.

*Overall Feedback:*
- Positive overall satisfaction with the test protocol.
- Slight improvement needed for better organization of tasks.

*Next Steps:*
- Review and refine the test protocol for future tests.
<table>
<thead>
<tr>
<th></th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
<th>10</th>
<th>11</th>
<th>12</th>
<th>13</th>
<th>14</th>
<th>15</th>
<th>16</th>
<th>17</th>
<th>18</th>
<th>19</th>
<th>20</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>6</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>7</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>8</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>9</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>10</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>11</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>12</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>13</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>14</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>15</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>16</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>17</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>18</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>19</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>20</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Task Test Protocol**
<p>| Paragraph | Completed | Timestamp | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 | 16 | 17 | 18 | 19 | 20 | 21 | 22 | 23 | 24 | 25 | 26 | 27 | 28 | 29 | 30 | 31 | 32 | 33 | 34 | 35 | 36 | 37 | 38 | 39 | 40 | 41 | 42 | 43 | 44 | 45 | 46 | 47 | 48 | 49 | 50 | 51 | 52 | 53 | 54 | 55 | 56 | 57 | 58 | 59 | 60 | 61 | 62 | 63 | 64 | 65 | 66 | 67 | 68 | 69 | 70 | 71 | 72 | 73 | 74 | 75 | 76 | 77 | 78 | 79 | 80 | 81 | 82 | 83 | 84 | 85 | 86 | 87 | 88 | 89 | 90 | 91 | 92 | 93 | 94 | 95 | 96 | 97 | 98 | 99 | 100 |</p>
<table>
<thead>
<tr>
<th>Evaluation Questions</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
</tr>
</thead>
<tbody>
<tr>
<td>enablers</td>
<td>3</td>
<td>2</td>
<td>4</td>
<td>1</td>
<td>5</td>
<td>7</td>
<td>6</td>
<td>8</td>
</tr>
<tr>
<td>the competencies</td>
<td>6</td>
<td>3</td>
<td>7</td>
<td>2</td>
<td>1</td>
<td>4</td>
<td>5</td>
<td>8</td>
</tr>
<tr>
<td>key themes</td>
<td>8</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
</tr>
<tr>
<td>8.3.2.3 Evaluation</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
<td>5</td>
<td>6</td>
<td>7</td>
<td>8</td>
</tr>
<tr>
<td>8.3.2.4 Collaborative Questions</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>---------------------------------</td>
<td>---</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Question</strong></td>
<td><strong>Response</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1. What is the problem you are trying to solve?</td>
<td>I am trying to improve collaboration in a team project.</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2. What is your main goal?</td>
<td>Increase productivity and efficiency.</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3. What are the constraints you face?</td>
<td>Budget limitations and time constraints.</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4. What are the potential benefits of implementing this solution?</td>
<td>Improved teamwork and better communication.</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5. What are the potential risks of implementing this solution?</td>
<td>Resistance to change and potential for miscommunication.</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>6. How will you measure success?</td>
<td>By tracking productivity metrics and team satisfaction surveys.</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>7. What is your timeline for implementation?</td>
<td>Within the next quarter.</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>8. What is your budget for this project?</td>
<td>$50,000.</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>9. What are the dependencies of this project?</td>
<td>Availability of team members and required software.</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>10. What is the scope of this project?</td>
<td>The scope includes improving communication tools and training team members.</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Day</td>
<td>Activity</td>
<td>Time</td>
<td>Location</td>
<td>Notes</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>-----</td>
<td>----------</td>
<td>------</td>
<td>----------</td>
<td>-------</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Tuesday</td>
<td>9:00 AM - 10:00 AM</td>
<td>Classroom</td>
<td>Lecture on Programming Fundamentals</td>
<td>Prepare for Introduction to Java</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Tuesday</td>
<td>10:00 AM - 11:00 AM</td>
<td>Lab</td>
<td>Hands-on Practice with Java</td>
<td>Bring laptop</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Tuesday</td>
<td>11:00 AM - 12:00 PM</td>
<td>Lunch</td>
<td>Cafeteria</td>
<td>Free pizza and soda</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Tuesday</td>
<td>12:00 PM - 1:00 PM</td>
<td>Discussion</td>
<td>Group Project Planning</td>
<td>Meet in small groups</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Tuesday</td>
<td>1:00 PM - 2:00 PM</td>
<td>Lab</td>
<td>Advanced Java Concepts</td>
<td>Optional for advanced students</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Tuesday</td>
<td>2:00 PM - 3:00 PM</td>
<td>Group Work</td>
<td>Project Development</td>
<td>Work on group project</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Tuesday</td>
<td>3:00 PM - 4:00 PM</td>
<td>Break</td>
<td>Outside on Campus</td>
<td>Socialize and relax</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Tuesday</td>
<td>4:00 PM - 5:00 PM</td>
<td>Lecture</td>
<td>Java Design Patterns</td>
<td>Keynote by guest speaker</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Tuesday</td>
<td>5:00 PM - 6:00 PM</td>
<td>Lab</td>
<td>Project Review and Feedback</td>
<td>Review group project and get feedback</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Wednesday</td>
<td>9:00 AM - 10:00 AM</td>
<td>Classroom</td>
<td>Review of Java Fundamentals</td>
<td>Review for Java Test</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Wednesday</td>
<td>10:00 AM - 11:00 AM</td>
<td>Lab</td>
<td>Quiz on Java Fundamentals</td>
<td>Quiz format - multiple choice</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Wednesday</td>
<td>11:00 AM - 12:00 PM</td>
<td>Lecture</td>
<td>Introduction to Data Structures</td>
<td>Key topics include arrays and lists</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Wednesday</td>
<td>12:00 PM - 1:00 PM</td>
<td>Discussion</td>
<td>Interactive Q&amp;A Session</td>
<td>Ask any questions about data structures</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Wednesday</td>
<td>1:00 PM - 2:00 PM</td>
<td>Lab</td>
<td>Practice with Data Structures</td>
<td>Bring laptop</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Wednesday</td>
<td>2:00 PM - 3:00 PM</td>
<td>Group Work</td>
<td>Data Structures Project</td>
<td>Work on project in groups</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Wednesday</td>
<td>3:00 PM - 4:00 PM</td>
<td>Break</td>
<td>Library</td>
<td>Study or meet with peers</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Wednesday</td>
<td>4:00 PM - 5:00 PM</td>
<td>Lab</td>
<td>Advanced Data Structures</td>
<td>Optional for advanced students</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Wednesday</td>
<td>5:00 PM - 6:00 PM</td>
<td>Lecture</td>
<td>Java and Data Structures Integration</td>
<td>Connect Java concepts with data structures</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Thursday</td>
<td>9:00 AM - 10:00 AM</td>
<td>Classroom</td>
<td>Review of Data Structures</td>
<td>Review for Data Structures Test</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Thursday</td>
<td>10:00 AM - 11:00 AM</td>
<td>Lab</td>
<td>Quiz on Data Structures</td>
<td>Quiz format - multiple choice</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Thursday</td>
<td>11:00 AM - 12:00 PM</td>
<td>Lecture</td>
<td>Preparation for End-of-Course Exam</td>
<td>Key topics include review of all concepts</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Thursday</td>
<td>12:00 PM - 1:00 PM</td>
<td>Discussion</td>
<td>Class Wrap-Up</td>
<td>Recap of course content and next steps</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Thursday</td>
<td>1:00 PM - 2:00 PM</td>
<td>Lab</td>
<td>Final Project Development</td>
<td>Work on final project in groups</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Thursday</td>
<td>2:00 PM - 3:00 PM</td>
<td>Group Work</td>
<td>Final Project Review</td>
<td>Review final project with peers</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Thursday</td>
<td>3:00 PM - 4:00 PM</td>
<td>Break</td>
<td>Park</td>
<td>Take a walk and enjoy the outdoors</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Thursday</td>
<td>4:00 PM - 5:00 PM</td>
<td>Lecture</td>
<td>Final Exam Preparation</td>
<td>Review course content for final exam</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Thursday</td>
<td>5:00 PM - 6:00 PM</td>
<td>Lab</td>
<td>Final Exam Simulation</td>
<td>Practice for final exam</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Collaborative Questionnaire**

- How did you enjoy the class in general?
- What did you like most about the class?
- What could be improved in the class?
8.3.3 Collaborative test user events
98