Requirements and Analysis Document for StudyBuddy

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

2020-10-02
version 1

1 Introduction

The project is a mobile application developed for Android mobile phones 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 purpose is to bring motivated students closer together in study sessions.

The beneficiaries are students.

1.1 Definitions, acronyms, and abbreviations

Abbreviation	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	A framework for Android development to make it easier to

[3]	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
Gradle [6]	A framework for managing dependencies
Git [7]	A version control manager
GitHub [8]	A repository for managing projects using Git

2 Requirements

2.1 User Stories

2.1.1 Done and in progress

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

2.1.2 Backlog

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: 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: 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)

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.

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

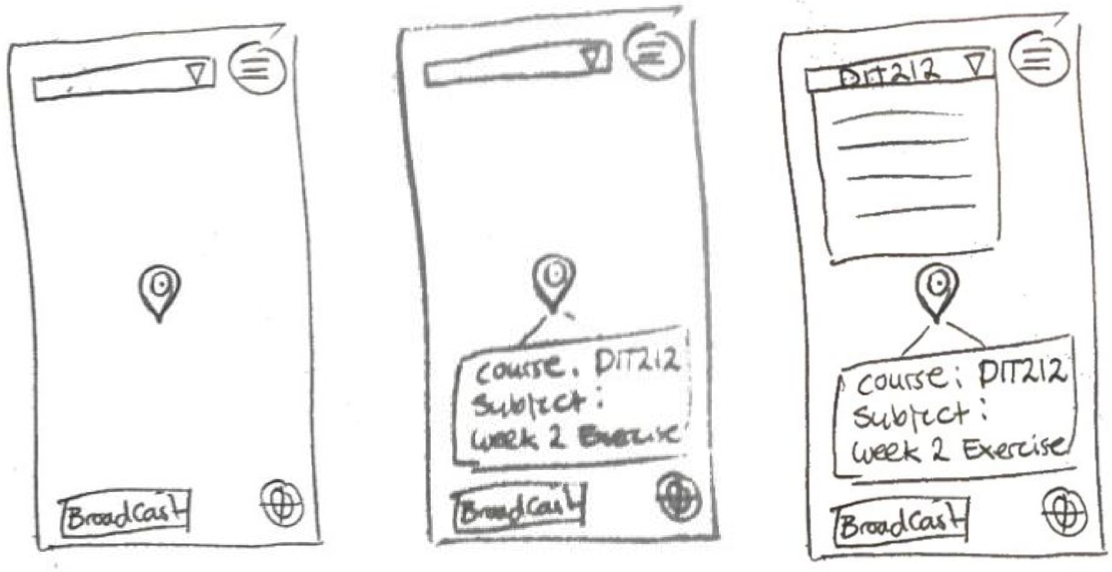

Figure 1. *Early sketches of the map screen GUI.*

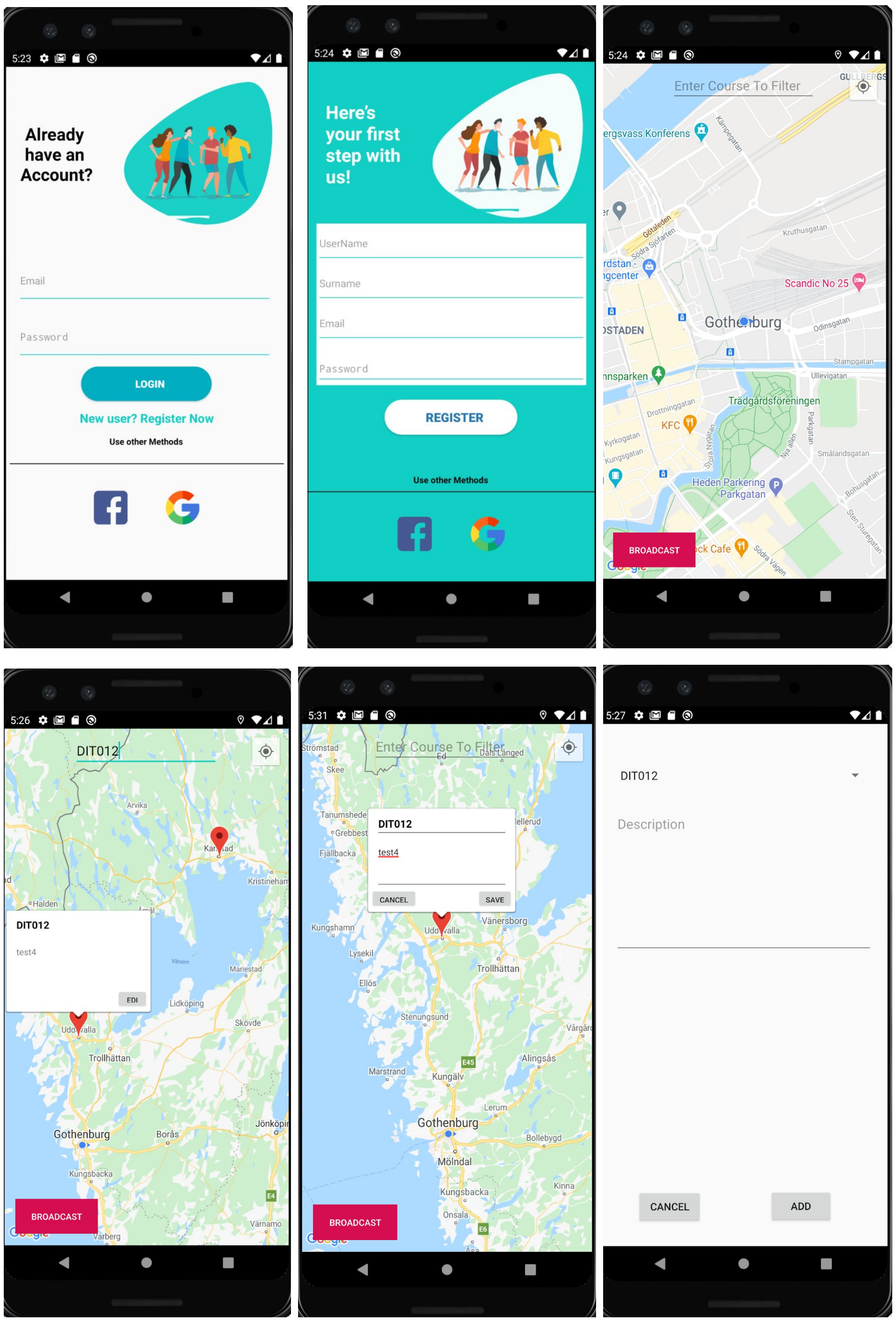

Figure 2. *Snapshots of the application's current GUI.*

The application currently has three primary screens: the login & registration screen, the map screen and the create broadcast screen. 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 it and pressing the edit button (not the geolocation, however, as that would call for a new study session).

3 Domain model

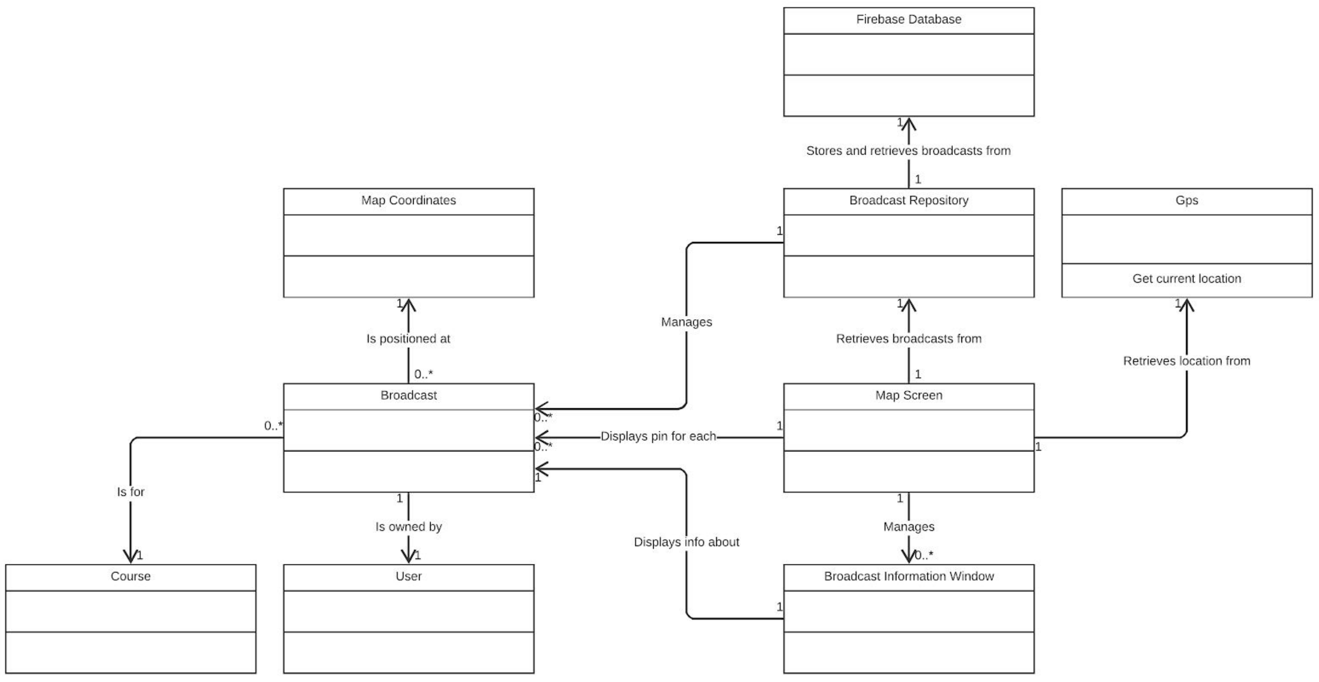

Figure 3. *Domain model UML diagram.*

3.1 Class responsibilities

Gps

The Gps class has the responsibility to communicate with Android's location services, and give the current or last known location to the other parts of the program that need it.

Firebase

Is a database that has the responsibility to store information about available broadcasts and other entities, to communicate them between different devices that use the program. The database is also meant to be storing login information for the devices who have registered users for the program. This is, however, not implemented yet.

Course

Handles information about "course" objects which get "published" within a broadcast.

User

Handles information about a single user of the system.

Broadcast

This class handles broadcasts which contain both a User (owner of the broadcast), and a course (the subject of the broadcast).

Map Coordinates

This class represents a value object which merely contains the longitude and latitude of a location on the map.

Broadcast Repository

This class communicates to the firebase broadcast objects created by this device, but also retrieves broadcast objects created by other devices from the firebase.

Broadcast Information Window

Displays information about a single broadcast on the Map Screen.

Map Screen

This is a central part of the domain which displays active broadcasts on a map. This part can publish broadcasts created by this device via the Firebase database and retrieve broadcasts created by other devices. This lets the device manage their own broadcasts, and broadcast them to other devices, while gathering information about broadcasts retrieved from other devices via the database. Setting this together with the GUI lets the program show a map with descriptions of other devices broadcast, in relation to where you are located, while letting the other devices access the broadcast of this device.

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] H. Dockter *et al.*, *Gradle Build Tool*. 2020.
[7] L. Torvalds, *Git*. .
[8] L. Torvalds, “Build software better, together,” *GitHub*. https://github.com (accessed Oct. 01, 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/)

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