Requirements and Analysis Document for StudyBuddy

Bashar Oumari, Benjamin Sannholm, Mathias Drage,
Pontus Nellgård, Sophia Pham

2020-10-21
version 2

1 Introduction

The project is a mobile application developed for Android smartphones using Android Studio. The main purpose of the app is to allow students to locate and interact with other students registered to the same courses and universities who want to study together or are looking for new study partners. The idea is loosely based on a Snapchat extension app called “Beer Buddy” in which people can see other friends and what they are currently drinking and request to join them.

The application provides a map where students can broadcast that they are currently studying at their current location. The map also allows the user to see other students on the map who are broadcasting their study session.

1.1 Definitions, acronyms, and abbreviations

Term	Definition
Activity	A user interface “scene” in which the current session is logged and the user can interact with different Android UI components.
Android Studio [1]	The framework used for developing the GUI, logic, and running the Android phone emulator.
Broadcast	The team’s technical term for the users to display that they are actively seeking a study partner. It holds information about the course, description, and the location of the study session (Broadcast). On the map, they are seen as pins.
Firebase [2]	A Google-developed “backend as a service” that works well with the Google Maps location integration and account setup.
Fragment	Similar to an Activity (scene) in which the user can interact with and navigate between different Android components (UI).
Jetpack Navigation [3]	A framework for Android development to make it easier to

	navigate between different UI screens.
GPS	A self-developed class to handle geolocation and requesting the user's device (and the user) to use geolocation activities (longitude and latitude).
Pin (see also "dot on the map")	A red Google Maps icon that displays where a specific broadcast is located.
Google Maps [4]	A framework provided by Google to use their Maps API in Android applications
MVC	Model View Controller, a design pattern
OOP	Object-Oriented Programming, a design pattern
GUI	Graphical User Interface
JUnit [5]	A framework used for testing
Espresso test recorder [6]	An android framework used for testing android UI interactions.
Gradle [7]	A framework for managing dependencies
Git [8]	A version control manager
GitHub [9]	A platform for managing projects using Git
AVD	Android Virtual Device, an android emulator mocking an android smartphone.
JaCoCo [10]	Java Code Coverage, a tool for Java which generates a code coverage report.
STAN [11]	A tool for creating dependency graphs.

2 Requirements

2.1 User Stories

2.1.1 Completed

Story Identifier: 4
Story Name: See Map

Description
As a user, I want to view the map so that I can see where I am.

Confirmation

Functional

- Can the user see the map?
- The user should be able to see a dot of where he/she is on a map.
- When the map initially shows, is the map centred on the user's location?
- When the user swipes on the screen the map pans accordingly

---

Story Identifier: 2
Story Name: Create broadcast study session

Description
As a user I want to broadcast that I'm looking for a study partner so that others can join me.

Confirmation

Functional

- Can the user add a broadcast?
- Does the location of the user issuing the broadcast get saved?
- Does the course the user entered get saved?
- Does the description the user entered get saved?

Story Identifier: 14
Story Name: See broadcasts on map

Description
As a user, I want to see where related broadcasts are located in relation to myself so that I can choose to join them.

Confirmation

Functional

- Can the user see active broadcasts with a dot on the map?
- Can the user press the broadcast to show the broadcast's course?

Story Identifier: 3
Story Name: Select Course to filter on

Description
As a user, I want to select a course to filter the map on so that I can see relevant broadcasts.

Confirmation

Functional

- Users can only see broadcasts related to the same courses they have selected.
- The user can select what courses should be visible using a UI.

Story Identifier: 6
Story Name: Edit existing broadcast

Description
As a user, I want to edit my broadcast so that I can change the subject studied, the description, and the timer.

Confirmation

Functional

- The broadcast should be updated for all users seeing it.
- There should be an edit button on the description info window.
- It should be possible to edit the description box.

Non-functional

- Only the creator of the broadcast should be able to edit the broadcast.

Story Identifier: 9
Story Name: Terminate broadcast if the user leaves the area

Description
As a user I want the broadcast to self-terminate if I walk too far from the designated area.

Confirmation

Functional

- If the user that made the broadcast goes too far from the designated area the broadcast is terminated.

Non-functional

- Once the broadcast has been terminated it will not show up again if the area is entered.

- Only the user who created the broadcast affects its termination.

Story Identifier: 12
Story Name: Add account

Description
As a user, I need to register an account so that I can interact with the application.

Confirmation

Functional

- Username is unique
- Firstname and surname are filled in.
- The user is registered
- Sign-in and register nice looking prototypes
- Register and sign-in with Google
- Register and sign-in with Facebook
- Register and sign-in with email and password
- Authentication via database
- The login screen is shown if the user is not logged in
- When a user successfully logs in the home screen is shown
- If the user is already logged in when the app starts, the map screen is displayed immediately

---

Story Identifier: 15
Story Name: Delete broadcast study session

Description
As a user, I want to be able to delete a broadcast announcement so that there would not come too many study buddies or if I regret making the announcement.

Confirmation

Functional

- There is a delete button in the broadcast info window.
- The broadcast is deactivated in the database when the button is pressed.
- The broadcast instantly disappears from the map.

Non-functional

- Only the owner of the broadcast can delete it.

2.1.2 Backlog

Story Identifier: 10
Story Name: Add timer to broadcast

Description
As a user I want to add a timer to the broadcast so that other users can see how long the study session will last.

Confirmation

Functional

- Other users should see the broadcast during the time specified.

Non-functional

- The broadcast should be terminated when the timer runs out.

Story Identifier: 1
Story Name: See other users on Map

Description
As a user I want to view where other users are in relation to myself.

Confirmation

Functional

- The user should see other users with a dot and nametag on the map.

Story Identifier: 5
Story Name: Subscribe to Courses

Description
As a user I want to subscribe to the relevant courses so that I can find other users taking those courses.

Confirmation

Functional

- The user should be able to add courses to a list of their current courses.

Story Identifier: 7
Story Name: Send friend request

Description
As a user I want to send a friend request to other users so that I can more easily study with them in the future.

Confirmation

Functional

- The user should be able to send a friend request.

- The user the friend request was sent to should be able to accept/deny the request.
- If accepted the user should be added to the contacts on both respective apps.

Story Identifier: 8
Story Name: Manage Friends

Description
As a user, I want to be able to manage the friends list so that I can manage who I want to interact with

Confirmation

Functional

- Add user
- Delete user
- The user is registered

Story Identifier: 13
Story Name: Filter users on Map

Description
As a user, I want to filter the users I see so that the map doesn't get clotted.

Confirmation

Functional

- The user should be able to filter the users seen on the map (related courses, university etc. more filters to be added)

2.2 Definition of Done

Our scrum board has six columns: Backlog, To do, In progress, Testing, Review, and Done. Our definition of done specifies what criteria have to be fulfilled for a user story to be moved from one column to the next column.

In progress → Testing

- The code should appear to work according to user story requirements.
- No unnecessary dependencies should be present.
- Any complicated parts of the code should be documented through comments.
- All preconditions of methods should be documented through comments above the method.
- The code infrastructure should strive to follow MVC and other OOP design principles and patterns where appropriate.
- The domain model and design model should be updated as well as possible to reflect the newly implemented code.

Testing → Review

- No known bugs should be present.
- The most important methods in the public interface of each module should be tested well.
- All public methods that call other methods should be tested.

Review → Done

- The code should work according to user story requirements.
- All tests should be passing.
- The code should be easily comprehensible.
- The code should be well documented through comments where appropriate.
- No unnecessary dependencies should be present.
- The code infrastructure should strive to follow MVC and other OOP design principles and patterns where appropriate.
- The domain model and design model should be updated to reflect the newly implemented code.

2.3 User interface

In this section, we will go through the different increments of the user interface of the application development. At the end is a detailed description of the user interaction between each of the main interfaces and how a normal interaction with the application might transpire. But first, a brief overview and an introduction of the application.

In the final version, there are three primary screens the user can interact with: the “Login & registration screen”, the “Map screen”, and “Create a broadcast” screen (described in greater detail below). Initially, the app starts at the login screen. If the user does not have an account, they can navigate to the registration screen. After logging in the user is directed to the map screen, which is the main screen of the app. While on the map, if the user decides to broadcast their study session, they can press the Broadcast button and the “Create broadcast screen” is opened from which they can select a course and set a description for the study session. When the broadcast is added it will be added to the Firebase database and be accessed from the map as the previously displayed broadcasts. The broadcast information can be edited by selecting the broadcast and pressing the edit button (not its location, however, as that would call for a new study session).

2.3.1 Development iterations

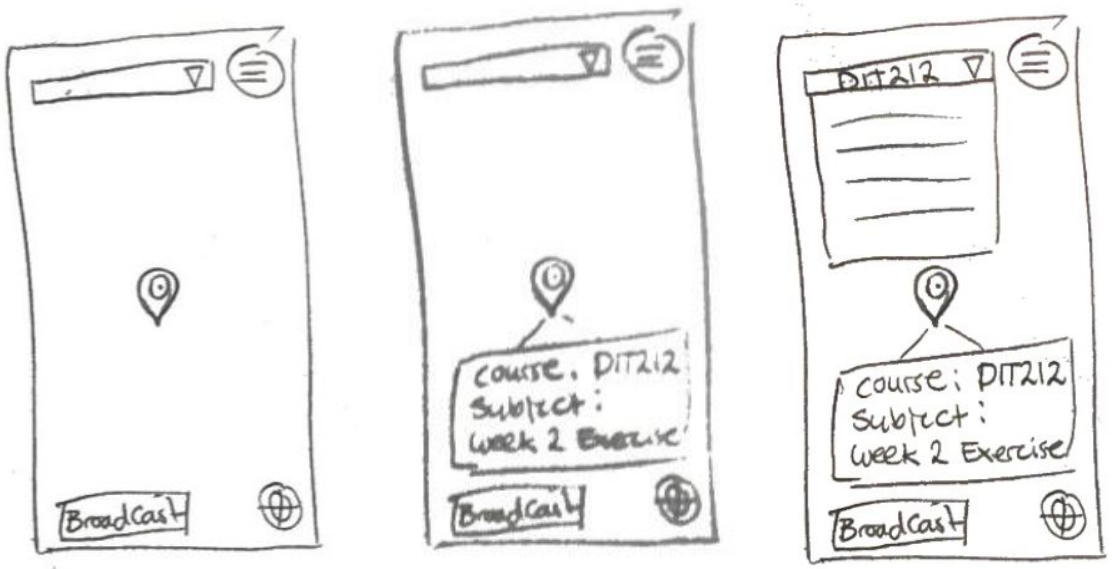

Figure 1. *Early sketches of the map screen GUI.*

The interface shown in Figure 1 was in the early stages of development. The highest priority functionality was to see broadcasts on a map and their description, the ability to add a broadcast, and the ability to filter the courses (pins) on the map based on a course code. The idea was to have a “Hamburger-icon”, or “Collapsed menu icon” button for potential additional settings but the idea was later scrapped for lack of features actually requiring additional separate interaction space. The “get current location” button was moved from the bottom right to the top right as it was included in the Google Maps API and there was no real motivation anymore to move it since the “hamburger” menu was not needed.

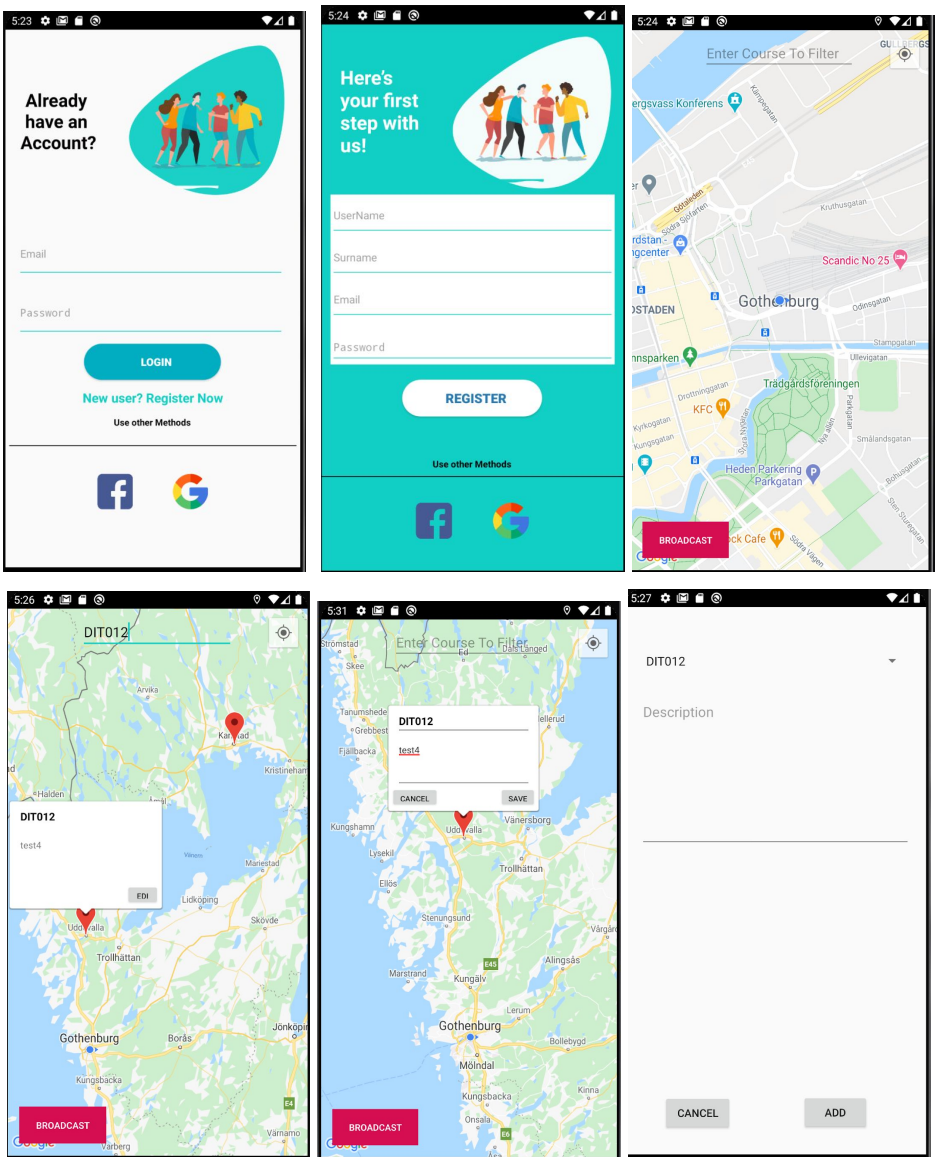

Figure 2. *Snapshots of the application's GUI during development.*

At this stage in development (see Figure 2), the support for different users was implemented, and the idea was that users should be able to register and log in using email, a Google-account or a Facebook-account. Once signed in the user is greeted with a map (Google Maps API). The user can pan around and see different broadcasts filtered by the course selected in the top search bar. The user can also add its own broadcast via the "Broadcast" button where they can select a course and set a description. More information about a broadcast could also be shown in the popup information window on the map, where it can also be edited.

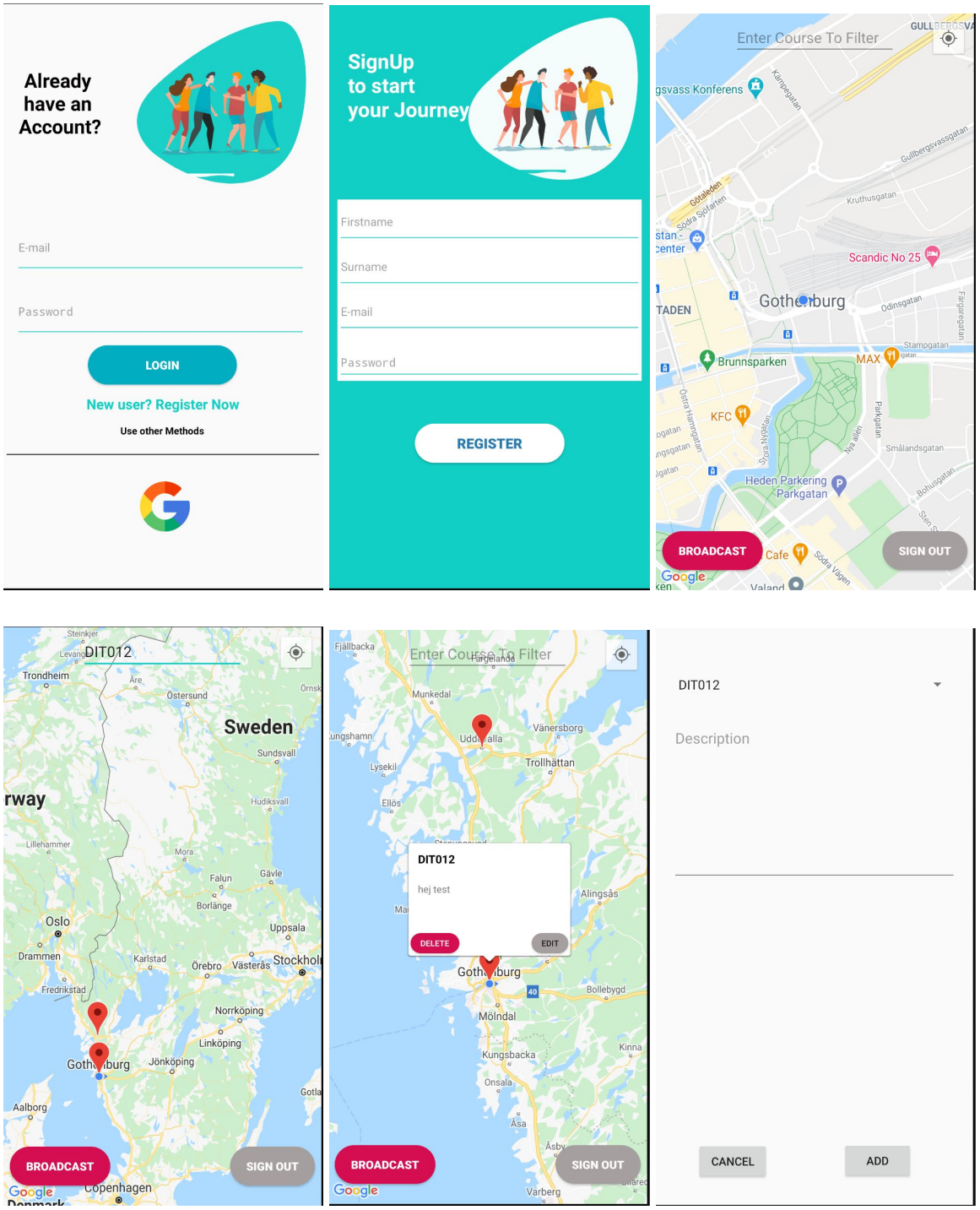

Figure 3. *Snapshots of the application's final GUI.*

Figure 3 shows the final version of the UI as of handing in the project. It is clearly shown that we had Facebook as a login button in the older version, but unfortunately, we faced a lot of difficulties in order to implement it. Therefore, the support for Facebook login was scrapped in favour of just email and Google-accounts. We also realised that on the registration page there is no need to have the Google sign-in button since it is enough to have it in the login screen because it automatically makes an account for the user and proceeds to the map if the registration succeeds. The support for Google-accounts was "easily" implemented as the

authentication service used is Firebase which is developed by Google and already supported this type of account registration. Furthermore, much functionality has been added "under-the-hood" and some of the interfaces have been "polished". The buttons now sport rounded corners and a new "Sign out" button has been added; allowing the user to sign out or switch between accounts. Additionally, a broadcast can be deleted by its owner through the popup information window on the map.

2.3.2 Login & registration screens

The application requires a user account in order to manage broadcasts. This is because only the specific user that created the broadcast (study session) should be able to delete and edit it. The application currently supports the user to register and log in using two methods; email or a Google account. There was a plan to support Facebook as a means to login but this was deemed to be a lesser priority due to time constraints. The button for registering with Facebook is, therefore, no longer included in the registration section in order not to confuse the user with functionality, not presently supported.

2.3.3 Map screen

Once the user is registered and logged in it is greeted with a map (using the Google Maps API). The map is designed to snap to where the user currently is located. The user's location is represented by a blue dot and is updated in real-time.

At the top of the screen, there is a search bar. Here the user can search for specific courses which the user is interested in finding study partners in. The search bar filters broadcasts and gives suggestions for courses to pick from. Once the user has specified a course to look for study partners in the relevant broadcasts are displayed as red pins or "red dots" on the map. The pins can then be interacted with by tapping on them. A small window then pops up with a description which the creator of the broadcast has written. If the user tapping the pin is the same as the one who created the broadcast the user can edit the broadcast's description by pressing the "Edit" button. The description text then becomes interactive and once the user is satisfied it can be saved and the broadcast is updated in the database which updates the broadcast on the map. The owner of the broadcast may also delete the broadcast by pressing the "Delete" button in the information window.

At the top right of the screen is a "location button" which automatically pans the map's view back to the user's current position. This can be useful if the user has panned across the map and wants to quickly return to its own position.

At the bottom right side of the screen is a grey button titled "Sign out" which logs the user out of the application. and takes the user back to the "Log in & registration" screen.

At the bottom left of the screen is a red button titled "Broadcast". If the user presses this button it is taken to the "Create a broadcast" window.

2.3.4 Create a broadcast

In the “Create a broadcast” window the top of the window has a drop-down list of courses to choose from. The courses are the same as the user could filter on in the “Map screen”. There is also a textbox where the user can add a description. For example, when GPS-coordinates are not enough as a descriptor of study location “I am studying in group-room R404 on the fourth floor”, or “in the back of Café C” would be a helpful description. The description could also be used if you want to include more information about the study session or maybe a description of who is hosting or participating. Selecting a course and adding a description is obligatory but there is no direct specification of what to write as we wanted it to feel natural and unforced. Once those parameters are set the user can press the button titled “Add” and it is added to the database and the user is taken back to the “Map screen” where the broadcast is displayed on the map at the user’s current location.

In the “Create a broadcast” window there is also a cancel button which also takes the user back to the “Map screen” but without adding the broadcast information to the database and thereby not creating a broadcast.

If the user has created a broadcast but strays too far from the area of the added broadcasts location it will be automatically deleted. This is a safety precaution in case the user simply forgets that there is an active broadcast session and decides to leave which might cause potential students eager to study at that broadcast location only to find it empty.

3 Domain model

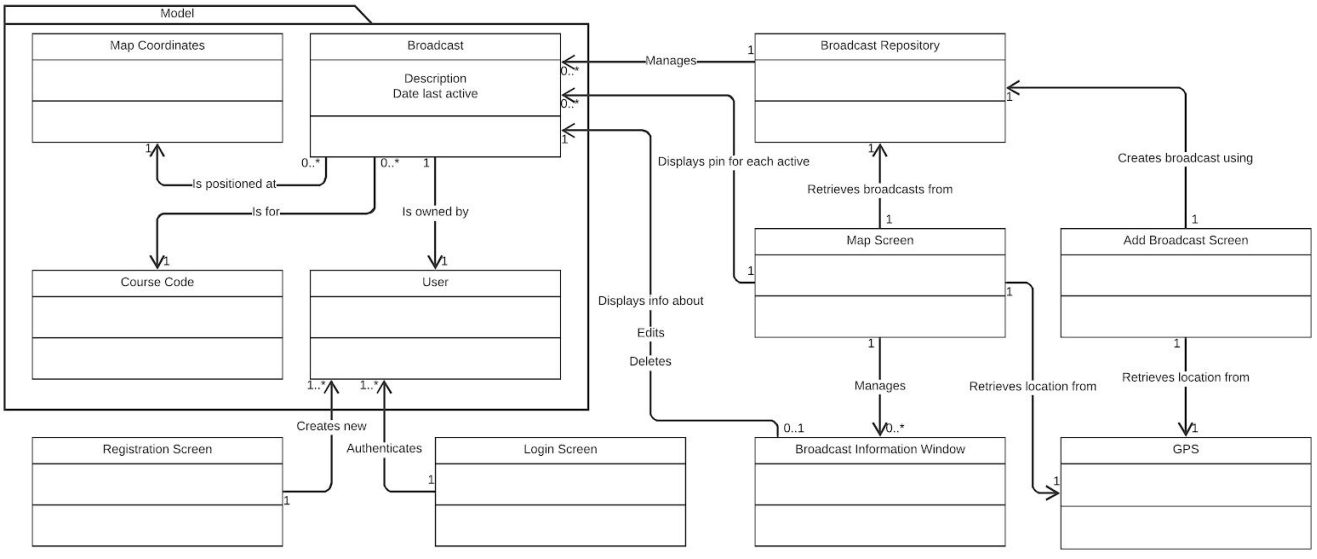

Figure 4. *Domain model UML diagram.*

3.1 Class responsibilities

- **User** — Represents a user and its attributes.

- **Course Code** — A unique identifier for a course used to reference what course a broadcast is published for.
- **Map Coordinates** — Represents a value object which merely contains the longitude and latitude of a location on the map.
- **Broadcast** — Represents a broadcast referencing an owner (User), a course code (the subject of the broadcast) and containing a location, a description and the date the broadcast was last active. This class also provides methods for updating a broadcast's course code and description, and handling whether the broadcast is active or not.
- **Broadcast Repository** — A database that has the responsibility to store and retrieve information about available broadcasts, and communicate them between different devices that use the application.
- **GPS** — The GPS class has the responsibility to communicate with Android's location services and provide the current or last known location.
- **Broadcast Information Window** — Displays information about a single broadcast on the Map Screen.
- **Map Screen** — A screen that displays a map with active broadcasts placed as pins on the map. The Broadcast Information Window may be opened from the screen for any visible broadcasts.
- **Login Screen** — A screen allowing the user to login with either email and password or a Google-account.
- **Registration Screen** — A screen allowing the user to register an account using email and password.
- **Add Broadcast Screen** — A screen where the user can choose a course and write a description and then press to create a broadcast that will be displayed on the Map Screen.

4 References

[1] *Android Studio and SDK tools*. Google, JetBrains, 2020.
[2] J. Tamplin and A. Lee, *Firebase*. Google, Alphabet.
[3] "Navigation," *Android Developers*. https://developer.android.com/guide/navigation (accessed Oct. 01, 2020).
[4] "Maps SDK for Android," *Google Developers*. https://developers.google.com/maps/documentation/android-sdk/overview (accessed Oct. 01, 2020).
[5] K. Beck, E. Gamma, D. Saff, and K. Vasudevan, *JUnit 5*. 2020.
[6] Android developers, "Espresso," *Espresso*. https://developer.android.com/training/testing/espresso (accessed Oct. 21, 2020).
[7] H. Dockter *et al.*, *Gradle Build Tool*. 2020.
[8] L. Torvalds, *Git*. .
[9] L. Torvalds, "Build software better, together," *GitHub*. https://github.com (accessed Oct. 01, 2020).
[10] EclEmma, "JaCoCo Java Code Coverage." https://www.eclemma.org/jacoco/.
[11] Bugan IT Consulting, "STAN, Report Generation." http://stan4j.com/reports/ (accessed Oct. 21, 2020).

Platforms:

- Android (https://www.android.com/)
- Firebase (https://firebase.google.com/)
- GitHub (https://github.com/)

External Tools:

- Android Studio (https://developer.android.com/studio)
- Gradle (https://gradle.org/)
- Git (https://git-scm.com/)
- JUnit (https://junit.org/junit4/)
- Espresso (https://developer.android.com/training/testing/espresso)
- JaCoCo (https://www.eclemma.org/jacoco/)
- STAN (http://stan4j.com/)

Libraries:

- Android SDK (Bundled with Android Studio)
- Jetpack AndroidX Support Libraries (https://developer.android.com/jetpack/androidx)
- Jetpack Navigation Graph (https://developer.android.com/guide/navigation)
- Jetpack Architecture Components (https://developer.android.com/topic/libraries/architecture)
- Google Play Services
    - Maps (https://developers.google.com/maps/documentation/android-sdk)
    - Auth (https://developers.google.com/identity/sign-in/android)

- InteractiveInfoWindowAndroid (https://github.com/Appolica/InteractiveInfoWindowAndroid)
- Firebase (https://github.com/firebase/firebase-android-sdk)
    - Realtime Database
    - Authentication
    - Analytics