

KAUNAS UNIVERSITY OF TECHNOLOGY

FACULTY OF INFORMATICS

T120B169 Fundamentals of App Development

Schedule app Study Easy

|  |
| --- |
| IFE-8, Vladas Bukinas  IFE-8, Martynas Kemežys  IFE-8, Augustė Viršilaitė |
| Date: 2021.12.06 |

Kaunas, 2021

**Table of contents**

[Description of the app 3](#_Toc89697062)

[Functionality of the app 4](#_Toc89697063)

[List of functions 4](#_Toc89697064)

[Solution 5](#_Toc89697065)

[Task #1: connect project to a remote database 5](#_Toc89697066)

[Task #2: add a calendar which displays events 5](#_Toc89697067)

[Task #3: create a function for adding one-time events 7](#_Toc89697068)

[Task #4: create a function for adding recurring events 10](#_Toc89697069)

[Task #5: add a function for deleting events in the calendar 12](#_Toc89697070)

[Task #6: implement app navigation through a sidebar menu 13](#_Toc89697071)

[Task #7 / defense task #1: implement the filtering of events by color 14](#_Toc89697072)

[Task #8: add a function for viewing events in the calendar 16](#_Toc89697073)

[Task #9: add authentication functions 17](#_Toc89697074)

[Task #10: add a function for viewing upcoming events 20](#_Toc89697075)

[Task #11: add a function for viewing profile 21](#_Toc89697076)

[Task #12 / defense task #2: add a function for an event search 23](#_Toc89697077)

[Task #13: add a function for registering COVID outbreaks 25](#_Toc89697078)

[Task #14: add a function for registering to events 27](#_Toc89697079)

[Task #15: add a function for displaying notifications 29](#_Toc89697080)

[Task #16: add a function for displaying settings 31](#_Toc89697081)

[Task #17: add a function for changing password 33](#_Toc89697082)

[Defense task #3: add a function for increasing events’ progress 34](#_Toc89697083)

[Defense task #4: add an ability to share via Facebook 35](#_Toc89697084)

[Reference list 36](#_Toc89697085)

# Description of the app

In times like these, when a global pandemic has interfered with everyone’s lives, it is now more useful than ever to have a day-to-day plan. Not only does knowing what to expect in the upcoming days help us keep up with our responsibilities, but it also lets us prepare to safely reenter society.

Places that are more susceptible to COVID-19 outbreaks include educational institutions [1], therefore, they can greatly benefit from organizing the flow of students. Keeping this in mind, our team has decided to create a schedule appcalled **Study Easy**, which mainly focuses on preventing the spread of COVID-19 in universities and schools.

The app has two types of users: lecturers and students. Lecturers are able to create recurring lectures and manipulate them. Created events are displayed in the calendar for students. They can register their attendance, which then becomes visible for the responsible lecturer. In case of a positive COVID-19 case, the lecturer, in whose lecture it took place, is able to notify the attendees and provide them with necessary information. Besides that, lecturers and students see upcoming lectures, which improves the ease of planning.

Lecturers and students can also create single (one-time) events, which are only visible to them, in order to customize their schedules and expand usability of the app.

The university schedule app is hoped to be a useful and easy-to-use tool for safely reopening educational institutions.

# **Functionality of the app**

## **List of functions**

1. connect project to a remote database;
2. add a calendar which displays events;
3. create a function for adding one-time events;
4. create a function for adding recurring events;
5. add a function for deleting events in the calendar;
6. implement app navigation through a sidebar menu;
7. implement the filtering of events by color;
8. add a function for viewing events in the calendar;
9. add authentication functions;
10. add a function for viewing upcoming events;
11. add a function for viewing profile;
12. add a function for an event search;
13. add a function for increasing events’ progress;
14. add a function for registering to events;
15. add a function for displaying notifications;
16. add a function for displaying settings;
17. add a function for changing password.

# Solution

## Task #1: connect project to a remote database

Originally, the project used *Android Room* [2]local database. However, since some data needs to be accessed by different users (e. g., students need to see the events added by lecturers), project was switched to *Firestore* [3] remote database. Code snippets, as seen in **Figure 1**, were added.



Figure 1. Example of code used for authentication of remote database

## Task #2: add a calendar which displays events

Calendar was implemented using the *Android Week View* library [4]. This library provides a component which shows the day of the week and the time of the day for an upcoming week. The component’s UI can be seen in **Figure 2** and parts of code in **Figure 3** and **Figure** **4**.

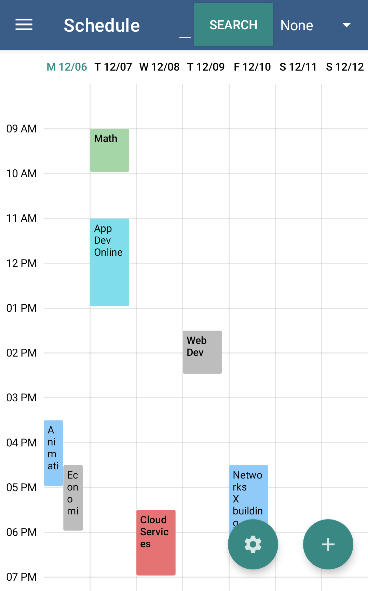


Figure 2. Screenshot of the *Android Week View* component

Paveikslėlis, kuriame yra žinutė

Automatiškai sugeneruotas aprašymas

Figure 3. Code of the *ScheduleFragment*

Paveikslėlis, kuriame yra žinutė

Automatiškai sugeneruotas aprašymas

Figure 4. Code of the *ScheduleAdapter*

## Task #3: create a function for adding one-time events

A function for adding events (lectures) was implemented by creating a local database with the help of the *Android Room* library [2], which simplifies manipulation of data. After that, a form with 6 input fields: event title, date, start time, duration, location and event color, was added. The form gets validated using the *Kotlin Flow* library [5], which asynchronously checks if the entered values are correct. Finally, the form can be accessed by clicking the “+” symbol that is visible on the bottom-right corner of the *Schedule* component (see **Figure 2**). The implemented function’s UI is displayed in **Figure 5** and parts of code in **Figure 6** and **Figure** **7**.

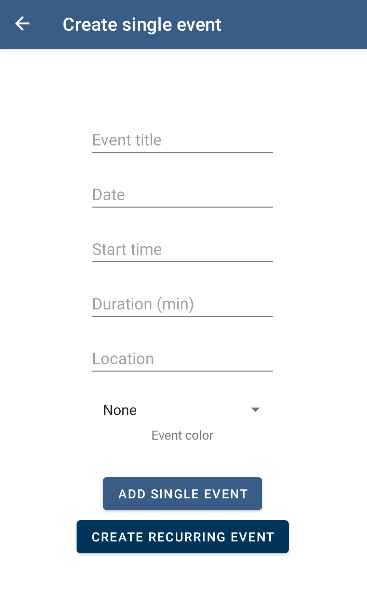


Figure 5. Screenshot of the *CreateEventFragment*



Figure 6. Code of the *CreateEventFragment*

Paveikslėlis, kuriame yra žinutė

Automatiškai sugeneruotas aprašymas

Figure 7. Code of the validation function

## Task #4: create a function for adding recurring events

Function for adding recurring events was implemented with the help of *Firestore* [3], which simplifies manipulation of data. After that, a form with 7 input fields: event title, day of the week, start time, duration, event recurrence, location and event color, was added. The form gets validated using the *Kotlin Flow* library [5], which asynchronously checks if the entered values are correct. Finally, the form can be accessed by clicking the “Create recurring event” button that is visible in the bottom of the *Create Event* component (see **Figure 4**). The implemented function’s UI is displayed in **Figure 8** and some of its code in **Figure 9**.

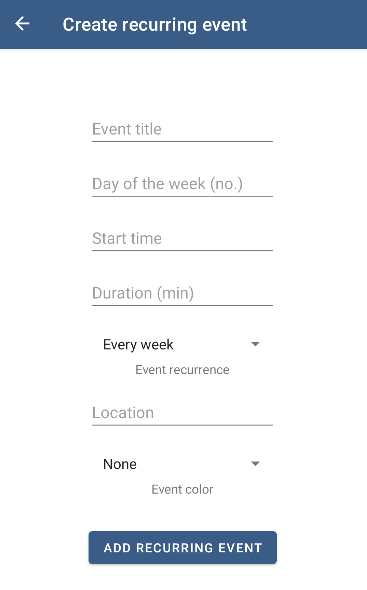


Figure 8. Screenshot of the *MassAddEvents* component



Figure 9. Code snippet of the *MassAddEvents*

## Task #5: add a function for deleting events in the calendar

A function for deleting events was implemented by using the previously mentioned *Android Room* [2]library’s database and the *Android Week View* [4]library, which provides a callback *onEventLongClick()*. It was used to trigger the delete function in the *ScheduleViewModel* class. The implemented function’s UI is displayed in **Figure 10** and the code in **Figure 11**.

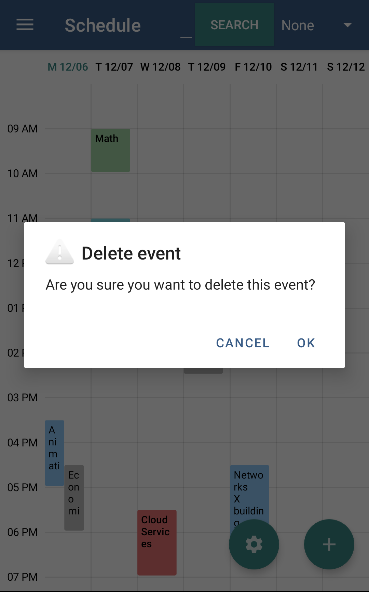


Figure 10. Screenshot of removing events

Paveikslėlis, kuriame yra žinutė, asmuo, ekrano nuotrauka

Automatiškai sugeneruotas aprašymas

Figure 11. Code of the function for removing events

## Task #6: implement app navigation through a sidebar menu

App navigation was implemented by using the *Android NavigationUI* library [7]. The navigation is done mostly through the *Drawer*, which is opened by sliding to the right or clicking on the “hamburger” icon. The implemented function’s UI is displayed in **Figure 12** and the code in **Figure 13.**

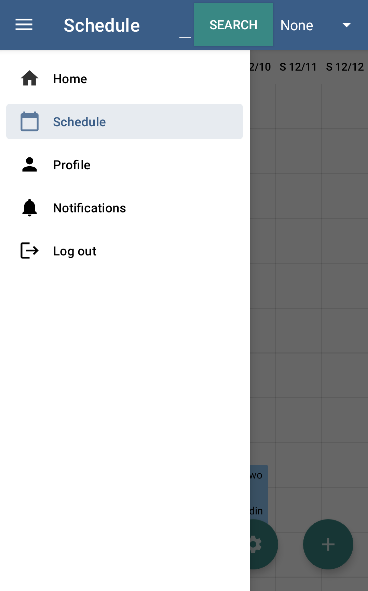


Figure 12. Screenshot of the sidebar menu

Paveikslėlis, kuriame yra žinutė

Automatiškai sugeneruotas aprašymas

Figure 13. Code of the app navigation

## Task #7 / defense task #1: implement the filtering of events by color

A function for filtering events (lectures) by color was implemented by a query, which selects events with a chosen color from the database. The implemented function’s UI is displayed in **Figure 14** and the code in **Figure 15**, **Figure 16** and **Figure 17**.

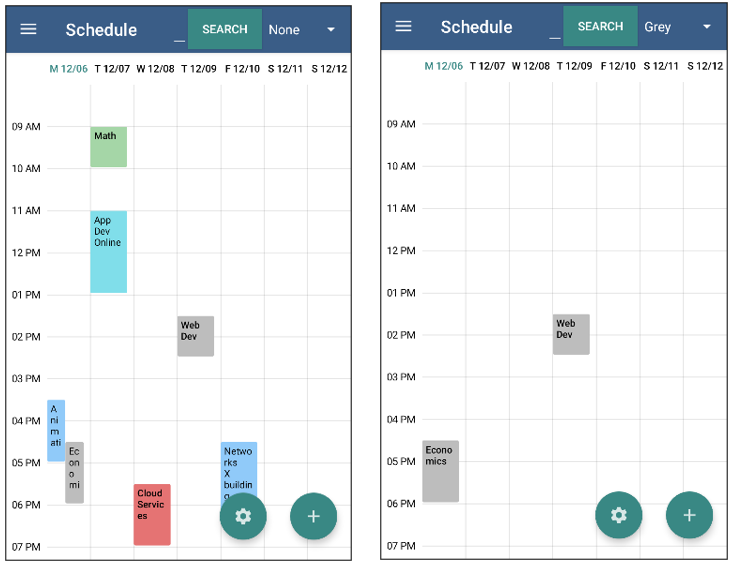


Figure 14. Screenshot of filtering events by color

Paveikslėlis, kuriame yra žinutė

Automatiškai sugeneruotas aprašymas

Figure 15. Code of the filtering of the events



Figure 16. Code of the query for filtering

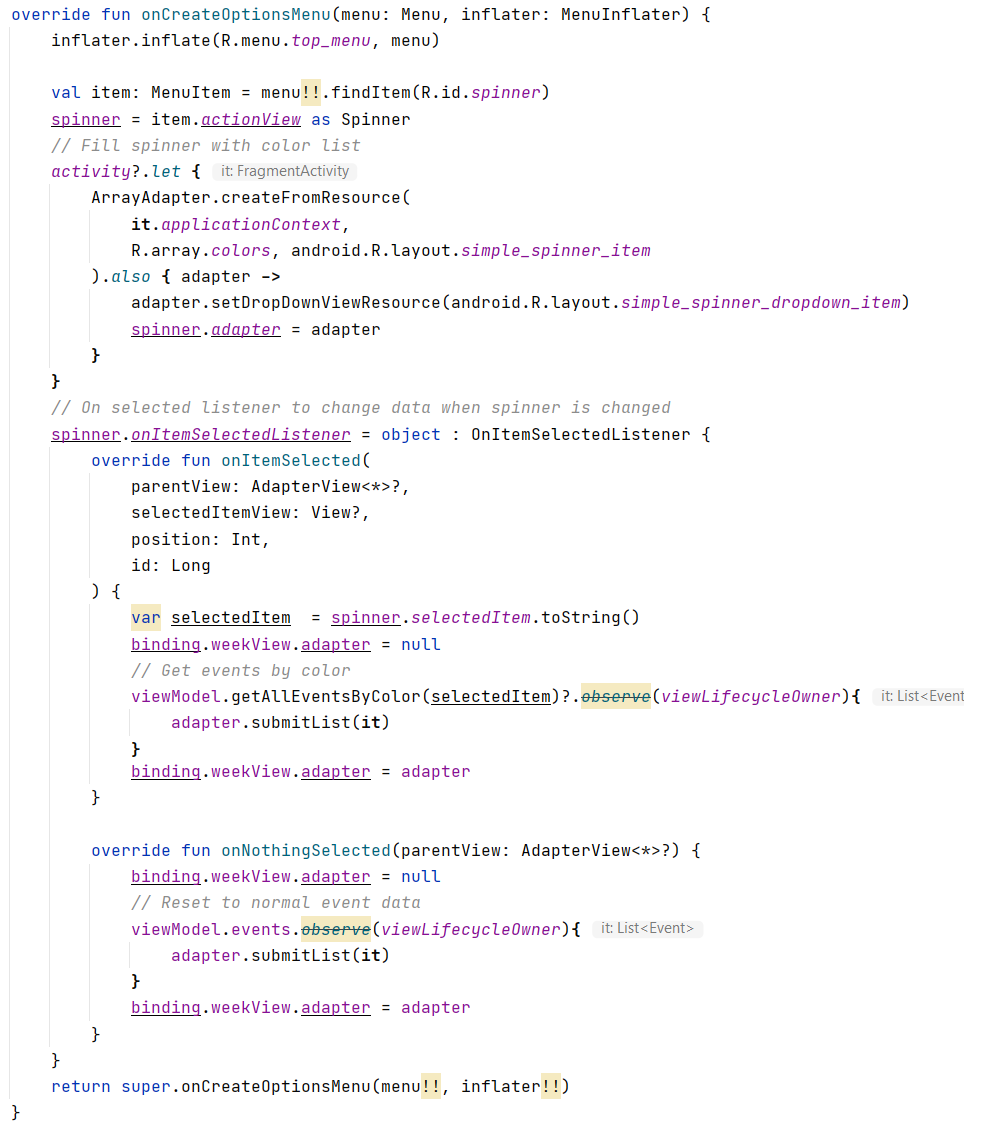


Figure 17. Code of the *ScheduleFragment* part for event filtering

## Task #8: add a function for viewing events in the calendar

A function for viewing events in the calendar was implemented by adding *EventFragment* and setting data to display. The implemented function’s UI is displayed in **Figure 18** and the code snippet in **Figure 19**.

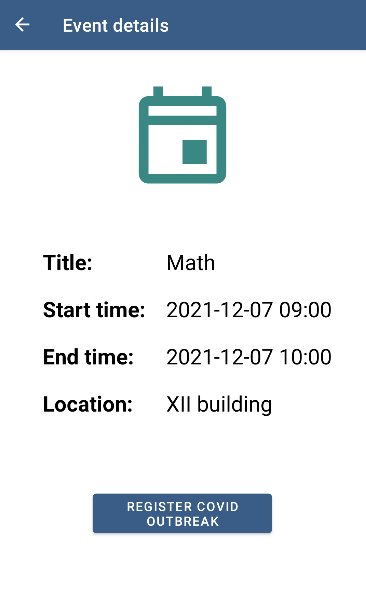


Figure 18. Screenshot of viewing an event

Paveikslėlis, kuriame yra žinutė

Automatiškai sugeneruotas aprašymas

Figure 19. Code of *EventFragment*

## Task #9: add authentication functions

Authentication functions, such as *sign up*, *log in* and *log out*, were implemented by using *Firebase* authentication system [7]. The implemented functions’ UI is displayed in figures **Figure** **20** and **Figure** **21,** and code snippets infigures **Figure** **22** and **Figure** **23**.

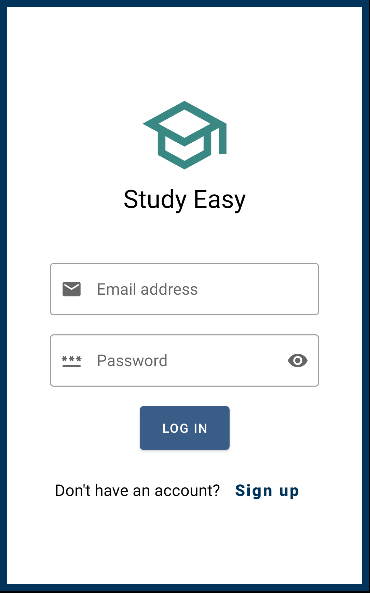


Figure 20. Screenshot of *LoginFragment*

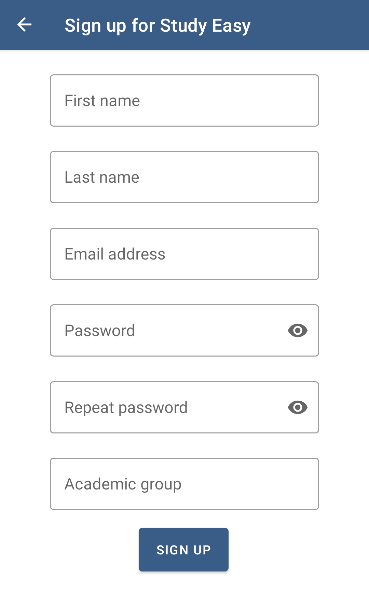


Figure 21. Screenshot of *RegisterFragment*



Figure 22. Code of *LoginFragment*



Figure 23. Code of *RegisterFragment*

## Task #10: add a function for viewing upcoming events

A function for viewing upcoming events was created by using an *Adapter* component and querying data from the *Firestore* [3] database. The upcoming event function is placed in the *HomeFragment* (see **Figure 24**). The implemented function’s code snippet is displayed in **Figure 25**.

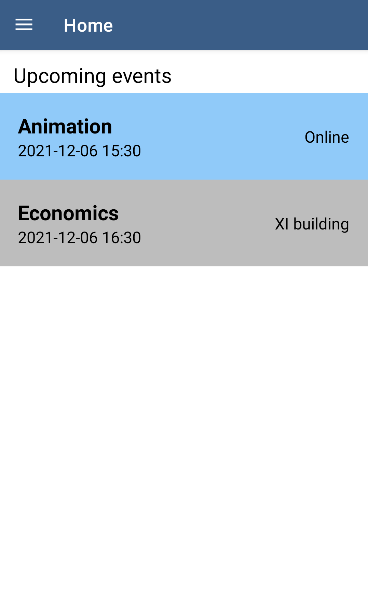


Figure 24. Screenshot of *HomeFragment*

Paveikslėlis, kuriame yra žinutė

Automatiškai sugeneruotas aprašymas

Figure 25. Code of *HomeFragment*

## Task #11: add a function for viewing profile

A function for viewing user’s profile page was implemented by querying data from *Firestore* [3]database. The profile page can be accessed through the sidebar menu (see **Figure 12**). The implemented function’s UI is displayed in **Figure 26** and the code snippet in **Figure 27**.

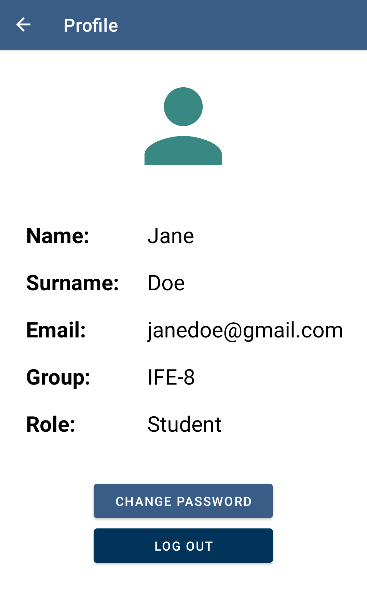


Figure 26. Screenshot of *ProfileFragment*



Figure 27. Code of *ProfileFragment*

## Task #12 / defense task #2: add a function for an event search

A function for an event search was implemented by querying data from *Firestore* [3]database and displaying events with titles that match looked up keywords. The search tab can be accessed above the calendar and besides filtering by color. The implemented function’s UI is displayed in **Figure 28** and the code snippet in **Figure 29**.

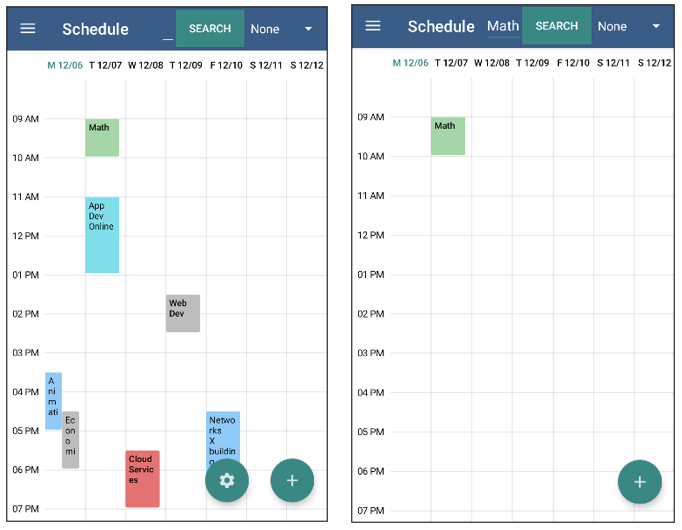


Figure 28. Screenshot of searching for events by title

Paveikslėlis, kuriame yra žinutė

Automatiškai sugeneruotas aprašymas

Figure 29. Code of *ScheduleModeView* part for an event search

## Task #13: add a function for registering COVID outbreaks

A function for registering a new COVID case was implemented by adding a button to each event details’ page and making it visible only for lecturers. The case is then shown as a notification for students. The implemented function’s UI is displayed in **Figure 30** and the code snippet in **Figure 31**.

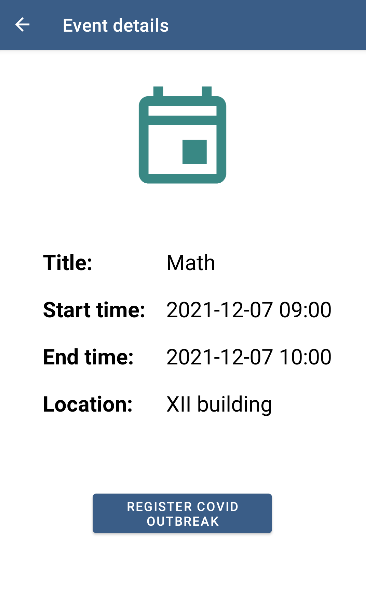


Figure 30. Screenshot of lecturer’s event details‘ page

Paveikslėlis, kuriame yra žinutė

Automatiškai sugeneruotas aprašymas

Figure 31. Code for alerting students about a COVID case

## Task #14: add a function for registering to events

A function for registering to an event (lecture) was implemented by adding a button to each event details’ page and making it visible only for students. The attendance is then saved in the *Firestore* [3]database. The implemented function’s UI is displayed in **Figure 32** and the code snippet in **Figure 33**.

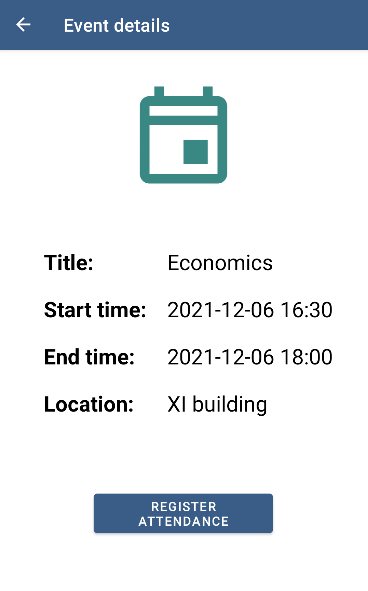


Figure 32. Screenshot of student’s event details‘ page



Figure 33. Code for registering event attendance

## Task #15: add a function for displaying notifications

A function for displaying notifications was implemented by creating a notification channel and a function for setting up notification’s attributes. By communicating with the *Firestore* [3]database, the system sends both students and lecturers notifications about upcoming events that day. Additionally, students receive notifications about registered COVID cases in the events they have attended, and those notifications are separately displayed in a new window accessed through the sidebar menu. The implemented function’s UI is displayed in **Figure 34**, **Figure 35** and the code snippet in **Figure 36**.

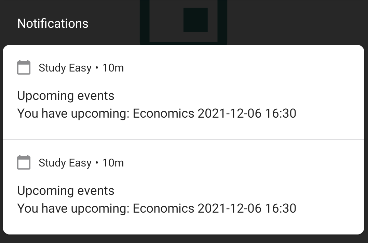


Figure 34. Screenshot of upcoming events‘ notifications

Paveikslėlis, kuriame yra žinutė

Automatiškai sugeneruotas aprašymas

Figure 35. Screenshot of COVID outbreaks‘ notifications



Figure 36. Code for creating notifications

## Task #16: add a function for displaying settings

A function for displaying settings was implemented by creating a new floating button that leads to a new window for lecturers only. Lecturers are able to set the beginning and the end dates of the current semester. This determines the limit of when events can take place and how far users can scroll horizontally in the calendar. The implemented function’s UI is displayed in **Figure 37** and the code in **Figure 38**.

Paveikslėlis, kuriame yra žinutė

Automatiškai sugeneruotas aprašymas

Figure 37. Screenshot of the Settings window



Figure 38. Code of *SettingsFragment*

## Task #17: add a function for changing password

A function for changing password was implemented by creating a new button in the user’s profile that leads to another window. There, by communicating with the *Firestore* [3]database, the system allows both lecturers and students to change their password by successfully entering the current one and the new one twice. The implemented function’s UI is displayed in **Figure 39** and the code snippet in **Figure 40**.

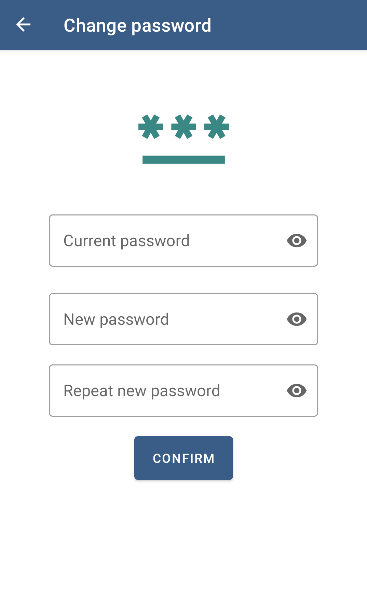


Figure 39. Screenshot of *ChangePwFragment*

Paveikslėlis, kuriame yra žinutė

Automatiškai sugeneruotas aprašymas

Figure 40. Code of *ChangePwFragment*

## Defense task #3: add a function for increasing events’ progress

A function for increasing events’ progress was implemented by adding a *drawable* component with a separate layout inside of it and increasing its height with a click of a button. The implemented function’s UI is displayed in **Figure 41** and the code snippet in **Figure 42**.

Paveikslėlis, kuriame yra stalas

Automatiškai sugeneruotas aprašymas

Figure 41. Screenshot of increasing event‘s progress

Paveikslėlis, kuriame yra žinutė

Automatiškai sugeneruotas aprašymas

Figure 42. Code snippet of increasing event‘s progress

## Defense task #4: add an ability to share via Facebook

A function for sharing notifications about a COVID case was implemented by adding a button which opens the app sharing overlay. That way you can share the alert not only via Facebook, but also via any app you choose. The implemented function’s UI is displayed in **Figure 43** and the code snippet in **Figure 44**.

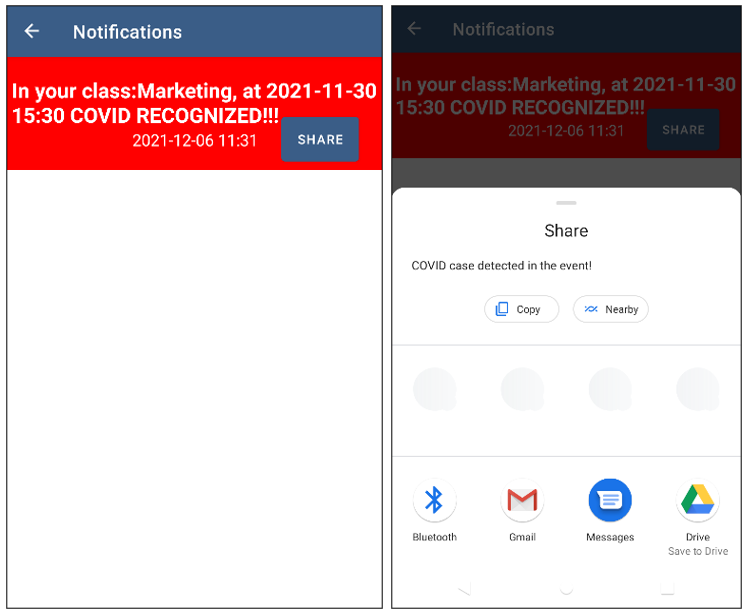


Figure 43. Screenshot of sharing notifications

Paveikslėlis, kuriame yra žinutė

Automatiškai sugeneruotas aprašymas

Figure 44. Code for sharing notifications

# Reference list

|  |  |
| --- | --- |
| [1] | I. D. S. o. America, „What the Experts Say About COVID-19 Risks,“ [Online]. Available: https://www.idsociety.org/globalassets/idsa/public-health/covid-19/activity-risk.pdf. |
| [2] | „Room,“ [Online]. Available: https://developer.android.com/jetpack/androidx/releases/room. |
| [3] | „Authenticate with Firebase on Android Using a Custom Authentication System,“ [Online]. Available: https://firebase.google.com/docs/auth/android/custom-auth. |
| [4] | T. Hellmund, „Android-Week-View,“ [Online]. Available: https://github.com/thellmund/Android-Week-View. |
| [5] | „Asynchronous Flow,“ [Online]. Available: https://kotlinlang.org/docs/flow.html. |
| [6] | „Navigation,“ [Online]. Available: https://developer.android.com/guide/navigation. |
| [7] | „Cloud Firestore,“ [Online]. Available: https://firebase.google.com/docs/firestore. |