Design Project

Department Of Computer Science

Design Report

Author:

Design Project Group 3

Liang Zhang(s2040093)
Shikuan Li(s2030640)
Zihan Wang(s2093685)

Supervisor:

Yeray del Cristo Barrios Fleitas

April 15, 2021

UNIVERSITY OF TWENTE

Abstract

This report is written as the main report of the “Design project” module at the University of Twente. In this report, An android application called “TravelSquad” has been designed and implemented. The development of this system was motivated by the following purpose: *“**The purpose of this app is to plan group trips. First, you create a group and invite your family/friends. Then you create a travel proposal. Optionally, you can add an itinerary, a date and a budget. The rest of the group will see your travel proposal and can vote on it and propose changes if, for example, somebody found a better hotel. Planning will be lighter, more inclusive and fun than ever**.”(cite from FMT: Android app for planning group trip).* In this report, four critical stages are elaborated: planning, design, development, and closure. It describes all essential phases of our project work: user stories and clarity of requirements, design of class diagram, database, and UI, choice of platform, etc. As this is only the earlier version of our product, which took ten weeks, we look forward to increasing its usability in later versions to fit our target audience or put it into commercial use.

Table of Contents

1 Introduction

Traveling is an integral part of life as it saves us from stress, stress, and depression that we have faced. It improves our mental and physical health by meeting new friends and broadening our horizons. However, all this presupposes that we have a reasonable travel plan.
Here is an example of how people think about travel and negotiate the details of the plan for it: *"**For example, for this Christmas, Pedro Flintstones would like to spend a warm Christmas somewhere in the Caribbean, but Popeye would prefer a place that doesn't involve spending so much money, since he has spent more than he expected on spinach this year. Scooby, however, doesn't care about the venue as long as the hotel has a buffet lunch included. A cheap and warm Christmas with an all-inclusive hotel is not easy to make everyone happy, unless the decisions are made in a simple way**. (cite from FMT: Android app for planning group trip)."* In this example, Pedro, Popeye, and Scooby, as our potential clients, have their requirements for the trips. Usually, Verbal agreements accompanied by bickering do not make for effective planning since every individual must engage in the planning process. A long journey may take a few days to schedule. Hence the plan should be recorded and updated timely.

There is some divergence between Pedro and its members in terms of location, budget, itinerary. Those travel details are required to be addressed and recorded effectively and systematically. Hence, we aim to design a system that allows users to make a plan that meets individual needs and engages its trip members.

2 Domain Analysis

In this chapter, the existing solutions are identities and compared. We have extracted the positive parts of them, such as comfortable but straightforward UI and Smooth and rational function switching but also noticed some related features such as high-customized travel plans which do not fit our product.

2.1 Existing Solutions

2.1.1 Flow of Wanderlog

We selected some popular travel planning apps from the market in the initial needs-finding and analysis stage of the software. Among these apps, the most similar to our project and the reference value is a travel planning app called "Walderlog." There are many recommended travel guides provided by other users and travel bloggers in this program's main interface. Users may create a trip plan or trip story for sharing their travel experience; also, they can look for guidance when they are hesitant about planning a trip. When a user wants to plan a journey, the destination and the trip's duration are needed to be filled in and invite the other users. Users who accepted the share may get access to the trip to view and edit the detailed information such as places to visit, itinerary activity, hotel, and transportation inside the trip.

2.1.2 UI and functionality

The overall UI design includes three parts: Home fragment, Create Trip fragment, and User Info fragment divided by a bottom navigation bar. Efficient, clean, and well-designed homepage consists of image, title, brief info of the trip, and publisher at the main activity; see Appendix C. In creating a trip interface, users are faced with three options, including trip plan, trip story, and guide. Three main steps include input the destination and its period and optionally invite your friends when users want to create a trip plan. The page displaying the trip is composed of three fragments, overview, itinerary, and exploration. The information in each area is reasonably distributed, and there is a linkage between each fragment. At the same time, the information displayed by each fragment is rich enough and not complicated. Inside the overview, a background indicates the trip and time slot title along with the profile picture to show how many users are in the trip also equipped with a button that can invite users. Such a UI design guarantees aesthetics to the greatest extent and fulfilled functionality at the same time to provide users with the most convenient experience.

As for the itinerary part, a horizontally scrollable date navigation bar for locating a specific date is an efficient design method for user experience. When there are many days in the trip, it is hard

for users to find the date he/she wants to edit the schedule, then this functionality provides the user with the easiest way to edit the agenda in a day. Although the distributed schedule design can make intuitive itinerary feedback to the user, many inspection summaries such as scenic spot information, business hours, and geographic location are also added simultaneously when entering a specific scenic spot. The abundance of information may make users ignore the attraction itself or the original intention of adding this attraction because the attention is distracted by this information.

Overall, Among the existing solutions, the functions of Wanderlog are most in line with expectations, especially its UI design. In our project, the efficient, concise, and practical UI design will be retained, and our unique utility will be added, such as voting and chat systems. Inside the itinerary page, we will modify with unique features based on the scrollable date navigation functionality and remove unuseful data to ensure that users are focused on the schedule. Users may also be concerned about the budget for the trip with building up an itinerary.

2.2 Our solutions

With such a demand, we plan to design an application that aims to provide a platform for users to arrange their trip remotely like they are planning face-to-face. The uniqueness and advantage of this application will be explained in the following sections.

First, having a central place where all group trip members can access the group trip plan is key to keeping everyone informed. When people are making a group trip plan, they may have disagreements about the time and budget. Also, everyone has different desires for the trip. Some may want a cultural trip with a lot of heritage to be visited, while others may just want a vacation with more leisure activities. Someone may just don’t want to spend too much money, and the time for everybody may also have conflict. There are always inconveniences during the whole process. So it seems that traveling is not just about choosing a destination and buying a ticket, the importance of making an itinerary goes without saying.

- It helps maximize your time and manages it perfectly.
- You can manage your expenditures and estimate your cost.
- Easier and quicker travel from one point to another.
- It helps in prioritizing your wants and needs.
- It makes sure that trip essentials are not forgotten.

Second, during our investigation, we find that most applications do not let every participant propose their idea well. The owner decides everything, which is not an appropriate way to plan a trip that everyone enjoys. Instead, we want to build a platform where all group members can discuss and finally create a plan together. Our idea is to provide more interaction functionalities to our users such that all the members can feel involved in making the plan.

Third, using a payment system can help manage all the money things and cut out all the troubles like who owes them. To solve this problem, we also add this feature to our application for convenience.

As mentioned in the introduction, instead of just creating an application for individuals to note their already planned trip, we aim to build a platform for our users to discuss and share their preferences. We solve the problem by determining some critical parts by voting, just like how people plan a trip face to face. First, the group owner creates a trip and invites his friends, which will be finished by sending an email and generating a link. If the user has not installed our app on his phone, it will ask to download the application first. Then the user can join this group by clicking the link. After all the members are invited to this group, the Budget and Time need to be discussed. In our application, we make this process easy by replacing the discussion with voting. Everyone inside this group can propose their desired time and budget. Then everyone chooses one of the options.

After the date and budget are decided, the owner can begin to add activities to the itineraries. The details of every activity need to be provided to members to determine whether they want to participate. To better improve the interaction between owner and members, we implement a "Like or dislike" system. We give up voting for the detailed itinerary because, considering the ease of use and the group owner is the person who knows most of the activities, he is the one in charge of making plans. We think that voting for every activity is troublesome, which also brings harmful user experiences. With this "Like or Dislike" system, members can express their attitude towards the same activity. Based on this result, the group owner can decide whether to make some changes to this activity. After all these details are determined, we think that some users have the requirement to ask every member to transfer money to a Travel account, in which way people do not need to care about the annoying money things. We design a function that can help to notify people to transfer money to that account.

3 Requirement Specification

The coordination of this project is using scrum methodology as our development method. We initially schedule one week for planning our working flow for the application and two weeks for design and drawing our user interface and coding scheme. In the following part, all requirements will be summarized.

3.1 functionality requirement

At first, we brainstormed the user stories (in Appendix A) with our clients about what a user and owner of the project must, should, and could achieve in different scenarios. After that, we analyzed and measured each requirement's difficulty and completion time and made the following table. This table describes the primary and secondary tasks to do in the limited time of 10 months.

Must have	**Should have**	**Could have**	**Won't have**
Data stored in the database	**email invitation for new users**	**Group discussion**	**Full recommendation for trips**
Different permission levels for users and admin	**password protection**	**Anonymous Login, Social media such as Facebook/Twitter Login**	
voting system	**High usability**	**Notification to transfer Money**	
Account manipulation and user management			
Group management			
Creation of travel			

proposals			

3.2 User Interface Requirement

After checking the existing solution in the domain analysis, the application's user interface performs quite well. So the overall graphical interface and operation flow of this application should upgrade from that base.

3.3 Data Manipulation

As a complete and persuasive product, data manipulation is an essential factor we need to take into consideration. Finally, we choose Firebase as our database instead of MySQL based on the following features:

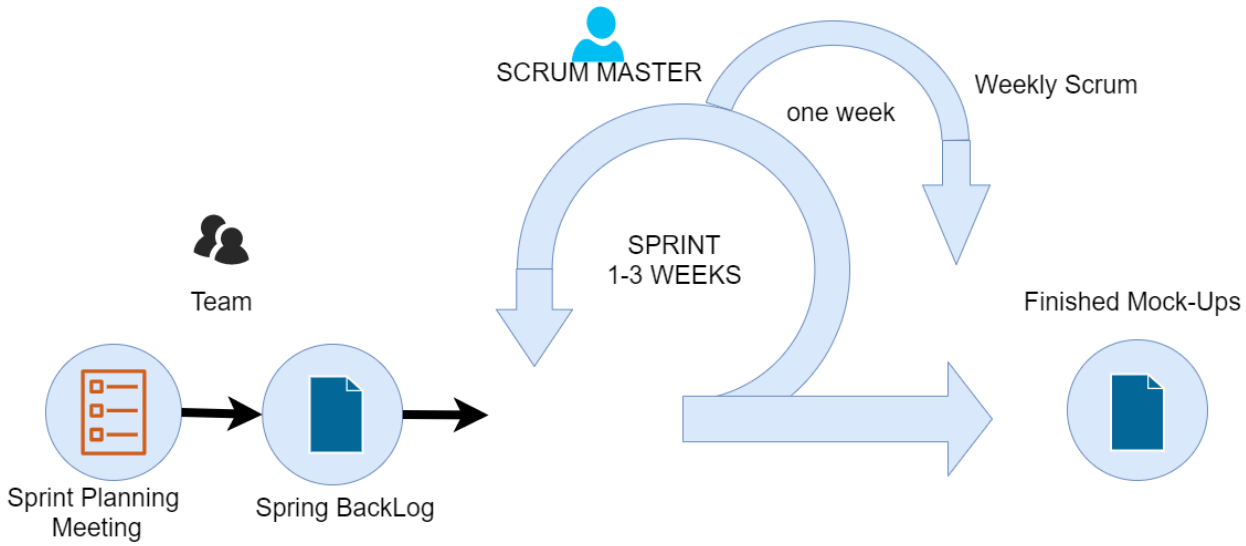
- Firebase stores and syncs data in real-time
- Firebase has dynamic schemas to facilitate unstructured data, which is extremely helpful when we only have limited time to implement our product and try to achieve our primary structure, such as store user data and allow the user to add trips. Instead, MySQL has a predefined schema.
- Firebase provides other functionality based on the stored user data. And this is elaborated in part 6.2.

4 System Proposal

The systematic design was carried out from the first week to the third week, and every Monday, we confirmed the requirements with the contact person and slowly revised our proposal. After three weeks' revision, we made our initial proposal as follows. Hence, there are some differences between our proposal and the final implementation.

4.1 Mock-ups Proposal(our drawing of UI)

The design and implementation of this prototype used the platform XiaoPiu and can be found in Appendix B. This task took three weeks to complete and follow Sprint cycle(Porras,2020):



4.2 Features Proposal(How features are organized in the UI page)

The features proposal consists of initial ideas and design on how this system will work. However, the actual implementation sometimes does not match our expectations. Hence the proposal might differentiate from our final implementation. The difficulties we face and the reasons we modify some features' architecture will be discussed in the Detailed Design(Part 6). Features are organized based on the systematic functionality requirements we have collected from our client(check detained in part 3.1 functionality requirement).

Must have features:

- **Data stored in the database:** firebase is used for storing our user-related data, trip data, and profiles. (check detail in 3.3 Data Manipulation)
- **Different permission levels for users and admin:** Admin of firebase backend is only accessible by the project owner using google account. Different permission levels for users are reflected in the system: the creator of the trip(owner) can decide which part of the trip is allowed for modification. The owner can make the final decision. In the meantime, participants could always determine whether or not to accept this trip offer.
- **Voting system: the** simple voting system does not mean everything is negotiable. Hence we aim to make a powerful product to provide choices for users to decide which part of this trip is applicable for voting.
- **Account manipulation and user management:** Account authentication and management can be handled by firebase, including Email address verification, password reset, email address change, and SMS verification. Details on how we implement will be explained in part 6.2.1
- **Group management: The** group is meant for the discussion group, same as a voting group created once a trip is made, and participants accept a trip link shared by the trip owner.
- **Creation of travel proposals:** this is achieved as our fundamental feature in the main activity, and the created trip will be stored in the database.

Should-Have Features:

- **Email invitation for new users:** this feature is inherited in the trip fragment and is achieved by clicking on the share button to spread the invitation message. New users will be invited to download the application while existing users will be asked for trip planning.
- **Password Protection:** Firebase provides Multi-factor authentication(MFA) to protect sensitive data.

- **High usability:** High useability includes Effectiveness, efficiency, engagement, error of tolerance, and ease of learning. Specifically, this should be achieved by optimizing the algorithm we fetch, store the data, and will be explained in detail in part 6.3.5

Could-have Features:

- **Group Discussion:** group discussion is combined into the trip fragment. Hence for each trip, there is a unique chat room for all participants to discuss their ideas.
- **Anonymous Login, popular Social media Login(Facebook, Twitter, etc.):** this feature could be implemented in the login page and match their social media accounts with our application account.

Won't-Have Features:

- **Full recommendation for trips:** some of the existing solutions mentioned in section 2.1 have journal recommendations(timetable, expenses estimation, duration, etc.), which is not the aim of this project.

4.3 Risk Analysis

4.3.1 Firebase

The first potential risk is Firebase. In the implementation of the application, We will make use of Firebase in terms of Authentication,dynamic links, hosting, realtime database, and file storages. In another word, the application cannot work remotely without the support from Google Firebase. Besides that, Firebase stores user information and files user upload. The security of user data is possibly a potential issue.

4.3.2 Workload

Second risk is the ability of group members to find out how to design and implement functionalities using android studio. None of us are experts on android studio, and even unfamiliar with conventional development framework of android studio.

5. Architectural Design

In this section, the architectural design of this project will be discussed in a global manner. Besides, the designed workflow of application and functionality will be displaced.

5.1 Global Design Choices

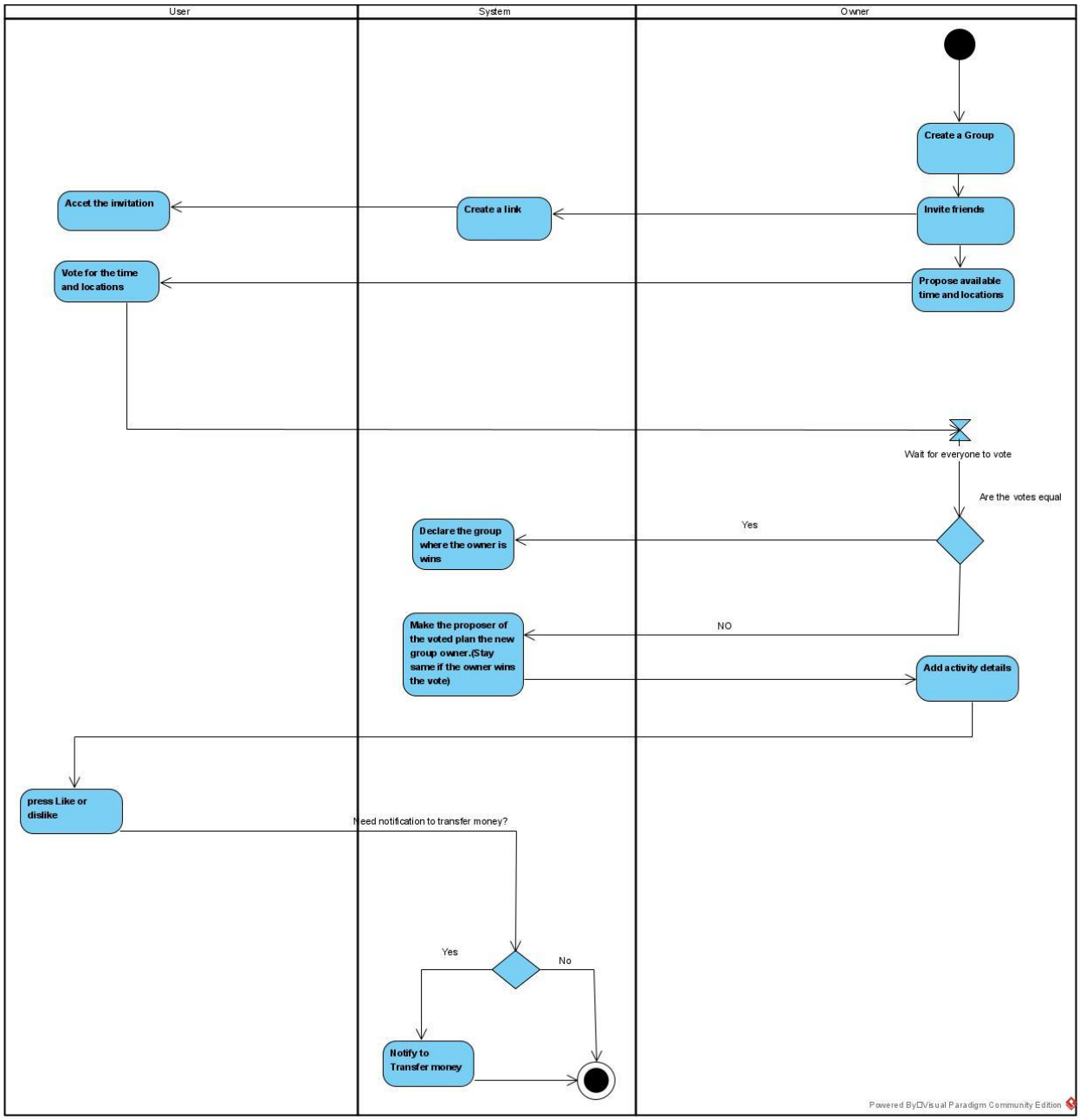
This project aims to create an appeal that is different from existing products on the market. Hence, we optimized the operation flow while keeping the UI that users are used to and leaving more options for users.

5.1.1 Key Design choices

After a clear understanding of requirements specifications, this application's basic and advanced functionality is relatively straightforward. However, some functionalities do not perfectly fit into the system, or our clients strongly recommend some extra functionalities. Hence, we applied **Agile Design** as our primary design model, which breaks our application into each functionality. This principle allows project developers to work on different functionalities separately. The advantage is that it will enable us to remove any functionality or add any functionality without influencing the long-term planning. On the other hand, this design involves customers who should regularly give us feedback to modify our short-term frames, mentioned as our Sprint Process in section 4.1.

5.1.2 Workflow design

This system's workflow mainly engages with three sections: Owner of this trip, participants, and system.



5.2 Preliminary Design Choices

5.2.1 Frameworks and Libraries

The development environment we choose is Android studio which is officially for Google's Android operating system, built on JetBrains' IntelliJ IDEA software and designed specifically for Android development. And we have applied the latest version(API level 30, Google Play 11.0) to build our emulator testing our application. Instead of the original Android Support Library, we applied android X to develop, test, and package our application. Android X has a significant improvement compared to the original version in terms of clear package structure and bundled classes. The old support libs are available until version 28, as the version newer than that will be released under the Android X project.

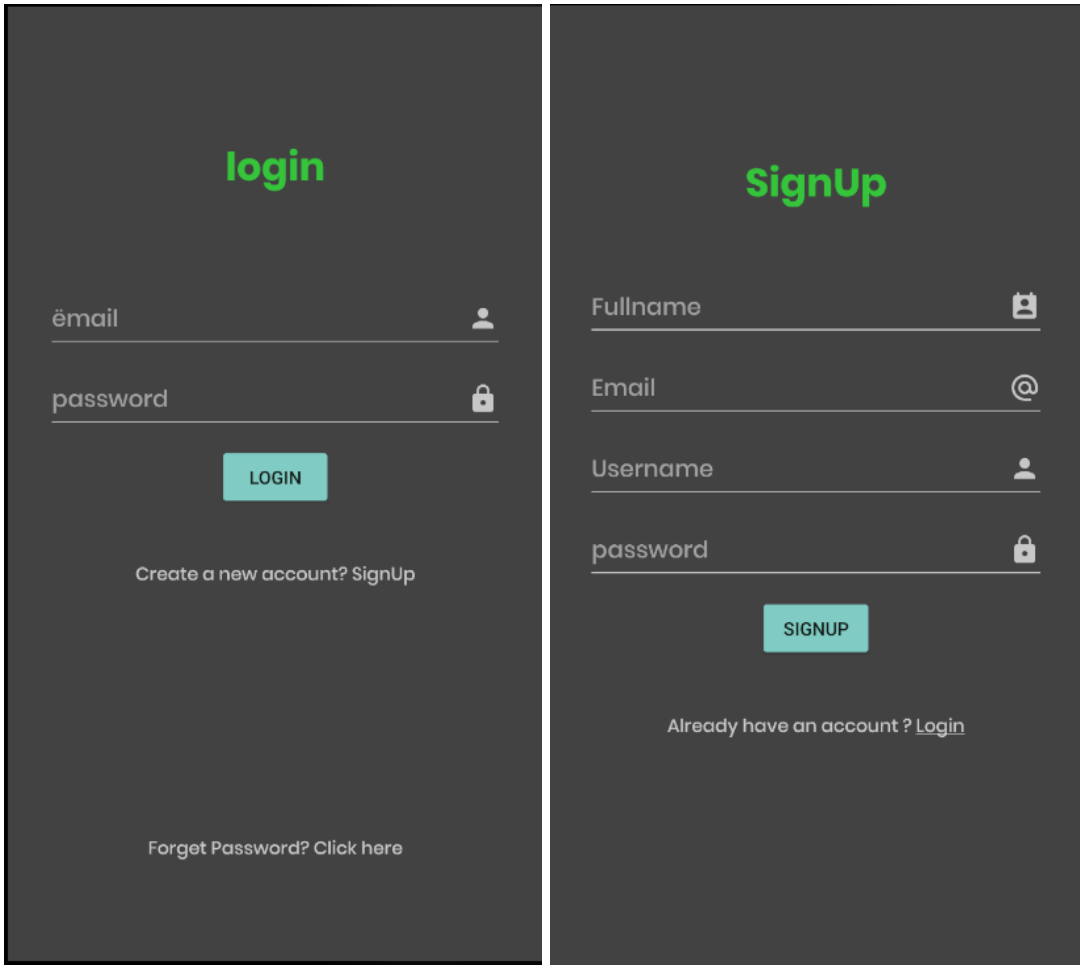
5.2.2 Programming Language

Android studio provides Java and Kotlin as the development language, and we choose Java since we are more experienced with Java development. However, Kotlin is suitable for developing server-side applications, which could help us focus more on the server-side. We will probably try to use Kotlin if more time and more server-related tasks are required, and it could happen in our future version.

The Java class files provide support for our application's backend, and XML is the markup language used in the android studio, much like HTML. As XML is a lightweight language, it makes our layout less complicated. Design and Code are two primary choices to build an XML file, and we mainly choose The Code option because it is more compatible with different devices in terms of different layouts.

5.3 Application Pages and Fragments Overview

5.3.1 Sign and SignUp Pages

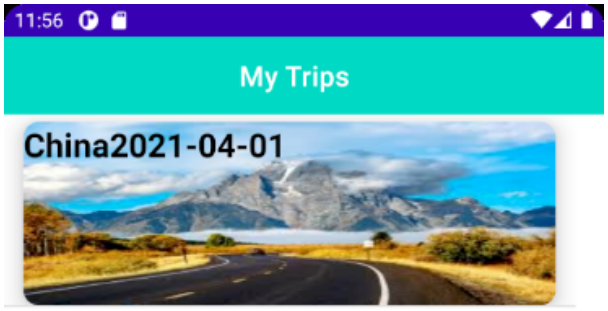


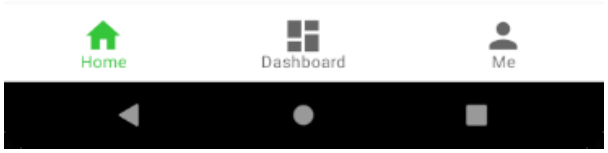
The fundamental pages for an application are **Login** and **SignUp**.

In the **SignUp page**, we collect the full name, Email, username, and password from users. The Firebase platform we have applied will check the validity of email addresses by sending verification emails. Only when the verification link is clicked will the account be authenticated to login in. Besides that, passwords will be stored in Firebase's Authentication, which means it only allows users to modify or verify.

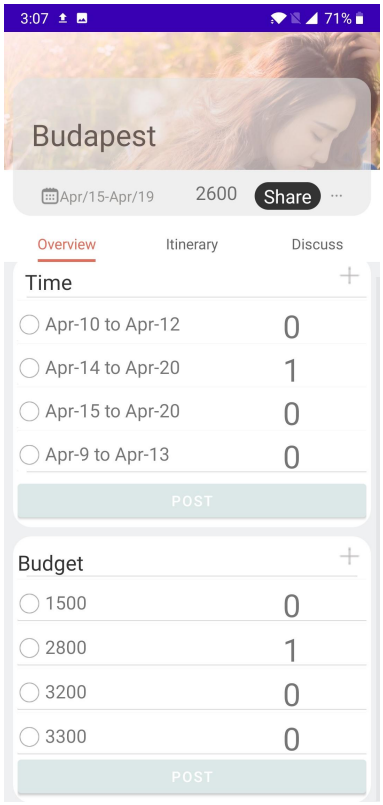
In the **Login Page**, only email is accessed to log in since logging in using a username or social media account such as a Facebook account is a "could-do" feature to be released in later versions.

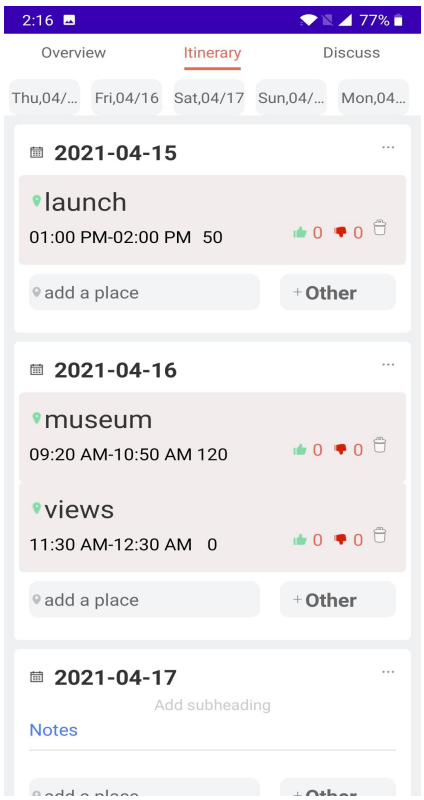
5.3.2 Home Page

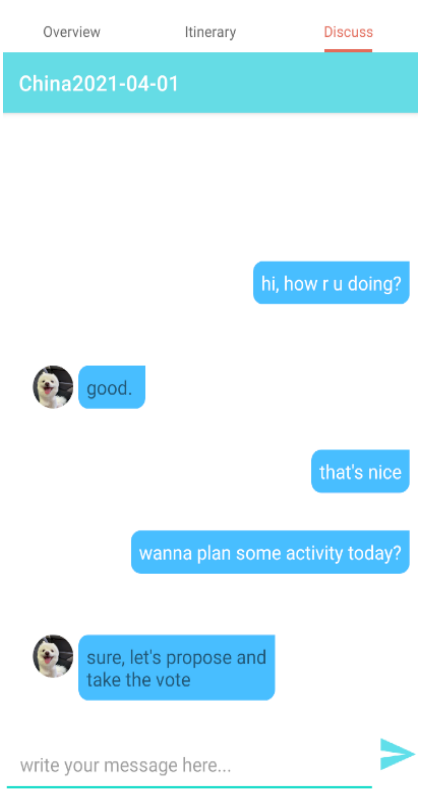




The main page is the **Home page**, which displays trip plannings we had created or participated in. The leading information in each card view including destination, duration, and budget. When users click one of the trips available, the details about this trip will show up with the following **fragments**: **Overview**, **Itinerary,** and **Discussion**.







(1) Overview Page **(2)Itinerary Page** **(3)Chat**

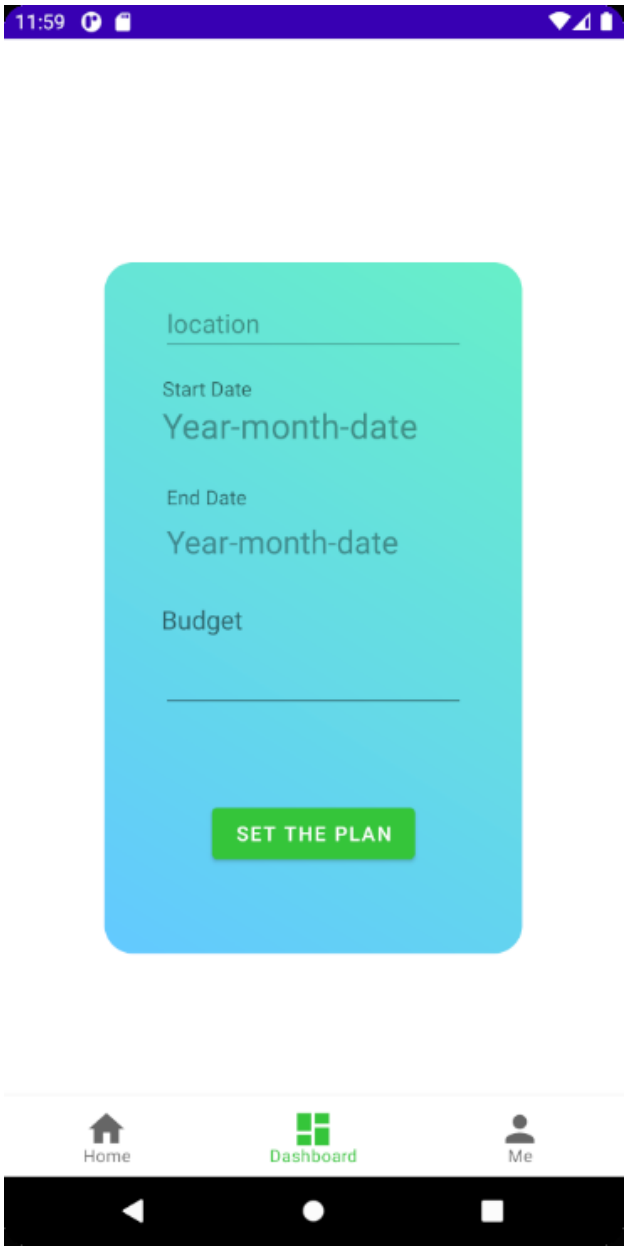
The overview includes **Time**, **Budget, and Share** Button for this trip:

- **Time** is used when participants have different desired durations for the trip, and they could post their opinion and vote for each other. There is one vote per person.
- **Budgets** are estimated and posted by the trip owner and the participants, which usually consists of multiple different numbers. This page allows the trip owner and participants to vote for their estimated budget and also proposed their desire. There is one vote per person.When the participants(not the owner) propose a suggestion,if there is a significant difference between their proposal and the owner's,they will receive a notification that according to the owner's suggestion,your proposal is not reasonable.
- **Share Button:** Clicking on this button will open a new window to search the latest and recommended applications for users to send invitation links. After the invited users click the link,they will see this trip on the **Home page** and participate in the voting process.

Itinerary includes a **grid view**, which separates the whole trip into every single day. This ensures the simplicity of the interface and allows the user to make quick changes to the plan. With a Top Navigation bar, users can easily locate the day they wish to find.For each activity, users can express their attitude

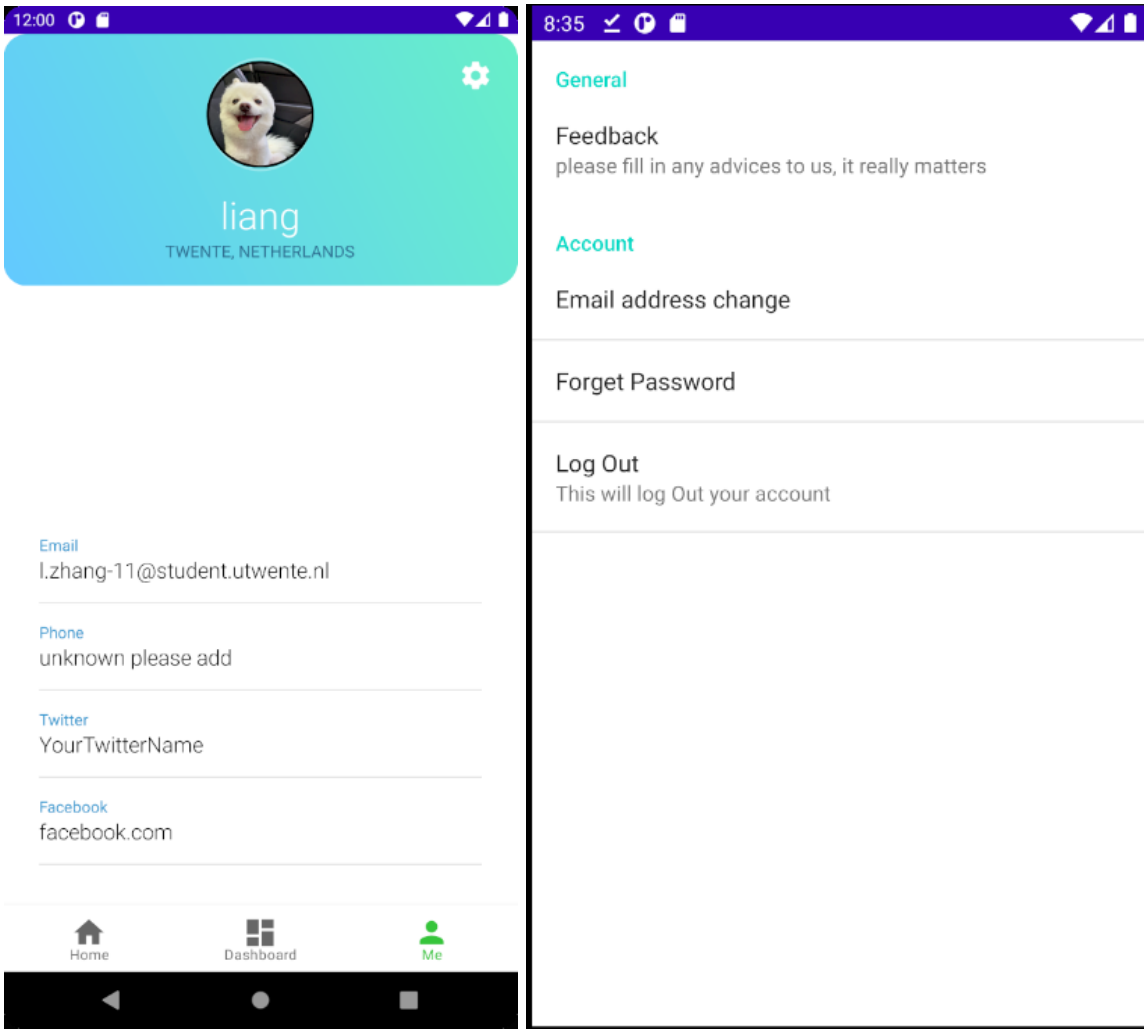
Discussion is unique for each trip: **Top bar** inside the discussion fragment will display the trip's destination and start date.

5.3.3 Add Plan Page



Adding Plan Page: This page requires the user to input **Destination**, **StartDate**, **EndDate,** and **Budget** as initial information. Every trip starts from this step, connecting to the database and recording any planning users make later.

5.3.4 Me Page



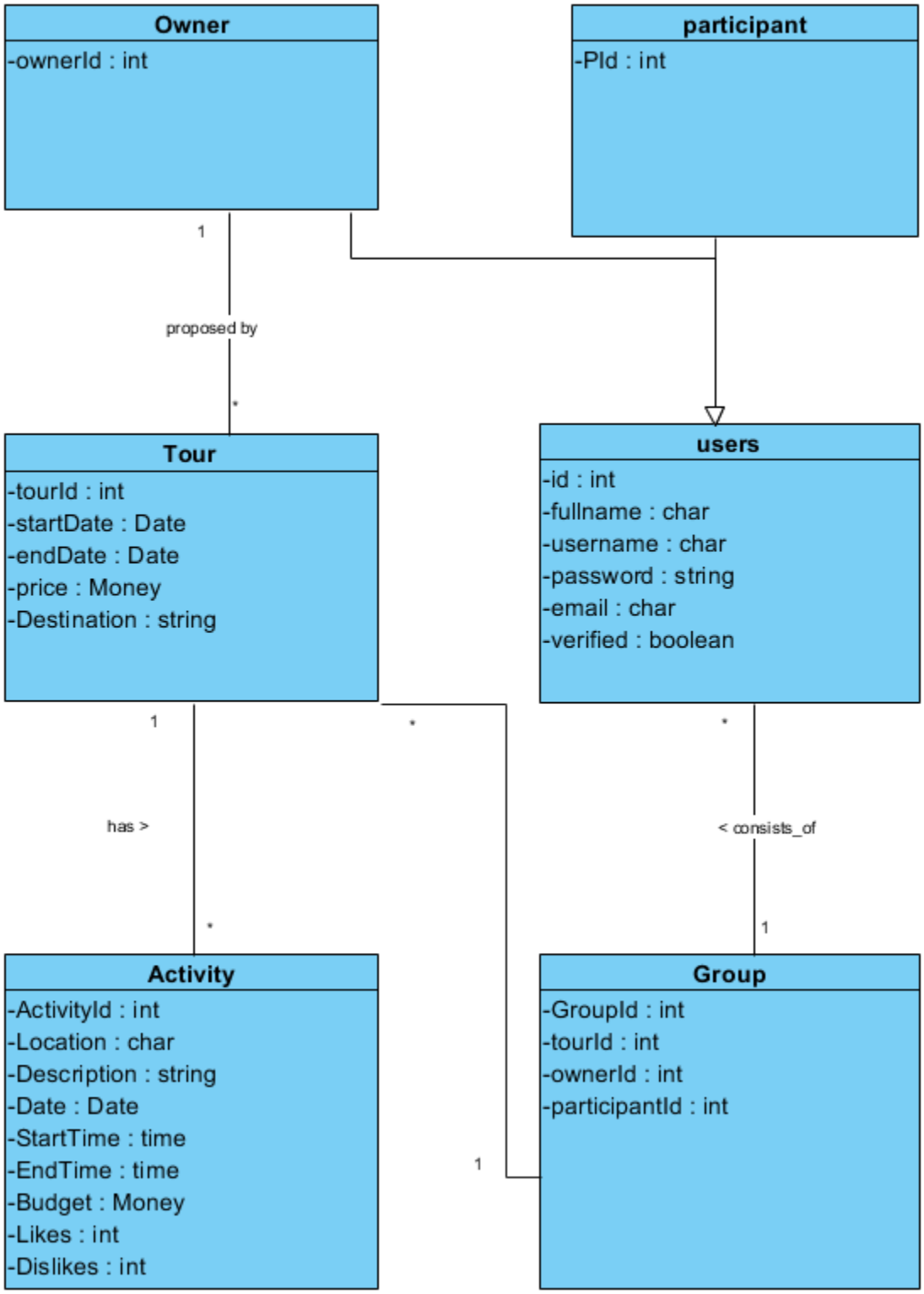
Me Page includes the personal information: username, location, email, phone number, and future connection to Twitter and Facebook if users choose to log in to the application using those social media accounts. However, this is a “could-have” functionality, which may be released in future versions.

Setting Image View is also on this page, in which users can accomplish fundamental account-related tasks: Email address change, Forget Password and Log Out.

6.Detailed Design

In this chapter, the detailed architecture of this application will be displayed. Besides that, the design brainstorming and reasons for modifying each component will be identified, explained, and justified.

6.1 System Description



We planned to use SQL as the database in the proposal, so we designed the class diagram above. It consists of mainly five classes:

The class **"Tour"** is used in the Add Plan Page, in which users input startDate, endDate, price, and destination as the initial variables of this trip. TourId is generated using the combination of destination and start date(For example, Twente-03-03-2021). And this tourId will be displayed on the Home Page. The purpose of this naming technique is to prevent users from confusing this trip with other trips which may have the same location or same duration.

The class **"Users"** is used in the SignIn and SignUp page. It mainly has two subclasses, Owner and Participant, both of which have the exact attributes "id," "fullname," "username," "password," "email," and "verified "as primary information. After users input all the information, that information will be uploaded to Firebase Realtime Database except "password," which will be protected by the Authentication function of Firebase (protection issue will be expanded in 6.2.1). Besides that, "verified" will be stored as "false" as the default value. It changes to "true" only after verifying their email via clicking the verification email sent by Firebase and login into the application. Using this technique, the admin could easily count the real active users from the backend.

The class **"Activity"** is used by Itinerary Fragment inside the Home Page, which has attributes(require users to input): "Location," "Description," "Date," "StartTime," "EndTime," "Budget," attributes(self-generated): "ActivityId" and voting attributes(displays as "Thumbs Up" and "Thumbs Down" emoji): "Likes," "Dislikes." Every participant has one vote on each activity: either "likes" or "dislikes." "Activity" is auto-generated using tourId and Location of this activity. (e.g., China-2021-03-03-Shanghai). This is the child class under Tour, in which the duration will be separated into every single day using grid view, and users could add and modify any activity under any day in this trip.

The class **"Group"** records the relationship between each tourId and trip owner and participants. The owner of the journey firstly establishes it. When the owner creates trip A, the owner is the only user who can view this trip's details. After this trip is shared with other users, they will be recorded under this class as participants' identities. Participants have the same right to propose, modify, and vote for activities except for the decision rights exclusive to the owner. This is "Group" class is not merged into the "Tour" class because a solo class will make it easier to search all participants in any specific trip.

The class **“Chats”** is used in the Discussion Fragment on the Home Page. This class has the same id as TourId. The discussion room is established at the same time this tour is created. It consists of attributes: “MessageId,” “Date,” “Message,” “SenderId,” “Time,” which is quite similar to our current social application(e.g., Whatsapp). “SenderId” is used for the application message adapter to recognize the message sender in this group discussion.

6.2 Firebase Service Analysis

Based on our knowledge of java, SQL database, and preliminary understanding of android development, we analyzed and implemented most of our functionality using the android development platform Firebase. The brief reason why we choose Firebase during the proposal stage is explained in part 3.3. Besides that, it is pretty significant to demonstrate the functionality of Firebase and explicitly elaborate how our project is rooted on Firebase.

6.2.1 Authentication & Protection Protocol

Firebase Authentication provides backend services to support user authentication. It supports authentication using passwords, email, phone numbers, popular social media identity providers such as Google, Facebook, GitHub, Anonymous, and more. It also provides easy-to-use SDKs such as Email added verification, Password reset, and SMS verification. Currently, we have implemented all except SMS verification.

According to the commitment from Google Support(Firebase Support, 2021), Google is committed to helping our project succeed under the EU GDPR and the California Consumer Privacy Act (CCPA) regulations, whether large software companies or independent developers.

6.2.2 Realtime Database

Instead of using SQL database, we had chosen Firebase Realtime Database.
Because Firebase uses JSON trees as a data structure, we made some modifications to the class diagram. Some of the essential changes are: (see details Referring to Appendix D)

- Instead of defining each table in SQL, we created different child nodes under the root project(e.g., Group, users, trip). Child nodes are not directly connected by Primary Key, but it is still possible to change two different child values by specifying the conditions.

- We add extra child nodes(classes) after finishing the class diagram, including Feedback and Chats, which did not come to our mind at the design stage. But this is the advantage of JSON tree structure, which allows us to add extra features easily without notifying any other child nodes or making any unnecessary connections.
- Compared to SQL databases, Realtime Database has Security Rules which can determine who has read and write access to the database. This could be integrated with Authentication to allow different authorization levels for different users.

6.2.3 Storage

Storage is utilized by our application to store the image uploaded by users. Hence users could use previously set photos even if they switch devices or reinstall the application. Besides that, we store some representative landscape photos, and this will allow users to have a preview of the view on the itinerary they designed.

6.3 Design Choices

Some designs are modified during the project implementation stage, while the application is revised each week by group members and clients. Basically, new designs follow most people's cognitive habits.

6.3.1 Discussion functionality

The discussion functionality is initially not in the discussion. We imagine users could chat on social applications such as Whatsapp. However, in order to save time and effort for users to switch different applications, the project group decided to implement Chat functionality with an elegant user interface. (see here)

6.3.2 Voting System

The critical difference between this application and other Google store applications is that we have designed a customized voting system for our clients. This is unique and customized. However, it is hard to fit everyone's preference. Initially, the brainstorming allows the owner and participants to vote for everything, such as location, duration, and daily schedule. But after some days' discussion with the group master, we all agreed it is too complex to handle, and it does not fit the condition in the real world. Hence, we decided to make the following modifications: the **Location**(country and city) is fixed for each trip. **Duration**, **Budgets** are offered by the trip

owner, and participants can only vote for their preferences. The owner will make the final decision. **Activities** on each day can be arranged and voted by both the owner and participants.

6.3.3 Duration modification

In the beginning,our idea is that when users plan to change their trip dates,if the duration is not consistent with the former one,they can choose the itineraries of the dates the want to remove.For example,they want to change their trip from 1.May - 5.May to 2.May - 4.May,they have to choose two days of itineraries .But during implementation phase,we find it difficult to implement during to our database schema,so we choose another way.We check whether the new dates have the same date with the old one,if so,we keep the itinerary at that day and remove the day that is not inside the new dates.For the new dates,we create new dates.

6.3.4 Verification of users

In section 6.1, there is a new attribute called “verified” in the “user” class. This feature is added when we receive new requirements to upload the application to the Google Store. The default value for this attribute is false. Only when users verify their email addresses and login into our application will it be changed to true. This will improve the usability when the project owner needs the number of users who truly use their account to access the application.

6.3.5 Increase Usability

In order to increase the usability of this application for users. We have reduced the number of steps that users need to take, both in account creation and creating a trip plan. The efficiency of using this application has increased by simply understandable operations and as few steps as possible to operate.

When users are filling in the budget and duration it is possible that users went in wrong information by accidentally filling, thus we have implemented the calendar picker dialog and limiting the input type of budget by numbers to achieve the error of tolerance design. At the same time it provides users an insight of what to do next when using this application.User friendly design with date navigation bar in the itinerary page, especially useful when a trip has multiple dates. Users may directly find out the date that they wish to edit by using this functionality.

Efficiency is a significant component of usability. In order to prevent users from bad user experience caused by inefficient data transmission, we have rigorously designed the structure of the database to ensure efficient and synchronous transmission of data, reducing the possibility of delayed display or loss of data at the same time.

For further versions, it is also possible to increase the usability of this application by linking the user account with their Facebook or Twitter account so that users no longer have to fill out email-address and passwords to log in, which is more convenient for users to use.

6.3.6 email invitation VS dynamic links

Inviting new users by email is a very popular business mode, and this is our initial plan. Second option is generating a dynamic link which is shared from application to other social applications. The advantage of the first option is that it provides more security for the invitation and users will have a clearer understanding on what they are invited to. The advantage of the second option is that a dynamic link allows multiple users to engage simultaneously. We assumed participants invited by the trip owner are known to each other, hence the second option is more convenient for inviting other users.

6.3.7 Change Owner

In our early design phase, we did not think of this functionality.This idea is proposed during the peer review and we adopt this opinion as we think of the case that the owner has to quit this group for any reasons.If we do not implement this feature, this group has to be removed and all the already inserted details will be gone,which is not user friendly.In this way,the owner can give out his ownership of the group before leaving the group.

7 Testing

In this chapter, description and results of tests are presented. The test is formatted based on the functionalities we had identified in section 3.1. Inside each test, we have a test approach, test criterias, test schedules and its potential risks.

Besides that, based on the results, the bugs or operation flaw would be identified and fixed. The tests basically consist of the following: unit testing, integration testing and usability testing.

7.1 Test Plan

7.1.1 Approach

There are three types of strategies for testing our application are implemented and recorded: Unit testing, Integration testing, and usability testing. While the first two parts concentrate on the codes of the application which is finished by the project members since we are both the developer and testing team, the usability test is achieved by the interaction between users and the application. In the following section, three strategies are discussed and justified individually.

7.1.1.1 Unit testing

Since this application is a big project which requires three members to accomplish codes together to make up the final product. We had an agreement that individual functionality is necessary to be tested individually before merging using Github. However, bugs still happen in the final product. Hence the first test strategy is unit testing which is implemented by all three members of the team to test each functionality individually.

7.1.1.2 Integration testing

Beyond unit testing, Integration testing also focuses on the functionality of the product. However integration testing will combine different modules in this application and test as a whole to check if everything works. Hence, integration testing is implemented before the final stage of the development.

7.1.1.3 Usability testing

Usability testing is basically conducted to test how easy a design is to use by representative users. This testing is done repeatedly throughout every two weeks' peer review and finally by our potential customers.

Before seeking feedback from our users, we had briefly explained the product's abstract and asked users to accomplish certain tasks. After users finished experiencing the product, we would ask some questions which were designed based on the Heuristic evaluation of user interfaces ([Nielsen and Molich(1990)]).

The questionnaire consists of the following questions:

- **Is dialogue simple and natural?**

  Aim: Dialogue should always inform users about the system's current state.

- **Content are consistent?**

  Aim: Content should be logical and self-consistent

- **Exits are marked clearly?**

  Aim: It should be straightforward for the user to exit any page or any fragment with exit mark or suggestion.

- **Shortcuts are well designed?**

  Aim: the procedures of inputting or switching with the system should be straightward to accomplish tasks.

- **Are errors well prevented?**

  Aim: Common errors such as Flashback, lag, input error and system not responding should be avoided.

- **Error messages are well indicated?**

Aim: The users' operating errors or the operation not allowed by the system should be pointed out with gentle error messages.

7.1.2 Functionality to be tested

After we introduced the abstract and basic functionalities of this tour application to users, we asked users to accomplish certain tasks. We inquire the users' viewpoints about the importance level of each functionality and made the following table using High(H), Medium(M), and Low(L):

Functionality involved	Level of importance
Register account, Login and Logout	H
Voting system	H
Creating travel plan	H
Invitation for friends/family	H
Add multiple budgets for voting	H
Modify duration of the travel	H
Change email address for login	H
Change password	H
Using popular social media to login	M
Recommendation for travel plans	M
Feedback option	M
Like or dislike marks for the itinerary	M
Group discussion	L

Besides the functionalities testing, we also received some feedback about the usability of the application.

7.2 Test Results

7.2.1 unit tests

All functionalities are tested individually to ensure there is no systematic error to prevent users from accomplishing the Must-have and Should-have features.

One bug was spotted in the Overview fragments. After trip owners have posted several budgets for voting, it occasionally happens that the trip owner fails to vote for budgets and application flashback to the main page. After some testing for the code, we identified the problem was from the fact that there is a conflict when the overview fragments is fetching data periodically and will prevent users from editing any data. To solve this problem, we replace the fetching data periodically by only fetching data when there is data change.

7.2.2 Integration tests

Multiple people including application developers had participated in the integration tests, there was a technical adaptation issue in the discussion page. Initially, Discussion fragment fetch basic information from Overview fragment and then use the information to create a discussion room. The code in the discussion section was responsible for opening an extra activity page to support discussion. This is designed to make the code more hierarchical and easy to maintain. However, there are some unknown rules which discourage frequent switching between fragments and activities, hence sometimes the interface automatically switches without touching. To solve this problem, we remove the intent switching hierarchy and place codes inside the discussion fragment section.

7.2.3 Usability tests

Methodologies used to prepare for the usability test is explained in section 7.1.2. The participants are asked to accomplish the following tasks without any help, and any feedback during the whole process is encouraged and recorded by the project members. Test scenarios are the following:

1. Test account management of the application: including registering an account, Login and Logout.
2. Testing voting system: including budgets and duration.
3. Creating travel plan
4. Invitation for friends/family via dynamic links

5. Add multiple budgets,durations for voting
6. Modify duration,budget of the travel
7. Change email address
8. Change password
9. Fill in feedback using the application
10. Like or dislike marks for the itinerary
11. Group discussion
12. Change Owner
13. Delete trip (as owner and also as member)

Many valuable feedback were received on the system's usability. The representative feedback and solutions we could come up with are listed below:

1. **The system should speak user's language**
   Solution: Multiple languages could be supported in later versions of the application, but it is hard to realize due to time factor for now.

2. **Privacy Policy is not clearly defined.**
   Solution: Privacy Policy is associated with many law knowledges, it normally took a long time to configure. Luckily, we have similar data collection items as Wanderlog, hence the application will follow the policy rules Wanderlog suggests.

3. **The Image at Overview Page can not be modified**
   Solution: The image in the overview should be consistent with the image at the overview page.

4. **Inconvenient of Trip Deletion**
   When users wish to delete a trip, they could be properly informed by the dialogue but it is hard for the user to figure out that the dialogue can only be shown when they long click the trip plan on the home page.

Solution: possible solutions like using a delete button instead of using dialogue, but in this way, it is possible to delete the trip by accidentally operating. Another suggestion would be to create a getting start notification to teach users the way to delete a trip is by long clicking the trip info on the homepage when they get started to use this application.

5. **Guidance not enough**

   The guidance for new users is not enough.During our testing we find that most of them do not know they can change the duration and budget simply by clicking them.

   Solution: In the later development, we plan to add a guidance when the user enters the app for the first time.And also make these items more clear so that the user can know they are clickable.

6. **The email inside our database should be encrypted**

   Solution: Consider the situation that our database is hacked, our users' data will leak.This is not an action that is responsible for our users.We plan to encrypt our users' email address in our database,in which way we can use it and also they will not be obtained by others.

7. **For itinerary, we can separate groups for activities.**

   Solution: Different people have their own desired places to visit, sometimes it is hard to make an agreement.In this situation, we plan to separate them into different groups for that specific day,and for different groups they have different itineraries.

8. **The unit of budget cannot be stored**

   The budget number could be added, but the currency is not clear.

   Solution: this can be solved by adding extra text input which allows users to choose the common currency.

8.Evaluation & Reflection

In this chapter, an evaluation for the rationality of the planning and execution power will be discussed and justified. Based on the evaluation, reflection on how to complete the similar product will be given. We would try to make the reflection applicable to more people, but it is not a formal requirement for everyone to follow.

8.1 Schedule

The Schedule of the whole project is divided on a weekly basis. At the start of each week, project members will have a meeting with the project master and elaborate what they had done last week and their planning this week. There is one week for the overall planning, two weeks for design and 6 weeks for development and final week for closure of the project.

Every two week, there is a peer review session. It is a precious opportunity for the whole group to listen to the feedback from outside the project. The group members will discuss and organize ideas after each feedback session and modify the project if necessary.

Besides the deliverables of the product, there are also deadlines for the reflection component. Every deadline was recorded on Trello Board and finished days before the deadlines. Hence everyone was engaged with the whole project.

8.2 Responsibilities

Delegating tasks to group members is an important part of the project. And this is how the project begins. Everyone has their interests and expertise. Luckily we all have some experience on android development and enjoy programming. Hence the tasks are divided as the following table:

	Liang Zhang	Shikuan Li	Zihan Wang
Group Contract	✓	✓	✓
Database Scheme	✓	✓	✓
Home Page			✓
Add Plan Page		✓	✓
Me Page & Login, Logout	✓		
Backend	✓	✓	✓
Testing	✓	✓	✓
Report	✓	✓	✓

The main principle for the tasks division is based on the interests, because it is a self-driven project. Fortunately, the workload is distributed quite evenly and we basically finish all the functionalities on schedule.

8.3 Reflection

The project schedule is quite reasonable, but we underestimated the time required to implement each functionality and the difficulty to handle the conflicts when merging functionalities. Ideally, I think it is better to assign two members to plan and design the project in the first three weeks, and the other member should look into the technical details and estimate the time for each functionality. After the estimations are done, task division based on this estimation would be more reasonable for every member to follow up the deadline for each section.

Another important lesson is a technical issue. During the development, we have to search many tutorials since none of us are experts for android development. Hence we found many old version videos from 2017,2018 and etc. Most of them used android instead androidX(latest version), hence it caused many compilation errors. Therefore we should try to find the latest tutorials for any programming project.

8.4 Result

The product requirements and goals were firstly proposed by the client, and the project developers discussed the feasibility of each functionality. During development progress, We have modified the product according to the feedback from customers. Finally, the customer, project manager and members are all satisfied with the results. Besides achieving the basic goals, we always try to add extra features and improve the final product. Hopefully this project will be released on the Google Play Store.

9. Future Work

Our product is not perfect. Based on the feedback, there is much more we would like to improve and extra functionalities we want to add.

9.1 Github release

During the development, we had also found many beautiful UI templates and similar functionality such as Chating, and Preference design on Github. Hence we really hope we could release a Github version for our work and share to others who are interested. However, since this is a design project, we are not sure if this is related to CopyRight or any code security issues. But we are welcomed to share our ideas to any future developers who are interested in this project.

9.2 More Language options

Both our project master and potential users encouraged us to add more popular languages such as Spanish and Chinese. This is an interesting and very friendly functionality, we would like to add this in our later versions.

9.3 Auto Recommendation

In the future, we hope to implement the functionality that can recommend places to visit to users,not only attractions but also when is the best time to visit some countries.For example,if the user wishes to go skiing, we can recommend that you can go in November and December to Switzerland and Germany.

9.4 Transfer Money notifications

As mentioned in the beginning, we plan to implement a functionality to notify group members to transfer money to a bank account if they need it.Due to time,we are not able to finish it yet.

9.5 The Images should correspond to the trip.

The image for every trip at the overview page should correspond to the location they are going.For now we only set some predefined images,in the future we hope to make use of some api to make the picture a photo of that location.

Reference

1.LovYtrip - Android app for planning group trips.pdf
https://canvas.utwente.nl/courses/7524/files/2155022?wrap=1

2.importance of itinerary
”https://www.template.net/business/itinerary-templates/itinerary-template-in-google-doc/#:~:text=An%20itinerary%20is%20a%20layout,plan%20for%20the%20trip%20ready.

3.choosing image feature in the “Me” page used open source code and idea from below resource <RAVI TAMADA> (FEBRUARY 7, 2019) <Android choosing Image from Camera / Gallery with Crop Functionality>
https://www.androidhive.info/2019/02/android-choosing-image-camera-gallery-crop-functionality/

4.Porras, A. (2020, April 16). *Scrum Methodology for Digital Product Development*. 4Geeks Blog. https://blog.4geeks.io/scrum-for-digital-product-development/

5.*What is Usability Testing?* (2020). The Interaction Design Foundation. https://www.interaction-design.org/literature/topics/usability-testing#:%7E:text=Usability%20testing%20is%20the%20practice,development%20until%20a%20product’s%20release.

6.[Nielsen and Molich(1990)] J. Nielsen and R. Molich, \Heuristic evaluation of user interfaces," 1990, pp. 249-256.

7.K. (2019). *KODDevYouTube/ChatAppTutorial*. GitHub. https://github.com/KODDevYouTube/ChatAppTutorial

8.Coding Cafe. (2018, August 25). *Android Group chat Application using Firebase 18 Create Group Chat Activity*. YouTube. https://www.youtube.com/watch?v=4RL85tdhCEU&list=PLxefhmF0pcPmtdoud8f64EpgapkclCllj&index=20

Appendices

Appendix A: User Stories

As the admin, I need the freedom of storing and modifying data from the backend.

As a owner of a itinerary, I would like my friend to vote for my itinerary proposed by the application

As the owner,I would like to invite my friends to the group.

As the user, I want a good itinerary after I select all the filters(time,location, blabla)

As a user, I want to create my personal account and modify my password

As the owner,I would like to add activities without voting.

As a user , I want to propose my preferencecs about the trip and i t should be seen by other members. (locartion,activeity...)

As the user,I want the final desination to be determined by the sytstem when opinions disagree.

As the user,I want to propose my favorite destination at the beginning.

As the user, I want the app to notify me when a new suggestion of an activity is proposed.

As the user,I want to vote to determine whether to use the new proposal to replace the old one.

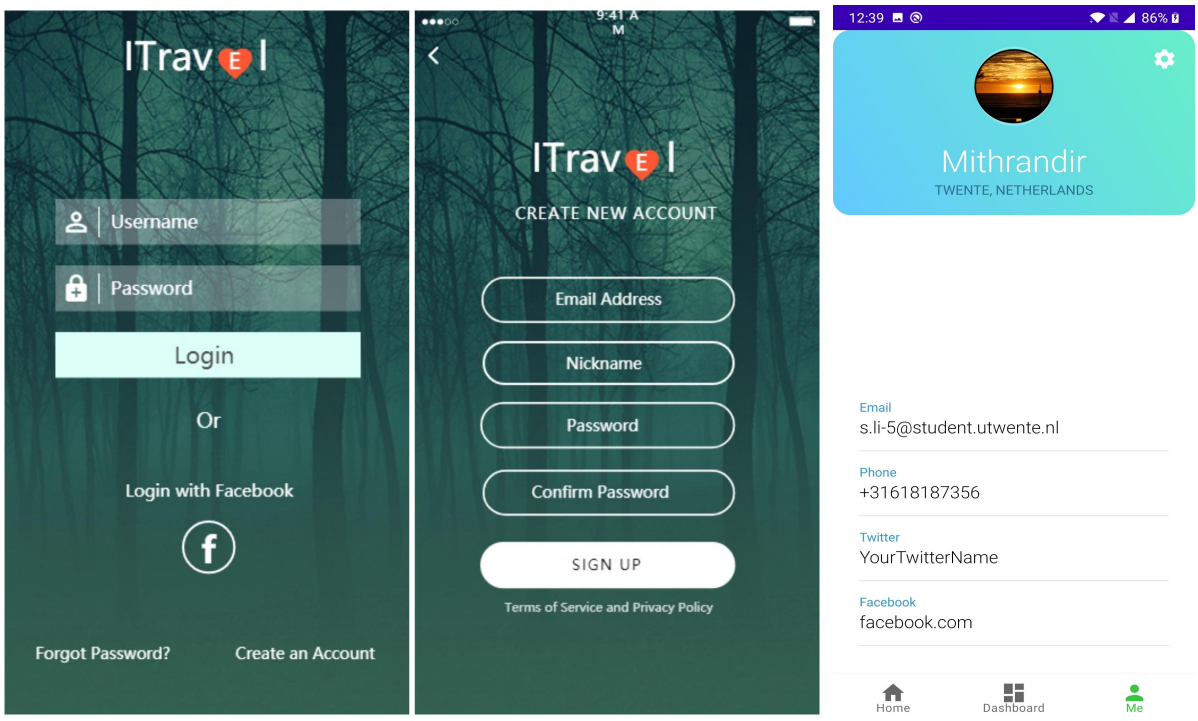
As both owner and member,we both want to propose my available time.

As a user,I want the activity to be sorted by time.

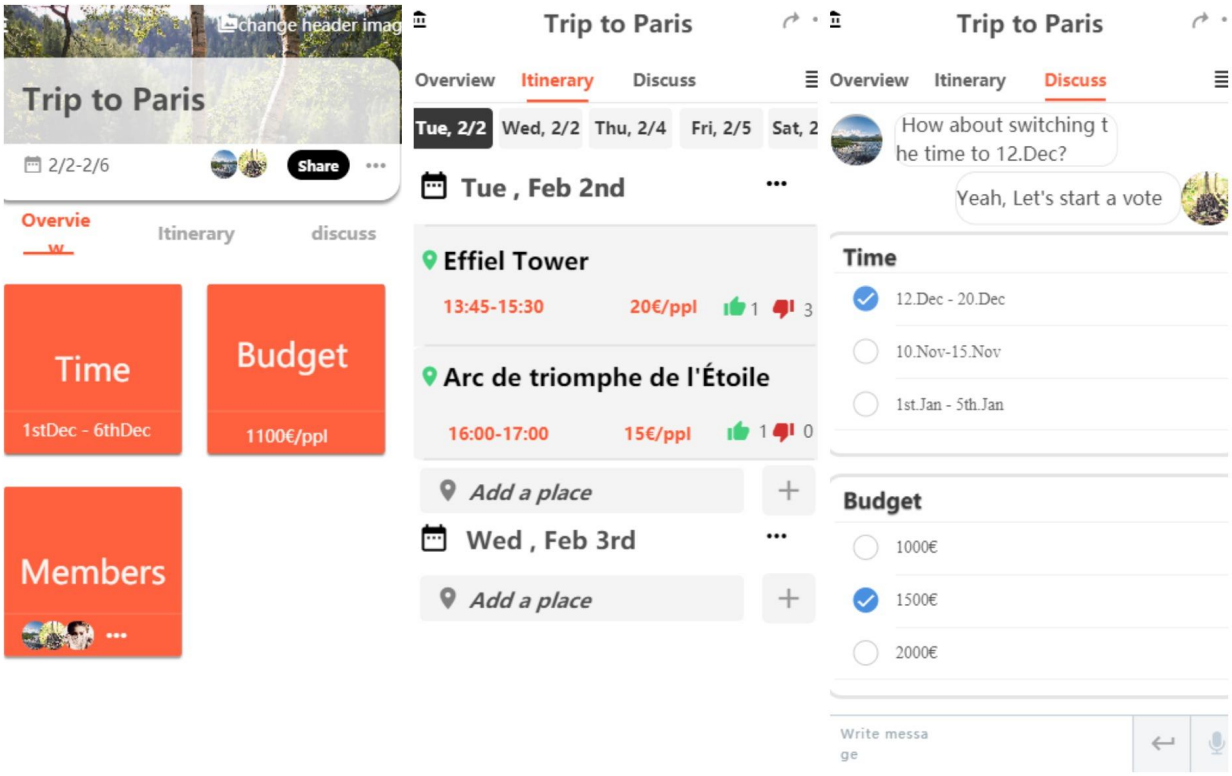
As an owner ,I want to decide which part is available for the member to vote.

Appendix B: Mock-ups

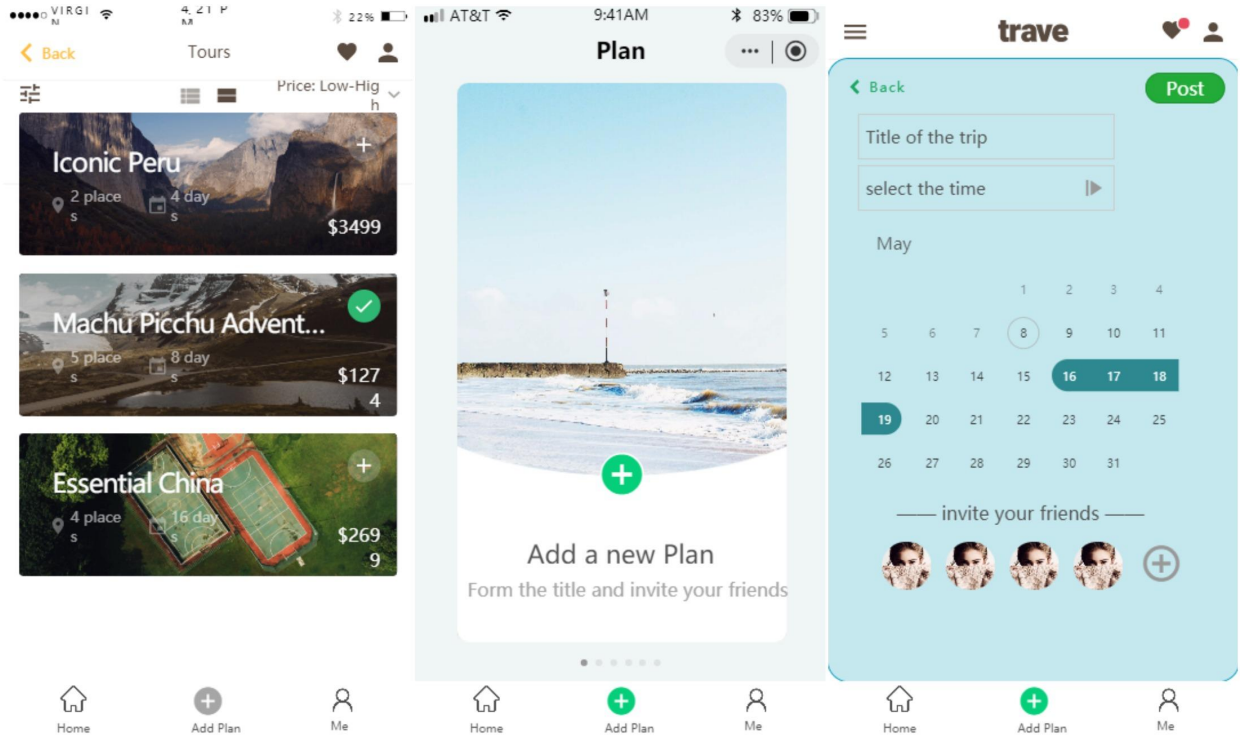
SignUp, Login and Profile Page



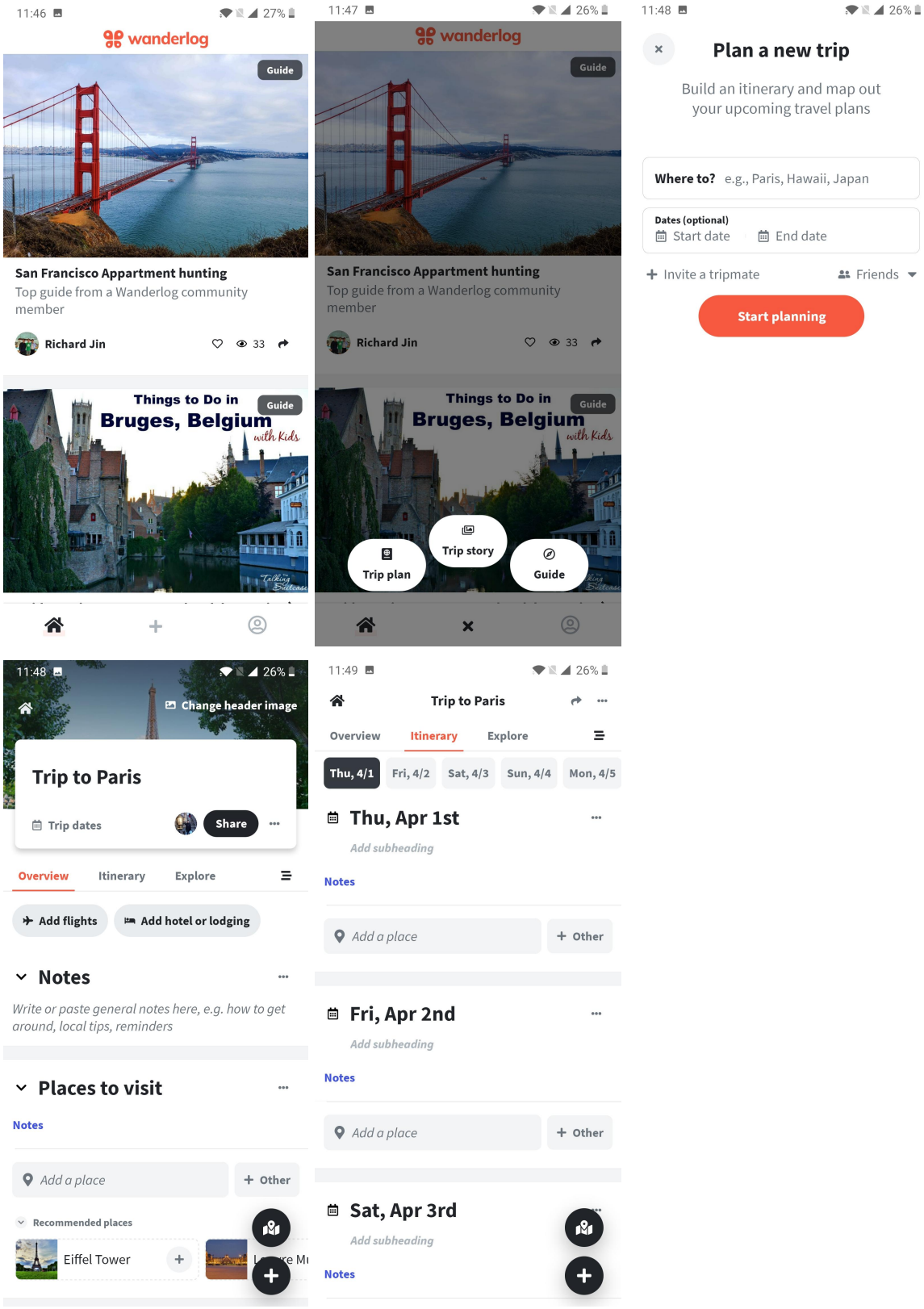
Trip page for the current trip



The mainActivity and Add Plan Page



Appendix C: UI of WanderLog

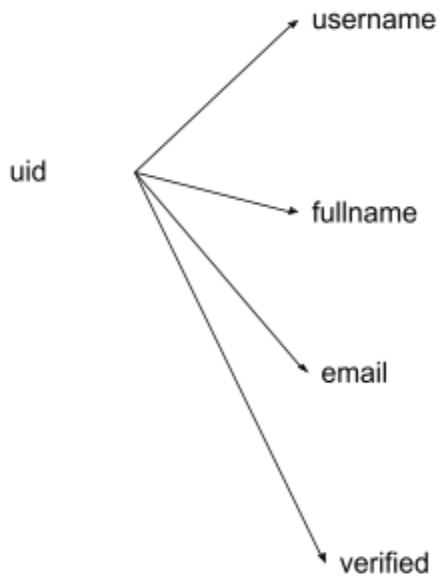


Appendix D: database Design in Firebase

Users:

Functionality:record users

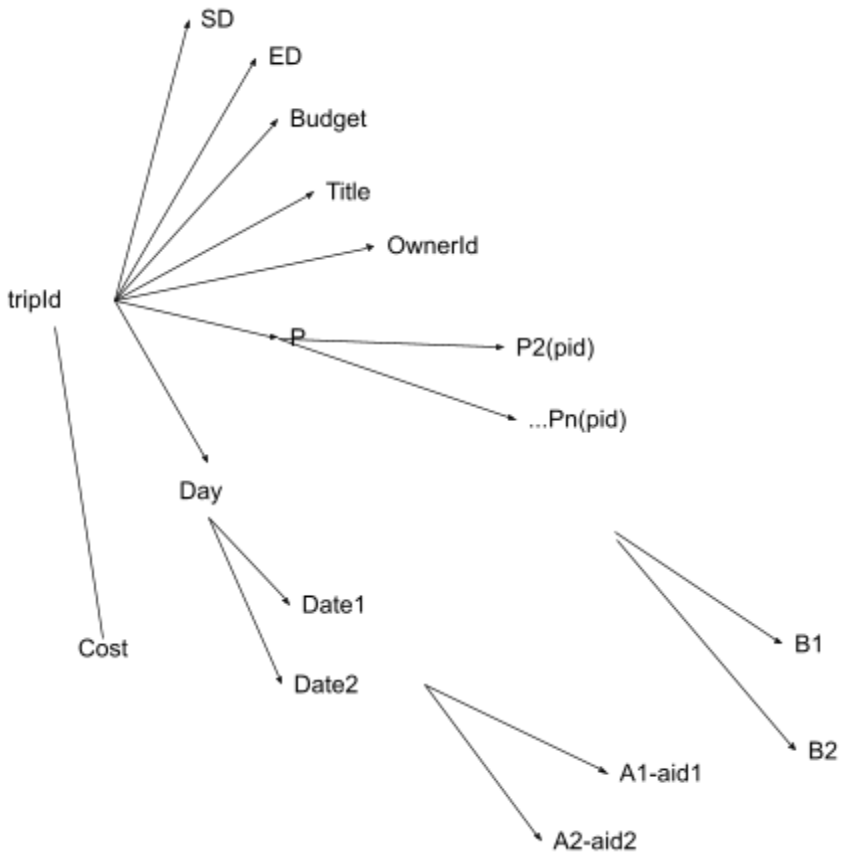
uid(Primary key)	int(automatically incremental)
fullname	text
username	Varchar(100)
password	text
email	Varchar(300)



Trips

Functionality: record all the information related to this trip

tripId(PK)(ownerId+location)	string
startDate	String
endDate	String
Budget(whole trip)	double
Title(example:Trip to Paris)	string
Activity(child)	child
Vote	boolean
ownerId	**string**
participantId--more than one	**child**



Activity:

Functionality: record activity detail inside the trip using Aid(each activity are different with different Aid even it is created by the same person with same input)

Aid(PK):tripId+title of this activity	string
Date	string
location	string
description	string
startTime	Time
endTime	Time
likes	Integer
dislikes	Integer
estimatedCost	double

GroupForVote:

WorkFlow: After the owner of a trip sends a shared link to participants,and participants click on the link, this table will be generated.
Functionality: find trip and participant by OwnerId
Participants: participants are child nodes of TripId
(participant accept invitation of trip)(owner create trip)will change this table

ownerId(uid of trip owner)	string
TripId(all trips this owner had created)--possibly more than one	string
participants(uid of participants)--more than one	string

GroupForChat

Functionality: this table is used to record the people in the chat box

ownerId(uid of owner)	string
ChatId(uid of owner+nameOfChat)	string
friends(uid of friends)--more than one	string