Aktivitetsklasser

package com.example.personalschema.activities;

import android.os.Bundle;
import android.view.Menu;
import android.view.MenuItem;
import androidx.appcompat.app.AppCompatActivity;
import com.alamkanak.weekview.WeekView;
import com.example.personalschema.R;
import com.example.personalschema.adapter.WeekViewAdapter;
import com.example.personalschema.model.MyEvent;
import com.example.personalschema.util.MyFunctions;
import com.example.personalschema.viewmodel.MyViewModel;
import java.util.Calendar;

/**
 * @author Kani & Mishu
 * A class to show the active calendar with the events saved in Firestore.
 */
public class ActiveCalendarActivity extends AppCompatActivity {

    // private final String TAG = "ActiveCalendarActivity";
    // Calendar UI
    private WeekView weekView;
    private WeekView.SimpleAdapter<MyEvent> adapter;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_calendar);
        // Create the WeekView and set the Date and Time formatter
        weekView = findViewById(R.id.weekView);
        MyFunctions.setupDateTimeInterpreter(weekView);
        // Call a new viewModel and adapter to supply data to the WeekView
        MyViewModel myViewModel = new MyViewModel();
        adapter = new WeekViewAdapter(this, true);
        weekView.setAdapter(adapter);
        myViewModel.event.observe(this, myEvents -> adapter.submitList(myEvents));
        myViewModel.fetchEvents();
    }

    @Override
    public boolean onCreateOptionsMenu(Menu menu) {
        // Inflate the menu, this adds items to the action bar if it is present.
        return false;
    }
}

Bilaga: java-kod
getMenuInflater().inflate(R.menu.menu_main, menu);
return true;
}

@Override
public boolean onOptionsItemSelected(MenuItem item) {
    int id = item.getItemId();
    if (id == R.id.getToday) {
        // Scroll to today's date
        if (weekView.getNumberOfVisibleDays() == 1) {
            weekView.scrollToDate(Calendar.getInstance());
        } else {
            weekView.scrollToDate(MyFunctions.week(Calendar.getInstance()));
        }
    } else if (id == R.id.getDayView) {
        // Change the WeekView to show one day
        weekView.setNumberOfVisibleDays(1);
    } else if (id == R.id.getWeekView) {
        // Change the WeekView to show seven days
        weekView.setNumberOfVisibleDays(7);
    }
    return super.onOptionsItemSelected(item);
}

package com.example.personalschema.activities;

import android.app.TimePickerDialog;
import android.content.Intent;
import android.graphics.Color;
import android.graphics.RectF;
import android.os.Bundle;
import android.text.InputType;
import android.util.Log;
import android.view.LayoutInflater;
import android.view.Menu;
import android.view.MenuItem;
import android.view.View;
import android.view.ViewGroup;
import android.widget.DatePicker;
import android.widget.EditText;
import android.widget.PopupMenu;
import android.widget.RadioButton;
import androidx.appcompat.app.AlertDialog;
import androidx.appcompat.app.AppCompatActivity;
import com.alamkanak.weekview.WeekView;
import com.example.personalschema.interfaces.EventInterface;
import com.example.personalschema.util.MyFunctions;
import com.example.personalschema.R;
import com.example.personalschema.adapter.WeekViewAdapter;
import com.example.personalschema.model.MyEvent;
import com.example.personalschema.ui.DisplayMessages;
import com.example.personalschema.viewModel.MyViewModel;
import com.google.firebase.firestore.FirebaseFirestore;
import java.util.Calendar;
import java.util.HashMap;
import java.util.Map;

/**
 * @author Kani & Mishu
 * A class for the users who have Admin accounts.
 * The class shows a calendar view and the users can create new events (shifts).
 */
public class AdminCalendarActivity extends AppCompatActivity implements EventInterface, PopupMenu.OnMenuItemClickListener {

    private final String TAG = "AdminActivity";
    // Calendar UI, ViewModel and adapter
    private WeekView weekView;
    private MyViewModel myViewModel;
    private WeekView.SimpleAdapter<MyEvent> adapter;
    private MyEvent ev;
    // Firebase
    private FirebaseFirestore fStore;
    private TimePickerDialog picker;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_admin);
        // Create the WeekView and set the Date and Time formatter
        weekView = findViewById(R.id.weekView);
        MyFunctions.setupDateTimeInterpreter(weekView);
        // Call a new viewModel and adapter to supply data to the WeekView
        myViewModel = new MyViewModel();
        adapter = new WeekViewAdapter(this, this);
        weekView.setAdapter(adapter);
        myViewModel.event.observe(this, myEvents -> adapter.submitList(myEvents));
        myViewModel.fetchEvents();
    }

    // The method is called when the floating button is clicked to create a new event
    public void createEvent(View v) {
        // Inflate the layout
        LayoutInflater li = LayoutInflater.from(getApplicationContext());
        View view = li.inflate(R.layout.dialog_new_event, null);
    
    }
// Create a new AlertDialog and connect it to the layout
AlertDialog.Builder builder = new AlertDialog.Builder(AdminCalendarActivity.this);
builder.setView(view);
builder.setTitle(getString(R.string.floating_description));

// UI buttons and texts
final DatePicker date = view.findViewById(R.id.datePicker);
final EditText startTimeEditText = view.findViewById(R.id.startTimeEditText);
startTimeEditText.setInputType(InputType.TYPE_NULL);
final EditText endTimeEditText = view.findViewById(R.id.endTimeEditText);
endTimeEditText.setInputType(InputType.TYPE_NULL);
final RadioButton sskRadioButton = view.findViewById(R.id.SSKRadioButton);
final RadioButton uskRadioButton = view.findViewById(R.id.USKRadioButton);
final EditText info = view.findViewById(R.id.infoEditText);

// Set setOnClickListener for the EditTexts to get the time user chooses from a TimePickerDialog
final int[] startH = {0};
final int[] startM = {0};
startTimeEditText.setOnClickListener(t -> {
    final Calendar calendar = Calendar.getInstance();
    int hour = calendar.get(Calendar.HOUR_OF_DAY);
    int minutes = calendar.get(Calendar.MINUTE);
    // Create a new TimePickerDialog
    picker = new TimePickerDialog(AdminCalendarActivity.this,
        (tp, sHour, sMinute) -> {
            String time = getString(R.string.time_interval_colon_text_view,
                sHour, sMinute);
            startTimeEditText.setText(time);
            startH[0] = sHour;
            startM[0] = sMinute;
        }, hour, minutes, true);
    picker.show();
});

final int[] endH = {0};
final int[] endM = {0};
endTimeEditText.setOnClickListener(t -> {
    final Calendar calendar = Calendar.getInstance();
    int hour = calendar.get(Calendar.HOUR_OF_DAY);
    int minutes = calendar.get(Calendar.MINUTE);
    // Create a new TimePickerDialog
    picker = new TimePickerDialog(AdminCalendarActivity.this,
        (tp, sHour, sMinute) -> {
            endTimeEditText.setText(getString(R.string.time_interval_colon_text_view,
                sHour, sMinute));
            endH[0] = sHour;
            endM[0] = sMinute;
        }, hour, minutes, true);
    picker.show();
});
// If the positive button of the dialog is clicked, save the event information to Firestore
builder.setPositiveButton(getString(R.string.dialog_positive_button_text), (dialog, id) -> {
    DisplayMessages.displayToast(getString(R.string.dialog_positive_text), AdminCalendarActivity.this);

    // Event title
    String title = "";
    if (sskRadioButton.isChecked()) {
        title = getString(R.string.ssk);
    } else if (uskRadioButton.isChecked()) {
        title = getString(R.string.usk);
    }

    // Create a random Id for the event
    String eventId = MyFunctions.getNewEventId();

    // Create a map vector with the information about the event
    Map<String, Object> EventInfo = new HashMap<>();
    EventInfo.put(getString(R.string.field_event_id), eventId);
    EventInfo.put(getString(R.string.field_title), title);
    EventInfo.put(getString(R.string.field_info), info.getText().toString());
    EventInfo.put(getString(R.string.field_year), date.getYear());
    EventInfo.put(getString(R.string.field_month), date.getMonth());
    EventInfo.put(getString(R.string.field_day), date.getDayOfMonth());
    EventInfo.put(getString(R.string.field_start_hour), startH[0]);
    EventInfo.put(getString(R.string.field_start_min), startM[0]);
    EventInfo.put(getString(R.string.field_end_hour), endH[0]);
    EventInfo.put(getString(R.string.field_end_min), endM[0]);

    // Store the new data in Firestore
    fStore = FirebaseFirestore.getInstance();
    fStore.collection(getString(R.string.event_collection)).document(eventId).set(EventInfo)
        .addOnSuccessListener(aVoid -> Log.d(TAG + Constant.WRITE_TAG, "DocumentSnapshot successfully written!"))
        .addOnFailureListener(e -> Log.i(TAG + Constant.WRITE_TAG, "Error writing document", e));

    // Update the events and the WeekView
    myViewModel.fetchEvents();
});

builder.setNegativeButton(getString(R.string.cancel_button), (dialog, which) -> {
    DisplayMessages.displayToast(getString(R.string.dialog_cancel_text), AdminCalendarActivity.this);
    builder.create().show();
});

/**
 * When a shift is clicked the user gets a menu to add or delete applicants chosen for the shift.
 */
@Override
public void eventClick(MyEvent ev, RectF bounds) {
    this.ev = ev;
    // Set the layout
    final ViewGroup root = findViewById(R.id.activity_admin);
    final View view = new View(getApplicationContext());
    view.setLayoutParams(new ViewGroup.LayoutParams(1, 1));
    view.setBackgroundColor(Color.TRANSPARENT);
    root.addView(view);
    view.setX(bounds.centerX());
    view.setY(bounds.centerY());
    // Create a new PopupMenu and inflate the menu
    PopupMenu popupMenu = new PopupMenu(this, view);
    popupMenu.setOnMenuItemClickListener(this);
    popupMenu.inflate(R.menu.menu_popup);
    popupMenu.show();
}

@Override
public boolean onMenuItemClick(MenuItem item) {
    if (item.getItemId() == R.id.chooseApplicant) {
        // If the menu item is clicked go to ChooseApplicantActivity
        Intent intent = new Intent(AdminCalendarActivity.this, ChooseApplicantActivity.class);
        intent.putExtra(getString(R.string.intent_event_id), ev.getId());
        intent.putExtra(getString(R.string.intent_date), ev.getDate(ev.getStartTime()));
        intent.putExtra(getString(R.string.intent_time), ev.getTime(ev.getStartTime()));
        startActivity(intent);
    } else if (item.getItemId() == R.id.deleteApplicant) {
        // If the menu item is clicked go to DeleteApplicantActivity
        Intent intent = new Intent(AdminCalendarActivity.this, DeleteApplicantActivity.class);
        intent.putExtra(getString(R.string.intent_event_id), ev.getId());
        intent.putExtra(getString(R.string.intent_date), ev.getDate(ev.getStartTime()));
        intent.putExtra(getString(R.string.intent_time), ev.getTime(ev.getStartTime()));
        startActivity(intent);
    } else if (item.getItemId() == R.id.deleteEvent) {
        // If the menu item is clicked show a dialog
        AlertDialog.Builder builder = new AlertDialog.Builder(AdminCalendarActivity.this);
        String timeInterval = getString(R.string.time_interval_text_view, ev.getTime(ev.getStartTime()), ev.getTime(ev.getEndTime()));
        builder.setTitle(getString(R.string.delete_event_title));
        builder.setMessage(ev.getDate(ev.getStartTime()) + "", + timeInterval);
        // If the positive button on the dialog is clicked, delete the selected event/shift
        builder.setPositiveButton(getString(R.string.delete_applicants_button_text), (dialog, which) -> {
            FirebaseFirestore.getInstance().collection(getString(R.string.event_collection)).document(ev.
getId().delete();
    myViewModel.fetchEvents();
});
    builder.setNegativeButton(getString(R.string.cancel_button), (dialog, which) ->
        DisplayMessages.displayToast(" ", AdminCalendarActivity.this));
    builder.create().show();
} return false;
}

@override
public boolean onCreateOptionsMenu(Menu menu) {
    // Inflate the menu, this adds items to the action bar if it is present.
    getMenuInflater().inflate(R.menu.menu_main, menu);
    return true;
}

@override
public boolean onOptionsItemSelected(MenuItem item) {
    int id = item.getItemId();

    if (id == R.id.getToday) {
        // Scroll to today's date
        if (weekView.getNumberOfVisibleDays() == 1) {
            weekView.scrollToDate(Calendar.getInstance());
        } else {
            weekView.scrollToDate(MyFunctions.week(Calendar.getInstance()));
        }
    } else if (id == R.id.getDayView) {
        // Change the WeekView to show one day
        weekView.setNumberOfVisibleDays(1);
    } else if (id == R.id.getWeekView) {
        // Change the WeekView to show seven days
        weekView.setNumberOfVisibleDays(7);
    }
    return super.onOptionsItemSelected(item);
}

@override
protected void onDestroy() {
    picker = null;
    super.onDestroy();
}

package com.example.personalschema.activities;

import android.content.Context;
import android.os.Bundle;
import android.util.Log;
import android.view.View;
import android.view.inputmethod.InputMethodManager;
import android.widget.EditText;
import androidx.annotation.Nullable;
import androidx.appcompat.app.AppCompatActivity;
import androidx.recyclerview.widget.LinearLayoutManager;
import androidx.recyclerview.widget.RecyclerView;
import com.example.personalschema.R;
import com.example.personalschema.adapter.ChatAdapter;
import com.firebase.ui.firestore.FirestoreRecyclerAdapter;
import com.google.firebase.auth.FirebaseAuth;
import com.google.firebase.firestore.FirebaseFirestore;
import com.google.firebase.firestore.Query;
import java.util.Objects;

/**
 * @author Kani & Mishu
 * A class to create a chat function.
 */
public class ChatActivity extends AppCompatActivity {

    private final String TAG = "ChatActivity";
    // Firebase
    private FirebaseAuth fAuth;
    private FirestoreFirestore fStore;
    // RecyclerView and adapter
    private RecyclerView recyclerView;
    private FirestoreRecyclerAdapter<MyChatMessage, ChatAdapter.MessageHolder> adapter;

    @Override
    protected void onCreate(@Nullable Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_chat);
        // Initiate firebase
        fAuth = FirebaseAuth.getInstance();
        fStore = FirestoreFirestore.getInstance();
        // Get a handle to the RecyclerView, and give the RecyclerView a default layout manager.
        recyclerView = findViewById(R.id.messagesRecyclerView);
        LinearLayoutManager linearLayout = new LinearLayoutManager(ChatActivity.this);
        linearLayout.setStackFromEnd(true);
        recyclerView.setLayoutManager(linearLayout);
        // Get the current data saved in Firestore Chat collection as a query
        Query query = fStore.collection(getString(R.string.message_collection))
                .orderBy(getString(R.string.field_message_time));
    }
}
// Create an adapter and supply the data to be displayed
adapter = new ChatAdapter(this, query, fAuth.getUid());
recyclerView.setAdapter(adapter);
}

// The method is called when the floating button is clicked
public void sendMessage(View view) {
    EditText messageText = findViewById(R.id.messageEditText);
    MyChatMessage chatMessage = new MyChatMessage(messageText.getText().toString(),
            getIntent().getStringExtra(getString(R.string.intent_name)),
            fAuth.getUid());
    // Save the data in the model type MyChatMessage to Firestore
    fStore.collection(getString(R.string.message_collection)).add(chatMessage)
            .addOnSuccessListener(documentReference ->
                    Log.d(TAG, "DocumentSnapshot written with ID: " +
                    documentReference.getId()))
            .addOnFailureListener(e -> Log.w(TAG, "Error adding document", e));
}

// Scroll the recyclerView to the last position
recyclerView.scrollToPosition(Objects.requireNonNull(recyclerView.getAdapter()).getItemC
ount() - 1);
// Clear the input and hide the keyboard
    messageText.setText("");
    hideKeyboard(view);
}

// Private function that can be called to hide the keyboard
public void hideKeyboard(View view) {
    try {
        InputMethodManager imm = (InputMethodManager)
                getSystemService(Context.INPUT_METHOD_SERVICE);
        imm.hideSoftInputFromWindow(view.getWindowToken(), 0);
    } catch (Exception ignored) {
    }
}

@Override
public void onStart() {
    super.onStart();
    // When the activity is started, start listening for database changes and notify the adapter
    if (adapter != null)
        adapter.startListening();
}

@Override
public void onStop() {
    super.onStop();
    // When the activity is stopped, stop the adapter from listening for database changes
    if (adapter != null)
adapter.stopListening();

package com.example.personalschema.activities;

import android.os.Bundle;
import android.os.Handler;
import android.os.Looper;
import android.util.Log;
import android.view.View;
import android.widget.TextView;
import androidx.appcompat.app.AppCompatActivity;
import androidx.recyclerview.widget.GridLayoutManager;
import androidx.recyclerview.widget.LinearLayoutManager;
import androidx.recyclerview.widget.RecyclerView;
import com.example.personalschema.interfaces.ApplicantInterface;
import com.example.personalschema.model.EventDocument;
import com.example.personalschema.R;
import com.example.personalschema.adapter.ApplicantAdapter;
import com.example.personalschema.model.MyApplicant;
import com.google.firebase.firestore.DocumentReference;
import com.google.firebase.firestore.DocumentSnapshot;
import com.google.firebase.firestore.FieldValue;
import com.google.firebase.firestore.FirebaseFirestore;
import java.util.LinkedList;
import java.util.List;
import java.util.Objects;

/**
 * @author Kani & Mishu
 * A class for the users who have Admin accounts.
 * The class shows a list of the applicants who have registered for the shift which the user can
 * choose from.
 */
public class ChooseApplicantActivity extends AppCompatActivity {

    private final String TAG = "ChooseApplicantActivity";
    // Adapter
    private ApplicantAdapter canApplicantAdapter;
    private ApplicantAdapter maybeApplicantAdapter;
    private ApplicantAdapter cannotApplicantAdapter;
    private final LinkedList<MyApplicant> canNameList = new LinkedList<>();
    private final LinkedList<MyApplicant> maybeNameList = new LinkedList<>();
    private final LinkedList<MyApplicant> cannotNameList = new LinkedList<>();
    // Firebase
    private FirebaseFirestore fStore;
private String eventId;
private Handler handler;

@Override
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_choose_applicant);
    eventId = getIntent().getStringExtra(getString(R.string.intent_event_id));
    // For work in the background
    handler = new Handler(Looper.getMainLooper());
    // Get the Firestore instance
    fStore = FirebaseFirestore.getInstance();
    // UI
    TextView time = findViewById(R.id.shiftTimeTextView);
    String timeString = getIntent().getStringExtra(getString(R.string.intent_date)) + ", " +
        getIntent().getStringExtra(getString(R.string.intent_time));
    time.setText(timeString);

    // Get a handle to the RecyclerViews
    RecyclerView canRecyclerView = findViewById(R.id.canRecyclerView);
    RecyclerView maybeRecyclerView = findViewById(R.id.maybeRecyclerView);
    RecyclerView cannotRecyclerView = findViewById(R.id.cannotRecyclerView);
    canRecyclerView.setHasFixedSize(true);
    maybeRecyclerView.setHasFixedSize(true);
    cannotRecyclerView.setHasFixedSize(true);

    // Create layout managers and connect it to the RecyclerViews
    LinearLayoutManager canLinearLayout = new GridLayoutManager(this, 3,
        GridLayoutManager.HORIZONTAL, false);
    canRecyclerView.setLayoutManager(canLinearLayout);
    LinearLayoutManager maybeLinearLayout = new GridLayoutManager(this, 3,
        GridLayoutManager.HORIZONTAL, false);
    maybeRecyclerView.setLayoutManager(maybeLinearLayout);
    LinearLayoutManager cannotLinearLayout = new GridLayoutManager(this, 3,
        GridLayoutManager.HORIZONTAL, false);
    cannotRecyclerView.setLayoutManager(cannotLinearLayout);

    // Create adapters, supply the data to be displayed and connect the adapter to the RecyclerViews
    canApplicantAdapter = new ApplicantAdapter(this, canNameList);
    canRecyclerView.setAdapter(canApplicantAdapter);
    maybeApplicantAdapter = new ApplicantAdapter(this, maybeNameList);
    maybeRecyclerView.setAdapter(maybeApplicantAdapter);
    cannotApplicantAdapter = new ApplicantAdapter(this, cannotNameList);
    cannotRecyclerView.setAdapter(cannotApplicantAdapter);

    // Get data from Firestore
    getData();
}

// Get data from Firestore from the collection Events,
private void getData() {
    DocumentReference dr =
    fStore.collection(getString(R.string.event_collection)).document(eventId);
    dr.get().addOnCompleteListener(task -> {
        if (task.isSuccessful()) {
            canNameList.clear();
            DocumentSnapshot document = task.getResult();
            assert document != null;
            if (document.exists() && document.get(getString(R.string.field_shift_can)) != null) {
                List<MyApplicant> myApplicantsList =
                Objects.requireNonNull(document.toObject(EventDocument.class)).shift_can;
                // Add the data to the list connected to the adapter
                canNameList.addAll(myApplicantsList);
            } else {
                Log.i(TAG, "(Can) Document or array does not exist:", task.getException());
            }
            // Notify the adapter about the changes in the list
            canApplicantAdapter.notifyDataSetChanged();
        } else {
            Log.i(TAG, "(Can) Error getting documents:", task.getException());
        }
    });
    dr.get().addOnCompleteListener(task -> {
        if (task.isSuccessful()) {
            maybeNameList.clear();
            DocumentSnapshot document = task.getResult();
            assert document != null;
            if (document.exists() && document.get(getString(R.string.field_shift_maybe)) != null) {
                List<MyApplicant> myApplicantsList =
                Objects.requireNonNull(document.toObject(EventDocument.class)).shift_maybe;
                // Add the data to the list connected to the adapter
                maybeNameList.addAll(myApplicantsList);
            } else {
                Log.i(TAG, "(Maybe) Document or array does not exist:", task.getException());
            }
            // Notify the adapter about the changes in the list
            maybeApplicantAdapter.notifyDataSetChanged();
        } else {
            Log.i(TAG, "(Maybe) Error getting documents:", task.getException());
        }
    });
    dr.get().addOnCompleteListener(task -> {
        if (task.isSuccessful()) {
            cannotNameList.clear();
            DocumentSnapshot document = task.getResult();
            assert document != null;
            if (document.exists() && document.get(getString(R.string.field_shift_cannot)) != null) {
                List<MyApplicant> myApplicantsList =
                Objects.requireNonNull(document.toObject(EventDocument.class)).shift_cannot;
                // Add the data to the list connected to the adapter
                cannotNameList.addAll(myApplicantsList);
            } else {
                Log.i(TAG, "(Cannot) Document or array does not exist:", task.getException());
            }
            // Notify the adapter about the changes in the list
            cannotApplicantAdapter.notifyDataSetChanged();
        } else {
            Log.i(TAG, "(Cannot) Error getting documents:", task.getException());
        }
    });
}
List<MyApplicant> myApplicantsList = Objects.requireNonNull(document.toObject(EventDocument.class)).shift_cannot;
// Add the data to the list connected to the adapter
cannotNameList.addAll(myApplicantsList);
} else {
    Log.i(TAG, "(Cannot) Document or array does not exist:", task.getException());
}
// Notify the adapter about the changes in the list
cannotApplicantAdapter.notifyDataSetChanged();
} else {
    Log.i(TAG, "(Cannot) Error getting documents: ", task.getException());
}
});

// The method is called when the user clicked on the done button
public void selectApplicants(View view) {
    // Update the fields in Firestore with the chosen applicants
    handler.post(() -> {
        for (int i = 0; i < canNameList.size(); i++) {
            if (canNameList.get(i).getSelected()) {
                canNameList.get(i).setSelected(false);
                fStore.collection(getString(R.string.event_collection)).document(eventId)
                    .update(getString(R.string.field_selected_applicants),
                        FieldValue.arrayUnion(canNameList.get(i)));
            }
        }
        for (int i = 0; i < maybeNameList.size(); i++) {
            if (maybeNameList.get(i).getSelected()) {
                maybeNameList.get(i).setSelected(false);
                fStore.collection(getString(R.string.event_collection)).document(eventId)
                    .update(getString(R.string.field_selected_applicants),
                        FieldValue.arrayUnion(maybeNameList.get(i)));
            }
        }
        for (int i = 0; i < cannotNameList.size(); i++) {
            if (cannotNameList.get(i).getSelected()) {
                cannotNameList.get(i).setSelected(false);
                fStore.collection(getString(R.string.event_collection)).document(eventId)
                    .update(getString(R.string.field_selected_applicants),
                        FieldValue.arrayUnion(cannotNameList.get(i)));
            }
        }
        // Update the chosen applicants information in Firestore
        getApplicants(myApplicants -> {
            for (int i = 0; i < myApplicants.size(); i++) {
                // Update the applicants information in Firestore with the eventId
                fStore.collection(getString(R.string.user_collection)).document(myApplicants.get(i).getUserUid())
private void getApplicants(ApplicantInterface myCallback) {
    fStore.collection(getString(R.string.event_collection)).document(eventId).get()
        .addOnCompleteListener(task -> {
            if (task.isSuccessful()) {
                List<MyApplicant> myApplicantsList = new LinkedList<>();
                DocumentSnapshot document = task.getResult();
                assert document != null;
                if (document.exists() &&
                    document.get(getString(R.string.field_selected_applicants)) != null) {
                    myApplicantsList =
                        Objects.requireNonNull(document.toObject(EventDocument.class))
                            .selected_applicants;
                } else {
                    Log.i(TAG, "Document or array does not exist:", task.getException());
                    myCallback.onCallback(myApplicantsList);
                } else {
                    Log.i(TAG, "Error getting documents:", task.getException());
                }
            } else {
            }
        });
    }

package com.example.personalschema.activities;

import android.os.Bundle;
import android.os.Handler;
import android.osLooper;
import android.util.Log;
import android.view.View;
import android.widget.TextView;
import androidx.appcompat.app.AppCompatActivity;
import androidx.recyclerview.widget.GridLayoutManager;
import androidx.recyclerview.widget.LinearLayoutManager;
import androidx.recyclerview.widget.RecyclerView;
import androidx.appcompat.app.AppCompatActivity;
import androidx.recyclerview.widget.GridLayoutManager;
import androidx.recyclerview.widget.LinearLayoutManager;
import androidx.recyclerview.widget.RecyclerView;
import androidx.appcompat.widget.AppCompatActivity;
import androidx.recyclerview.widget.widget.GridLayoutManager;
import androidx.recyclerview.widget.widget.LinearLayoutManager;
import androidx.recyclerview.widget.widget.RecyclerView;
import androidx.appcompat.widget.AppCompatActivity;
import androidx.recyclerview.widget.widget.GridLayoutManager;
import androidx.recyclerview.widget.widget.LinearLayoutManager;
import androidx.recyclerview.widget.widget.RecyclerView;
import com.example.personalschema.R;
import com.example.personalschema.adapter.ApplicantAdapter;
import com.example.personalschema.interfaces.ApplicantInterface;
import com.example.personalschema.model.EventDocument;
import com.example.personalschema.model.MyApplicant;
import com.google.firebase.firestore.DocumentSnapshot;
import com.google.firebase.firestore.FieldValue;
import com.google.firebase.firestore.FirebaseFirestore;

import java.util.LinkedList;
import java.util.List;
import java.util.Objects;

/**
 * @author Kani & Mishu
 * A class for the users who have Admin accounts.
 * The class shows a list of chosen applicants for the shift which the user can delete from.
 */
public class DeleteApplicantActivity extends AppCompatActivity {

    private final String TAG = "DeleteApplicantActivity";
    // Adapter
    private ApplicantAdapter applicantAdapter;
    private final LinkedList<MyApplicant> nameList = new LinkedList<>();
    // Firebase
    private FirebaseFirestore fStore;
    // Intent data
    private String eventId;
    // Handler
    private Handler handler;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_delete_applicant);
        eventId = getIntent().getStringExtra(getString(R.string.intent_event_id));
        // For work in the background
        handler = new Handler(Looper.getMainLooper());
        // Get the Firestore instance
        fStore = Firebase.firestore.getInstance();
        // UI
        TextView time = findViewById(R.id.shiftTimeTextView);
        String timeString = getIntent().getStringExtra(getString(R.string.intent_date)) + ", " +
            getIntent().getStringExtra(getString(R.string.intent_time));
        time.setText(timeString);

        // Get a handle to the RecyclerView
        RecyclerView recyclerView = findViewById(R.id.recyclerView);
        recyclerView.setHasFixedSize(true);
        // Create layout manager and connect it to the RecyclerView
        LinearLayoutManager linearLayout = new GridLayoutManager(this, 8,
            GridLayoutManager.HORIZONTAL, false);
        recyclerView.setLayoutManager(linearLayout);
        // Create an adapter, supply the data to be displayed and connect the adapter to the RecyclerView
        applicantAdapter = new ApplicantAdapter(nameList);
        recyclerView.setAdapter(applicantAdapter);
    }
}
applicantAdapter = new ApplicantAdapter(this, nameList);
recyclerView.setAdapter(applicantAdapter);

// Get data from Firestore
getData();
}

// Get data from Firestore from the collection Events
private void getData() {
  fStore.collection(getString(R.string.event_collection)).document(eventId).get()
    .addOnCompleteListener(task -> {
      if (task.isSuccessful()) {
        nameList.clear();
        DocumentSnapshot document = task.getResult();
        assert document != null;
        if (document.exists() &&
            document.get(getString(R.string.field_selected_applicants)) != null) {
          List<MyApplicant> myApplicantsList = Objects.requireNonNull(document.toObject(EventDocument.class)).selected_applicants;
          // Add the data to the list connected to the adapter
          nameList.addAll(myApplicantsList);
        } else {
          Log.i(TAG, "Document or array does not exist:", task.getException());
        }
        // Notify the adapter about the changes in the list
        applicantAdapter.notifyDataSetChanged();
      } else {
        Log.i(TAG, "Error getting documents: ", task.getException());
      }
    });
}

// The method is called when the user clicked on the done button
public void deleteApplicants(View view) {
  // Get the chosen applicants for the shift
  handler.post(() -> getApplicants(myApplicants ->
    for (int i = 0; i < myApplicants.size(); i++) {
      if (nameList.get(i).getSelected()) {
        // If an applicant is chosen update the event in Firestore and remove the applicant
        fStore.collection(getString(R.string.user_collection)).document(myApplicants.get(i).getUserUid()).
          update(getString(R.string.field_shifts), FieldValue.arrayRemove(eventId))
            .addOnCompleteListener(task -> {
              if (task.isSuccessful()) {
                for (int i1 = 0; i1 < nameList.size(); i1++) {
                  if (nameList.get(i1).getSelected()) {
                    // If the applicant is removed from the event,
                  }
                }
              }
            });
    }
  });
}
// update their information in Firestore
nameList.get(i1).setSelected(false);

fStore.collection(getString(R.string.event_collection)).document(eventId)
    .update(getString(R.string.field_selected_applicants),
    FieldValue.arrayRemove(nameList.get(i1)));
else {
    Log.i(TAG, "Error updating documents: ", task.getException());
}
});
finish();

// Method to get the chosen applicants for an event (shift)
private void getApplicants(ApplicantInterface myCallback) {
    fStore.collection(getString(R.string.event_collection)).document(eventId).get()
        .addOnCompleteListener(task -> {
            if (task.isSuccessful()) {
                List<MyApplicant> myApplicantsList = new LinkedList<>();
                DocumentSnapshot document = task.getResult();
                assert document != null;
                if (document.exists() &&
                    document.get(getString(R.string.field_selected_applicants)) != null) {
                    myApplicantsList = Objects.requireNonNull(document.toObject(EventDocument.class))
                        .selected_applicants;
                } else {
                    Log.i(TAG, "Document or array does not exist: ", task.getException());
                }
                myCallback.onCallback(myApplicantsList);
            } else {
                Log.i(TAG, "Error getting documents: ", task.getException());
            }
        });
}

package com.example.personalschema.activities;

import android.content.Intent;
import android.os.Bundle;
import android.util.Log;
import android.view.View;
import android.widget.EditText;
import androidx.appcompat.app.AppCompatActivity;

import com.example.personalschema.R;
import com.example.personalschema.ui.DisplayMessages;
import com.google.firebase.auth.FirebaseAuth;

/**
 * @author Kani & Mishu
 * A class to sign in a user.
 */
public class LoginActivity extends AppCompatActivity {

    private final String TAG = "LoginActivity";
    // UI
    public EditText email;
    public EditText password;
    // Firebase
    private FirebaseAuth fAuth;

    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_login);
        // Set the text views and the buttons
        email = findViewById(R.id.emailEditText);
        password = findViewById(R.id.passwordEditText);
        // Firebase authentication
        fAuth = FirebaseAuth.getInstance();
    }

    // The method is called when the button resetPassword is clicked to go to the
    // ResetPasswordActivity
    public void startResetPasswordActivity(View view) {
        startActivity(new Intent(LoginActivity.this, ResetPasswordActivity.class));
        finish();
    }

    // The method is called when the button Login is clicked to login the user
    public void login(View view) {
        // Call a function to check if the fields are filled
        if (checkFields()) {
            // Sign in user and continue to the main activity if the log in is successful
            login();
        }
    }

    // The method is called when the button Register is clicked to go to the RegisterActivity
    public void startRegisterActivity(View view) {
        startActivity(new Intent(LoginActivity.this, RegisterActivity.class));
        finish();
    }

// Method to check if the fields are filled
private Boolean checkFields() {
    boolean filled = true;
    if (email.getText().toString().isEmpty()) {
        email.setError(getString(R.string.missing_email));
        filled = false;
    }
    if (password.getText().toString().isEmpty()) {
        password.setError(getString(R.string.missing_password));
        filled = false;
    }
    return filled;
}

// Method to login a user with Firebase given email and password
private void login() {
    fAuth.signInWithEmailAndPassword(email.getText().toString(),
    password.getText().toString())
        .addOnCompleteListener(this, task -> {
            if (task.isSuccessful()) {
                // Sign in success, update UI with the signed-in user's information
                Log.d(TAG, "Login:success");
                startMainActivity();
            } else {
                // If sign in fails, display a message to the user.
                Log.w(TAG, "createUserWithEmail:failure", task.getException());
                DisplayMessages.displayToast(getString(R.string.login_failed),
                getApplicationContext());
            }
        });
}

// A function that can be called to start the main activity
private void startMainActivity() {
    startActivity(new Intent(LoginActivity.this, MainActivity.class));
    finish();
}

package com.example.personalschema.activities;

import android.content.Intent;
import android.os.Bundle;
import android.util.Log;
import android.view.View;
import android.widget.TextView;
import androidx.appcompat.app.AppCompatActivity;

import androidx.appcompat.app.AppCompatActivity;
import com.example.personalschema.R;
import com.example.personalschema.interfaces.UserListInterface;
import com.example.personalschema.ui.DisplayMessages;
import com.example.personalschema.util.MyFunctions;
import com.google.firebase.auth.FirebaseAuth;
import com.google.firebase.auth.FirebaseUser;
import com.google.firebase.firestore.DocumentSnapshot;
import com.google.firebase.firestore.FirebaseFirestore;
import java.util.HashMap;
import java.util.Map;
import java.util.Objects;

/**
 * @author Kani & Mishu
 * The main class of the application, the user can click on the images to start the other
 * activities.
 */
public class MainActivity extends AppCompatActivity {

    private final String TAG = "MainActivity";
    // UI
    private TextView welcomeText;
    // Firebase
    private FirebaseUser fAuthCurrentUser;
    private FirebaseFirestore fStore;
    // Intent data
    private Boolean personnel = true;
    private String name = "";
    private String competencies = "";
    private Intent personnelPlanScheme;
    private Intent chatIntent;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        // Intents
        personnelPlanScheme = new Intent(MainActivity.this, PersonnelCalendarActivity.class);
        chatIntent = new Intent(MainActivity.this, ChatActivity.class);
        // UI
        welcomeText = findViewById(R.id.navigationHeaderTextView);
        // Initiate firebase
        fAuthCurrentUser = FirebaseAuth.getInstance().getCurrentUser();
        fStore = FirebaseFirestore.getInstance();
        // Call the method getData to get the users information
        getData(userInfoList -> {
            name = Objects.requireNonNull(userInfoList.get(getApplication().getString(R.string.field_name))).toString();
// Set the users name to the activity header
welcomeText.setText(getString(R.string.welcome_name_text_view, name));
personnelPlanScheme.putExtra(getString(R.string.intent_name), name);
chatIntent.putExtra(getString(R.string.intent_name), name);

// Send the users competencies with intent
competencies = Objects.requireNonNull(userInfoList.get(getApplication()
    .getString(R.string.field_competencies))).toString();

// Check if the user is personnel or admin
personnel = (Boolean) userInfoList.get(getString(R.string.field_is_personnel));

/**
 * Method to get user data from Firestore using a callback. A callback is used to notify the
 * method calling it once the data from Firestore is available.
 */
private void getData(UserListInterface myCallback) {
    // Get the collection Users from Firestore and get the document with the users Uid
    // to get the users information

    fStore.collection(getString(R.string.user_collection)).document(fAuthCurrentUser.getUid()).get()
        .addOnCompleteListener(task -> {
            if (task.isSuccessful()) {
                Map<String, Object> userInfo = new HashMap<>();
                DocumentSnapshot document = task.getResult();
                assert document != null;
                if (document.exists()) {
                    userInfo.put(getString(R.string.field_name),
                        document.getString(getString(R.string.field_name)));
                    userInfo.put(getString(R.string.field_competencies),
                        document.getString(getString(R.string.field_competencies)));
                    userInfo.put(getString(R.string.field_is_personnel),
                        document.getBoolean(getString(R.string.field_is_personnel)));
                } else {
                    Log.d(TAG, "Document does not exist:", task.getException());
                }
                myCallback.onCallback(userInfo);
            } else {
                Log.i(TAG, "Error getting documents: ", task.getException());
            }
        });
}

// If the user clicks on the application icon the onBoarding screen will be started
public void startOnBoardingActivity(View view) {
    startActivity(new Intent(MainActivity.this, OnBoardingActivity.class));
finish();

// If the user clicks on the Active calendar image the ActiveCalendarActivity will be started
public void startActiveCalendar(View view) {
    startActivity(new Intent(MainActivity.this, ActiveCalendarActivity.class));
}

// If the user clicks on the Plan calendar image, wither the AdminCalendarActivity
// or PersonnelCalendarActivity will be started
public void startPlanCalendarActivity(View view) {
    if (personnel) {
        startActivity(personnelPlanScheme);
    } else {
        startActivity(new Intent(MainActivity.this, AdminCalendarActivity.class));
    }
}

// If the user clicks on the My shifts image the MyShiftsActivity will be started
public void startMyShiftsActivity(View view) {
    if (personnel) {
        startActivity(new Intent(MainActivity.this, MyShiftsActivity.class));
    } else {
        DisplayMessages.displayToast(getString(R.string.future_features_text), MainActivity.this);
    }
}

// If the user clicks on the Chat image the ChatActivity will be started
public void startChatActivity(View view) {
    startActivity(chatIntent);
}

// If the user clicks on the Settings image the SettingsActivity will be started
public void startSettingsActivity(View view) {
    startActivity(new Intent(MainActivity.this, SettingsActivity.class));
}

@Override
protected void onStart() {
    super.onStart();
    // Check if internet is available when the activity starts
    if (!MyFunctions.checkInternet(this)) {
        DisplayMessages.displayDialog(getString(R.string.dialog_internet_title),
               getString(R.string.dialog_internet_message), this);
    }
}
package com.example.personalschema.activities;

import android.os.Bundle;
import android.util.Log;

import androidx.appcompat.app.AppCompatActivity;
import androidx.recyclerview.widget.LinearLayoutManager;
import androidx.recyclerview.widget.RecyclerView;
import com.example.personalschema.R;
import com.example.personalschema.adapter.ShiftsAdapter;
import com.example.personalschema.interfaces.ShiftListInterface;
import com.example.personalschema.model.EventDocument;
import com.example.personalschema.model.MyEvent;
import com.example.personalschema.model.MyEventType;
import com.example.personalschema.model.UserDocument;
import com.google.firebase.auth.FirebaseAuth;
import com.google.firebase.auth.FirebaseUser;
import com.google.firebase.firestore.DocumentSnapshot;
import com.google.firebase.firestore.FirebaseFirestore;
import java.util.ArrayList;
import java.util.Calendar;
import java.util.LinkedList;
import java.util.List;
import java.util.Objects;

/**
 * @author Kani & Mishu
 * A class to show a list with the users upcoming shifts.
 */
public class MyShiftsActivity extends AppCompatActivity {

    private final String TAG = "MyShiftsActivity";
    // Adapter
    private ShiftsAdapter shiftsAdapter;
    private final List<MyEvent> shiftsList = new LinkedList<>();
    // Firebase
    private FirebaseUser fAuthCurrentUser;
    private FirebaseFirestore fStore;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_my_shifts);
        // Get a handle to the RecyclerView, and give the RecyclerView a default layout manager.
        RecyclerView recyclerView = findViewById(R.id.shiftsRecyclerView);
        recyclerView.setLayoutManager(new LinearLayoutManager(this));
        // Create an adapter, supply the data to be displayed and connect the adapter to the
RecyclerView

    shiftsAdapter = new ShiftsAdapter(this, shiftsList);
    recyclerView.setAdapter(shiftsAdapter);

// Initiate firebase
fAuthCurrentUser = FirebaseAuth.getInstance().getCurrentUser();
fStore = Firestore.getInstance();
// Call the method getData to get the users shifts
getData(shiftList -> {
    for (int i = 0; i < shiftList.size(); i++) {
        // Get the information about the each shift
        getEvent(shiftList.get(i));
    }
});

/**
 * Method to get user data from Firebase using a callback. The data that is collected is
 * the users shifts.
 */
private void getData(ShiftListInterface myCallback) {

    fStore.collection(getString(R.string.user_collection)).document(fAuthCurrentUser.getUid()).get()
        .addOnCompleteListener(task -> {
            if (task.isSuccessful()) {
                List<String> userInfo1 = new ArrayList<>();
                DocumentSnapshot document = task.getResult();
                assert document != null;
                if (document.exists()) {
                    if (document.get(getString(R.string.field_shifts)) != null) {
                        userInfo1.addAll(Objects.requireNonNull(document.toObject(UserDocument.class)).shifts);
                    }
                } else {
                    Log.d(TAG, "Document does not exist:", task.getException());
                }
                myCallback.onCallback(userInfo1);
            } else {
                Log.i(TAG, "Error getting documents: ", task.getException());
            }
        });

    // Method to get information about an events from Firestore
    private void getEvent(String eventId) {
        fStore.collection(getString(R.string.event_collection)).document(eventId).get()
            .addOnCompleteListener(task -> {
                if (task.isSuccessful()) {
                    DocumentSnapshot document = task.getResult();
                    assert document != null;
                }
            });
    }
}
if (document.exists()) {
    final MyEventType[] ev = new MyEventType[1];
    ev[0] = new MyEventType(document.getString(document.getString(R.string.field_event_id)),
                              document.getString(document.getString(R.string.field_title)),
                              document.getString(document.getString(R.string.field_info)),
                              Objects.requireNonNull(document.getLong(document.getString(R.string.field_year))).intValue(),
                              Objects.requireNonNull(document.getLong(document.getString(R.string.field_month))).intValue(),
                              Objects.requireNonNull(document.getLong(document.getString(R.string.field_day))).intValue(),
                              Objects.requireNonNull(document.getLong(document.getString(R.string.field_start_hour))).intValue(),
                              Objects.requireNonNull(document.getLong(document.getString(R.string.field_start_min))).intValue(),
                              Objects.requireNonNull(document.getLong(document.getString(R.string.field_end_hour))).intValue(),
                              Objects.requireNonNull(document.getLong(document.getString(R.string.field_end_min))).intValue(),
                              Objects.requireNonNull(document.toObject(EventDocument.class)).selected_applicants);
    if (ev[0].getEndTime().getTimeInMillis() > Calendar.getInstance().getTimeInMillis()) {
        // Add the new data to the list connected to the adapter
        shiftsList.add(new MyEvent(ev[0].getEvent_id(), ev[0].getTitle(),
                                   ev[0].getInfo(), ev[0].getStartTime(), ev[0].getEndTime(),
                                   ev[0].getSelected_applicants()));
        // Notify the adapter about the changes in the list
        shiftsAdapter.notifyDataSetChanged();
    } else {
        Log.d(TAG, "Document does not exist:", task.getException());
    }
} else {
    Log.d(TAG, "Error getting documents: ", task.getException());
}
import android.widget.Button;
import android.widget.LinearLayout;
import android.widget.TextView;

import androidx.annotation.Nullable;
import androidx.appcompat.app.AppCompatActivity;
import androidx.viewpager.widget.ViewPager;

import com.example.personalschema.R;
import com.example.personalschema.adapter.OnBoardingAdapter;
import com.example.personalschema.model.OnBoardingItem;

import java.util.ArrayList;

/**
 * @author Kani
 * Class to create and show an onBoarding screen to give the user information about the application.
 */
public class OnBoardingActivity extends AppCompatActivity {
    private boolean isFirstRunLogin = true;

    private LinearLayout pageIndicator;
    private int dotsCount;
    private Animation animation;

    // UI
    private ViewPager viewPager;
    private Button letsGetStarted;
    private Button skipButton;
    private Button nextButton;

    private final ArrayList<OnBoardingItem> onBoardItems = new ArrayList<>();

    @Override
    protected void onCreate(@Nullable Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_on_boarding);
        // Save boolean data that the onBoarding screen has been shown, because it should only be shown once.
        isFirstRunLogin = getSharedPreferences("LoginFirstTime", MODE_PRIVATE).getBoolean("isFirstRun", true);

        // Set the UI
        viewPager = findViewById(R.id.slider);
        pageIndicator = findViewById(R.id.dots);
        letsGetStarted = findViewById(R.id.getStartedButton);
        skipButton = findViewById(R.id.skipButton);
        nextButton = findViewById(R.id.nextButton);

        // Call method to load data
        loadData();
// Create an adapter to show the views
OnBoardingAdapter onBoardingAdapter = new
OnBoardingAdapter(OnBoardingActivity.this,
onBoardItems);
viewPager.setAdapter(onBoardingAdapter);
viewPager.setCurrentItem(0);
addDots(0);
viewPager.addOnPageChangeListener(new ViewPager.OnPageChangeListener() {
    @Override
    public void onPageScrolled(int position, float positionOffset, int positionOffsetPixels) {
    }
    @Override
    public void onPageSelected(int position) {
        // Customize the pages
        addDots(position);
        dotsCount = position;
        // If the position is at the last screen show the letsGetStarted button
        if (position == 7) {
            animation = AnimationUtils.loadAnimation(OnBoardingActivity.this,
R.anim.slide_up_anim);
            letsGetStarted.setAnimation(animation);
            letsGetStarted.setVisibility(View.VISIBLE);
            skipButton.setVisibility(View.INVISIBLE);
            nextButton.setVisibility(View.INVISIBLE);
        } else {
            letsGetStarted.setVisibility(View.INVISIBLE);
            skipButton.setVisibility(View.VISIBLE);
            nextButton.setVisibility(View.VISIBLE);
        }
    }
    @Override
    public void onPageScrollStateChanged(int state) {
    }
});

letsGetStarted.setOnClickListener(v -> {
    startNextActivity();
    // Save the information that the screen has been shown
    getSharedPreferences("OnBoardingFirstTime", MODE_PRIVATE).edit()
        .putBoolean("isFirstRun", false).apply();
});
skipButton.setOnClickListener(v -> {
    startNextActivity();
    // Save the information that the screen has been shown
    getSharedPreferences("OnBoardingFirstTime", MODE_PRIVATE).edit()
        .putBoolean("isFirstRun", false).apply();
});
// Method to create an intent to get to start a new activity
private void startNextActivity() {
    if (isFirstRunLogin) {
        startActivity(new Intent(OnBoardingActivity.this, RegisterActivity.class));
    } else {
        startActivity(new Intent(OnBoardingActivity.this, MainActivity.class));
    }
    finish();
}

// Method to create the dots on the screen
private void addDots(int position) {
    TextView[] dots = new TextView[8];
    pageIndicator.removeAllViews();

    for (int i = 0; i < dots.length; i++) {
        dots[i] = new TextView(this);
        dots[i].setText(Html.fromHtml("•"));
        dots[i].setTextSize(35);
        pageIndicator.addView(dots[i]);
    }
    dots[position].setTextColor(Color.BLACK);
}

// Method to load the images and text data
private void loadData() {
    // Get the texts and images from the resource file
    String[] title = getResources().getStringArray(R.array.title);
    String[] description = getResources().getStringArray(R.array.description);
    int[] imageId = {R.drawable.main_activity, R.drawable.active_calendar_week,
                     R.drawable.active_calendar_day, R.drawable.plan_admin_new_shift,
                     R.drawable.plan_admin_applicant, R.drawable.plan_personnel_apply,
                     R.drawable.plan_personnel_shifts, R.drawable.chat_activity};

    for (int i = 0; i < imageId.length; i++) {
        OnBoardingItem item = new OnBoardingItem();
        item.setImageID(imageId[i]);
        item.setTitle(title[i]);
        item.setDescription(description[i]);

        onBoardItems.add(item);
    }
}

nextButton.setOnClickListener(v -> viewPager.setCurrentItem(dotsCount + 1));
package com.example.personalschema.activities;

import android.graphics.RectF;
import android.os.Bundle;
import android.util.Log;
import android.view.LayoutInflater;
import android.view.Menu;
import android.view.MenuItem;
import android.view.View;
import android.widget.TextView;
import android.widget.ToggleButton;
import androidx.appcompat.app.AlertDialog;
import androidx.appcompat.app.AppCompatActivity;
import com.alamkanak.weekview.WeekView;
import com.example.personalschema.model.MyApplicant;
import com.example.personalschema.util.MyFunctions;
import com.example.personalschema.interfaces.EventInterface;
import com.example.personalschema.R;
import com.example.personalschema.adapter.WeekViewAdapter;
import com.example.personalschema.model.MyEvent;
import com.example.personalschema.ui.DisplayMessages;
import com.example.personalschema.viewModel.MyViewModel;
import com.google.firebase.auth.FirebaseAuth;
import com.google.firebase.firestore.DocumentReference;
import com.google.firebase.firestore.DocumentSnapshot;
import com.google.firebase.firestore.FieldValue;
import com.google.firebase.firestore.FirebaseFirestore;
import java.util.Calendar;
import java.util.Objects;

/**
 * @author Kani & Mishu
 * A class for the users who have Personnel accounts.
 * The class shows a calendar view and the users can register their interest for available shifts.
 */
public class PersonnelCalendarActivity extends AppCompatActivity implements EventInterface {

    private final String TAG = "PersonnelActivity";
    // Calendar UI and adapter
    private WeekView weekView;
    private WeekView.SimpleAdapter<MyEvent> adapter;
    // Firebase
    private FirebaseFirestore fStore;
    private MyApplicant myApplicant;

@Override
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_calendar);

    // Initiate firebase and get the users Uid to create a new model of the type MyApplicant
    FirebaseAuth fAuth = FirebaseAuth.getInstance();
    String userUid = Objects.requireNonNull(fAuth.getCurrentUser()).getUid();
    MyApplicant app = new MyApplicant(userUid,
            getIntent().getStringExtra(getString(R.string.intent_name)),
            getIntent().getStringExtra(getString(R.string.intent_competencies)), false);

    // Create the WeekView and set the Date and Time formatter
    weekView = findViewById(R.id.weekView);
    MyFunctions.setupDateTimeInterpreter(weekView);
    // Call a new viewModel and adapter to supply data to the WeekView
    MyViewModel myViewModel = new MyViewModel();
    adapter = new WeekViewAdapter(this, this);
    weekView.setAdapter(adapter);
    myViewModel.event.observe(this, myEvents -> adapter.submitList(myEvents));
    myViewModel.fetchEvents();
}

/**
 * When a shift is clicked the user gets a dialog that shows information about the event
 * and the user can register their interest for the shift.
 */
@Override
public void eventClick(MyEvent event, RectF bounds) {

    // Inflate the layout
    LayoutInflater li = LayoutInflater.from(getApplicationContext());
    View view = li.inflate(R.layout.dialog_event_info, null);

    // Create a new AlertDialog and connect it to the layout
    AlertDialog.Builder builder = new AlertDialog.Builder(PersonnelCalendarActivity.this);
    builder.setView(view);
    builder.setTitle(event.getTitle() + "-pass");

    // UI buttons and texts
    final TextView start_date = view.findViewById(R.id.dateTextView);
    final TextView time = view.findViewById(R.id.timeTextView);
    final TextView info = view.findViewById(R.id.infoTextView);
    final ToggleButton canButton = view.findViewById(R.id.canButton);
    final ToggleButton maybeButton = view.findViewById(R.id.maybeButton);
    final ToggleButton cannotButton = view.findViewById(R.id.cannotButton);
    start_date.setText(event.getDate(event.getStartTime()));
    String timeInterval = getString(R.string.time_interval_text_view, event.getTime(event.getStartTime()),
            event.getTime(event.getEndTime()));
    time.setText(timeInterval);
    info.setText(event.getInfo());

    // Get the Firestore instance
    fStore = FirebaseFirestore.getInstance();
// Set setOnCheckedChangeListener for the buttons to check if the other buttons are checked

canButton.setOnCheckedChangeListener((buttonView, isChecked) -> {
    if (isChecked) {
        checkIfDataExists(getString(R.string.field_shift_maybe), event.getId());
        checkIfDataExists(getString(R.string.field_shift_cannot), event.getId());
        maybeButton.setChecked(false);
        cannotButton.setChecked(false);
    }
});

maybeButton.setOnCheckedChangeListener((buttonView, isChecked) -> {
    if (isChecked) {
        checkIfDataExists(getString(R.string.field_shift_can), event.getId());
        checkIfDataExists(getString(R.string.field_shift_cannot), event.getId());
        canButton.setChecked(false);
        cannotButton.setChecked(false);
    }
});

cannotButton.setOnCheckedChangeListener((buttonView, isChecked) -> {
    if (isChecked) {
        checkIfDataExists(getString(R.string.field_shift_can), event.getId());
        checkIfDataExists(getString(R.string.field_shift_maybe), event.getId());
        canButton.setChecked(false);
        maybeButton.setChecked(false);
    }
});

// If the positive button of the dialog is clicked, save the users button choice to Firestore

builder.setPositiveButton(getString(R.string.dialog_positive_per_button_text), (dialog, id) -> {
    if (canButton.isChecked()) {
        fStore.collection(getString(R.string.event_collection)).document(event.getId()).
            update(getString(R.string.field_shift_can), FieldValue.arrayUnion(myApplicant));
    } else if (maybeButton.isChecked()) {
        fStore.collection(getString(R.string.event_collection)).document(event.getId()).
            update(getString(R.string.field_shift_maybe), FieldValue.arrayUnion(myApplicant));
    } else if (cannotButton.isChecked()) {
        fStore.collection(getString(R.string.event_collection)).document(event.getId()).
            update(getString(R.string.field_shift_cannot), FieldValue.arrayUnion(myApplicant));
    }
    DisplayMessages.displayToast(getString(R.string.dialog_positive_per_text), PersonnelCalendarActivity.this);
});

builder.setNegativeButton(getString(R.string.cancel_button), (dialog, which) -> {
    DisplayMessages.displayToast(getString(R.string.dialog_cancel_text),
});
PersonnelCalendarActivity.this); 
builder.create().show();
}

// Method to check if the user has already registered interest to the shift
private void checkIfDataExists(String array_name, String id) {
    DocumentReference df =
fStore.collection(getString(R.string.event_collection)).document(id);
df.get().addOnCompleteListener(task -> {
    if (task.isSuccessful()) {
        DocumentSnapshot document = task.getResult();
        assert document != null;
        if (document.exists() &&
            Objects.requireNonNull(document.getData()).containsKey(array_name)) {
            if (Objects.requireNonNull(document.getData().get(array_name)).toString()
                .contains(myApplicant.getName())) {
                // If the array exists, remove it
                df.update(array_name, FieldValue.arrayRemove(myApplicant));
                Log.i(TAG, "Array field is deleted");
            }
        } else {
            Log.d(TAG, "Document or array does not exist.", task.getException());
        }
    }
}
else {
    Log.i(TAG, "Error getting documents: ", task.getException());
});

@Override
public boolean onCreateOptionsMenu(Menu menu) {
    // Inflate the menu, this adds items to the action bar if it is present.
    getMenuInflater().inflate(R.menu.menu_main, menu);
    return true;
}

@Override
public boolean onOptionsItemSelected(MenuItem item) {
    int id = item.getItemId();

    if (id == R.id.getToday) {
        // Scroll to today's date
        if (weekView.getNumberOfWeeksVisible() == 0) {
            weekView.scrollToDate(Calendar.getInstance());
        } else {
            weekView.scrollToDate(MyFunctions.week(Calendar.getInstance()));
        }
    } else if (id == R.id.getDayView) {
        // Change the WeekView to show one day
        weekView.setNumberOfVisibleDays(1);
    }
else if (id == R.id.getWeekView) {
    // Change the WeekView to show seven days
    weekView.setNumberOfVisibleDays(7);
}
return super.onOptionsItemSelected(item);

package com.example.personalschema.activities;

import android.content.Intent;
import android.os.Bundle;
import android.util.Log;
import android.view.View;
import android.widget.EditText;
import android.widget.RadioButton;
import androidx.appcompat.app.AppCompatActivity;
import com.example.personalschema.R;
import com.example.personalschema.ui.DisplayMessages;
import com.google.firebase.auth.FirebaseAuth;
import com.google.firebase.auth.FirebaseUser;
import com.google.firebase.firestore.DocumentReference;
import com.google.firebase.firestore.FirebaseFirestore;
import java.util.HashMap;
import java.util.Map;

/**
 * @author Kani & Mishu
 * A class to register a new user.
 */
public class RegisterActivity extends AppCompatActivity {

    private final String TAG = "RegisterActivity";
    // UI
    private EditText name;
    private EditText email;
    private EditText password;
    private EditText competencies;
    private RadioButton adminCheckBox;
    private RadioButton personnelCheckBox;
    // Firebase
    private FirebaseAuth fAuth;
    private FirebaseFirestore fStore;

    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_register);
    }
}
// Set the text views and the buttons
name = findViewById(R.id.nameEditText);
email = findViewById(R.id.emailEditText);
password = findViewById(R.id.passwordEditText);
competencies = findViewById(R.id.competenciesEditText);
adminCheckBox = findViewById(R.id.adminRadioButton);
personnelCheckBox = findViewById(R.id.personnelRadioButton);
// Firebase authentication
fAuth = FirebaseAuth.getInstance();
fStore = FirebaseFirestore.getInstance();

// The method is called when the button Register is clicked to register a new user
public void register(View view) {
    // If the text fields are not filled, inform the user
    if (checkFields()) {
        // Create a new account
        createAccount();
    }
}

// The method is called when the button hasAccountButton is clicked to go back to the LoginActivity
public void startLoginActivity(View view) {
    // Save the information the the screen has been shown
    getSharedPreferences("LoginFirstTime", MODE_PRIVATE).edit().putBoolean("isFirstRun", false).apply();
    startActivity(new Intent(RegisterActivity.this, LoginActivity.class));
    finish();
}

// Method to check if the fields are filled
private Boolean checkFields() {
    boolean filled = true;

    if (name.getText().toString().isEmpty()) {
        name.setError(getString(R.string.missing_name));
        filled = false;
    }
    if (email.getText().toString().isEmpty()) {
        email.setError(getString(R.string.missing_email));
        filled = false;
    }
    if (password.getText().toString().isEmpty()) {
        password.setError(getString(R.string.missing_password));
        filled = false;
    }
    if (competencies.getText().toString().isEmpty()) {
        competencies.setError(getString(R.string.missing_competencies));
        filled = false;
    }
    return filled;
}
return filled;
}

private void createAccount() {
    fAuth.createUserWithEmailAndPassword(email.getText().toString(),
    password.getText().toString())
        .addOnCompleteListener(this, task -> {
            if (task.isSuccessful()) {
                // Sign in success, update UI with the signed-in user's information
                Log.d(TAG, "createUserWithEmail:success");
                FirebaseUser user = fAuth.getCurrentUser();

                // Create a collection for the users data
                assert user != null;
                DocumentReference sf = fStore.collection(getString(R.string.user_collection))
                    .document(user.getUid());
                Map<String, Object> userInfo = new HashMap<>();
                userInfo.put(getString(R.string.field_name), name.getText().toString());
                userInfo.put(getString(R.string.field_email), email.getText().toString());
                userInfo.put(getString(R.string.field_competencies),
                    competencies.getText().toString());
                // If the user is admin set isPersonnel as false otherwise true
                if (adminCheckBox.isChecked()) {
                    userInfo.put(getString(R.string.field_is_personnel), false);
                } else if (personnelCheckBox.isChecked()) {
                    userInfo.put(getString(R.string.field_is_personnel), true);
                }
                // Save the user data in Firestore
                sf.set(userInfo)
                    .addOnSuccessListener(aVoid -> {
                        Log.d(TAG, "DocumentSnapshot successfully written!");
                        // If the new account is created, continue to the main activity
                        startMainActivity();
                    })
                    .addOnFailureListener(e -> {
                        Log.w(TAG, "Error writing document", e);
                        DisplayMessages.displayToast(getString(R.string.user_info_failed),
                            this);
                    });
            } else {
                // If sign in fails, display a message to the user.
                Log.w(TAG, "createUserWithEmail:failure", task.getException());
                DisplayMessages.displayToast(getString(R.string.login_failed), this);
            }
        });
}
private void startMainActivity() {
    // Save the information the the screen has been shown
    getSharedPreferences("LoginFirstTime", MODE_PRIVATE).edit().putBoolean("isFirstRun", false).apply();
    startActivity(new Intent(RegisterActivity.this, MainActivity.class));
    finish();
}

package com.example.personalschema.activities;

import android.content.Intent;
import android.os.Bundle;
import android.widget.Button;
import android.widget.EditText;
import androidx.appcompat.app.AppCompatActivity;
import com.example.personalschema.R;
import com.example.personalschema.ui.DisplayMessages;
import com.google.firebase.auth.FirebaseAuth;

/**
 * @author Kani & Mishu
 * A class to reset password to Firebase account.
 */
public class ResetPasswordActivity extends AppCompatActivity {

    // private final String TAG = "ResetPasswordActivity";
    // UI
    private EditText email;
    // Firebase
    private FirebaseAuth fAuth;

    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_reset_password);
        // Set the text views and the buttons
        email = findViewById(R.id.emailEditText);
        Button nextButton = findViewById(R.id.nextButton);
        Button backToLoginButton = findViewById(R.id.toLoginButton);
        // Firebase authentication
        fAuth = FirebaseAuth.getInstance();

        // Set actions for when the buttons are clicked
        nextButton.setOnClickListener(v -> {
            // Check if the email field is filled
            if (!email.getText().toString().isEmpty()) {
                // Reset the password with Firebase
            }
        });
    }
}
resetPassword();
}
else {
    email.setError(getString(R.string.missing_email));
}
);
backToLoginButton.setOnClickListener(v -> {
    // When the button is clicked go back to the LoginActivity
    startActivity(new Intent(ResetPasswordActivity.this, LoginActivity.class));
    finish();
});
}

// The method is called to reset a password given an account email
private void resetPassword() {

    fAuth.sendPasswordResetEmail(email.getText().toString()).addOnCompleteListener(task -> {
        if (task.isSuccessful()) {
            DisplayMessages.displayToast("Ett mejl har skickats till dig for att skapa nytt lösenord",
            ResetPasswordActivity.this);
        } else {
            DisplayMessages.displayToast("Ingen konto är kopplat till e-post adressen",
            ResetPasswordActivity.this);
        }
    });
}

package com.example.personalschema.activities;

import android.content.Intent;
import android.os.Bundle;
import android.view.View;
import androidx.appcompat.app.AlertDialog;
import androidx.appcompat.app.AppCompatActivity;
import com.example.personalschema.R;
import com.example.personalschema.ui.DisplayMessages;
import com.google.firebase.auth.FirebaseAuth;

/**
 * @author Kani & Mishu
 * Settings class to show and create actions for the settings buttons.
 */
public class SettingsActivity extends AppCompatActivity {

    // private final String TAG = "SettingsActivity";

    @Override
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_settings);
}

// If the user clicks on the button Logout the method is activated
public void logout(View view) {

    AlertDialog.Builder builder = new AlertDialog.Builder(SettingsActivity.this);
    builder.setTitle(getString(R.string.logout_title));
    builder.setPositiveButton(getString(R.string.logout_item), (dialog, which) -> {
        FirebaseAuth.getInstance().signOut();
        Intent loginIntent = new Intent(SettingsActivity.this, LoginActivity.class);
        startActivity(loginIntent);
        finish();
        finishAffinity();
    });
    builder.setNegativeButton(getString(R.string.cancel_button), (dialog, which) -> DisplayMessages.displayToast(" ", SettingsActivity.this));
    builder.create().show();
}

package com.example.personalschema.activities;

import android.content.Intent;
import android.os.Bundle;
import android.os.Handler;
import androidx.appcompat.app.AppCompatActivity;
import com.example.personalschema.R;
import com.google.firebase.auth.FirebaseAuth;

/**
 * @author Kani
 * Class to show a splash screen when the application has started.
 */
public class SplashScreenActivity extends AppCompatActivity {

    private static boolean loggedIn = false;
    private boolean isFirstRunOnBoarding;
    private boolean isFirstRunLogin;
    private static boolean splashScreenLoaded = false;

    @Override
public void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);

    // The on boarding screen and the login screen should only show once on start
    isFirstRunOnBoarding = getSharedPreferences("OnBoardingFirstTime", MODE_PRIVATE)
        .getBoolean("isFirstRun", true);
    isFirstRunLogin = getSharedPreferences("LoginFirstTime", MODE_PRIVATE)
        .getBoolean("isFirstRun", true);
    // If the splash screen have not been shown, show it before showing the next activity,
    // otherwise show the next activity
    if (!splashScreenLoaded) {
        setContentView(R.layout.activity_splash);
        int secondsDelayed = 1;
        new Handler().postDelayed(this::startNextActivity, secondsDelayed * 1500);
        splashScreenLoaded = true;
    } else {
        startNextActivity();
    }
}

private void startNextActivity() {
    if (isFirstRunOnBoarding && isFirstRunLogin) {
        startActivity(new Intent(SplashScreenActivity.this, OnBoardingActivity.class));
    } else {
        if (loggedIn) {
            // Go directly to the next activity if the on boarding screen and register screen
            // has been shown before and if the user is already signed in
            startActivity(new Intent(SplashScreenActivity.this, MainActivity.class));
        } else {
            // If the on boarding screen and register screen has been shown before but
            // if the user is not signed in, go to login activity
            startActivity(new Intent(SplashScreenActivity.this, LoginActivity.class));
        }
    }
    overridePendingTransition(R.anim.fade_in_anim, R.anim.fade_out_anim);
    finish();
}

@Override
public void onStart() {
    super.onStart();
    // Check if user is signed in (non-null), if yes, continue to the main activity
    if (FirebaseAuth.getInstance().getCurrentUser() != null) {
        loggedIn = true;
    }
}

@Override
protected void onResume() {

super.onResume();
// Check if user is signed in (non-null), if yes, continue to the main activity
if (FirebaseAuth.getInstance().getCurrentUser() != null) {
    loggedIn = true;
}

@Override
protected void onRestart() {
    super.onRestart();
    // Check if user is signed in (non-null), if yes, continue to the main activity
    if (FirebaseAuth.getInstance().getCurrentUser() != null) {
        loggedIn = true;
    }
}
Adapter-klasser

package com.example.personalschema.adapter;

import android.content.Context;
import android.graphics.Color;
import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;
import android.widget.TextView;
import androidx.cardview.widget.CardView;
import androidx.recyclerview.widget.RecyclerView;
import com.example.personalschema.R;
import com.example.personalschema.model.MyApplicant;
import org.jetbrains.annotations.NotNull;
import java.util.List;

/**
 * @author Kani & Mishu
 * An implementation of RecyclerView.Adapter for applicant items
 */
public class ApplicantAdapter extends RecyclerView.Adapter<ApplicantAdapter.ApplicantViewHolder> {

    private final LayoutInflater mInflater;
    private final List<MyApplicant> applicantList;

    // Constructor
    public ApplicantAdapter(Context context, List<MyApplicant> applicantList) {
        mInflater = LayoutInflater.from(context);
        this.applicantList = applicantList;
    }

    /**
     * Called when RecyclerView needs a new ViewHolder of the given type to represent an item.
     * The new view holder displays the items using onBindViewHolder.
     */
    @NotNull
    @Override
    public ApplicantViewHolder onCreateViewHolder(@NotNull ViewGroup parent, int viewType) {
        // Inflate an item view
        View mItemView = mInflater.inflate(R.layout.applicant_item, parent, false);
        // Create a new viewHolder for the items and put in the items
        return new ApplicantViewHolder(mItemView);
    }
}

private final LayoutInflater mInflater;
private final List<MyApplicant> applicantList;

// Constructor
public ApplicantAdapter(Context context, List<MyApplicant> applicantList) {
    mInflater = LayoutInflater.from(context);
    this.applicantList = applicantList;
}

/**
 * Called when RecyclerView needs a new ViewHolder of the given type to represent an item.
 * The new view holder displays the items using onBindViewHolder.
 */
@NotNull
@Override
public ApplicantViewHolder onCreateViewHolder(@NotNull ViewGroup parent, int viewType) {
    // Inflate an item view
    View mItemView = mInflater.inflate(R.layout.applicant_item, parent, false);
    // Create a new viewHolder for the items and put in the items
    return new ApplicantViewHolder(mItemView);
}
/**
 * Called to display the data at the given position by binding values to the viewHolder.
 */
@Override
public void onBindViewHolder(ApplicantViewHolder holder, int position) {
    final MyApplicant applicant = applicantList.get(position);

    holder.applicantName.setText(applicant.getName());
    holder.applicantCompetencies.setText(applicant.getCompetencies());

    // Set a click listener to save when the applicant is chosen
    holder.applicantItemCardView.setOnClickListener(view -> {
        applicant.setSelected(!applicant.getSelected());
        holder.itemView.setBackgroundColor(applicant.getSelected() ? Color.LTGRAY : Color.WHITE);
    });
}

/**
 * Returns the total number of items in the data set.
 */
@override
public int getItemCount() {
    return applicantList.size();
}

/**
 * ViewHolder class for the applicant items
 */
static class ApplicantViewHolder extends RecyclerView.ViewHolder {
    public final TextView applicantName;
    public final TextView applicantCompetencies;
    public final CardView applicantItemCardView;

    /**
     * A custom view holder to hold the view to display in the RecyclerView,
     * this views the applicant items.
     */
    public ApplicantViewHolder(View itemView) {
        super(itemView);
        applicantName = itemView.findViewById(R.id.applicantNameTextView);
        applicantCompetencies = itemView.findViewById(R.id.applicantCompetenciesTextView);    
        applicantItemCardView = itemView.findViewById(R.id.applicant_item);
    }
}
package com.example.personalschema.adapter;

import android.content.Context;
import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;
import android.widget.TextView;
import androidx.annotation.NonNull;
import androidx.recyclerview.widget.RecyclerView;
import com.example.personalschema.R;
import com.example.personalschema.model.MyChatMessage;
import com.firebase.ui.firestore.FirestoreRecyclerAdapter;
import com.firebase.ui.firestore.FirestoreRecyclerOptions;
import com.google.firebase.firestore.Query;
import java.text.SimpleDateFormat;
import java.util.Locale;

/**
* @author Kani & Mishu
* An implementation of FirestoreRecyclerView.Adapter for chat items
*/
public class ChatAdapter extends FirestoreRecyclerAdapter<MyChatMessage, ChatAdapter.MessageHolder> {

    private final LayoutInflater inflaterIn;
    private final LayoutInflater inflaterOut;
    private final String userId;
    private final int MESSAGE_IN_VIEW_TYPE = 1;
    private final int MESSAGE_OUT_VIEW_TYPE = 2;

    // Constructor
    public ChatAdapter(Context context, Query query, String userId) {
        super(new FirestoreRecyclerOptions.Builder<MyChatMessage>()
                .setQuery(query, MyChatMessage.class)
                .build());

        inflaterIn = LayoutInflater.from(context);
        inflaterOut = LayoutInflater.from(context);
        this.userId = userId;
    }

    /**
     * Called when RecyclerView needs a new ViewHolder of the given type to represent an item.
     * The new view holder displays the items using onBindViewHolder.
     */

}
@NonNull
@Override
public MessageHolder onCreateViewHolder(@NonNull ViewGroup parent, int viewType) {
    // If the view type if of type MESSAGE_IN_VIEW_TYPE then inflate message receive layout,
    // otherwise inflate message send layout
    View view;
    if (viewType == MESSAGE_IN_VIEW_TYPE) {
        view = inflaterIn.inflate(R.layout.message_recieve_item, parent, false);
    } else {
        view = inflaterOut.inflate(R.layout.message_send_item, parent, false);
    }
    // Create a new viewHolder for the items and put in the items
    return new MessageHolder(view);
}

/**
 * Called to display the data at the given position by binding values to the viewHolder.
 */
@Override
protected void onBindViewHolder(@NonNull MessageHolder holder, int position,
@NonNull MyChatMessage model) {
    final TextView mText = holder.messageText;
    final TextView mUsername = holder.messageUsername;
    final TextView mTime = holder.messageTime;

    SimpleDateFormat sdf = new SimpleDateFormat("dd MMM, HH:mm", new Locale("sv"));

    mUsername.setText(model.getMessage_user());
    mText.setText(model.getMessage_text());
    mTime.setText(sdf.format(model.getMessage_time()));
}

/**
 * Returns the total number of items in the data set.
 */
@Override
public int getItemCount() {
    return super.getItemCount();
}

/**
 * Returns the view type, if the message userId equals the current userId return
 * MESSAGE_OUT_VIEW_TYPE otherwise return MESSAGE_IN_VIEW_TYPE
 */
@Override
public int getItemViewType(int position) {
if (getItem(position).getUser_id().equals(userId)) {
    return MESSAGE_OUT_VIEW_TYPE;
}
return MESSAGE_IN_VIEW_TYPE;

/**
 * ViewHolder class for the message items
 */
public static class MessageHolder extends RecyclerView.ViewHolder {
    final TextView messageText;
    final TextView messageUsername;
    final TextView messageTime;

    /**
     * A custom view holder to hold the view to display in the RecyclerView,
     * this views the message items.
     */
    public MessageHolder(View itemView) {
        super(itemView);
        messageText = itemView.findViewById(R.id.messageTextTextView);
        messageUsername = itemView.findViewById(R.id.messageUserTextView);
        messageTime = itemView.findViewById(R.id.messageTimeTextView);
    }
}

package com.example.personalschema.adapter;

import android.content.Context;
import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;
import android.widget.TextView;
import androidx.recyclerview.widget.RecyclerView;
import androidx.recyclerview.widget.widget.RecyclerView;
import com.example.personalschema.R;
import com.example.personalschema.model.MyEvent;
import org.jetbrains.annotations.NotNull;
import java.util.List;

/**
 * @author Kani & Mishu
 * An implementation of RecyclerView.Adapter for shift items
 */
public class ShiftsAdapter extends RecyclerView.Adapter<ShiftsAdapter.ShiftViewHolder> {
private final Context context;
private final LayoutInflater mInflater;
private final List<MyEvent> shiftsList;

// Constructor
public ShiftsAdapter(Context context, List<MyEvent> shiftsList) {
    mInflater = LayoutInflater.from(context);
    this.shiftsList = shiftsList;
    this.context = context;
}

/**
 * Called when RecyclerView needs a new ViewHolder of the given type to represent an item.
 * The new view holder displays the items using onBindViewHolder.
 */
@NotNull
@Override
public ShiftsAdapter.ShiftViewHolder onCreateViewHolder(@NotNull ViewGroup parent, int viewType) {
    // Inflate an item view
    View mItemView = mInflater.inflate(R.layout.shift_item, parent, false);
    // Create a new viewHolder for the items and put in the items
    return new ShiftViewHolder(mItemView);
}

/**
 * Called to display the data at the given position by binding values to the view holder.
 */
@Override
public void onBindViewHolder(ShiftsAdapter.ShiftViewHolder holder, int position) {
    final MyEvent ev = shiftsList.get(position);
    holder.shiftTitle.setText(ev.getTitle());
    holder.shiftDate.setText(ev.getDate(ev.getStartTime()));
    holder.shiftTime.setText(context.getResources().getString(R.string.time_interval_text_view,
        ev.getTime(ev.getStartTime()),
        ev.getTime(ev.getEndTime())));
}

/**
 * Returns the total number of items in the data set.
 */
@Override
public int getItemCount() {
    return shiftsList.size();
}

/**
ViewHolder class for the shift items
*/

static class ShiftViewHolder extends RecyclerView.ViewHolder {
    public final TextView shiftTitle;
    public final TextView shiftDate;
    public final TextView shiftTime;

    /**
     * A custom view holder to hold the view to display in the RecyclerView,
     * this views the shift items.
     */
    public ShiftViewHolder(View itemView) {
        super(itemView);
        shiftTitle = itemView.findViewById(R.id.titleShiftTextView);
        shiftDate = itemView.findViewById(R.id.dateShiftTextView);
        shiftTime = itemView.findViewById(R.id.timeShiftTextView);
    }
}

package com.example.personalschema.adapter;

import android.content.Context;
import android.graphics.Color;
import android.graphics.RectF;

import com.alamkanak.weekview.WeekView;
import com.alamkanak.weekview.WeekViewEntity;
import com.example.personalschema.R;
import com.example.personalschema.interfaces.EventInterface;
import com.example.personalschema.model.MyApplicant;
import com.example.personalschema.model.MyEvent;

import org.jetbrains.annotations.NotNull;

import java.util.List;
import javax.annotation.NonNull;

/**
 * @author Kani & Mishu
 * Implementation of simpleAdapter to submit a single list of elements to WeekView.
 */
public class WeekViewAdapter extends WeekView.SimpleAdapter<MyEvent> {
    private final Context context;
    private EventInterface listener;
    private boolean activeCalendar = false;

    // Constructor
public WeekViewAdapter(Context context) {
    this.context = context;
}

// Constructor
public WeekViewAdapter(Context context, boolean activeCalendar) {
    this.context = context;
    this.activeCalendar = activeCalendar;
}

// Constructor
public WeekViewAdapter(Context context, EventInterface listener) {
    this.context = context;
    this.listener = listener;
}

/**
 * The method returns event data with the type WeekViewEntity which then is send to the
 * WeekView.
 */
@override
@NonNull
public WeekViewEntity onCreateEntity(MyEvent item) {
    int eventColor;
    // Set the color of the event
    if (item.getTitle().equals(context.getString(R.string.ssl))) {
        eventColor = Color.argb(100, 41, 155, 207);
    } else {
        eventColor = Color.argb(100, 127, 239, 131);
    }
    // Set the style of the event
    WeekViewEntity.Style style = new WeekViewEntity.Style.Builder()
        .setTextColor(Color.BLACK)
        .setBackgroundColor(eventColor)
        .setBorderWidth(2)
        .setBorderColor(Color.BLACK)
        .build();

    // Create a new WeekViewEntity.Event depending if the event is shown on the
    // Active calendar or Plan calendar
    WeekViewEntity eventEntity;
    List<MyApplicant> applicants = item.getSelected_applicants();
    StringBuilder sb = new StringBuilder();
    if (activeCalendar && applicants != null) {
        for (int i = 0; i < applicants.size(); i++) {
            sb.append(applicants.get(i).getName());
            sb.append("\n");
        }
        eventEntity = new WeekViewEntity.Event.Builder<>((item)
.setId(Long.parseLong(item.getId()))
.setTitle((item.getTitle() + "-pass"))
.setSubtitle(sb.toString())
.setStartTime(item.getStartTime())
.setEndTime(item.getEndTime())
.setStyle(style)
.build();
} else {
    eventEntity = new WeekViewEntity.Event.Builder<>((item)
        .setId(Long.parseLong(item.getId()))
        .setTitle((item.getTitle() + "-pass"))
        .setSubtitle(item.getInfo())
        .setStartTime(item.getStartTime())
        .setEndTime(item.getEndTime())
        .setStyle(style)
        .build();
    }
    return eventEntity;
}

/**
 * If an event is clicked, the method returns the event data with the model type MyEvent
 * and the position, bounds, of the event
 */
@Override
public void onEventClick(MyEvent data, @NotNull RectF bounds) {
    if (listener != null) {
        (listener).eventClick(data, bounds);
    }
}
package com.example.personalschema.interfaces;

import com.example.personalschema.model.MyApplicant;
import java.util.List;

public interface ApplicantInterface {
    void onCallback(List<MyApplicant> myApplicants);
}

package com.example.personalschema.interfaces;
import android.graphics.RectF;
import com.example.personalschema.model.MyEvent;

public interface EventInterface {
    void eventClick(MyEvent ev, RectF bounds);
}

package com.example.personalschema.interfaces;
import com.example.personalschema.model.MyEvent;
import java.util.List;

public interface EventListInterface {
    void onCallback(List<MyEvent> userInfoList);
}

package com.example.personalschema.interfaces;
import java.util.List;

public interface ShiftListInterface {
    void onCallback(List<String> shiftList);
}

package com.example.personalschema.interfaces;
import java.util.Map;

public interface UserListInterface {
    void onCallback(Map<String, Object> userInfoList);
}
Model-klasser

package com.example.personalschema.model;

import java.util.List;

public class EventDocument {
    public List<MyApplicant> shift_can;
    public List<MyApplicant> shift_maybe;
    public List<MyApplicant> shift_cannot;
    public List<MyApplicant> selected_applicants;
    public String event_id;
    public String title;
    public String info;
    public int year;
    public int month;
    public int day_of_month;
    public int start_hour;
    public int start_min;
    public int end_hour;
    public int end_min;

    public EventDocument() {
    }

    package com.example.personalschema.model;

    /**
     * A model class to get MyApplicant type data
     */
    public class MyApplicant {
        private String userUid;
        private String name;
        private String competencies;
        private Boolean isSelected = false;

        // Constructor
        public MyApplicant() {
        }

        // Constructor
        public MyApplicant(String userUid, String name, String competencies, Boolean isSelected)
        {
            this.userUid = userUid;
            this.name = name;
            this.competencies = competencies;
            this.isSelected = isSelected;
        }

        public String getUserUid() {
            return userUid;
        }
    }
return userUid;
}

public void setUserUid(String userUid) {
    this.userUid = userUid;
}

public String getName() {
    return name;
}

public void setName(String name) {
    this.name = name;
}

public String getCompetencies() {
    return competencies;
}

public void setCompetencies(String competencies) {
    this.competencies = competencies;
}

public Boolean getSelected() {
    return isSelected;
}

public void setSelected(Boolean selected) {
    isSelected = selected;
}

package com.example.personalschema.model;

import java.util.Date;

/**
 * A model class to get MyChatMessage type data
 */
public class MyChatMessage {
    private String message_text;
    private String message_user;
    private String user_id;
    private long message_time;

    // Constructor
    public MyChatMessage() {
    }

    // Constructor
public MyChatMessage(String message_text, String message_user, String user_id) {
    this.message_text = message_text;
    this.message_user = message_user;
    this.user_id = user_id;

    // Initialize to current time
    message_time = new Date().getTime();
}

public String getMessage_user() {
    return message_user;
}

public void setMessage_user(String message_user) {
    this.message_user = message_user;
}

public String getMessage_text() {
    return message_text;
}

public void setMessage_text(String message_text) {
    this.message_text = message_text;
}

public String getUser_id() {
    return user_id;
}

public void setUser_id(String user_id) {
    this.user_id = user_id;
}

public long getMessage_time() {
    return message_time;
}

public void setMessage_time(long message_time) {
    this.message_time = message_time;
}

package com.example.personalschema.model;

import java.text.DateFormat;
import java.text.SimpleDateFormat;
import java.util.Calendar;
import java.util.List;
import java.util.Locale;
/**
 * A model class to get MyEvent type data
 */

public class MyEvent {
    private String id;
    private String title;
    private String info;
    private Calendar startTime;
    private Calendar endTime;
    private List<MyApplicant> selected_applicants;

    // Constructor
    public MyEvent() {
    }

    // Constructor
    public MyEvent(String id, String title, String info, Calendar startDate, Calendar endDate, List<MyApplicant> selected_applicants) {
        this.id = id;
        this.title = title;
        this.info = info;
        this.startTime = startDate;
        this.endTime = endDate;
        this.selected_applicants = selected_applicants;
    }

    public String getId() {
        return id;
    }

    public void setId(String id) {
        this.id = id;
    }

    public String getTitle() {
        return title;
    }

    public void setTitle(String title) {
        this.title = title;
    }

    public String getInfo() {
        return info;
    }

    public void setInfo(String info) {
        this.info = info;
    }
}
public Calendar getStartTime() {
    return startTime;
}

public void setStartTime(Calendar startTime) {
    this.startTime = startTime;
}

public Calendar getEndTime() {
    return endTime;
}

public void setEndTime(Calendar endTime) {
    this.endTime = endTime;
}

public String getDate(Calendar cal) {
    DateFormat weekDay = new SimpleDateFormat("EEEE", new Locale("Sv"));
    DateFormat date = new SimpleDateFormat("dd MMMM", new Locale("Sv"));
    String strWeekDay = weekDay.format(cal.getTime());
    String strDate = date.format(cal.getTime());
    String strWeekDayFirstLetter = String.valueOf(strWeekDay.charAt(0)).toUpperCase() + strWeekDay.substring(1) + strDate;
    return strWeekDayFirstLetter.toUpperCase() + strWeekDay.substring(1) + strDate;
}

public String getTime(Calendar cal) {
    DateFormat dateFormat = new SimpleDateFormat("HH:mm", new Locale("Sv"));
    return dateFormat.format(cal.getTime());
}

public List<MyApplicant> getSelected_applicants() {
    return selected_applicants;
}

public void setSelected_applicants(List<MyApplicant> selected_applicants) {
    this.selected_applicants = selected_applicants;
}

package com.example.personalschema.model;

import java.util.Calendar;
import java.util.List;

/**
 * A model class to get MyEventType type data
 */
public class MyEventType {

}
private String event_id;
private String title;
private String info;
private int year;
private int month;
private int day_of_month;
private int start_hour;
private int start_min;
private int end_hour;
private int end_min;
private List<MyApplicant> selected_applicants;

// Constructor
public MyEventType() {
}

// Constructor
public MyEventType(String event_id, String title, String info, int year, int month, int day_of_month, int start_hour, int start_min, int end_hour, int end_min, List<MyApplicant> selected_applicants) {
    this.event_id = event_id;
    this.title = title;
    this.info = info;
    this.year = year;
    this.month = month;
    this.day_of_month = day_of_month;
    this.start_hour = start_hour;
    this.start_min = start_min;
    this.end_hour = end_hour;
    this.end_min = end_min;
    this.selected_applicants = selected_applicants;
}

public String getEvent_id() {
    return event_id;
}

public String getTitle() {
    return title;
}

public String getInfo() {
    return info;
}

public Calendar getStartTime() {
    Calendar cal = Calendar.getInstance();
    cal.set(year, month, day_of_month, start_hour, start_min, 0);
    return cal;
}
public Calendar getEndTime() {
    Calendar cal = Calendar.getInstance();
    cal.set(year, month, day_of_month, end_hour, end_min, 0);
    return cal;
}

public List<MyApplicant> getSelected_applicants() {
    return selected_applicants;
}

public void setSelected_applicants(List<MyApplicant> selected_applicants) {
    this.selected_applicants = selected_applicants;
}

package com.example.personalschema.model;

/**<*
 * A model class to get OnBoardingItem type data
 */
public class OnBoardingItem {
    int imageID;
    String title;
    String description;

    // Constructor
    public OnBoardingItem() {
    }

    public int getImageID() {
        return imageID;
    }

    public void setImageID(int imageID) {
        this.imageID = imageID;
    }

    public String getTitle() {
        return title;
    }

    public void setTitle(String title) {
        this.title = title;
    }

    public String getDescription() {
        return description;
    }

}
    public void setDescription(String description) {
        this.description = description;
    }
}

package com.example.personalschema.model;

import java.util.List;

public class UserDocument {
    public List<String> shifts;
    public String name;
    public String email;
    public String competencies;
    public boolean is_personnel;

    public UserDocument() {
    }
}
ViewModel-klass

package com.example.personalschema.viewModel;

import android.util.Log;

import androidx.lifecycle.MutableLiveData;
import androidx.lifecycle.ViewModel;

import com.example.personalschema.interfaces.EventListInterface;
import com.example.personalschema.model.EventDocument;
import com.example.personalschema.model.MyEventType;
import com.example.personalschema.model.MyEvent;
import com.google.firebase.firestore.FirebaseFirestore;
import com.google.firebase.firestore.QueryDocumentSnapshot;
import java.util.ArrayList;
import java.util.List;
import java.util.Objects;

/**
 * @author Kani & Mishu
 * Class that gets event data from Firebase to then submit them to an adapter which are used to show the events in the WeekView.
 */
public class MyViewModel extends ViewModel {

    private final String TAG = "BasicViewModel";

    public MutableLiveData<List<MyEvent>> event = new MutableLiveData<>();
    private FirebaseFirestore fStore;

    // Method to get the events from Firestore
    public void fetchEvents() {
        fStore = FirebaseFirestore.getInstance();
        getData(userInfoList -> {
            for (int i = 0; i < userInfoList.size(); i++) {
                if (event == null) {
                    event = new MutableLiveData<>();
                } else {
                    event.setValue(userInfoList);
                }
            }
        });
    }

    /**
     * Method to get event data from Firebase using a callback. A callback is used to notify the method calling it once the data from Firestore is available.
     */
private void getData(EventListInterface myCallback) {
    // Get the collection Events from Firestore to get the saved event data
    fStore.collection("Events").get()
        .addOnCompleteListener(task -> {
            if (task.isSuccessful()) {
                if (!Objects.requireNonNull(task.getResult()).isEmpty()) {
                    // Create an empty list to put the data in
                    List<MyEvent> eventsList = new ArrayList<>();
                    for (QueryDocumentSnapshot document : task.getResult()) {
                        final MyEventType[] ev = new MyEventType[1];
                        // Create a new model of the type MyEventType
                        ev[0] = new MyEventType(document.getString("event_id"),
                                                  document.getString("title"),
                                                  document.getString("info"),
                                                  Objects.requireNonNull(document.getLong("year")).intValue(),
                                                  Objects.requireNonNull(document.getLong("month")).intValue(),
                                                  Objects.requireNonNull(document.getLong("day_of_month")).intValue(),
                                                  Objects.requireNonNull(document.getLong("start_hour")).intValue(),
                                                  Objects.requireNonNull(document.getLong("start_min")).intValue(),
                                                  Objects.requireNonNull(document.getLong("end_hour")).intValue(),
                                                  Objects.requireNonNull(document.getLong("end_min")).intValue(),
                                                  document.toObject(EventDocument.class).selected_applicants);
                        // Save the event data with model type MyEvent
                        eventsList.add(new MyEvent(ev[0].getEvent_id(), ev[0].getTitle(),
                                                   ev[0].getInfo(), ev[0].getStartTime(), ev[0].getEndTime(),
                                                   ev[0].getSelected_applicants()));
                        myCallback.onCallback(eventsList);
                    }
                } else {
                    Log.d(TAG, "empty: ", task.getException());
                }
            } else {
                Log.d(TAG, "Error getting documents: ", task.getException());
            }
        });
}
package com.example.personalschema.util;

import android.content.Context;
import android.net.ConnectivityManager;
import android.net.NetworkInfo;
import com.alamkanak.weekview.WeekView;
import java.text.SimpleDateFormat;
import java.util.Calendar;
import java.util.Locale;
import java.util.Random;

/**
 * @author Kani & Mishu
 * Class with global functions
 */
public class MyFunctions {

/***
 * Called to change calendar view month and time format
 */
public static void setupDateTimeInterpreter(WeekView weekView) {

    // Set the date formatter
    SimpleDateFormat sdf = new SimpleDateFormat("EEE
        \ndd/M", Locale("sv"));
    return sdf.format(weekView.getDateFormatter().format());
}

    // Set the time formatter
    SimpleDateFormat sdf = new SimpleDateFormat("HH",
        Locale("sv"));
    return sdf.format(weekView.getTimeFormatter());
}

/**
 * Called to check internet connection
 */
public static boolean checkInternet(Context context) {
    boolean isInternetAvailable = true;
    ConnectivityManager connMgr = (ConnectivityManager)
            context.getSystemService(Context.CONNECTIVITY_SERVICE);
    NetworkInfo networkInfo = connMgr.getActiveNetworkInfo();

...
if (networkInfo == null || networkInfo.isConnected()) {
    isInternetAvailable = false;
}
return isInternetAvailable;

/**
 * Called to get a new random value
 */
public static String getNewEventId() {
    Random r = new Random();
    return String.valueOf(r.nextInt((10000000 - 1) + 1) + 1);
}

/**
 * Called to change calendar view layout
 */
public static Calendar week(Calendar calendar) {
    calendar.clear();
    calendar.setFirstDayOfWeek(Calendar.MONDAY);
    calendar.setTimeInMillis(Calendar.getInstance().getTimeInMillis());
    // If the day of week is Sunday, first subtract 5 days
    if (calendar.get(Calendar.DAY_OF_WEEK) == Calendar.SUNDAY) {
        calendar.add(Calendar.DATE, -5);
    }
    while (calendar.get(Calendar.DAY_OF_WEEK) > calendar.getFirstDayOfWeek()) {
        calendar.add(Calendar.DATE, -1); // Subtract 1 day until first day of week
    }
    return calendar;
}

package com.example.personalschema.ui;

import android.content.Context;
import android.widget.Toast;
import androidx.appcompat.app.AlertDialog;

public class DisplayMessages {

    /**
     * Can be called to display a message
     */
    public static void displayToast(String message, Context context) {
        Toast.makeText(context, message, Toast.LENGTH_SHORT).show();
    }

    /**
public static void displayDialog(String title, String msg, Context context) {
    AlertDialog.Builder builder = new AlertDialog.Builder(context);
    builder.setTitle(title);
    builder.setMessage(msg);
    builder.setPositiveButton("Ok", (dialog, id) -> {
        // do nothing, just close the alert
    });
    builder.create().show();
}