

TravelBuddy

Maciej Traczyk

Department of Computer Science
Aberystwyth University
Wales

September
2024

This thesis is submitted in partial fulfilment of the requirements for
the degree of Master of Science

Degree: MSc Advanced Computer Science
Module: MSC Project (CHM9360)
Supervisor: Dr Yasir Saleem Shaikh

Declaration of originality

I confirm that:

- This submission is my own work, except where clearly indicated.
- I understand that there are severe penalties for Unacceptable Academic Practice, which can lead to loss of marks or even the withholding of a degree.
- I have read the regulations on Unacceptable Academic Practice from the University's Academic Quality and Records Office (AQRO) and the relevant sections of the current Student Handbook of the Department of Computer Science.
- In submitting this work, I understand and agree to abide by the University's regulations governing these issues.

Name: Maciej Traczyk

Date: 05/09/2024

Abstract

The creation of TravelBuddy, a smartphone app meant to improve traveller experiences by offering tailored travel advice, is described in this project. Through an analysis of user preferences and related products, the research investigates how smartphones and mobile apps affect travel. The app's features, which included map integration with APIs, data storage, and authentication, were developed using the Agile approach. The idea accomplishes its goals well, providing a practical answer for contemporary travellers while considering potential enhancements for the future.

Acknowledgements

I am extremely grateful to my project supervisor, Dr Yasir Saleem Shaikh for his constant and extremely valuable help during the creation of my final work and for dedicating his time during weekly meetings during, which allowed me to receive insightful and accurate suggestions.

In addition, I would like to express my appreciation to my friends and relatives who supported me in moments of doubt which allowed me to complete my work.

Contents

Chapter 1. Introduction	9
1.1. Project Background	9
1.2. Project Aims and Objectives	9
1.2.1. Aim of the Project	9
1.2.2. Objectives of the Project	10
1.3. Motivation	10
Chapter 2. Literature review	11
2.1. Tourist Experiences in the Era of Smartphones and Mobile Applications	11
2.2. User Preferences for Mobile Travel Applications	13
2.3. Life Satisfaction and Travelling	14
2.4. Similar Products	16
Chapter 3. Tool Selection	17
3.1. Project Management and Version Control	17
3.1.1. Code Repository	17
3.1.2. Kanban Board	17
3.2. Design of the User Interface	18
3.3. Managing the Data and Authentication	18
3.3.1. Persistent Data Storage	18
3.3.2. Authentication	18
3.4. Code handling	19
3.4.1. Integrated Development Environment (IDE)	19
3.4.2. Programming Language and UI framework	19
Chapter 4. Process	19
4.1. Adopting Agile Methodology	20
4.2. Weekly Cycles	20
4.3. Writing Minutes	20
4.4. Kanban Board	21
4.5. MoSCoW Prioritization	22
Chapter 5. Design	23
5.1. Overall Architecture	23
5.2. Detailed Design	24
5.2.1. Database Design	24
5.2.2. APIs Usage	25
5.2.3. Authentication	26
5.3. User Interface Design	27

5.3.1.	Colour Scheme of TravelBuddy.....	28
5.3.3.	Authentication and Registration Screens	30
5.3.4.	Main screens	31
5.3.5.	Top App Bars	34
5.3.6.	Navigation Bar	35
5.3.7.	Bottom Sheet for Adding Data	35
5.3.8.	User Account Panel	36
5.3.10.	Other Screens.....	37
5.3.11.	Dialogs	39
5.3.12.	Snackbars (Toasts)	40
Chapter 6.	Implementation.....	41
6.1.	Database Integration	41
6.2.	Authentication handling.....	42
6.3.	APIs integrations	43
6.3.1.	Retrofit	43
6.3.2.	OSMDroid.....	43
6.4.	Friends' Requests.....	44
6.5.	Encountered Issues	44
6.5.1.	Firebase Rules	44
6.5.2.	Username Validation	45
6.5.3.	Autocompletion API	45
6.5.4.	Language State	45
6.5.5.	System Notifications.....	45
Chapter 7.	Testing	46
7.1	Manual Testing	46
7.2	User Testing	46
7.3.	Automated Testing	47
Chapter 8.	Critical Evaluation.....	48
8.1	Project Overview and Initial Objectives	48
8.2	Methodology	49
8.3.	Design and Implementation	49
8.4.	Testing, Quality Assurance and User Feedback.....	50
8.5.	Reflection on Outcomes	51
Chapter 9.	Conclusion	51
10.	References	51
11.	Appendices.....	54

Appendix A. Ethics Form	54
Appendix B: Manual Test Table.....	57
Appendix C: Users Feedback	65
Figure 1: How using the MTAs affected the experience.....	12
Figure 2: Main benefits associated with travel apps.	12
Figure 3: Influence of the MTAs on the travel experience.	12
Figure 4: Main motivations for the use of travel apps.....	12
Figure 5: If the trip has the same price where do you prefer to buy?.....	13
Figure 6: Do you use your Mobile Travel Application when you are:.....	14
Figure 7: What is the best service when using a Mobile Travel Application?	14
Figure 8: Respondents' Profile	15
Figure 9: Results of Confirmatory Factor Analysis (CFA).	16
Figure 10: Example of weekly minutes.	21
Figure 11: Part of Kanban Board in GitHub repository.	21
Figure 12: MoSCoW prioritisation file.	22
Figure 13: Example screens of the design of TravelBuddy in Figma.....	28
Figure 14: Dark Scheme table from Material Theme Builder.	29
Figure 15: Light Scheme table from Material Theme Builder.....	29
Figure 16: Icon of TravelBuddy[41].....	30
Figure 17: Password reset dialog.	30
Figure 18: Log in screen.	30
Figure 19: Verification dialog.	31
Figure 20: Sign up screen.	31
Figure 21: My Trips screen.....	32
Figure 22: Dialog for trip plan adding.	33
Figure 23: Explore screen with fetched data.	33
Figure 24: Explore screen with no data fetched.	33
Figure 25: Tab "Phrases" on Social screen.....	34
Figure 26: Tab "Trips" on Social screen.....	34
Figure 27: Main top app bar.	34
Figure 28: App bar with arrow back.....	35
Figure 29: Navigation bar.....	35
Figure 30: Bottom sheet.	35
Figure 31: About screen.....	36
Figure 32: Profile screen.	36
Figure 33: Account Panel.	36
Figure 34: Empty notification centre.	37
Figure 35: Notification centre with the notification.	37
Figure 36: Map view screen.	38
Figure 37: Checklist screen.	38
Figure 38: Trip details screen.....	38
Figure 39: Add a friend screen.....	39
Figure 40: Add a phrase screen.....	39
Figure 41: Add a trip screen.....	39
Figure 42: Confirm all data removal screen.....	39

Figure 43: Confirm removal dialog.	39
Figure 44: Theme selection dialog.	40
Figure 45: Verification dialog.	40
Figure 46: Examples of Toasts.....	41
Figure 47: Unit tests results.	48

Chapter 1. Introduction

1.1. Project Background

Travelling has become a common and significantly popular activity all over the world. People, by discovering unknown corners of the world, experience new sensations and pleasures, but also learn numerous useful things. A person who is travelling or planning to travel should, first of all, demonstrate resourcefulness and self-discipline to organise a detailed trip plan, and what one should remember before setting off on a journey. In addition to strengthening soft skills, a traveller learns a lot about new, foreign cultures and customs, which is a great way to learn acceptance and appreciation through experiencing and observing the lives of other people[1]. At the same time, travellers learn more practical skills like speaking in foreign languages and make new acquaintances that complement the idea of travelling. It also has a positive psychological dimension because, as studies show, it also improves mood and reduces the risk of suffering from depression[2].

However, for some people, visiting new places can be associated with a stressful and difficult atmosphere before and during the trip. This may be related to the situations from the past, negative feelings from previous expeditions or significant events taking place in the world, such as armed conflicts or pandemics. Some of the best and proven techniques that will allow a fearful person to overcome their fear of travelling include thoroughly familiarising themselves with the characteristics of the destination and carefully planning the trip, which can be a good way to visualise the future trip[3]. However, regardless of whether a person is worried about travelling or not, the mentioned techniques are good practices that will not only allow the trip to go in a smoother manner but also help avoid disappointment or dissatisfaction resulting from false expectations regarding the destination. Good planning and research also potentially allow saving money by searching for the types of attractions and accommodations that will best fit the specific financial situation of the traveller.

There are many different ways to accurately perform these procedures. Due to the wide availability of smartphones, having a mobile application that will facilitate activities related to planning and actually going for a trip is both a convenient and useful solution. At the user's fingertips, they can have the most important functions strictly related to travelling to assist during the expedition. For this purpose, "TravelBuddy" was created. It is a mobile application for the Android platform[4], which, due to its extensive functions, is able to assist travellers and aid in activities related to managing and planning their trips.

1.2. Project Aims and Objectives

1.2.1. Aim of the Project

The main goal of this project is to create a mobile application for the Android operating system that will let the user enter and display trip-related information. This will assist in simplifying the process of organising and getting ready for the trip. In this way, it will facilitate the process of planning and preparing for the process of travelling. It should also be able to search for places based on the user's desired location and category so that the user can draw ideas for a trip plan and discover new places to visit. Moreover, it must also allow for adding other users of the application as friends so that users can share their travel plans and also post useful phrases from foreign languages. This app has to incorporate a login system due to the core value of this functionality, and it must be able to store data

in the cloud so that users can access it on any compatible Android device by logging in. To create an account and log in to the application, a network connection will be required.

1.2.2. Objectives of the Project

In order to achieve the intended aim of the project, it is necessary to specify the functional requirements that will determine the most important elements that need be included in the application.

- Configuration of a service enabling management of login data and secure user authentication to their account.
- Setting up a cloud-based database to enable real-time access to user data after logging into the application installed on a compatible device.
- Integration of API to request information about places, addresses, images, and maps.
- Designing an intuitive process for adding "trips" to make it easier for the user to enter and manage data.
- Displaying places on a list after specifying the location and category to browse potential places of interest.
- Sending notifications for notifying the user
- Using a map to search for places, visualize the trip plan and display the location of places.
- Designing a "friends" system that will allow adding other users registered in the application to the friends list.
- Sharing trip plans and phrases with previously added friends in the application and interacting with them by leaving likes and comments.

Fulfilment of the above-mentioned functional requirements will ensure the practicality of the application in terms of assisting the traveller, in order to facilitate the organising and planning of trips and related activities.

1.3. Motivation

The motivation that led to the selection of the described project consists of several factors. The first is that travelling has become an everyday occurrence for many people for recreational, business, or educational reasons. Year by year, a growing trend can be observed in the number of tourists in the world [5]. This is all due to the wide variety of vacation packages that also offer budget options, which makes them affordable for more people. Many people are looking for a dedicated tool that will allow them to collect data on their trips and more.

This is combined with another factor, which is the creation of a mobile application[6], i.e. a software program that can be installed on a smartphone. Due to the huge popularity of smartphones in the world, this platform is the most attractive to users, and specifically the Android system, which operates at a level of slightly over 70%[7], which gives a big advantage over the competitors. For several years, a growing interest in smartphones has been observed. Smartphones have become an almost integral part of everyday life for the whole world, offering applications for communication, entertainment and work.

Having applications for the activities needed by a person is a convenient way to manage them from anywhere on earth where the Internet is available. It can be concluded that mobile development is the future that will accompany people for many years to come.

Chapter 2. Literature review

The literature review explores various aspects related to travel apps, aiming to understand how they affect users' experiences and gather their opinions on this impact. It also includes a brief analysis of similar products and examines various influences that travelling has on individuals.

2.1. Tourist Experiences in the Era of Smartphones and Mobile Applications

In today's world, people are very attached to technology. Almost every person has at least a few smart devices. The most popular type of device in the world is a smartphone[8]. It allows for consumption of content and connection to the outside world while maintaining a pocket-sized dimension. In order to increase the range of smartphone capabilities, mobile applications are created to facilitate and simplify various types of operations through their user-friendly interface. Mobile app development has become a daily occurrence as every brand wants to optimise the user experience by designing their app in a way that most of the potential problems and issues are avoided. In the tourism sector, there are also numerous demands to facilitate many aspects of planning trips for tourists.

In the article by Sonia Dias et al.[9], a study, conducted between August and September in 2018, was described. The methodology consisted of conducting an online survey in which participants had to answer questions on travel apps that concerned four issues, namely: knowledge and use, motivations for use, moments of use and degree of satisfaction, impact on experience and value to

obtain their point of view. One hundred and ten valid responses were obtained from Portuguese residents who used mobile technologies and also desired travel apps.

Table 3: Influence of the MTAs on the travel experience.

Influence	Percentage
Has not changed	0.9%
Has changed little	4.5%
It had no impact	22.5%
Amended	41.4%
It has changed a lot	14.4%
Does not use MTAs	16.2%

Figure 3: Influence of the MTAs on the travel experience.

Table 1: Main motivations for the use of travel apps.

Motivations	Percentage
Performance	28%
Satisfaction	23%
Utility/perceived value	21%
Habit	10%
Price	7%
Rating of the app	6%
Management expectations	3%
Social value (everyone I know uses)	2%
Emotional value (feelings)	0%

Figure 4: Main motivations for the use of travel apps.

Table 2: Main benefits associated with travel apps.

Benefits	Percentage
Ubiquity	23%
Immediate availability	20%
Access to information	20%
Convenience	15%
Organisation/capacity planning	12%
Price	6%
Pragmatism	3%
Personalisation	1%
Innovation	0%
Entertainment	0%
Security	0%

Figure 2: Main benefits associated with travel apps.

Table 4: How it affected the experience.

Influence	Percentage
Planning during trip	27%
More convenience	18%
Greater security	11%
Greater connectivity	11%
Greater productivity	9%
Greater quality	9%
Less previous planning	6%
Sharing experiences during travel	5%
More fun	4%

Figure 1: How using the MTAs affected the experience.

After receiving the results, the following conclusions could be drawn:

- The main motivations for using travel apps were convenience, effectiveness, and productivity
- The main benefits of using this type of app were ubiquity, access to information and convenience
- Travel apps had an impact on their travel experience.

To sum up the final results, the study showed that mobile apps for travel and related purposes change the user's experience while travelling, by offering tools that facilitate connection and access to information, which also increased the sense of security of travellers. Considering the continuous development and evolution of mobile and smartphone apps, there is a potential for even better travel experiences that allow travellers to manage their trips, however, there is a continuous need for research on this topic.

2.2. User Preferences for Mobile Travel Applications

The mobile app market is very extensive. In the Google Play store alone, over two million different types of apps can be found (as of December 2023)[10]. They are all divided into a variety of categories. Apps that fall into the "Travel" category may have different functionalities and be responsible for various activities related to travelling. For example, there are apps for booking accommodation, booking flights, renting cars, displaying maps, and helping with organizing trips that are included.

Laura Parro's thesis[11] analysed the impact of technology on the tourism industry. A significant part of its focus is concentrated on smartphones, as devices for operating the Internet on them and applications that play different roles at different stages of the trip. The study used a questionnaire consisting of nineteen questions. The number of valid answers varied depending on the question.

Question number eleven, which also asked about travel apps, referred to booking trips when the price for it would be the same regardless of the form of booking. In this category, travel apps received the lowest score, while the Internet in general received the highest score. The poor results of mobile apps in this aspect may be caused by the lack of trust in individual apps due to concerns about the validity of the purchase. Official websites or other trusted services seem to have a better reputation, which attracts more potential customers.

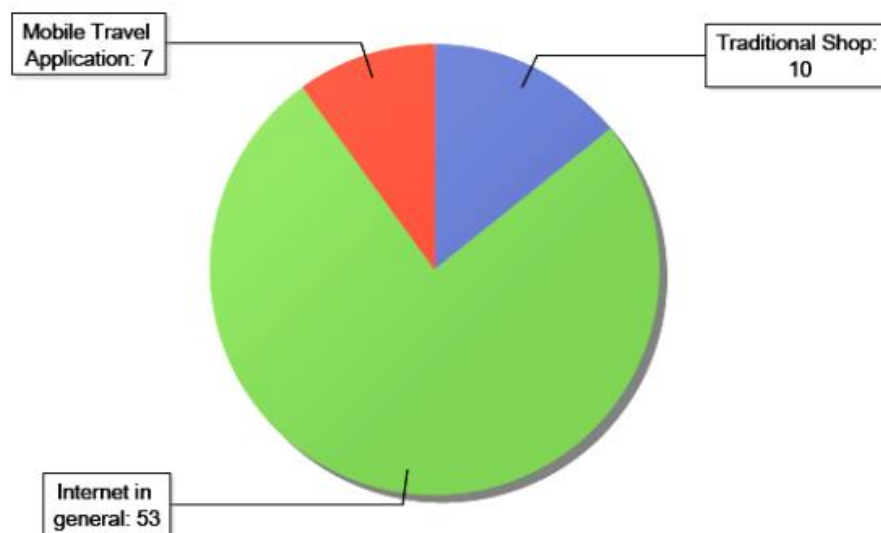


Figure 5: If the trip has the same price where do you prefer to buy?

Question number fourteen concerned the selection of time stages of the trip when using travel apps. The first and second most frequently selected issues were the planning stage of the trip and during the trip. This result suggests that users of these apps value them the most before and during the trip. This may be due to the help in organising and managing the trip and also assistance during it.

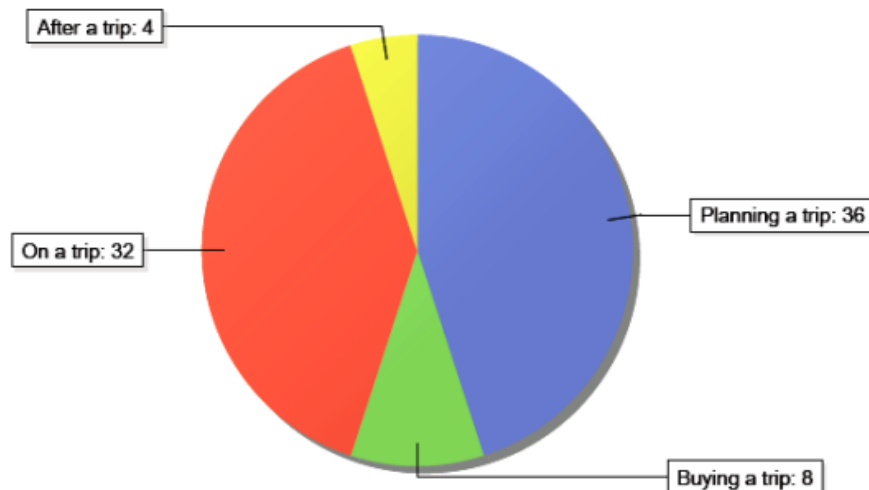


Figure 6: Do you use your Mobile Travel Application when you are:

Question number seventeen concerned the functionality of the app during its use. The first two answers with the highest score were "Destination map" and "General information". These are the most valued functions by users of travel apps. Users value the simplicity of the interface and the ease of use. They also pay attention to the operability of the app in offline mode so that in the event of a lack of network connection, they still have at least the basic functionality of the app that they can rely on in such situations.

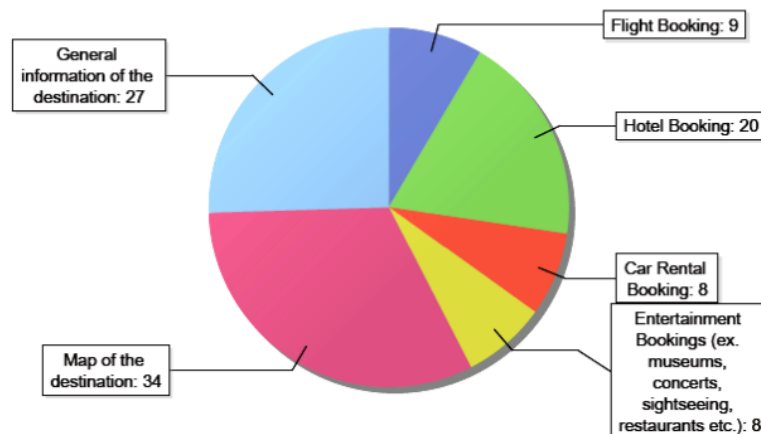


Figure 7: What is the best service when using a Mobile Travel Application?

The study contains useful results that can potentially guide mobile application developers to increase the popularity and trust of users and strive to create an ideal product that will have all the desired functions, work in offline mode, and ensure the security of data and transactions.

2.3. Life Satisfaction and Travelling

Tourism is a highly popular activity among people nowadays. Going on a trip is associated with relaxation during free time. Studies show that there are positive effects on a person during a trip in terms of stress relief and mental health.

In a study conducted by Chun-Chu Chen et al.[12], the focus was to find out whether more frequent travel would have an impact on the greater life satisfaction of a traveller. The study involved collecting responses from respondents from Taiwan through an online survey. Five hundred results were collected. Participants were first asked to fill in information about their gender, age, education, monthly income, and the number of pleasant trips in the last twelve months.

Variables	n (%)
Gender	
Male	250 (50.0%)
Female	250 (50.0%)
Age	
18–24 years	84 (16.8%)
25–34 years	138 (27.6%)
35–44 years	139 (27.8%)
45–54 years	139 (27.8%)
Education	
High school	130 (26.0%)
College	266 (53.2%)
Graduate school	104 (20.8%)
Monthly income	
Under NT\$19,999 (<US\$660)	122 (24.4%)
NT\$20,000–49,999 (US\$660–1,650)	250 (50.0%)
NT\$50,000–69,999 (US\$1,650–2,300)	67 (13.4%)
NT\$70,000–99,999 (US\$2,300–3,295)	37 (7.4%)
More than NT\$100,000 (>US\$3,295)	24 (4.8%)
Pleasure trips in the past 12 months	
Zero	34 (6.8%)
One trip	50 (10.0%)
Two trips	82 (16.4%)
Three trips	72 (14.4%)
Four trips	36 (7.2%)
Five trips	64 (12.8%)
Six trips or more	162 (32.4%)

Figure 8: Respondents' Profile

Then, they were asked to fill in questions about their experiences of travelling. The first section of questions was about the importance of tourism in their lives in order to determine the personal significance of travel for the respondents. The next set of questions was about the attention to information that was related to their trips. This was to determine their interest in their trip. The next set of questions asked whether and how often they talked or discussed their travel plans with other people in order to determine the social aspect of this issue. The fourth set of questions asked about the actual amount of travel in a given period to then compare it with the last questions about the respondents' life satisfaction.

Factors/Items	Factor Loadings	Error Variances	CR	AVE
Travel importance			0.94	0.85
1. How important is travel/tourism to your life?	0.91	0.18		
2. How important is travel/tourism to you relative to other issues in your life?	0.92	0.16		
3. How much do you personally care about travel/tourism?	0.94	0.12		
Attention to information about future travels			0.95	0.85
1. How much attention do you generally pay to information you came across regarding future travels?	0.92	0.16		
2. How much attention do you pay to future travels relative to other issues?	0.95	0.11		
3. How much attention do you pay to news articles and televised new stories about future travels?	0.91	0.18		
Frequency of discussion about future travels			0.94	0.85
1. How frequently do you discuss potential vacations with other people?	0.93	0.13		
2. How often do potential vacations come up in your conversations with others?	0.95	0.10		
3. How much time do you spend talking about potential vacations relative to other issues?	0.88	0.22		
Frequency of travel			0.89	0.73
1. Total number of pleasure trips	0.80	0.36		
2. Number of pleasure trips that were more than 75 miles	0.92	0.16		
3. Number of pleasure trips that were overnight	0.84	0.29		
Life satisfaction			0.93	0.74
1. In most ways my life was close to my ideal.	0.87	0.24		
2. The conditions of my life were excellent.	0.93	0.14		
3. I was satisfied with my life.	0.94	0.12		
4. I felt I had the important things I wanted in life.	0.82	0.33		
5. If I could have lived my life over, I would change almost nothing.	0.72	0.48		

Figure 9: Results of Confirmatory Factor Analysis (CFA).

After obtaining the results, several issues were established. People who actively sought information about travel and discussed their travel plans were much more likely to initiate actions aimed at realising a potential trip. The study also confirmed a (relatively small due to many factors responsible for general well-being) positive relationship between the frequency (and satisfaction) of travel and general life satisfaction as people who travelled more often in their lives also reported higher levels of life satisfaction. The researchers emphasise the need for future studies to obtain more precise data. For this reason, travelling should be promoted not only as a recreational experience, but also as an experience that has a real impact on the tourist in aspects that affect their entire life.

2.4. Similar Products

There are several well-known applications on the mobile app market whose main task is to manage trips and discover new travel suggestions. Below, there is a short analysis of three highly rated travel planners with a large number of downloads on Google Play Store.

a) Wanderlog - It is an application focused on helping users organise their trips by creating itineraries. It has over one million downloads, with an average rating of 4.7/ 5 (Google Play Store) [13].

b) Polarsteps - An app that helps to organise and document the user's travel. It has over five million downloads, with an average rating of 4.8/ 5 (Google Play Store) [14].

c) Triplt - An application for managing a user's itinerary by consolidating information from emails. It has over five million downloads and an average rating of 4.5/ 5 (Google Play Store) [15].

Managing itineraries, visualising trips on maps, and sharing information with others are some of the features that are the most common to all of these applications. In addition, they also offer numerous other features that are impressive but may be considered as too overwhelming and distracting for some users who value clearer and more intuitive user interfaces with key functionalities exposed. Too many functions on the screen can significantly complicate navigation in the application, which can lead to user discouragement and eventually to complete abandonment of the application. The complexity of these applications can be problematic, especially for people who are less familiar with

technology, as they may have issues with navigating the application and accessing the desired functions.

The creator of TravelBuddy designed the application with an emphasis on providing the most important, key functionalities that are available in an intuitive interface. This choice is caused by the fact that travellers value the ease of use that will accompany them throughout the entire cycle of using the application.

Chapter 3. Tool Selection

The selection of tools that were used during the project creation process took place after defining the goal and assumptions that should be achieved by executing the project. It was decided that the project would be based mainly on creating software called a mobile application for the Android operating system. Therefore, it was necessary to select tools that would help in the process of creating a mobile application at the stages of planning and management, visual design, managing data, and writing and testing the code. The selection of tools was preceded by research on practices that are used when working with this type of software. This was aimed at selecting the most appropriate tools in terms of work efficiency but also selecting tools that would meet the personal preferences of the project creator.

3.1. Project Management and Version Control

3.1.1. Code Repository

The GitHub[16] repository is a virtual space where user-selected files related to the project are stored. These files can include source code, documentation, and support files. It also serves as a kind of backup for storing important files. In addition, the repository allows many useful operations that make working with the code easier in many respects. The most important of them is enabling version control. This facilitates users to view versions of the code that have been previously uploaded to the repository using commits, i.e. identifiers that contain changes to the previous version and optional messages describing the changes made. Authorised users can easily locate changes in the code relative to other versions as they are visually compared on the website or command line console. A useful function in repositories is the creation of branches. This allows working on new features while not affecting the main (default) branch, which prevents the code from mixing with the tested product. In the event of the code being unavailable locally on the device due to unforeseen circumstances, the code can be restored to the device in the version that the user has selected. As long as the code is in the repository, it is available to authorised users.

3.1.2. Kanban Board

GitHub additionally offers tools that do not directly deal with code. One of them allows the user to create a board similar to a Kanban board. This tool is used to visualise the work that is planned for a specific date and to manage the workload. The board allows the user to divide it into the number of columns with according to names that suit the user's plan of work. Columns usually represent the status of the workflow. In the columns, there are elements that symbolise a task (ticket), which can be moved between columns to reflect the current state of progress. This is a great tool for people who follow agile methodologies as it visualises tracking sprints and the overall progress of the project and helps with project management.

3.2. Design of the User Interface

Before the beginning of the implementation process of the code responsible for the interface components that the user interacts with while using the application began, a prototype of the application interface was created using a tool called Figma[17]. This allows the production of high-quality interface design due to the many options that this tool offers. The creator can use built-in shapes to best reflect the credibility of the final product by cropping, resizing, colouring and many others. However, there are also official design packs that contain components that are used by real applications/interfaces. In the case of the interface design of this project, the "Material 3 Design Kit" [18] was used. It contains official components that are utilised in the Material Design 3 design framework to create mobile and online applications. Figma allows support for many plugins that help in the best possible reproduction of the design. For this purpose, the "Material Theme Builder"[19] plugin was used, a tool gives the appropriate colour palette to the application components based on the dominant colour of the application selected by the user, granting the application a pleasant look and feel. The last plugin that optimises work is a plugin called "Material Design Icons"[20]. It has a huge collection of icons that are partially supported by Jetpack Compose. This is another tool that gives the design prototype a feeling very close to the appearance of the actual application.

3.3. Managing the Data and Authentication

3.3.1. Persistent Data Storage

TravelBuddy mobile application's database was chosen as a cloud-hosted NoSQL database called Cloud Firestore[21] from Firebase. This solution offers a lot of useful features that work on numerous platforms. Key-value pairs are used to store data in collections of documents. Since the document structure is not required to be predefined, the data model allows a greater flexibility in data collection while developing new application capabilities. Real-time data synchronisation with the database grants nearly instantaneous response for all clients that are connected. Another beneficial feature of this solution is supporting offline mode, i.e. when the application does not have access to the Internet. In this case, the data is cached on the user's device, which allows viewing the data after the last synchronisation with the database, and therefore for immediate synchronisation of local changes with those in the cloud after restoring the network connection. This ensures continuous consistency. All data is available for management in the Firebase console, which also allows editing security rules. This facilitates specifying which users can read previously designated data and which can write it. Firestore integrates with Firebase Authentication, a project-specific login data management system that allows for additional data restrictions based on user authentication.

3.3.2. Authentication

An authentication system for users is part of the application. The service that manages this functionality is also provided by Firebase, called Firebase Authentication[22]. It enables to implement a secure and user-friendly SDK (Software Development Kit) in the program of the creator's choice. The system can authenticate users by utilising a variety of ways, including conventional methods such as using a phone number, integration with well-known social media sites, and email and password. It also allows the user to call the method that is responsible for sending an email with a request to confirm the login address to the address that the user used in the user registration process in the system. If the email address is not validated, the user will not be able to access the account. The ability to reset a user's password when necessary is another important feature. The email address that the user provides is used by Firebase to send an email containing password reset instructions. The

Firebase console also enables managing accounts. Firebase Authentication ensures the safe handling and protection of user credentials from common threats, in accordance with the best practices of this type of service.

3.4. Code handling

3.4.1. Integrated Development Environment (IDE)

The programming environment is the basic tool for software development. In the case of creating applications for the Android system, one IDE stands out significantly from the others. Android Studio (version: Koala | 2024.1.1)[22] is the official IDE for developing applications for Android, which is a recommendation made by Google. It is based on IntelliJ IDEA, a very popular Java IDE, but it is prepared specifically for development dedicated to Android. The code editor offered in Android Studio is highly advanced because it has all the desired functions that facilitate and accelerate work with the code, such as syntax highlighting, code refactoring or real-time error detection. The structure of the project is organised in a transparent way, which allows for easy navigation through files. Additionally, Android Studio offers the creation of a virtual device called an emulator, which will be used to run and test applications directly. It can be chosen from a range of devices to test applications on different screen sizes and hardware configurations. The described IDE also offers a possibility that enables the project to be connected to the Firebase platform while also offering instructions for implementing the code of Firebase services. Android Studio is a complete tool that contains everything needed to create mobile applications, and continuous updates expand the functionality of this environment.

3.4.2. Programming Language and UI framework

The main programming language of this project was Kotlin[24]. It is a modern programming language that was developed by JetBrains[25] (a company offering a wide range of IDEs) since it has great support and introduction of the latest innovations. It is fully interoperable with the Java language. Kotlin is a language widely used to create applications for the Android system. As a modern language, it reduces boilerplate code through modern functions. The powerful tools it offers are coroutines, functions that can suspend and resume execution at a different time. This is especially useful when performing many asynchronous operations[26].

When creating applications for the Android system, Kotlin is complemented by a framework that is based on this language. It is called Jetpack Compose[27] and was developed by Google. This toolkit simplifies and speeds up the creation of the user interface through a declarative approach. This involves describing the structure of the component and its styling with themes, typography and shapes, and then programming the function that it will perform in the user interface. Built-in support for Material Design components helps the creator design interfaces according to Google Material Design guidelines[28]. Using Kotlin with Jetpack Compose guarantees programmers to achieve more with less code. As a result, the code is clean, readable, and maintainable.

Chapter 4. Process

This section presents the development of methodology of this project and describes all the activities that made up the entire process.

4.1. Adopting Agile Methodology

At the very beginning of the project, it was decided that the agile methodology would be adopted to ensure a smooth development process. This is a particularly effective approach when working on software where the requirements may change during the process, which is problematic when an appropriate strategy for active response to changes has not been adopted in advance. Agile methodology (Scrum)[29] allows for flexibility and quick adaptation to evolving project needs, minimising the risks associated with unexpected shifts in direction. This was due to several key factors.

- a) Iterative development - all work was divided into cycles (Sprints), which allowed for frequent revisions and application of functionality improvements.
- b) Continuous feedback - supervisor feedback was integrated into each Sprint, allowing for weekly product refinements.
- c) Transparency - regular documentation of the entire process and decisions made during development

By enforcing agile principles, it was possible to complete the work on the project without major problems in the originally planned time.

4.2. Weekly Cycles

Each cycle called Sprint began with presenting the progress that took place during the previous week to the project supervisor during in-person meetings in Supervisor's office and online meetings via Microsoft Teams. It consisted of presenting new functionalities in the application on the emulator in Android Studio. The project supervisor gave feedback for each functionality, which allowed for improving the application towards a product that meets the intended goals. Then, using minutes, the state of completion of existing tasks was determined and new tasks were created for the beginning sprint. After identifying the tasks, the obstacles encountered during the implementation of the functionality were discussed and potential solutions were considered to prevent downtime in the project progress. After each meeting, the recorded minutes were uploaded to the git repository to collect meeting documentation, and the Kanban board was updated to reflect the project completion status.

4.3. Writing Minutes

Minutes from each weekly meeting have been written and subsequently submitted to the GitHub repository in order to track the project's and the meetings' advancement. Noting details about the meeting, including information on participants, time and place of the meeting, who wrote it, and its version, was how the document started. These details determined the convenience of referring to the paper. Work that had been presented at earlier sessions was included in the "Matters Arising" section. Descriptions of tasks helped to identify possible delays by outlining what needed to be done and how the job was advancing. There were tasks under the "Other tasks" section as well, although these were often the less urgent, secondary kind. The new tasks that were discovered during the day were mentioned in the "New Business" section. The effectiveness of work was greatly impacted by following the minutes-specified duties. By organising the work in stages, they were able to prevent downtime.

```

Project: MSc Project
Meeting: Weekly Project Meeting
People present: Maciej Traczyk [mat78] and Yasir Saleem [yssi]
Place and date of meeting: MS Teams, Thursday, 20th June 2024
Circulation list: mat78 and yssi
Author: Yasir Saleem [yssi]
Date of minutes: Thursday, 20th June 2024
Version: 1.0

Matters arising
=====
1. Finish UI prototype development using Figma (https://www.figma.com/design/42eGAI5qXsYGTWkXsZInE/Travel-App-UI-Design?node-id=0-1&t=rc7gdPP9sGMkrScf-0).
Status: Done

2. Identify "will not have" functional requirements
ACTION: mat78
Status: Done

3. Add multi language support in 'could have' MoSCoW requirements
ACTION: mat78
Status: Done

4. Invite Yasir's personal email (shaikh.yasir.saleem@gmail.com) on GitHub repository.
ACTION: mat78
Status: Done

Other Tasks
=====
1. Set up your project on Android Studio and add into Git repository.
ACTION: mat78
Status: Done

2. Generate a theme for Android Studio that matches with Figma theme.
ACTION: mat78
Status: Done

3. Created an icon for the app
ACTION: mat78
Status: Done

New business
=====
1. Develop the skeleton of the app (simple components)
ACTION: mat78

2. Implement and test firebase functionality
ACTION: mat78

AOB
===
None

```

Figure 10: Example of weekly minutes.

4.4. Kanban Board

The Kanban board[30] is a tool for visualising the progress of the project. It was located on GitHub and contained three columns containing tasks in the form of cards. They could be dragged to other columns so that each of them accurately reflected the status of the task completion. The first column was called "To Do" and collected tasks that had already been identified but had not yet been started, and their presence here usually meant that they were prioritised for the next work session. The second column called "In Progress" contained tasks that were currently being worked on. Then the cards went to the last column called "Done".



Figure 11: Part of Kanban Board in GitHub repository.

4.5. MoSCoW Prioritization

MoSCoW[31] requirements model was created during the first sprint. This technique was used to categorise the application features based on their importance and necessity. The document has four sections. The first one called “Must-Have” contains features that are essential for the application, i.e. those that must be guaranteed as minimum user requirements. The second section called “Should-Have” also contains important features, but not those without which the application could not function. This is a place for features that will enhance the user experience by adding significant value. The next category called “Could-Have” contains features that are desirable but not necessary. Their implementation is time and resource dependent. They will not compromise the application in any way if they are not incorporated. The last category “Won’t-Have” collects features that are consciously not included in the project in the current scope and time. The purpose of identifying this section is to avoid less important features at the current state of the app’s development. Introducing this analysis at the beginning of this project allowed imposing priorities in terms of implementing functionality in the application.

```
1. MUST-HAVE
a) Authentication to allow users to create and manage accounts for a tailored experience.
b) Cloud database to store user's information and entries.
c) API integration for displaying maps, weather forecasts, and destination information.
d) Offline access to previously created itineraries.
e) Creation and management of personalized travel itineraries (choosing destination,
   accommodation and transport details, and specifying tasks before the journey).

2. SHOULD-HAVE
a) Planning their trips by searching and adding desired places based user preferences and interests.
b) Adding different types of media, reviews, and notes to create trips and visited places.
c) Visualizing trips on an interactive map.
d) Sharing travel journals with friends who can interact with them by leaving reviews and comments.
e) Collaborative feature that allows app users to add commonly used useful phrases for
   different countries. Plus having a feature to like and comment by other users.

3. COULD-HAVE
a) Reminding users about previously added checklists/to-do lists a few days before a trip.
b) Possibility to change a language of the app (English and Polish).
c) Proposing recommended destinations, attractions, accommodations, and activities near
   the user's destination.
d) Dedicated section with contact details for emergency personnel, hospital information,
   and local embassy and travel advisories.

4. WON'T-HAVE
a) IOS and web version of the app.
b) The app will not be deployed to Play Store (due to limitations of free plans).
c) Multi-language support (just English and Polish).
d) Adding expanses to a trip to split it between previously added friends.
```

Figure 12: MoSCoW prioritisation file.

Chapter 5. Design

5.1. Overall Architecture

The TravelBuddy application was designed in the Model-View-ViewModel (MVVM) architecture[32]. This is a design pattern that divides applications into three main components that are mentioned in the name. The reason for this separation is the clean structure that allows for easy maintenance and testing of the code. As a result, it is possible to separate the interface and software logic.

The Model represents the application's data layer, i.e. data sources, business logic and all API calls that interact with external services []. In this case, this component will include: - Firebase Firestore database responsible for collecting application data.

- Firebase Authentication that manages the login data of application users
- Retrofit used for interaction with external APIs (Pixabay API, Geoapify API)
- OSMDroid that enables displaying a map working with the model layer to retrieve data from the map
- Coil[50] is used to load and display images imported from Pixabay API

The View is a layer that represents what is shown on the screen for the user to see. It does not contain any business logic but reacts to changes occurring in the ViewModel. Jetpack Compose is entirely responsible for this layer because as a user interface framework it is able to directly bind the functionality of components to their state which is managed by the ViewModel. Then the components observe the state which is provided by the ViewModel and are updated with the change of states. This will be noticeable in these cases:

- Selected components present data which is fetched from Firestore using implemented methods.
- Components responsible for composable maps are integrated with OSMDroid to render the map based on the ViewModel data.
- Coli is implemented with interface elements which are a place for images loaded via Pixabay API.

The ViewModel is the element responsible for the user's interaction with the application to update the model accordingly []. It is a mediator that communicates between the view and the model. It provides data from StateFlows which represent the application state and all changes that occur in them.

- It also coordinates all calls to the previously mentioned services to update the UI accordingly based on the received data. A good example here would be navigating the user to the main application screens after entering valid credentials in the login screen.
- Data received from API calls is prepared for presentation in the View layer, e.g. displaying the desired destinations on a map.
- The ViewModel also covers the availability of cached data that was previously loaded from Firestore, which allows for offline access to data.

The architecture used in this project also allowed for easy localisation and fix of errors that occurred on different layers of the application.

5.2. Detailed Design

5.2.1. Database Design

Cloud Firestore database is organised into three main collections: "friendRequests", "friends", and "users".

a) "friendRequests" has a sub-collection under which it stores data relating to any friend request in the application. Documents (friend requests) appear in the database only when someone sends an invitation, and it is neither accepted nor rejected. This means that if a document exists in this collection, the status of this friend request can be defined as "pending". In case of rejecting the invitation, the document is deleted from the database. In case of accepting the invitation, it is also deleted from the database, but new documents responsible for matching users as pairs of friends appear in a new collection called "friends".

- Each document has its unique ID representing the currently existing friend request that was sent by any user. Each document contains four fields. The "receiverId" field stores the unique ID of the user to whom the invitation is directed, to identify the recipient. The remaining three fields concern the user sending the invitation to present the data of the recipient of the invitation. The "senderEmail" field contains the email address that was imported from the user who sent the report. The "senderUsername" field representing the username of the given user works on the same principle. The last field called "senderId" stores the unique ID of the user sending the invitation. This is a variable that identifies the sender and associates him with the receiver account.

b) "friends" is a top-level collection containing a document named "friends_list" which acts as a list of users as friends.

- In the document, there is an array named "list_of_friends" which is the main structure that collects users who are connected as friends after accepting the request. The array collects two structures in the form of a 'map'. The first one is named "friend" and collects information about the user who accepted the invitation. The second map named "user" contains information about the user who sent the friend request, and it was accepted by the user from the map "friend". Both maps have three fields namely "email", "userId" and "username". "user" collects the data of the sender of the friend request and "friend" the receiver of it.

- When a user accepts the invitation, the two above-described map structures are created simultaneously. This means that the database will contain information that user A is a friend of user B, and also that user B is a friend of user A. As a result of this design, by accepting the invitation, users A and B will automatically be added to their friend lists. This is an intentional action to prevent both users from having to send an invitation to each other.

c) "users" has a sub-collection under it that collects application user data. Each user is identified by their UID that matches the one in Firebase Authentication which means that access is secured and integrated. In addition, the user also has a field that stores the email address associated with the account and a field that stores the username that the user gives themselves. The user has two sub-collections, "trips" which contains all the data related to the user's trips and "phrases" which contains all the data related to the user's phrases. This simplifies obtaining data that belongs to individual users.

- In the "trips" sub-collection, there are documents with the UID identifier of each trip added by the user. Each of these documents contains fields in which data related to the details of the trip is collected. "author" is the user who is the creator of the given trip. "checklist" is an array that has maps

containing three strings: "checked" to indicate the state of completion of the item, "id" for individual identification of items and "task" which is the content of the item that is displayed in the checklist. The "destination" field is a place where the location entered or selected by the user in the application is stored. "title" is a field where the user's name for the trip is stored. Two fields "startDate" and "endDate" which store the dates specified by the user in the format DD-MM-YYYY. The field named "shared" is a boolean that indicates whether the trip is visible to friends. The last field is "tripPlans" which is also an array of maps that have three strings named "dateOfVisit" (the same date format as "startDate" and "endDate"), "id" for individual identification of items and the "place" field which is entered or selected by the user in the application.

- The "phrases" sub-collection contains documents that are identified by UID and represent phrases and related data added by the user. It contains five fields, three of which store data entered by the user. These are "language" to specify the language of the phrase, "phrase" to enter the actual phrase, and "translation" which is the translated version of the phrase. The "username" field is filled in automatically with the username of the user who created the phrase. The last field "id" contains a unique identifier (the same as the document name) to identify phrases individually.

5.2.2. APIs Usage

a) Retrofit

Two Retrofit[33] APIs are integrated in Travel Buddy to implement services and as a result the application will have functionalities that enhance the user experience of this application.

- Pixabay API[34] is used to fetch high-quality images that are related to the selected query, in this case, these are queries that concern the destination of the trip and then the images are displayed on the application interface using Coil, a library used in Jetpack Compose to load the image. Pixabay API integration is done using Retrofit which manages the queries and responses. This interface is accessed using a function whose parameter corresponds to the keyword which is the destination of the trip. The application is able to load multiple image requests so that the user having several created trips can see images corresponding to each of them on the interface.

- Geoapify API[35] provides several significant functionalities in TravelBuddy. The first one is Geocoding which offers obtaining geographic coordinates from the places entered by the user. This allows locating the place and displaying it on the map if needed. The second feature is Address Autocomplete which displays suggestions for locations around the world based on characters entered by the user in the text field. This is very useful when searching for a place on a map or list or entering the user's travel destination. The third feature is called Places and allows loading places and points of interest based on selecting predefined categories in the app and also the location where the user wants to explore these places. In TravelBuddy, there is a whole screen called "Explore" which utilises this functionality by displaying suggestions to the app user. In this case, Retrofit API was also used to integrate it with multiple endpoints for these features.

Retrofit integration for both of these APIs provides an efficient solution by handling network requests which increases the overall functionality and usability of the application.

b) OSMDroid

Integrating a map into a travel app is an important part of its functionality. OSMDroid [] was used for implementation, which is a flexible and extensible API that allows for the introduction of map-related features.

- Initialization of the map starts with the use of "AndroidView" that is composable in Jetpack Compose which is used to embed the map into the application structure. The map uses Mapnik tiles, a known source for working on maps from OpenStreetMap[37]. The user can control the map using swipes to move it and pinches to manage the map zoom.

- The map supports two types of markers. They were created using the Vector Asset creator in Android Studio using a template from svgrepo.com[38]. The first one is red and is a central marker that represents the user's current location or destination depending on the functionality selected in the application. The second type of marker is blue and represents a point of interest or a search result depending on the map type selected in the application. Markers are extremely important for the user's orientation in the context of the map. Clicking on each of them shows information about what each marker indicates to improve the user's orientation on the map.

- The map search functionality is also available in TravelBuddy. Clicking the button brings up a text field that allows entering a location (address completion from Geoapify API is also available here). Then, after clicking the "Search" icon button, the result is converted to GeoPoint [] (a data structure representing a geographic location using longitude and latitude) and then the map is centred on the search result with the marker indicating it.

- After obtaining permission from the user, the application is able to show the user's last known location, which facilitates navigating on the map and search for places located next to the user by browsing the map next to the marker indicating the last known location.

The above-mentioned decisions regarding the map functionality give the user the possibility of using the map robustly in case of real-life needs.

5.2.3. Authentication

After confirming user's data in the application, the user authentication system is implemented to ensure safe and trouble-free access to their data from any compatible device. As part of the project, several functionalities have been introduced that are closely related to the concept of authentication.

- a) The first functionality is to allow the user to register (create an account) in the application using a selected email address and password (and optionally username). Along with the registration in the Firebase console, next to the user's email, which serves as an identifier, the date of account creation in the application and a unique ID are also saved. After this process, the user appears as registered, but in order to log in to the application, he must first verify the email address.

- b) Another method is to make sure the application is only used if the email address is confirmed and that requires the user to manually verify the email. This procedure ensures that there are no unauthorised users in the database, and that is why this stage of the user registration procedure is essential. An email with a verification link is automatically sent out to the email address provided at user registration. After clicking the link, the user is able to log in to their account in the application.

c) Once the account is registered and verified, the user is granted the ability to log in to the application. Following logging in using their email address and password, they are navigated to the application's main screens, where they can access all available features.

d) In the event of forgetting the content of the account password, it is possible to reset it. After entering the email address located in the Firebase console, the application sends an email with a link allowing them to enter a new password, which is assigned to the associated account. As a result, the user is able to regain access to the application despite forgetting the current password.

e) The application allows for a safe logout operation in order to potentially change the user. After clicking the button responsible for this functionality, the user is no longer able to use a given account until they log in to the application again.

f) The user's authentication status in the application is constantly monitored. This is because the appropriate screens are displayed based on this status. If the user is not logged in to the application, after turning it on, it will display the "Sign In" screen, which allows entering the user's login details. However, when the user has completed the login process and has not logged out, the status is shown as logged in, which means that until the user is logged out, the application will display the main screens with the logged-in user's details. This provides an intuitive experience and at the same time prevents unauthorised access after logging out

The login and registration process also implements error handling, which consists of informing the user via a component called "Toast" about an error that occurred during these processes. Errors that the user may be informed about if they occur are:

a) Registration

- "User already exists"
- "Password is too short"

b) Login

- "User is not verified"
- "Wrong password or email"
- "Account does not exist"

c) During both

- "Wrong email format"
- "Unknown error"

This allows the user to diagnose more easily what went wrong during both of these processes.

5.3. User Interface Design

The user interface is an equally important issue in the development of an application, next to its functionality. An easy-to-use, intuitive and eye-catching interface directly affects the user experience of the software, making a good first impression. This can have a key impact on whether the user decides to stay with a given application or to use competitive solutions that are available on the market.

The first step in creating the UI was to design it in Figma, in order to visualize and assess whether the decisions made during this process were well thought out before implementing the code responsible for the application interface[39].

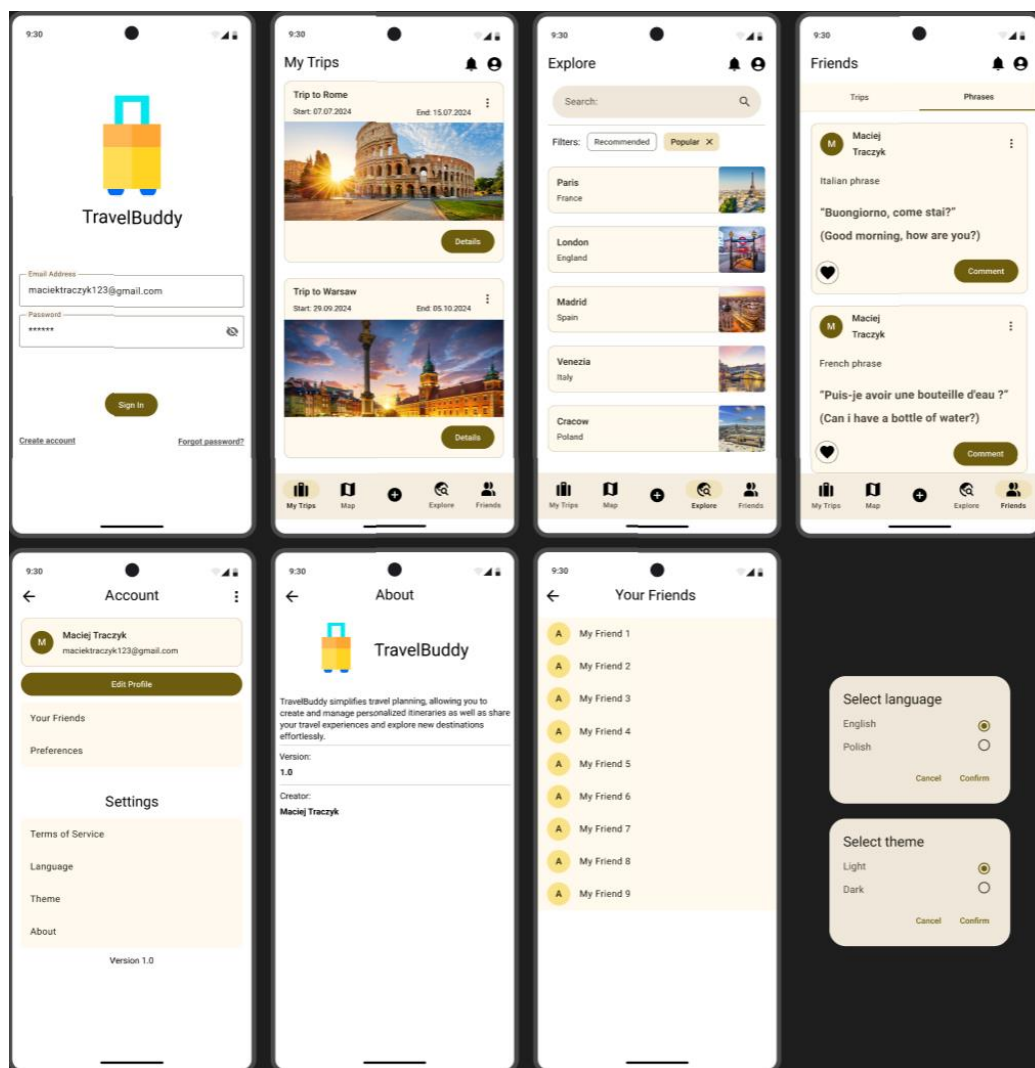


Figure 13: Example screens of the design of TravelBuddy in Figma

Material 3 was used as the system according to which the TravelBuddy interface was created, i.e. the latest design system developed by Google, among others for mobile applications. It enables the creation of visually attractive interfaces by using consistent colours, components and fonts, ensuring a cohesive experience.

5.3.1. Colour Scheme of TravelBuddy

The colours in TravelBuddy are not random. Each colour was generated based on a source colour (Hex: #6E5D0E) by a tool called Material Theme Builder[40] which allows for a provisional visualisation of colours in the application. After selecting the preferences for colours and fonts, a file is generated that contains the code representing them. The next step is to locate it in the appropriate place in the project files so that the changes affect the appearance of the application. After the code is properly integrated, all components, backgrounds and fonts will take on the style that was imposed by the file. The generated file has versions of colours that are displayed during the light mode, but also versions

during the dark mode. It requires slightly different shades of colours due to the presentation of the content in better contrast that will allow for better readability on a background dominated by dark colours.

Light Scheme									
Primary P-40	Secondary S-40		Tertiary T-40		Error E-40				
On Primary p-100	On Secondary s-100		On Tertiary t-100		On Error E-100				
Primary Container P-90	Secondary Container S-90		Tertiary Container T-90		Error Container E-90				
On Primary Container p-10	On Secondary Container s-10		On Tertiary Container t-10		On Error Container E-10				
Surface Dim N-87	Surface N-98		Surface Bright N-98		Inverse Surface N-20				
Surf. Container Lowest N-100	Surf. Container Low N-96	Surf. Container N-94	Surf. Container High N-92	Surf. Container Highest N-90	Inverse On Surface N-95				
On Surface N-10	On Surface Var. NV-30	Outline NV-50	Outline Variant NV-80	Inverse Primary P-80					
					Scrim N-0	Shadow N-0			

Figure 15: Light Scheme table from Material Theme Builder.

Dark Scheme									
Primary P-80	Secondary S-80		Tertiary T-80		Error E-80				
On Primary p-20	On Secondary s-20		On Tertiary t-20		On Error E-20				
Primary Container P-30	Secondary Container S-30		Tertiary Container T-30		Error Container E-30				
On Primary Container p-90	On Secondary Container s-90		On Tertiary Container t-90		On Error Container E-90				
Surface Dim N-6	Surface N-6		Surface Bright N-24		Inverse Surface N-90				
Surf. Container Lowest N-4	Surf. Container Low N-10	Surf. Container N-12	Surf. Container High N-17	Surf. Container Highest N-24	Inverse On Surface N-20				
On Surface N-90	On Surface Var. NV-90	Outline NV-60	Outline Variant NV-30	Inverse Primary P-40					
					Scrim N-0	Shadow N-0			

Figure 14: Dark Scheme table from Material Theme Builder.

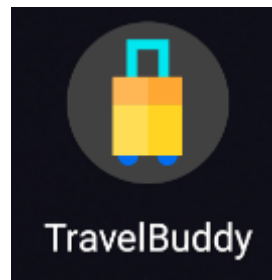


Figure 16: Icon of TravelBuddy[41].

5.3.3. Authentication and Registration Screens

a) Login screen appears after the first launch of the application when the user has not yet logged into their account. This screen allows the user to log in to the application. It has two text fields that enable entering the user's email address and account password. The text field in which the password is entered has a button icon on the right that can reveal the entered password, which is by default presented as dots to cover it. In the case of entering incorrect login data, they change the colour accent to red and a component called Toast is displayed that informs the user about the login error. Below the text fields is a button that, when pressed, triggers the user login attempt function. The button is enabled only when the two above text fields have been filled with login data. Below in the row there are two text buttons. On the left "Create an account" which navigates to the user registration screen in the application. On the right side "Forgot password?" when pressed, invokes the Dialog component that displays a text field to enter the email address for which user wants to reset the password.

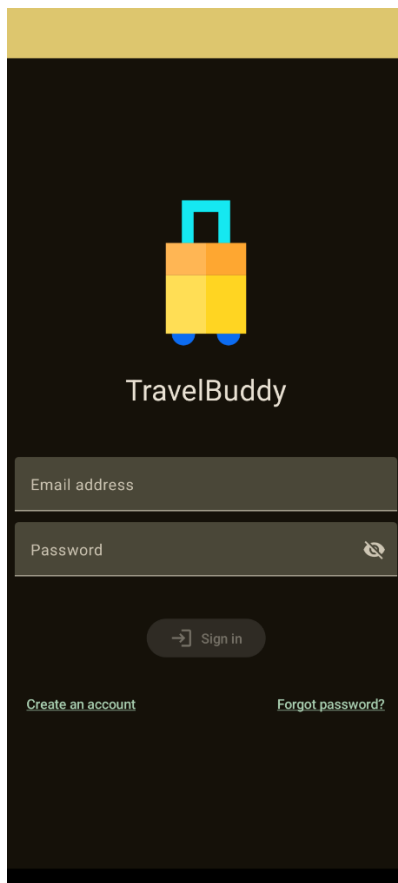


Figure 18: Log in screen.

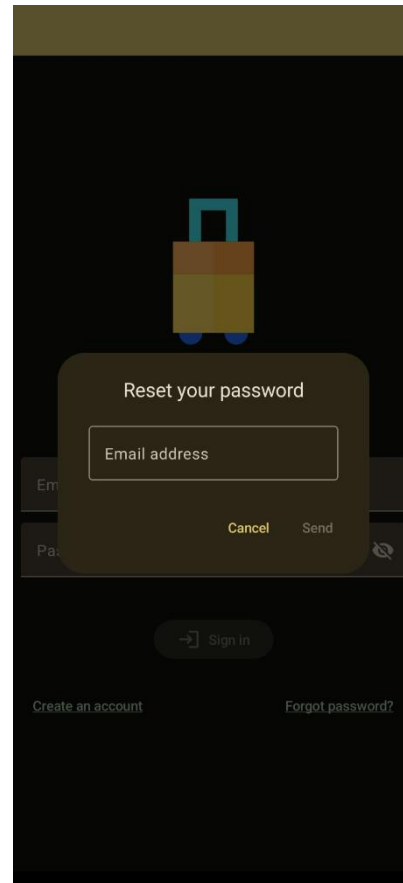


Figure 17: Password reset dialog.

b) After navigating to the screen for registering a user account in the application, three text fields appear that allow the user to enter their email address, password and password confirmation. The last two also have an icon to reveal the password. The button is located below them and enables starting the account creation process. It is enabled when all text fields are filled in. After pressing it, if all the data is correct, a "Dialog" appears informing about the need to verify the user's email to enable logging into the given account. In the case of providing incorrect data, Toast will inform the user about the error. In the top app bar, there is an icon button on the left to navigate to the previous screen.

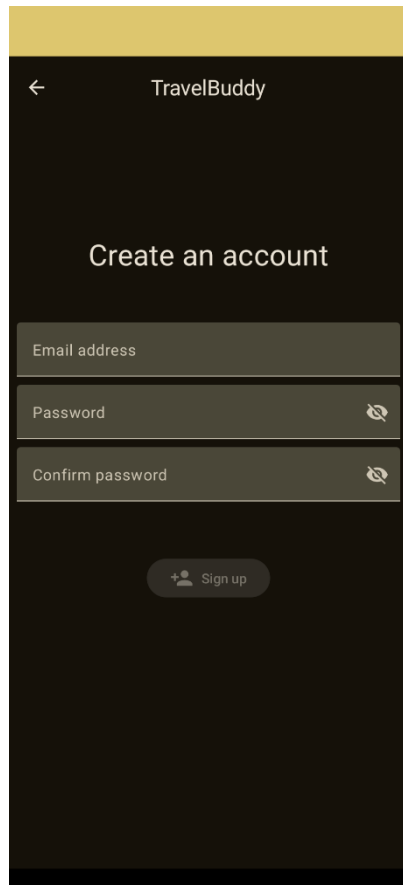


Figure 20: Sign up screen.

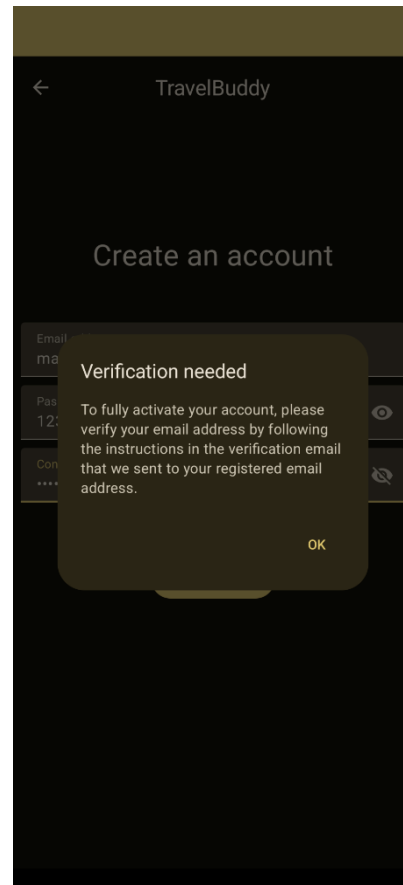


Figure 19: Verification dialog.

5.3.4. Main screens

The main screens of the application are the ones that allow users to navigate to them by clicking on icons on the navigation bar.

a) The screen that is displayed immediately after logging into the application is the "My Trips" screen. This is a screen displaying user-created trips in the form of components called Cards that are displayed one below the other. Each Card has the title of the trip at the very top, below in one row there are the start and end dates of the trip. Then, a photo is displayed that is fetched based on the destination of the trip entered by the user. Below the photo in one row on the left there is a button that navigates to display the details of this trip, and on the right is a button that sets the status of the trip to be visible to friends.

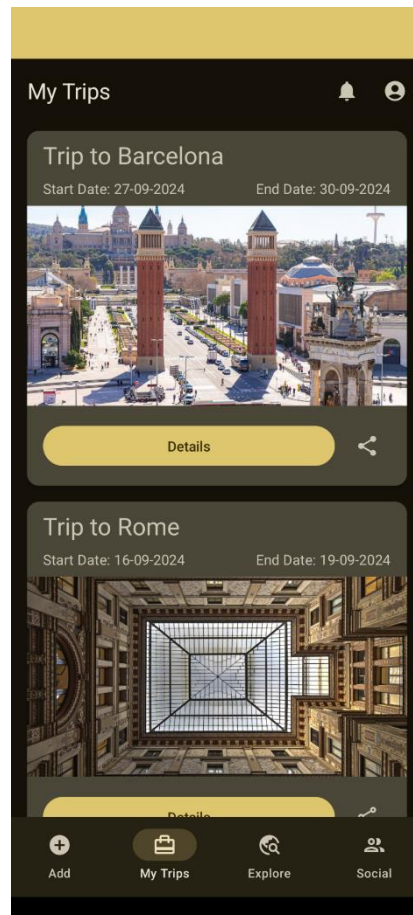


Figure 21: My Trips screen.

b) The screen called "Explore" is a screen with two text fields in which the user enters or selects destinations from the list which appears after and selects the category of the place from the list. Loading places is triggered by pressing the search button under the text fields on the left. On the right side, there is a button that navigates to the place search screen using the map search engine. Places are displayed below one under the other in the form of a "Card" with the name and address of the place which are clickable. After clicking on them, they navigate to the screen allowing adding the chosen place to the existing user's trip and indicating its place on the map. When places have been fetched by the application, the text fields are hidden, and the search button changes its function to make them visible to perform another search and the map button changes its function to display loaded places on the map along with the indicated destination.

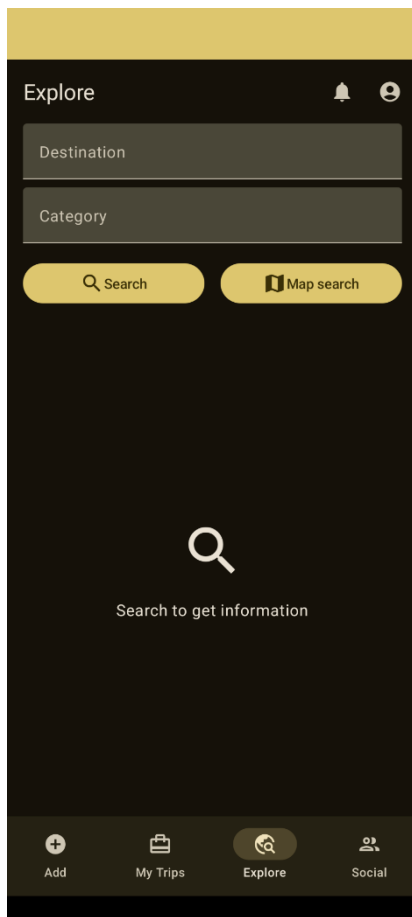


Figure 24: Explore screen with no data fetched.

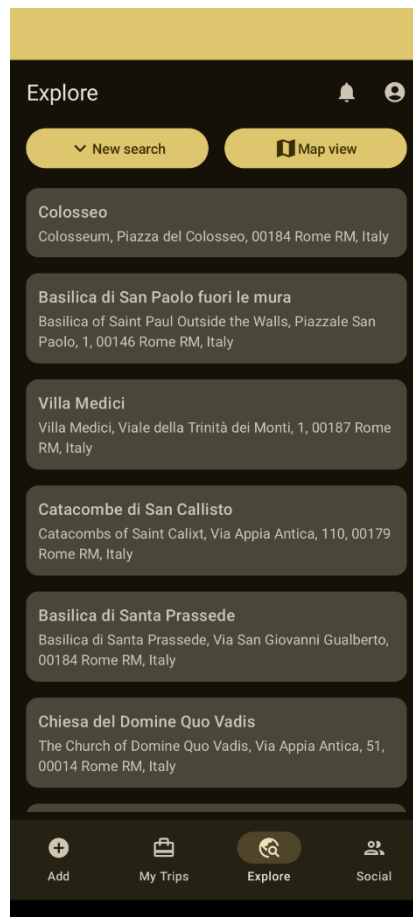


Figure 23: Explore screen with fetched data.

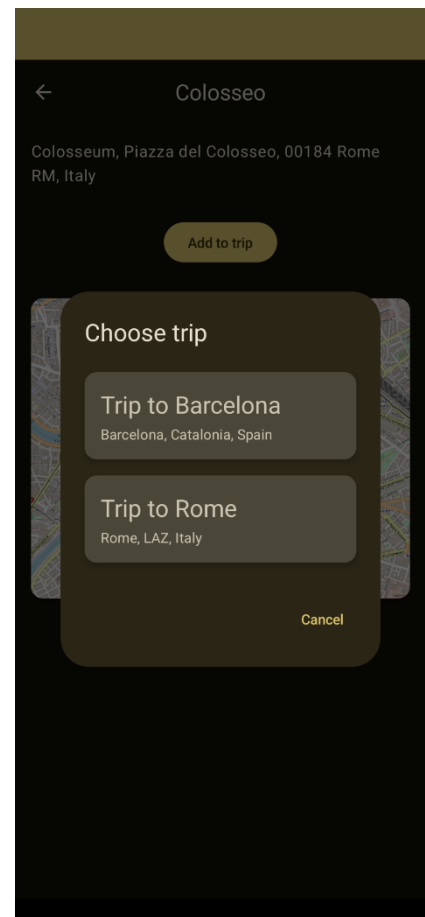


Figure 22: Dialog for trip plan adding.

c) The last screen is dedicated to the social section, which displays content shared and visible to the user's friends. It is divided into two tabs. On both of them, it is possible to perform a swipe-down gesture to perform a content re-fetch. The first one displays trips that are shared by the user and their friends. They are presented in the same way as trips in the "My Trips" screen, with the difference that instead of a button to share it, the author of the trip is displayed. The second tab contains phrases added by users one below the other. They are displayed in the form of Card components that contain the author, language, content of the phrase and also the translation. In the case of phrases shared by the user, they can also edit or delete them using the menu displayed after clicking the icon button on the right.



Figure 26: Tab "Trips" on Social screen.



Figure 25: Tab "Phrases" on Social screen

5.3.5. Top App Bars

This is a component located at the top of the screen. It collects icon buttons that perform various functions in the application. There are two types of these components in TravelBuddy:

a) Main top app bar is located at the top of the main screens. On the left side, it has a text that represents the name of the currently displayed screen in the application. On the right side, there are two icon buttons. The first one navigates to the notification centre. The second one navigates to the user account panel.



Figure 27: Main top app bar.

b) App bar with an arrow back is the second type of top app bar that differs slightly from the main one. On the left side, there is always an icon button that allows the user to navigate to the previous screen. In the middle, there is a text that represents the name of the currently displayed screen. The

right side of the component displays an icon button that, depending on the displayed screen, will perform the function that is most needed or will not be displayed at all. In the case of the user account panel, the icon will symbolise and execute logging out of the application. When adding entries to the application, an icon will be displayed that enables saving them. The screen where the trip details are shown will display an icon button that triggers a menu with the functionalities available on that screen.

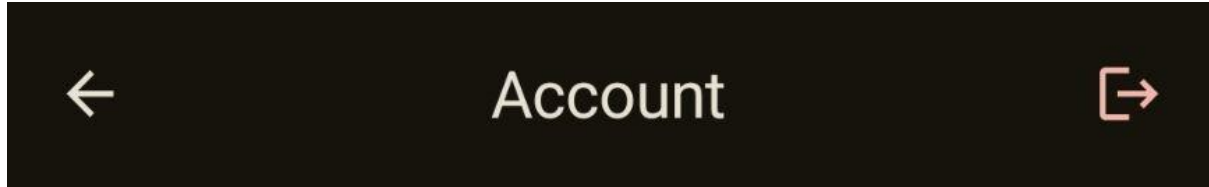


Figure 28: App bar with arrow back.

5.3.6. Navigation Bar

This is a component that is located at the bottom of the main screens. It does not appear in any other part of the application. It enables navigation between various screens by clicking on the icons that are located there and also to perform certain actions. In TravelBuddy, this component contains four icons. On the left side, there is an icon labelled "Add" and after clicking it, it triggers the appearance of the bottom sheet component for adding entries. The next three icons after clicking display the named main screens ("My Trips", "Explore", "Social").

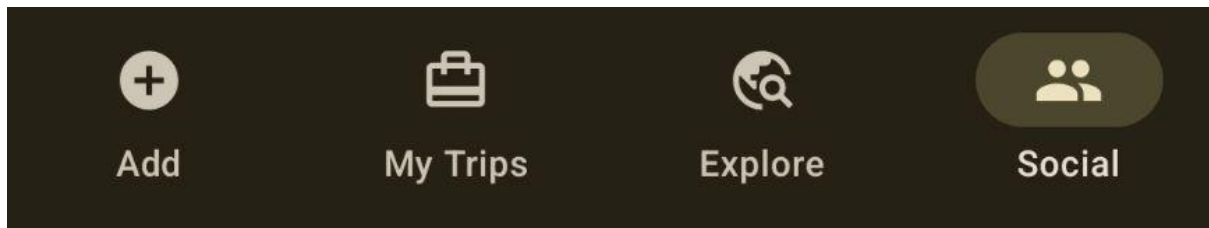


Figure 29: Navigation bar.

5.3.7. Bottom Sheet for Adding Data

This component contains three buttons in a row that allow navigation to screens that will allow the user to add entries (trips, phrases), as well as send friend requests within the application.

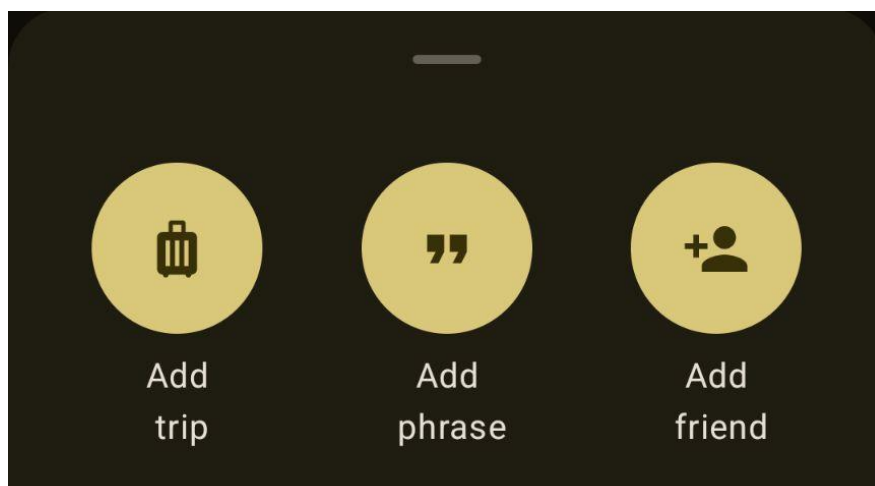


Figure 30: Bottom sheet.

5.3.8. User Account Panel

This screen is for account management and also for application settings. In the upper section of the screen, a component is displayed that displays the display name, the user's email address and also a monogram that displays the first letter of the user's email address. Below is a button that navigates the user to the profile section and also one that navigates to the friends list. Below is a section that concerns the application settings which starts with the text "Settings" for indication. Below is a list of four clickable elements. The first one is "Terms of Service" which navigates to a screen that is not expanded at the moment. It is there in case of future development of the application which assumes introducing terms from this service to inform the application users about the rules of using it. The next two elements enable calling up the Dialog components. The first one is used to select the application language. The second one allows for the selection of the application theme preferences. The last element navigates to a screen called "About" which contains information about TravelBuddy. At the very bottom is a text that displays the application version which in its current form is 1.0.

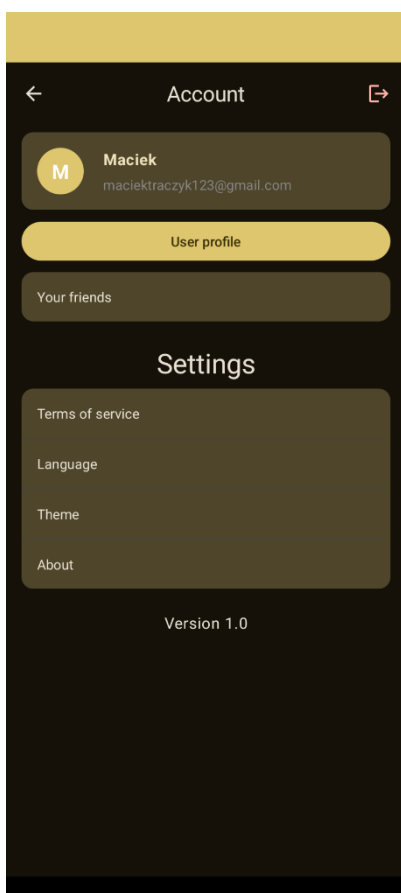


Figure 33: Account Panel.

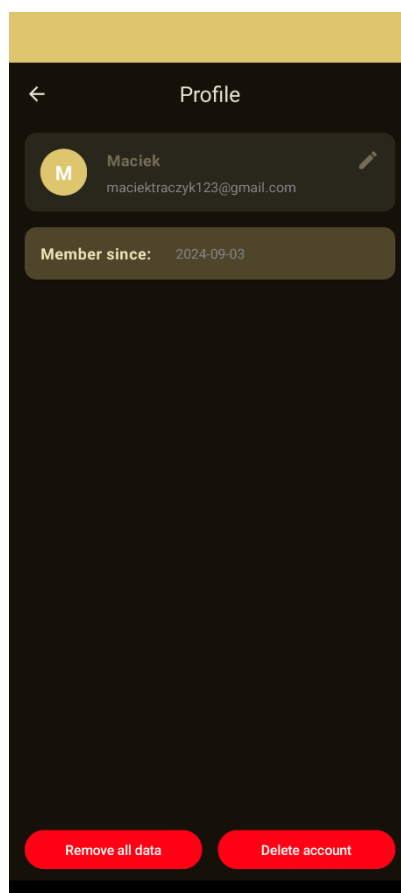


Figure 32: Profile screen.

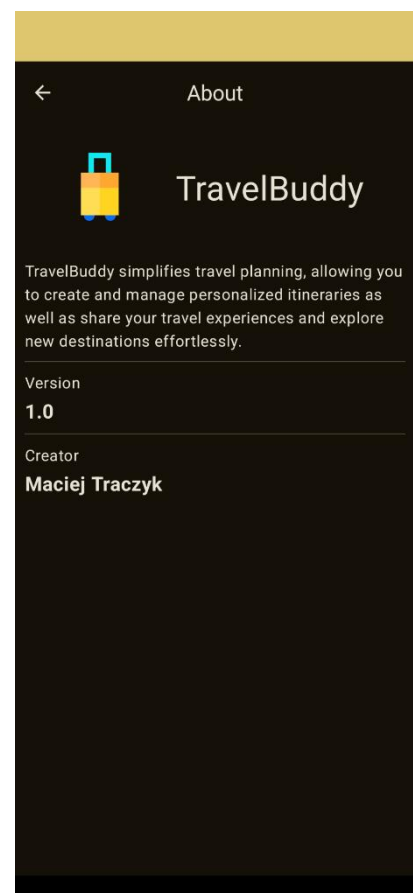


Figure 31: About screen.

5.3.9. Notification Centre

Notification centre is intended for in-app friend requests that accumulate on this screen. When a user receives a friend request, a Card component is displayed that contains text with information about which user the request comes from, and below it, there are two text buttons that allow the user to accept or reject it. In the case of several notifications, Cards are displayed one below the other. In the

case of further development of the application, the screen will also collect other in-app notifications, e.g. information about upcoming trips or commenting on and liking friends' trips.

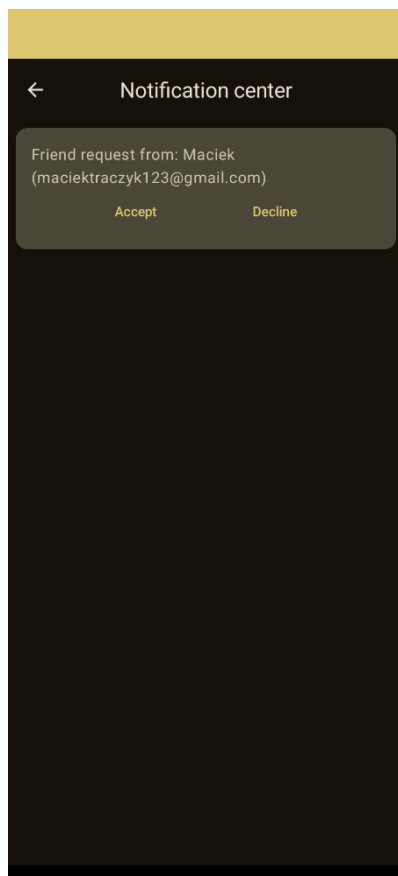


Figure 35: Notification centre with the notification.

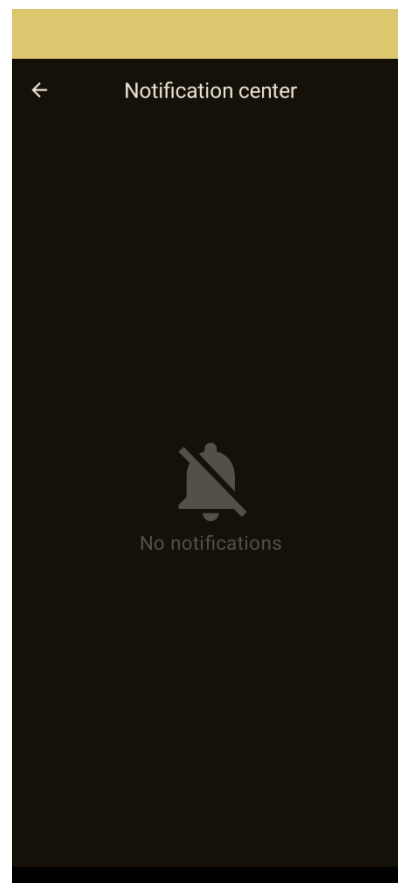


Figure 34: Empty notification centre.

5.3.10. Other Screens

a) Trip details screen presents the details of the chosen trip. It has fields which contain texts displaying information related to the given trip. At the top, there is the start and end date of the trip in the formats DD-MM-YYYY placed in a row. Below it, the Card component displays the full destination of the trip, and then in the row are two cards. The first one allows navigation to the map screen which displays the map of the destination and places marked with markers that have been included in the field containing the trip plans. The second one navigates the user to the web browser in which the phrase "(destination) weather" is automatically searched, which allows a quick view of the weather at the destination online. Then, below there are two cards which represent the checklist and the trip plans. Both of them display the first five items from the lists for a quick view. Clicking on them redirects us to screens allowing the user to edit these lists. The checklist gives the possibility to add items using a button which can be freely named using the text field. The icon button on the right side of the item allows removing it when clicked on the text field associated with it. In the list of trip plans, Cards are displayed that show the name, address of the place and the date of the planned visit, and on the right side of the icon button, it brings up a menu that enables editing the given item (edit, removal). The button under the last item brings up a screen to add another place to this list. At the very bottom of

the trip details screen is text displaying the status of the trip visibility to friends and below it, there is a button that brings up a dialogue to set the trip to private mode.

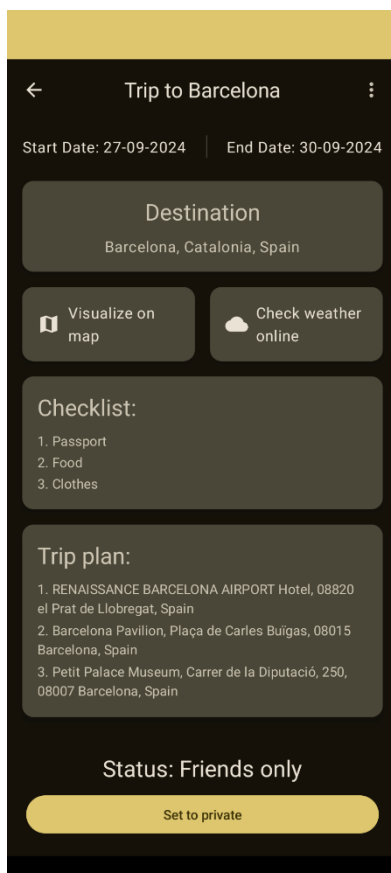


Figure 38: Trip details screen.

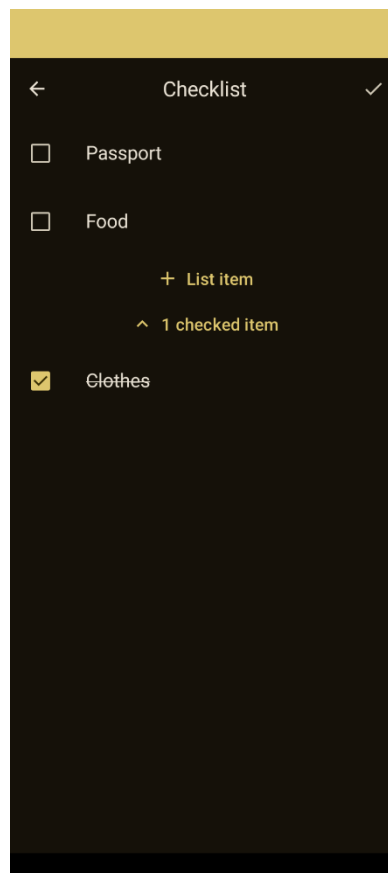
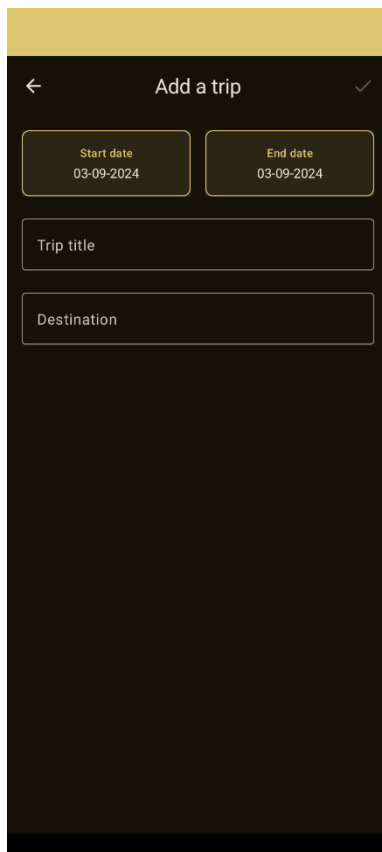
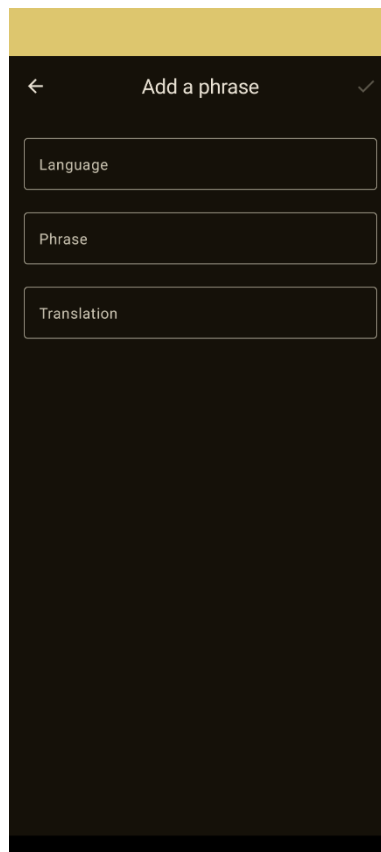
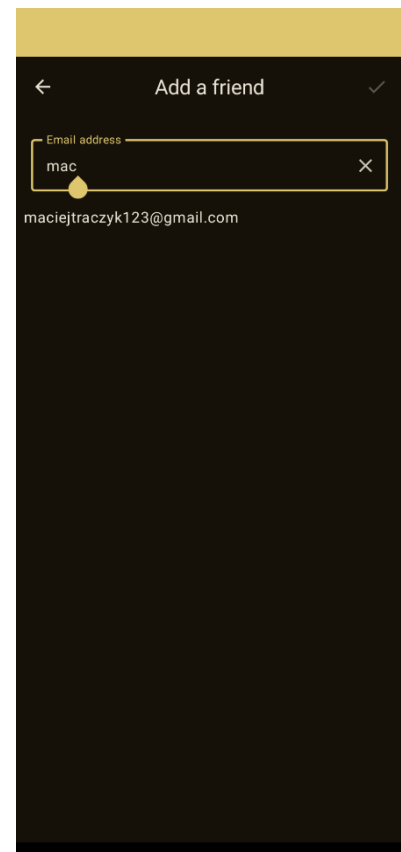


Figure 37: Checklist screen.



Figure 36: Map view screen.

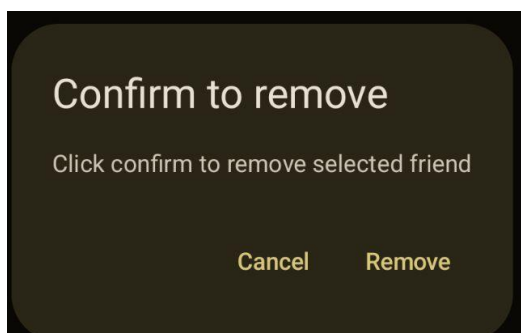
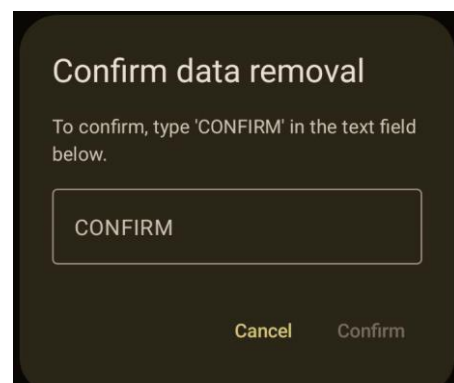
b) Add/edit data screens are used to add trips, phrases, friends and display name (username). The screen for adding trips has two elements in one row displaying the start and end date. Clicking on them invokes the "DatePicker" which allows for the selection of a date and guarantees that the user will enter the format imposed on it[42]. Below there are text fields for entering the title and destination of the trip from which, after entering at least three characters, a list appears from which the user can select suggestions and place them in the text field. The screen for adding phrases has only text fields for entering the phrase, translation and language. The screen for setting the display name is a screen that appears only when the user has not specified it. At the moment, the username cannot be edited to avoid a situation where user A changes the display name, and it is not updated on the friends list of user B. The screen for searching for friends and sending them friend requests contains one text field which, after entering the first character, displays the list of users who are already registered in the application.

The 'Add a trip' screen features a dark theme with a yellow header. It includes a back arrow and a checkmark icon. The form contains two date pickers for 'Start date' and 'End date', both set to '03-09-2024'. Below these are text input fields for 'Trip title' and 'Destination'.*Figure 41: Add a trip screen.*The 'Add a phrase' screen has a dark theme with a yellow header. It includes a back arrow and a checkmark icon. The form consists of three text input fields labeled 'Language', 'Phrase', and 'Translation'.*Figure 40: Add a phrase screen.*The 'Add a friend' screen has a dark theme with a yellow header. It includes a back arrow and a checkmark icon. The form features an 'Email address' input field with a yellow border and a close button (X). Below the input field, the email address 'maciejtraczyk123@gmail.com' is displayed.*Figure 39: Add a friend screen.*

5.3.11. Dialogs

Dialog components are windows that are displayed in front of the current screen content. They have text buttons at the bottom, and text in the main part. There are three types of Dialogs in TravelBuddy.

a) The first one is used to confirm the execution of high-risk action, such as deleting entries, in case the button responsible for these actions is accidentally pressed. The dialog will display a text asking if the user wants to proceed with this action and using the appropriate text buttons it will be possible to perform the desired operation or cancel it. In the case of a dialog that confirms the deletion of all entries from the user's account or the entire account, it also requires entering the word "CONFIRM" in capital letters in the text field to protect against accidental loss of all data in the account.

The 'Confirm to remove' dialog is a dark-themed window with rounded corners. It has a title 'Confirm to remove' and a subtitle 'Click confirm to remove selected friend'. At the bottom, there are two buttons: 'Cancel' and 'Remove'.*Figure 43: Confirm removal dialog.*The 'Confirm data removal' screen is a dark-themed window with rounded corners. It has a title 'Confirm data removal' and a subtitle 'To confirm, type 'CONFIRM' in the text field below.'. Below the subtitle is a text input field containing the word 'CONFIRM'. At the bottom, there are two buttons: 'Cancel' and 'Confirm'.*Figure 42: Confirm all data removal screen.*

b) The second type enables selecting options that appear in the form of Radio Buttons. One is used to select the application theme preferences and the other to select the language in the application. After selecting the option, the user needs to click the appropriate Radio Button and press "Save" to confirm the changes made or cancel them.

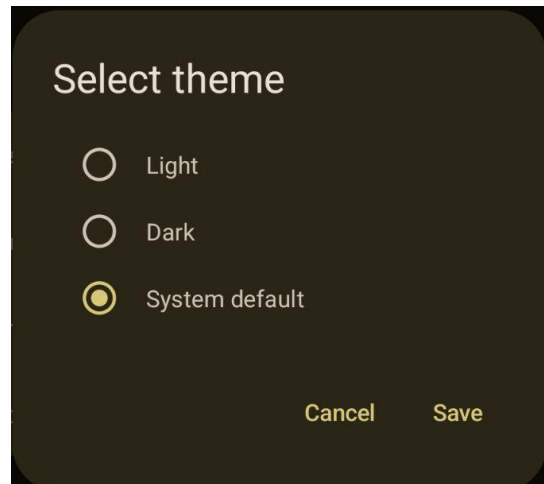


Figure 44: Theme selection dialog.

c) The last type informs the user about important events. In case of a successful registration in the application, a dialogue will be displayed that will inform the user about the need to verify the email address by clicking the link in the message sent to the email box.

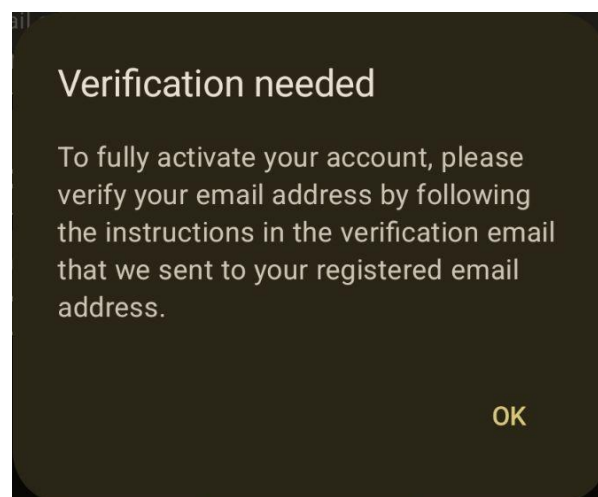


Figure 45: Verification dialog.

5.3.12. Snackbars (Toasts)

These are components that are used to provide the user with a short update at the bottom of the screen. They are also displayed in front of screen content, but they are much smaller and do not require the user to disable them. In TravelBuddy, they are used to inform about various updates. For example, they will inform the user why they are unable to create an account in the application or log in to it, for example, "Password is too short", "Wrong email format" or "Password mismatch". They will also inform about the action that has been accomplished, e.g. "Signed out", "Logged in", "Friend

request sent". This is a reliable way to provide feedback from the application to the user in a non-invasive way.

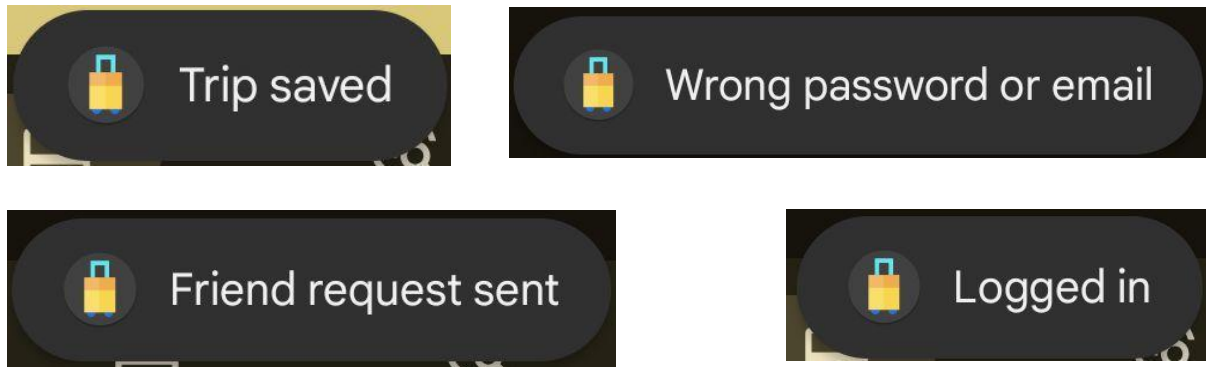


Figure 46: Examples of Toasts.

Chapter 6. Implementation

6.1. Database Integration

The "FirebaseViewModel" class contains code that manages user data integrating Cloud Firestore service in the application. Firestore instance is created to interact with the service. It contains several data holders (MutableStateFlow) to hold data related to usernames, emails, phrases, trips, and friends, among others. The "init" code block calls methods related to trips, phrases and the user. This ensures the fetching of selected data when the ViewModel instance is created. The class employs CRUD operations (Create, Read, Update, Delete) to manage user data efficiently within the application. Management of specific data types is done through fetch, add, update and remove operations. Each data type has functions that are designed to handle them separately, providing real-time synchronisation between the service and the UI.

- The fetch function retrieves data from Firestore. This data is user data and selected friends' data that are available for reading. Its implementation typically starts with verifying user authentication and then launching a coroutine to perform asynchronous operation. Then, a query is made for the appropriate collection or subcollection to retrieve the specified data. The results are stored in StateFlow variables to update the UI in real-time.

- The add functions are used to add new entries to the Firestore. The functions do the same thing, i.e. check the user's authentication and launch the coroutine. Then the data is added to the relevant subcollections. In the case of trips and phrases, they also contain additional data such as the user's username and their unique ID. Finally, they trigger the "fetch" functions to ensure that the UI is updated with the latest data.

- The update functions allow for modifications to data that is already in the database. The initial implementation also includes verification of the user's authentication and the launch of the coroutine. Typically, they involve constructing a Map from the fields to be updated, which is then applied to the corresponding document using the unique ID. Finally, they also call fetch functions and have an onSuccess callback to allow the user to be notified about this operation using the UI.

- The remove functions remove selected existing data from the database. The structure also starts with authentication verification and launches a coroutine for asynchronous deletion. The selected document is identified by its unique ID and then removed from the Firestore subcollection. The

functions also contain a callback to use this information to pass to the UI layer to inform the user about the operation. The "removeAllUserData" function is also implemented. It removes all data related to the currently logged in user.

The described data provides consistent data management throughout the application, allowing the application user to have complete control over the application regardless of the data type.

6.2. Authentication handling

The code responsible for the Firebase Authentication service logic is also in a class "FirebaseViewModel" class. This is a class that helps separate the UI logic from the data processing. It serves as a communicator between the UI and Firebase Authentication layers.

- To interact with the service, a "FirebaseAuth" instance is created. The class uses an observable data holder (MutableStateFlow) that holds the current state of the authenticated user to allow the UI to reactively update the UI based on this state.

- The "observeAuthState" function monitors changes in the authentication state. In the case of a sign-in or sign-out, the current Firebase user is automatically updated in the StateFlow (_authState) which ensures that the UI reflects on the state in an appropriate way.

- The "signInWithEmailAndPassword" function allows for logging in using an email address and password. It is wrapped in a coroutine to avoid blocking the UI thread. Then the helper function checks the success of the authentication. In case of success, it checks the email address verification status and starts to call the appropriate callback containing feedback information.

- The "signUpWithEmailAndPassword" function registers the user with an email and password. It uses the coroutine and the helper function in the same way as the previously described function (to determine the success of the operation). After successful registration, it creates a new document in Cloud Firestore in the "users" collection containing the registered email. Then the function responsible for sending the email verification to the new user. If the registration process fails at any point, it provides an error code via callback.

- The "signOut" function logs out the currently logged-in user and also clears the authentication status by setting its value to null.

- The "sendVerificationEmail" function sends an email with a verification link to the indicated user and reports the success or failure of the operation.

- The "isUserLogged" function checks if the user is currently logged in. This is used to conditionally display the login screen or main screens when the application is launched.

- The user also has the option to completely get rid of the Firebase Authentication service account. This is done with the "deleteUserAccount" function. The function calls "removeAllUserData" to first get rid of all data associated with the user and then it deletes the document in the "users" collection. Finally, it removes the user from the Firebase Authentication record.

Account management methods give the user full control to decide about the status of data entered into the application.

6.3. APIs integrations

6.3.1. Retrofit

a) "GeoapifyApi" interface defines API endpoints used to interact with the service. "searchPlaces" is a method that searches for a place based on a text query and returns search results in a GeoapifyResponse which contains a list of features that match the search criteria. The second method is "geocodeCity" which retrieves geocoding information for a specified place. Data classes in the interface define the JSON response structures returned by the API.

RetrofitViewModel is a file that coordinates API interactions with the UI state. It has three methods related to Geoapify API.

- The "searchPlaces" function is used to load places using the city name and category. It uses a helper function to retrieve the bounding box for the specified city and then filters out the result which is stored in the list. The loading state is included to display the progress bar in the UI.
- The "getBoundingBoxForCity" function fetches the bounding box of the city to determine the area of the searched place. It calls the appropriate endpoint and extracts the response if possible.
- This function is used to display suggestions to fill the text fields with. The endpoint is queried with characters and then returns a list of location names formatted to match the characters entered. The results are stored so that the UI provides suggestions each time the user types a character.

b) The "PixabayApi" interface is responsible for defining endpoints for interaction with a given service. It has a "searchPhotos" method that searches for a photo using a string in the query. It returns images that most closely match the query.

RetrofitViewModel coordinates API interactions with the UI state using a method for fetching images related to the trip destination.

- The "fetchImage" function initiates an image search via the endpoint and sets the loading state to true to display a circular progress bar at the location where the image appears to indicate the ongoing process. Then it checks the availability of photos that match the query. If the image is available, it provides the first available image on the list. Otherwise, the returned entry is null, which allows it to display information about the lack of availability of the photo in the given query. The process ends by setting the loading state to false to indicate that the operation has been completed.

The implementation of these functions (except getBoundingBoxForCity) also starts with the utilisation of a coroutine to allow a background process without blocking the main thread, which affects immediate changes in the UI.

6.3.2. OSMDroid

The OSMDroid library allows for the integration of the map and its interactions within the application.

- The "MapViewComposable" function initialises an instance where the tile sources and other map settings are configured. The "Configuration.getInstance().load()" method loads the settings from the configuration using shared preferences to maintain a consistent map appearance. Then a request for location permissions is called when they have not yet been granted.
- "AndroidView" is used to integrate the map with Jetpack Compose, allowing the "MapView" to handle user interactions like panning, zooming, and marker tapping.

- The map configuration contains the tile source as MAPNIK and also allows multi-touch control of the map to facilitate interactions with it on the screen. The map's zoom level and centre coordinates are preserved across sessions by the "MapAdapter", ensuring a seamless experience. Performance is optimised for efficient marker management, maintaining smooth interactions even with numerous markers.

- The ViewModel is also used to manage the map logic. It has variables holding the centre coordinates, zoom level and markers to allow the map to be reactive depending on its usage. It also contains functions to update the map based on the user's current location and to manage markers and their titles. They are managed by getting a list of GeoPoints that represent the locations (as latitude and longitude) of markers on the map. The titles, however, are updated using the "updateMarkerTitles" function that associates each GeoPoint with its title. This allows for viewing the title of the place by clicking on any marker on the map.

6.4. Friends' Requests

"FirebaseViewModel" has also a code that is responsible for handling the entire friend request process. MutableStateFlow properties store a list of the currently logged-in user, and the friend requests received by them. A data class called "FriendRelation" is used to enclose the relationship between user A (user) and user B (user) which are represented as User objects.

- The "addFriend" function adds another user to the list of friends by launching a coroutine for asynchronous operation. A "FriendRelation" object is created that represents the friendship and adds it to the document containing the friends list.

- The "getFriendsOfUser" function loads the user's friends by querying the "friends" collection in Firestore. The asynchronously fetched data is parsed into a list of User objects which are stored in "_friends".

- The "sendFriendRequest" function sends a friend request to another user. It checks if the user is not already on the list of friends or if the request has already been sent. If these conditions are not met, it creates an object that is stored in the "friendRequest" collection and waits for either acceptance or rejection.

- The "acceptFriendRequest" and "declineFriendRequest" functions are used to confirm or terminate the process of adding a friend. In both cases, the friend request is removed from the Firestore collection, but in case of acceptance, users A and B are added to the "friends" collection in both friendship directions (A to B's friend and B to A's friend) so that both appear on their friends lists.

- The "fetchFriendRequests" function monitors the status of incoming requests by using a snapshot listener in the "friendRequest" collection to filter the user ID.

- The "removeFriend" function is used to remove friendships from the database. The "FriendsRelation" object created represents the friendship and is removed from the Firestore document. Then, the reverse relationship is also removed so that both users are not on their friends lists.

6.5. Encountered Issues

6.5.1. Firebase Rules

Updating Firebase rules was often problematic. Each rule had to be defined very precisely so as not to affect the others moreover, and the testing process was tedious and time-consuming. This caused

especially considerable problems during the implementation of the friend request system. It contained data relations that conflicted with the others. Before modifying them, it was necessary to properly design how the friend request system would work. Then, a complete refactor of Firebase rules started to write them in a more understandable and organised way. After a thorough analysis of the code, the intended effect was achieved.

6.5.2. Username Validation

Having a field that records the author in both journeys and phrases should have been solved differently. It was added to check and grant access to data authors and block it for friends. The application checks if the logged-in user has the same username as in a given journey or phrase. Based on that, it displays UI components that allow modifications. Having fields representing the author of this data should be removed in future work due to unnecessary data storage, and the process of checking the author of an item should be based on something else. This way, the user could also freely change the username since it would not be dependent on access to other people's data. This problem was noticed by one of the testers. Due to lack of time to implement another solution, the option to modify usernames was disabled. Additionally, when setting a username, the application checks if it already exists in the database and informs the user about it, forcing them to set a different one.

6.5.3. Autocompletion API

The first solution for place autocompletion was the implementation of the "GeoDB" service[43]. After the correct implementation of the API, it was noticed that the service did not work properly in every scenario. For example, when entering the full text "Rome" in the text field, it did not suggest this city on the list. It can therefore be assumed that it has more deficiencies like this. The error was checked using Logcat, in which the loaded places were verified, and it was confirmed that it was not the UI's fault. For this reason, the decision was made to find a new API that also has this functionality. It was a good decision because the new implementation works fully, and no errors were noticed with the current solution.

6.5.4. Language State

After investigating the problem, it turns out that the value stored in DataStore[44] that is responsible for the Polish language is correctly retrieved, but after passing it to the "setLocale" function, changing the locale does not affect the current state of the application. This may be due to the fact that changing the locale settings requires a restart, and the changes are applied after the application is initialised. Despite this, the Polish language in the application can be set, but currently, it has to be done every time after starting the application.

6.5.5. System Notifications

An attempt was made to create a system of notifications that would notify the user about an upcoming trip when it was three days away[45]. A notification was successfully created that was triggered when the start date of the trip was three days or less days away. However, the notification only worked while the application was running. After deleting it from the cache, the notification would only pop up the next time the application was launched. For this reason, the idea was abandoned. The key to achieving the desired effect is the integration of the notification manager with a process that runs in the background.

Chapter 7. Testing

The application testing stage is crucial in the application development process by identifying potential bugs and issues that, although unintended, may be encountered during use. Fixing them will make the application more stable and the overall user experience will be enhanced. The entire testing process was based on three techniques described below.

7.1 Manual Testing

This process is based on manual testing of the application by manually executing test cases. No tools are used for this purpose to automate this activity. The testing plan started with creating a table with six columns named: "Test Ref", "Figure Ref", "Description", "Input", "Output" and "Pass/Fail". The tests cover most of the possible user interactions with the interface. The entire manual application test was performed using a physical Google Pixel 7[46] device running on the fourteenth version of Android, using mobile data to connect to the Internet and with a high battery charge level.

Seventy-seven tests were performed, seventy-six of which passed. One test marked as "Fail" concerned the lack of use of the Polish language in the application after its restart (it is described in the section on encountered issues). Each test case was noted after any interaction with the application interface to ensure the widest possible coverage. The process started with the application being launched and every possible operation that had not been tested yet was performed. Each test case containing interactions with the database was simultaneously verified by displaying the Firebase console to confirm its success. Some of the tests required testing the scenario several times due to the specificity of the test case, e.g. testing whether the application, after receiving credentials during login/registration, informs the user in an appropriate way via the toast component. All the tests that were run are listed in the table placed in Appendix B: Manual Test Table.

7.2 User Testing

The described phase was conducted to test the application on various Android devices and gather feedback that introduced useful insights regarding usage, features and the overall user experience. The process consisted of three main elements.

a) Application distribution was carried out by sharing a link to a OneDrive[47] drive containing the TravelBuddy APK file. This ensured that each tester would receive the same version of the application. This allowed for quick and easy distribution of the file and then a smooth installation on devices by the testers.

b) A questionnaire containing 10 questions (nine regarding user experience, one in which one could specify suggestions for the development of the application and report bugs) was constructed to gather feedback from testers after interacting with the application. The link to the survey was distributed to testers shortly after the APK file was made available (survey available in Appendix C).

c) The entire survey was managed by Google Forms[48], which ensured automatic data collection after sending and visual presentation of statistics regarding the survey.

Three bugs were reported. The first one concerned the ability to change the username to an existing one in the system, which allowed editing user data with the same username. The problem was solved by changes described in the "Encountered issues" section.

The other two concerned errors in displaying data on the "Social" screen in the "Trips" tab. One of the testers rightly pointed out the need to refresh the screen by swiping down every time the user wants to see trips shared with friends. This was solved by an update that introduced an automatic refresh of the screen along with navigation to it, so the user no longer has to perform this gesture every time. The third bug concerns displaying a shared trip of a friend multiple times on the screen in the case of having more than one friend. An investigation was conducted to check what is responsible for the duplicate data. It turned out that the data is fetched several times but only in some cases. The data is not duplicated in the database. This issue has not been resolved and requires future investigation. However, it does not disrupt the application in a noticeable way, it only affects the user experience.

The combination of rating based questions and an open-ended final question was an accessible and effective way to assess the strengths and weaknesses of the application but also to indicate areas for improvement that will positively impact the product in future development.

7.3. Automated Testing

The process of creating automated tests was carried out by creating sixteen instrumented tests using Jetpack Compose and the Android Studio testing environment. The elements that were tested concern four areas of the application. The main goal of the performed tests is to ensure the functionality of navigation using UI elements. "AndroidJUnit4"[49] was used to write these tests.

- a) At first, the tests concerning the main top app bar verify the presence of the notification icon and the account icon in this UI element. Then, the interaction with the interface is simulated by pressing these icons to check if they navigate to the appropriate screens (notification centre, account panel)
- b) The first test that was performed on the navigation bar component and it concerns the verification of the items that are in it. Then, the interaction with them is tested, namely the items "My Trips", "Explore", "Social" navigate to their corresponding screens, and the item "Add" calls the bottom sheet component that displays three buttons "Add trip" "Add phrase" and "Add friend".
- c) Tests related to the user login screen involved ensuring that the login button is disabled when nothing has been entered the one or both text fields and checking if the button is enabled after entering text into both text fields.
- d) The first two tests of the registration screen involved checking if the sign up button is disabled when the text fields are empty, and if the button is enabled when the fields are filled in. Then, the reaction after pressing the button is tested when the entered data symbolises password mismatch or invalid email format. This checks that the application does not proceed with the registration process when incorrect data is entered.

All sixteen tests were tested in 44 seconds on the emulated Pixel 7 device, as shown in the table below.

Tests	Duration	Pixel_7_API_34
✓ ✓ Test Results	44 s	16/16
✓ ✓ MainTopAppBarTest	12 s	4/4
✓ notificationIcon_isDisplayed	5 s	✓
✓ clickingAccountIcon_navigatesToAccountScreen	2 s	✓
✓ accountIcon_isDisplayed	2 s	✓
✓ clickingNotificationIcon_navigatesToNotificationScreen	1 s	✓
✓ ✓ MyNavigationBarTest	9 s	5/5
✓ navigationBarItems_areDisplayed	1 s	✓
✓ clickingMyTrips_navigatesToMyTripsScreen	1 s	✓
✓ clickingExplore_navigatesToExploreScreen	1 s	✓
✓ clickingAddSheetButton_showsBottomSheet	2 s	✓
✓ clickingSocial_navigatesToSocialScreen	1 s	✓
✓ ✓ SignInScreenTest	7 s	3/3
✓ signInButton_shouldBeEnabled_whenTwoFieldsEntered	3 s	✓
✓ signInButton_shouldBeDisabled_whenFieldsAreEmpty	1 s	✓
✓ signInButton_shouldBeDisabled_whenOneFieldEntered	2 s	✓
✓ ✓ SignUpScreenTest	13 s	4/4
✓ clickingSignUpButton_showsPasswordMismatch_onDifferentPasswc	5 s	✓
✓ clickingSignUpButton_staysOnSignUpScreen_onInvalidEmail	3 s	✓
✓ signUpButton_shouldBeEnabled_whenDataEntered	3 s	✓
✓ signUpButton_shouldBeDisabled_whenFieldsAreEmpty	1 s	✓

Figure 47: Unit tests results.

Each of these tests provides comprehensive validation of the described application functionalities by interacting with them and checking the reaction. Due to limited time, no more tests were performed regarding interaction with the UI. There could be many more of them, and they would be much more in-depth. Ultimately, every possible component of the application should be tested. An attempt was made to perform tests that would check the functionality of Firebase services by creating mock objects, however, due to the large number of errors and limited time, they were not implemented in the final version of the project.

Chapter 8. Critical Evaluation

8.1 Project Overview and Initial Objectives

The aim of the project was to create a mobile application that would accompany the user during the planning of the trip and its course. In order to guarantee wide and easy access for a large group of users, it was decided that the application would be created for smartphones with the Android system. All functional requirements of the project were placed in the MoSCoW prioritisation, which lists and determines the importance of each of the features.

All features from the "Must-have" section have been implemented in full. The user is able to register for the application to have their own account, which allows them to collect data and have access to it on any device on which the application is supported. Access to data is provided by a cloud database, which also supports adding and reading data in offline mode. In addition, the application is able to provide external data on places located in the world by completing the text search for geographic locations, displaying places of interest in these locations, displaying images related to the

destination and also allowing the user to interact with the map, which is enriched with several useful features. The features specified in the "Should-have" section have been mostly implemented. The user can plan trips along with its visualisation on the map. He is also able to share trips and phrases with his friends. However, the possibility of reacting and commenting on shared items has not been implemented, and the user is also not able to publish media in the application. The "Could-have" section has features that have not been implemented in most cases. The application does not have system notifications to notify the user about upcoming trips, a trip recommendation system and a section containing information useful in an emergency. However, the functionality of changing the language in the application to Polish has been achieved.

During the continuous development of the project, limitations related to the availability of solutions were also learned, which had an impact on the abandonment of certain ideas, as well as the limited time for their implementation. Some of the features in relation to the initial assumptions have not been implemented, however, despite the lack of these functions, the created product is a usable application that is able to have a real impact on the issues for which it is intended.

8.2 Methodology

Choosing the right agile methodology for project development was one of the most important elements of the entire process. Due to the characteristics of the project implementation, which required weekly meetings with the supervisor, the Scrum framework was chosen intentionally as an agile methodology that provides flexibility and adaptability.

The idea of working in weekly sprint cycles contributed to the progressive development of the project, preventing the buildup of hard-to-fix bugs. The iterative approach, which consisted of frequent delivery of new functional code and receiving feedback, allowed for taking effective steps that prevented longer downtimes in the case of encountered problems and allowed for focusing on priorities that may change from week to week. It also opened up the possibility of developing several features at the same time, by setting smaller goals in relation to planned functionalities. The supervisor's weekly review ensured that the project was going in the right direction and prevented late-stage rework that could delay the implementation of the planned features.

Using tools visualising the project's progress also had a huge impact on identifying delays. This gave the opportunity to plan activities in such a way as to prioritise those that must be applied in order to enable the implementation of the next ones. Recording decisions at weekly meetings in "minutes" not only served as a document to track progress but also provided transparency and allowed to avoid misunderstandings with the supervisor. The entire process required constant coordination and commitment from the supervisor and the project author throughout the development process. Without adopting such a development strategy, the project could have been completed in a much more limited state.

8.3 Design and Implementation

The implementation of the application in the MVVM architecture was a good choice considering the solutions used. It gave an advantage in the application's maintainability and testability by separating the UI and business logic. However, in the case of further development of the application, this approach could prove complex. A more modular approach should be considered by breaking down the ViewModels into smaller units, which would make the application easier to manage and more readable.

Using Jetpack Compose to create the UI significantly sped up the code implementation process, which allowed us to gain time for implementing more complex functionalities. At the same time, the created interface seems intuitive and easy to use, which is confirmed by the results of the user experience survey sent by testers.

The authentication system performed by Firebase Authentication worked without any problems and fully fulfilled its role. The implemented functions give the user a sense of control over the account and their data. In future work, it would also be necessary to implement functions to assign a current account to another email in case of such a need.

The Firestore database design is well-structured and functional. However, it has one major drawback, namely the collection of the data author's name for each trip and phrase. The reason for the solution of this issue is described in the "Encountered Issues" section. In future work, this solution should be changed to a more thoughtful and efficient one.

The used APIs fulfil their functions as a helpful tool that helps in travelling. However, their selection was very limited because many more advanced APIs require paid plans that allow for greater possibilities. In the absence of these limitations, it would be possible to create an application that would use more advanced solutions that would offer more advanced tools, which would automatically increase the attractiveness of the application.

8.4. Testing, Quality Assurance and User Feedback

Application testing began in the final phase of development and consisted of three phases.

Manual testing contributed to the in-depth testing of the application in numerous possible scenarios. In this way, the entire interface and the operation of authentication along with interactions with the database and APIs were tested. This is a method that is not as efficient as automated testing, but its execution also affects the identification of errors and is worth documenting.

User testing, which was carried out by the testers, allowed the identification of several bugs, the resolution of which improved the user experience. Along with testing, insights were also collected regarding the general use of TravelBuddy. Nine questions answered by five users along with the results are available in Appendix 3.

The overall feedback that was collected indicates a positive reception of TravelBuddy. Considering all the answers collected, the aspects of intuitive interface and ease of navigating the UI have been achieved. The functionalities regarding authentication and the friend request system also meet the requirements of the testers. The questionnaire had a positive impact on the development of the project, but it could be divided into two parts: questions regarding user experience and questions typically constructed to carry out testing activities in various aspects.

The application also has instrumented tests to test some components found in the application. They consist of verifying the correct functionality of components in the UI. However, performing sixteen automated tests for an application such as TravelBuddy is definitely not enough. Each screen and each component included should be thoroughly tested, including checking the services that have been implemented in the application. This would provide faster and more reliable verification of the application's operation.

8.5. Reflection on Outcomes

The entire process of creating the project was highly organized in terms of the agile methodology and the correct selection of tools. It contributed to the optimisation of the creation process, and as a result, final product mostly meets the assumptions that were presented at the beginning of development. The cost of refining the core functionalities was the abandonment of plans for certain other application features, the implementation of which would make the product even more complete and functional. However, despite the shortcomings due to which the application loses some of its usefulness, it is still a complete tool that will help optimise the user's journey experience by assisting him during various stages of the journey.

Chapter 9. Conclusion

Travelling has become a popular activity that is practised by almost every person. It has an impact on many positive aspects of a person's life that can contribute to improving well-being and personal development. Despite these benefits, travelling can be challenging for some in various aspects. The key to getting rid of any fears associated with travelling is a thorough preparation for it.

Nowadays, a smartphone is a device that is always at the user's fingertips, ready to quickly present a solution that will help in any situation. The TravelBuddy smartphone application was created to help the user plan and manage their trips. Its development methodology provided flexibility and adaptability during its execution. It uses proven solutions in terms of authentication and database that are securely managed through Cloud Firestore and Firebase Authentication. Its functionality is enriched by the support of API services, which increases its attractiveness in the eyes of the user. In addition, it is created for Android devices, which guarantees a wide range of potential users.

Despite the successful implementation of essential user features, the app still has a lot of potential that could be revealed in future updates. They would focus on integrating new features and improving the current ones to create the ultimate travel assistant.

10. References

- [1] L. Polok, "Benefits of Travel: 14 Things Travel Teaches You," *www.europelanguagejobs.com*, Sep. 12, 2023. <https://www.europelanguagejobs.com/blog/benefits-of-travel> (accessed Jun. 02, 2024).
- [2] L. Health, "The (Mental) Health Benefits of Traveling," *www.leehealth.org*, Nov. 09, 2021. <https://www.leehealth.org/health-and-wellness/healthy-news-blog/mental-health/the-mental-health-benefits-of-traveling> (accessed Jun. 02, 2024).
- [3] S. Bhandari, "What to Know About Hodophobia," *WebMD*, Feb. 24, 2024. <https://www.webmd.com/anxiety-panic/trip-a-phobia> (accessed Jun. 02, 2024).
- [4] Android, "Android," *Android*, 2016. https://www.android.com/intl/en_uk/ (accessed Jul. 05, 2024).
- [5] Statista, "International tourist arrivals by region," *Statista*, Oct. 05, 2022. <https://www.statista.com/statistics/186743/international-tourist-arrivals-worldwide-by-region-since-2010/> (accessed Jun. 02, 2024).

- [6] "Global: mobile app downloads by segment 2017-2025," *Statista*, Nov. 16, 2023.
<https://www.statista.com/forecasts/1262881/mobile-app-download-worldwide-by-segment>
(accessed Jun. 04, 2024).
- [7] StatCounter, "Mobile Operating System Market Share Worldwide," *StatCounter Global Stats*, 2024. <https://gs.statcounter.com/os-market-share/mobile/worldwide> (accessed Jun. 01, 2024).
- [8] Oberlo, "Most Popular Electronics Worldwide [July 2022 Update]," *www.oberlo.com*.
<https://www.oberlo.com/statistics/most-popular-electronics> (accessed Jun. 03, 2024).
- [9] S. Dias and V. A. Afonso, "Impact of Mobile Applications in Changing the Tourist Experience," *European Journal of Tourism, Hospitality and Recreation*, vol. 11, no. 1, pp. 113–120, Dec. 2021, doi: <https://doi.org/10.2478/ejthr-2021-0011>.
- [10] Statista, "Google Play Store: number of apps 2019 | Statista," *Statista*, 2019.
<https://www.statista.com/statistics/266210/number-of-available-applications-in-the-google-play-store/> (accessed Jun. 06, 2024).
- [11] L. Parro, "Bachelor's thesis Degree program in Hospitality Management Production in Management of Services 2013," 2013. Accessed: Jun. 05, 2024. [Online]. Available: <https://core.ac.uk/download/pdf/38098263.pdf>
- [12] C.-C. Chen, "Would You Be More Satisfied with Your Life If You Travel More Frequently?," *ResearchGate*, Jan. 2020.
https://www.researchgate.net/publication/347386688_Would_You_Be_More_Satisfied_with_Your_Life_If_You_Travel_More_Frequently (accessed Jun. 01, 2024).
- [13] "Wanderlog - Trip Planner App – Apps on Google Play," *play.google.com*.
https://play.google.com/store/apps/details?id=com.wanderlog.android&hl=en_GB&pli=1 (accessed Jun. 13, 2024).
- [14] "Polarsteps - Travel Tracker – Apps on Google Play," *play.google.com*.
https://play.google.com/store/apps/details?id=com.polarsteps&hl=en_GB
- [15] "Triplt: Travel Planner – Apps on Google Play," *play.google.com*.
https://play.google.com/store/apps/details?id=com.tripit&hl=en_GB (accessed Jun. 13, 2024).
- [16] GitHub, "GitHub," *GitHub*, 2024. <https://github.com/>
- [17] Figma, "Figma: the Collaborative Interface Design tool.," *Figma*, 2016. <https://www.figma.com/>
- [18] "Material 3 Design Kit | Figma Community," *Figma*, 2022.
<https://www.figma.com/community/file/1035203688168086460> (accessed Jun. 01, 2024).
- [19] "Material Theme Builder | Figma Community," *Figma*, 2023.
<https://www.figma.com/community/plugin/1034969338659738588/Material-Theme-Builder>
(accessed Jun. 02, 2024).
- [20] "Figma," *Figma*, 2022.
<https://www.figma.com/community/plugin/740272380439725040/Material-Design-Icons> (accessed Jun. 02, 2024).
- [21] Google, "Cloud Firestore | Firebase," *Firebase*, 2019.
<https://firebase.google.com/docs/firestore> (accessed Jun. 13, 2024).

- [22] Google, "Firebase Authentication | Firebase," *Firebase*, 2019. <https://firebase.google.com/docs/auth> (accessed Jun. 03, 2024).
- [23] "Android Studio Koala | 2024.1.1 (June 2024)," *Android Developers*, 2024. <https://developer.android.com/studio/releases/past-releases/as-koala-release-notes> (accessed Jun. 03, 2024).
- [24] *Kotlin*, 2020. <https://kotlinlang.org/> (accessed Jun. 03, 2024).
- [25] "JetBrains: Developer Tools for Professionals and Teams," *JetBrains*. <https://www.jetbrains.com/> (accessed Jun. 04, 2024).
- [26] M. Moskała, "Why using Kotlin Coroutines?," *Kt.academy*, 2023. <https://kt.academy/article/cc-why> (accessed Jun. 15, 2024).
- [27] "Jetpack Compose UI App Development Toolkit," *Android Developers*. <https://developer.android.com/compose> (accessed Jun. 04, 2024).
- [28] Google, "Material Design," *Material Design*, 2023. <https://m3.material.io/> (accessed Jun. 05, 2024).
- [29] Scrum.org, "What is Scrum?," *Scrum.org*, 2020. <https://www.scrum.org/resources/what-scrum-module> (accessed Jun. 04, 2024).
- [30] Atlassian, "What is a Kanban Board?," *Atlassian*. <https://www.atlassian.com/agile/kanban/boards#:~:text=A%20kanban%20board%20is%20an> (accessed Jun. 03, 2024).
- [31] "What is MoSCoW Prioritization? | Overview of the MoSCoW Method," *www.productplan.com*. <https://www.productplan.com/glossary/moscow-prioritization/#:~:text=MoSCoW%20prioritization%2C%20also%20known%20as> (accessed Jun. 10, 2024).
- [32] Ramotion, "Understanding MVVM: Model-View-ViewModel Architecture," *Web Design, UI/UX, Branding, and App Development Blog*, May 01, 2023. <https://www.ramotion.com/blog/what-is-mvvm/> (accessed Jul. 04, 2024).
- [33] "Retrofit," *Github.io*, 2013. <https://square.github.io/retrofit/> (accessed Jun. 28, 2024).
- [34] "Pixabay API Documentation," *pixabay.com*. <https://pixabay.com/api/docs/> (accessed Jun. 27, 2024).
- [35] "API documentation and Playground for Geoapify maps and components," *apidocs.geoapify.com*. <https://apidocs.geoapify.com/> (accessed Jul. 15, 2024).
- [36] "osmdroid-android 6.1.20 javadoc (org.osmdroid)," *Javadoc.io*, 2024. <https://javadoc.io/doc/org.osmdroid/osmdroid-android/latest/index.html> (accessed Jul. 06, 2024).
- [37] "OpenStreetMap," *OpenStreetMap*, 2024. <https://www.openstreetmap.org/#map=6/54.91/-3.43> (accessed Jul. 04, 2024).
- [38] "Map Marker SVG." <https://www.svgrepo.com/svg/302636/map-marker> (accessed Aug. 04, 2024).

[39] "TravelBuddy Figma Design."

<https://www.figma.com/design/42eGAI5qXsYTGTWKXsZInE/Travel-App-UI-Design?node-id=0-1&node-type=CANVAS&t=OOcCtwkqQOiMSFTD-0>

[40] "Material Theme Builder," *material-foundation.github.io*. <https://material-foundation.github.io/material-theme-builder/> (accessed Jun. 07, 2024).

[41] *Flaticon*. https://www.flaticon.com/free-icon/luggage_144384 (accessed Jun. 10, 2024).

[42] R. Se, "Jetpack Compose - DatePicker Easy to Implement," *Programming Headache*, Nov. 28, 2023. <https://programmingheadache.com/2023/11/28/jetpack-compose-datepicker-easy-to-implement/> (accessed Jul. 24, 2024).

[43] "GeoDB Cities API," *Wirefreethought.com*, 2024. <http://geodb-cities-api.wirefreethought.com/> (accessed Jun. 15, 2024).

[44] "App Architecture: Data Layer - DataStore," *Android Developers*. <https://developer.android.com/topic/libraries/architecture/datastore> (accessed Jul. 16, 2024).

[45] Meet Patadia, "Local notification in Android with Jetpack compose - Meet Patadia - Medium," *Medium*, Jan. 28, 2024. <https://meetpatadia9.medium.com/local-notification-in-android-with-jetpack-compose-437b430710f3> (accessed Jul. 16, 2024).

[46] "Google Pixel 7 - Full phone specifications," *www.gsmarena.com*. https://www.gsmarena.com/google_pixel_7-11903.php (accessed Aug. 15, 2024).

[47] "OneDrive for Business | Secure Cloud Storage | Cloud Backup," *www.microsoft.com*. <https://www.microsoft.com/en-gb/microsoft-365/onedrive/onedrive-for-business> (accessed Aug. 29, 2024).

[48] Google, "Google Forms: Free Online Surveys for Personal Use," *Google.co.uk*, 2019. <https://www.google.co.uk/forms/about/> (accessed Aug. 25, 2024).

[49] "AndroidJUnit4 | Android Developers," *Android Developers*, 2024. <https://developer.android.com/reference/androidx/test/runner/AndroidJUnit4> (accessed Aug. 27, 2024).

[50] "Coil," *coil-kt.github.io*. <https://coil-kt.github.io/coil/> (accessed Jul. 05, 2024).

11. Appendices

Appendix A. Ethics Form

03/07/2024

For your information, please find below a copy of your recently completed online ethics assessment

Next steps

Please refer to the email accompanying this attachment for details on the correct ethical approval route for this project. You should also review the content below for any ethical issues which have been flagged for your attention

Staff research - if you have completed this assessment for a grant application, you are not required to obtain approval until you have received confirmation that the grant has been awarded.

Please remember that collection must not commence until approval has been confirmed.

In case of any further queries, please visit www.aber.ac.uk/ethics or contact ethics@aber.ac.uk quoting reference number 29940.

Assessment Details

AU Status

Undergraduate or PG Taught

Your aber.ac.uk email address mat78@aber.ac.uk

Full Name

Maciej Traczyk

Please enter the name of the person responsible for reviewing your assessment.

Reyer Zwiggelaar

Please enter the aber.ac.uk email address of the person responsible for reviewing your assessment

rrz@aber.ac.uk

Supervisor or Institute Director of Research Department

cs

Module code (Only enter if you have been asked to do so)

Proposed Study Title

Android application for travelling

Proposed Start Date

04.06.2024

Proposed Completion Date

04.09.2024

Are you conducting a quantitative or qualitative research project?

Quantitative

Does your research require external ethical approval under the Health Research Authority?

No

Does your research involve animals?

No

Does your research involve human participants?

Yes

Are you completing this form for your own research?

Yes

Does your research involve human participants?

Yes

Institute

IMPACS

Please provide a brief summary of your project (150 word max)

The project will be focused on creating an Android application which accompanies users before, during and after their trips. Every user will be able to create an account in the application and start using the tools to plan and manage expeditions, explore recommended places around the world and share memories with their friends.

I can confirm that the study does not involve vulnerable participants including participants under the age of 18, those with learning/communication or associated difficulties or those that are otherwise unable to provide informed consent?

Yes

I can confirm that the participants will not be asked to take part in the study without their consent or knowledge at the time and participants will be fully informed of the purpose of the research (including what data will be gathered and how it shall be used during and after the study). Participants will also be given time to consider whether they wish to take part in the study and be given the right to withdraw at any given time.

Yes

I can confirm that there is no risk that the nature of the research topic might lead to disclosures from the participant concerning their own involvement in illegal activities or other activities that represent a risk to themselves or others (e.g. sexual activity, drug use or professional misconduct).

Yes

I can confirm that the study will not induce stress, anxiety, lead to humiliation or cause harm or any other negative consequences beyond the risks encountered in the participant's day-to-day lives.

Yes

Please include any further relevant information for this section here:

The participants' feedback will be taken into account and will enable the development of the application thanks to the proposed new and improved current functionalities

Where appropriate, do you have consent for the publication, reproduction or use of any unpublished material?

Not applicable

Will appropriate measures be put in place for the secure and confidential storage of data?

Yes

Does the research pose more than minimal and predictable risk to the researcher?

Not applicable

Will you be travelling, as a foreign national, in to any areas that the UK Foreign and Commonwealth Office advise against travel to?

No

Please include any further relevant information for this section here:

Is your research study related to COVID-19?

No

If you are to be working alone with vulnerable people or children, you may need a DBS (CRB) check. Tick to confirm that you will ensure you comply with this requirement should you identify that you require one.

Yes

Declaration: Please tick to confirm that you have completed this form to the best of your knowledge and that you will inform your department should the proposal significantly change.

Yes

Please include any further relevant information for this section here:

Appendix B: Manual Test Table

Test Ref	Description	Input	Output	Pass/Fail
1	(When user is not signed in) Clicking TravelBuddy icon, displays "Sign In" screen	Clicking the app icon	"Sign In" screen appears	Pass
2	(When user is signed in) Clicking TravelBuddy icon displays "My Trips" screen	Clicking the app icon	"My Trips" screen appears	Pass
3	Clicking "VisibilityOff" icon button in text fields meant for password, reveals the password	Clicking the "Visibility off" icon button	Password gets revealed	Pass
4	Clicking "Create an account" text button navigates to "Sign up" screen	Clicking the "Create an account" text button	"Sign up" screen appears	Pass
5	Clicking "Forgot password?" text button invokes "Reset your password" dialog	Clicking the "Forgot password?" text button	"Reset your password" dialog appears	Pass

6	(When valid credentials are provided) Clicking "Sign in" button logs in the user and navigates to "My Trips" screen	Providing valid credentials and clicking the "Sign in" button	User is logged in to the app and "My Trips" screen appears	Pass
7	(When invalid credentials are provided) Clicking the "Sign in" or "Sign up" button invokes a toast with the information about the error	Providing invalid credentials and clicking the "Sign in" or "Sign up" button	A toast with the appropriate information about the error appears	Pass
8	"Sign in" and "Sign up" buttons are enabled only when all text fields on the screen are filled with text	Typing text in all text fields	"Sign in" or "Sign up" button gets enabled	Pass
9	(When valid credentials are provided) Clicking the "Sign up" button registers the user in the Firebase, sends the email with the verification link and invokes the dialog with verification information	Providing valid credentials and clicking the "Sign up" button	User is registered in the Firebase, email is sent, and the dialog appears	Pass
10	(When user is not verified) Clicking the "Sign in" button invokes the dialog with the information about verification	Providing valid credentials of not verified user and clicking the "Sign in" button	The dialog appears	Pass
11	The "Password reset" dialog after providing an email sends the email with the password reset link	Providing email address and clicking "Send" text button	The email is sent	Pass
12	If trip exists in the database, trip card is displayed in the "My Trips" screen	(Existing trip data)	Card is displayed in the screen	Pass
13	Clicking "Details" button on the trip card navigates to "Trip details" screen with appropriate data	Clicking "Details" button	"Trip details" screen appears with appropriate data	Pass
14	Clicking "Share" icon button, invokes confirming dialog which sets the "shared" to true	Clicking "Shared" icon button and "Confirm" text button	The value is set to "true"	Pass
15	Clicking "Add" button in the navigation bar, triggers "Add data" bottom sheet	Clicking "Add" button	The bottom sheet appears	Pass
16	Clicking one of three buttons on the bottom sheet ("Add trip", "Add phrase", "Add	Clicking one of three buttons	Desired screen appears	Pass

	friend”) navigates to the desired screen			
17	Clicking the card with start date or end date in “Add a trip” screen, invokes the date picker	Clicking the card	The date picker appears	Pass
18	Clicking on the date in the date picker and clicking “OK” text button, sets the chosen date	Clicking the date and “OK” text button	The date gets updated with the chosen date	Pass
19	Text field in “Add a friend” screen, after providing a character, displays filtered users in the system	Providing a character in the text field	Lazy column with filtered users appears	Pass
20	Clicking the icon button “Done” in the top app bar with arrow back, is enabled when text field are filled with data and executes desired operation.	Providing required data in the text fields	“Done” icon button gets enabled and executes desired operation	Pass
21	Clicking the card “Visualise on map” navigates to the map, and marks the destination (and the trip plans) with the red marker (blue markers)	(Existing trip plan data) Clicking the “Visualise on map” card	Map is displayed with desired markers	Pass
22	Clicking the card “Check weather online” navigates to the web browser and searches for “(destination) weather”	Clicking the “Check weather online” card	Web browser with desired search appears	Pass
23	“Checklist:” card displays the first five unchecked items as preview	(Existing checklist “unchecked” items)	The card displays up to five unchecked items	Pass
24	Clicking “Checklist:” card navigates to “Checklist” screen	Clicking “Checklist:” card	“Checklist” screen appears	Pass
25	“List item” text button creates an item in the checklist	Clicking “List item” text button	Item is created	Pass
26	Clicking the item enables editing the text and shows the “Delete” icon button which enables removal of item	Clicking the text field item and “Delete” icon button	Item is enabled for editing and gets removed after clicking the icon button	Pass
27	Checking the checkbox next to the item, moves the item to the checked items section	Clicking the checkbox	Item gets placed in the checked items section	Pass
28	Clicking the “(number) checked item” text button shows the hidden list	Clicking the “(number)”	The list is visible	Pass

		checked item” text button		
29	“Trip plan:” card displays the first five items as preview	(Existing trip plan items)	The card displays up to five items	Pass
30	Clicking the “Trip plan” card navigates to “Trip plan” screen	Clicking “Trip plan:” card	“Trip plan” screen appears	Pass
31	“Trip plan” screen displays trip plans as cards and enables for editing and removal of this data by clicking the menu item under the “MoreVert” icon button	(Existing trip plan data)	Cards displays the data	Pass
32	“List item” text button navigates to “Add trip plan” screen	Clicking the “List item” text button	“Add trip plan” screen appears	Pass
33	Clicking “Set to private” button invokes confirmation dialog used for setting “shared” to “false”	Clicking the “Set to private” button and “Confirm” in the dialog	The value is set to “false”	Pass
34	“Set to private” button is disabled when “shared” is “false”	“shared” = false	The button is disabled	Pass
35	Text “Status: Private/ Friends only” indicates the state of the “shared”	“shared” = true/ false	“Status: Friends only/ Private”	Pass
36	Clicking the “MoreVert” icon button in the top app bar, triggers the trip menu	Clicking the “MoreVert” icon button	Menu gets invoked	Pass
37	Clicking the “Edit details” menu item navigates to “Edit details” screen	Clicking “Edit details” menu item	“Edit details” screen appears	Pass
38	Clicking the “Remove trip” menu item invokes the dialog which removes the trip from the database	Clicking “Remove trip” menu item	The dialog appears and after clicking “Confirm”, the trip gets removed	Pass
39	Clicking the “Notification” icon button in the main top app bar navigates to “Notification centre”	Clicking the “Notification” icon button	“Notification centre” appears	Pass
40	Clicking the “Account” icon button in the main top app bar navigates to “Account panel”	Clicking the “Account” icon button	“Account panel” appears	Pass

41	Clicking the "Sign out" icon button in the top app bar with arrow back navigates to "Sign in screen" and signs out the user from the application	Clicking the "Sign out" icon button	"Sign in" screen appears, and user is logged out	Pass
42	Clicking the "User profile" button navigates to "Profile" screen	Clicking the "User profile" button	"Profile" screen appears	Pass
43	Clicking "Remove all data" button removes all user data from the database	Clicking the "Remove all data" button	All data gets removed	Pass
44	Clicking "Delete account" button removes all data and removes the user from the system. After that the user is navigated to the "Sign in" screen	Clicking the "Delete account" button	The data and the user get removed from the Firebase, and the "Sign in" screen appears	Pass
45	Clicking the "Arrow back" icon button in the top app bar with arrow back navigates back to the previous screen	Clicking the "Arrow back" icon	The previous screen appears	Pass
46	Clicking "Your friends" list item navigates to "Your friends" screen	Clicking the "Your friends" list item	"Your friends" screen appears	Pass
47	"Your Friends" screen displays the current state of the friends of the user by displaying cards with users' data	(Existing data of user friends)	Cards with all friends appears	Pass
48	Clicking the "Delete" icon button in the card invokes the confirmation dialog for friend removal which removes a friend from the friend list	Clicking the "Delete" icon button and "Confirm" text button in the dialog	Friend gets removed and the card disappears	Pass
49	Clicking "Language" list item invokes the "Language selection" dialog	Clicking the "Language" list item	The dialog for language selection appears	Pass
50	Selecting the desired language option in the dialog and clicking "Save" text button resets the application and applies the chosen language in the app	Clicking the appropriate radio button and "Save" text button	Activity gets restarted and the chosen language is set in the application	Pass
51	After relaunching the app, the language stays Polish	Relaunching the app after selecting Polish	After the relaunch, the app goes back to the	Fail

		language as a preference	default language (English)	
52	Clicking "Theme" list item invokes the "Theme selection" dialog	Clicking the "Theme" list item	The dialog for theme selection appears	Pass
53	Selection the desired theme option in the dialog and clicking "Save" text button applies the chosen theme setting in the app	Clicking the appropriate radio button and "Save" text button	The chosen theme preference is set in the application	Pass
54	After relaunching the app, the theme preference is retrieved	Relaunching the app after setting the theme preference	After the relaunch, the app retrieves the last chosen theme preference	Pass
55	Clicking "About" list item navigates to "About" screen	Clicking the "About" list item	"About" screen appears	Pass
56	Clicking the "Map search" button navigates to "Map View" screen	Clicking the "Map search" button	"Map View" screen appears	Pass
57	Clicking the "Search" FAB in the "Map View" screen enables search bar on top of the screen	Clicking "Search" FAB	Search bar appears	Pass
58	Clicking the "Location" FAB in the "Map View" screen navigates to the last known location of the user	Clicking "Location" FAB	The marker which represents the last known location of the user is in the center of the map	Pass
59	After providing the valid address in the search bar, the map navigates to the desired place after clicking the "Search" icon button	Providing a valid address and clicking "Search" icon button	The map shows the searched place in the center	Pass
60	After providing valid data in the "Destination" and "Category" text fields, clicking "Search" button, the searched places appear as cards on the screen (Two text fields disappear to make the scrollable area larger)	Providing a valid data in the text fields and clicking "Search" button	The data is fetched in the cards and two text fields disappear	Pass
61	When the places are fetched, clicking "Map view" navigates to "Map view" screen with all fetched places and the	Clicking "Map view" button, after fetching the places	The "Map view" screen appears with the places and the	Pass

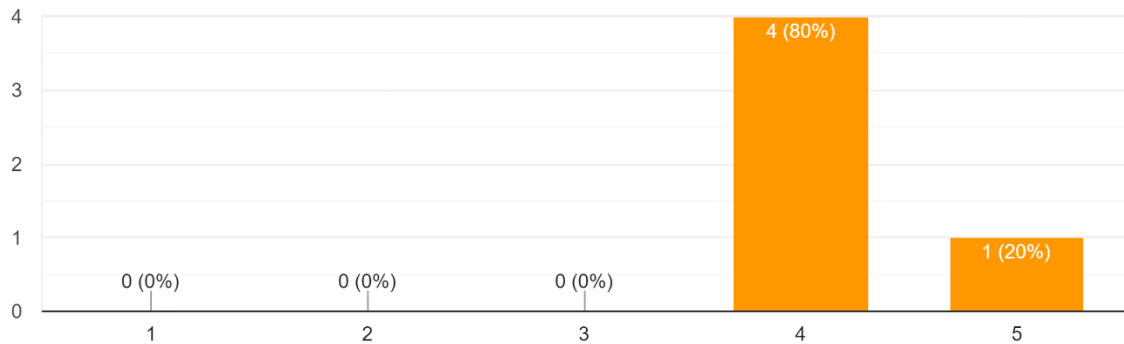
	destination marked on the map as markers		destination placed as the markers on the map	
62	Clicking on card of the place in "Explore" screen navigates to "Place details" screen	Clicking on the place card	"Place details" screen appears with the appropriate data	Pass
63	"Place details" screen displays the map in the card which shows the place location with the marker	Navigating to "Place details" screen	The map in the card displays the appropriate location	Pass
64	Clicking "Add to trip" button in "Place details" screen invokes a dialog which contains every trip of the user	Clicking "Add to trip" button	The dialog with the trips of the user appears	Pass
65	Clicking on the trip in the dialog navigates to add trip plan screen and fills the data of this place in the text field	Clicking on the trip of the user	"Add trip plan" screen appears	Pass
66	When fetching places in "Explore" screen and data is not available, the text informing the user about it appears	Fetching data but no data of the desired place available	"No places found. Try a different search" text appears	Pass
67	After sending a friend request, the chosen user receives it in the notification panel	Sending a friend request to another user	The notification appears in another user account	Pass
68	Clicking "Accept"/"Decline" text button in the notification, accepts/declines friend request	Clicking "Accept/Decline" text button	The friend request gets accepted/declined	Pass
69	The user is able to see their friends' trips if they are set to be visible for them	Friend's trip is set to be shared	The trip is visible in the "Trips" tab	Pass
70	After clicking on "Details" button on friend trip card, it navigates to "Trip details" screen but the components which allow to modify the data are disabled	Clicking "Details" button on the friend's trip card	The screen does not display elements of UI for editing	Pass
71	Swiping down on "Trips" and "Phrases" tabs in "Social" screen invokes refresh function to fetch the data on demand	Swiping down on "Trips"/"Phrases" tabs	The data gets refreshed	Pass

72	The trip cards display the shared trips of friends and the author of it in the "Social" screen ("Trips" tab)	(Existing trips of friends)	The cards display friends' trips and display name on the author in it	Pass
73	The cards display phrases of the user and their friends in the "Social" screen ("Phrases" tab)	(Existing phrases data of the user and their friends)	The cards display user and friends' phrases	Pass
74	The phrases cards of the user have "MoreVert" icon button to invoke a menu for phrase editing	(Existing phrase of the authenticated user)	The card has a "MoreVert" icon button which invokes a menu	Pass
75	Clicking "Edit phrase" menu item navigates the user to "Edit Phrase" screen	Clicking "Edit phrase" menu item	"Edit phrase" screen appears	Pass
76	Clicking "Remove phrase" menu item invokes the dialog which enables deletion of the phrase	Clicking "Remove phrase" menu item and "Confirm" in the dialog	The phrase gets removed	Pass
77	After first login to the app, "Set up your display name" screen appears which allows username set up	Existing user username = "(empty string)"	"Set up your display name" appears	Pass
78	(When the device has no internet access) After launching the application, the dialog appears which informs user about limited app functionality	(No internet access) Launching the application	The "Offline mode" dialog appears	Pass

Appendix C: Users Feedback

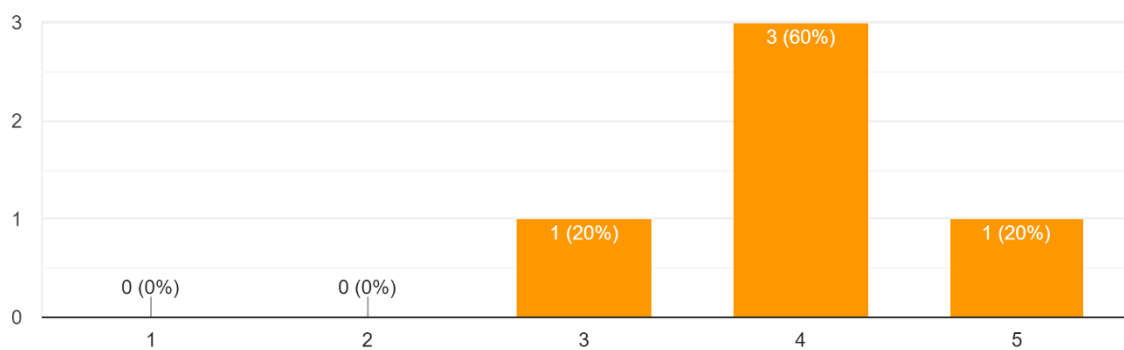
1. How would you rate your overall experience with the TravelBuddy app?

5 responses



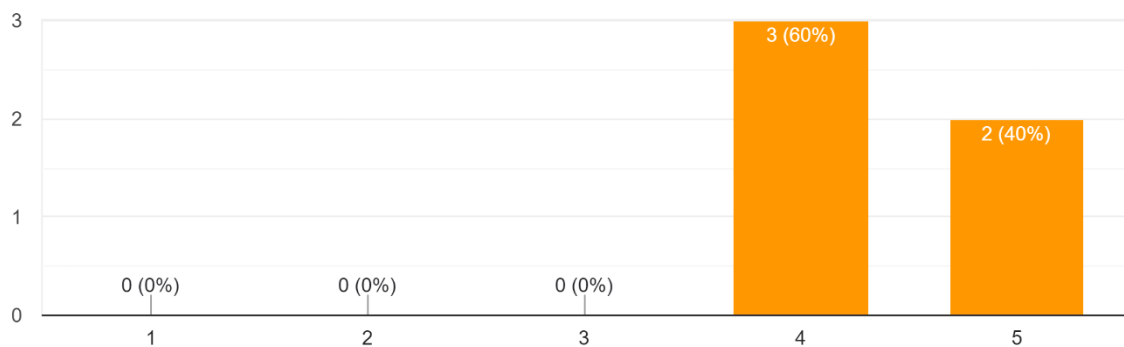
2. How easy was it to navigate through the TravelBuddy app?

5 responses



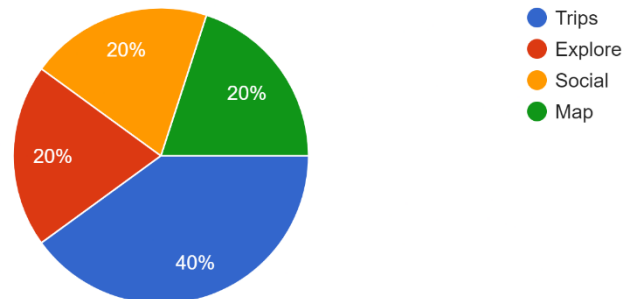
3. How would you rate the visual design and aesthetics of the app?

5 responses



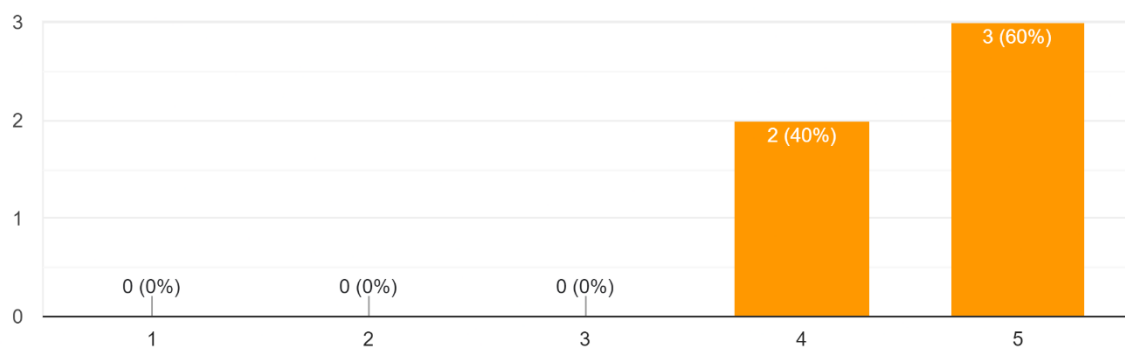
4. Which feature of the TravelBuddy app did you find most useful?

5 responses



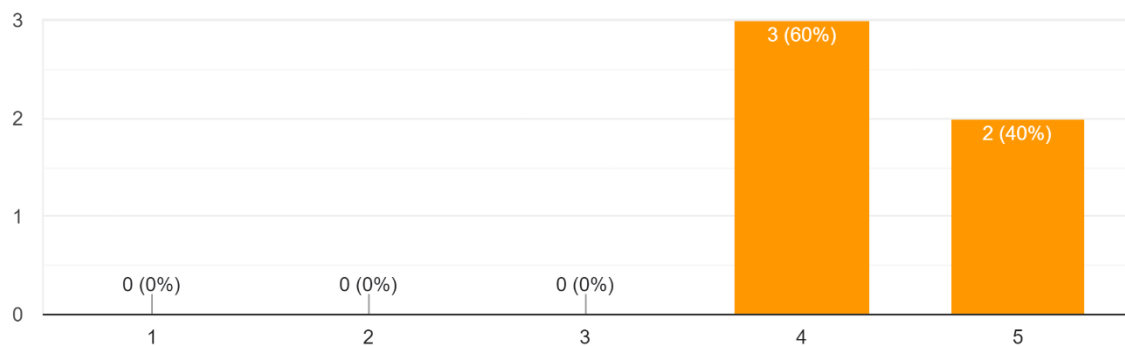
5. How satisfied are you with the process of sending and receiving friend requests in the app?

5 responses



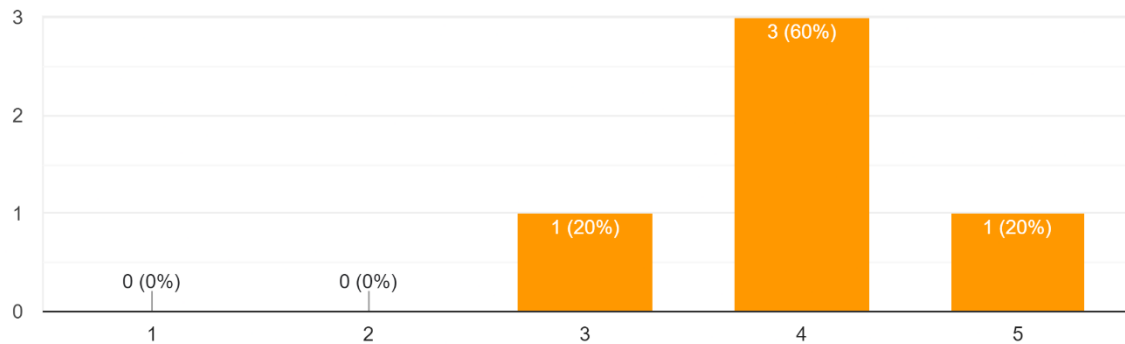
6. How effective do you find the trip management features?

5 responses



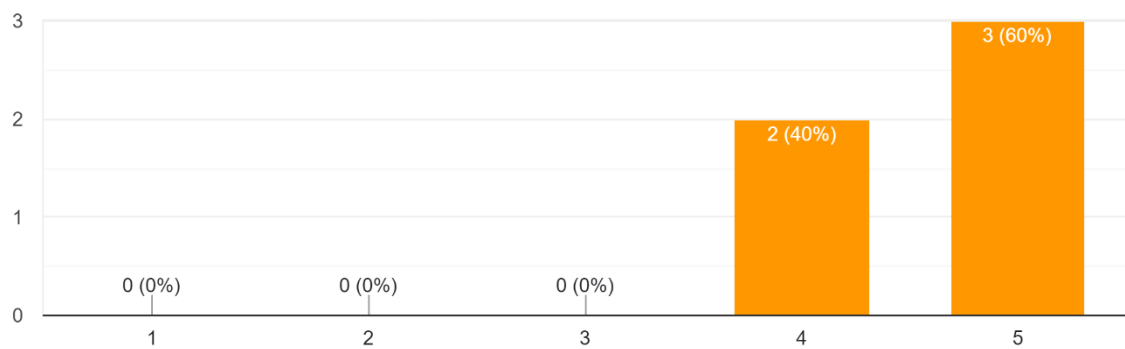
7. How well does the map functionality (search, markers, user location) meet your needs?

5 responses



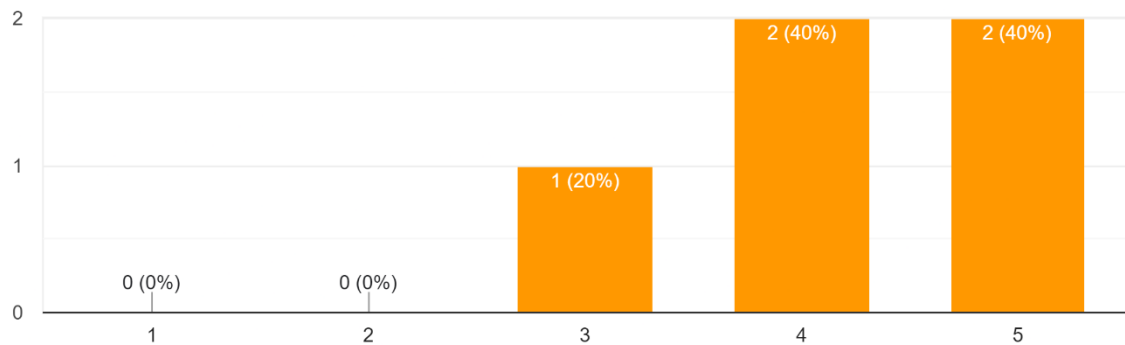
8. How secure and straightforward did you find the authentication and account management process?

5 responses



9. How likely are you to recommend TravelBuddy to a friend or family member?

5 responses



10. Is there anything specific you would like to see improved or any additional features you would like to suggest for the TravelBuddy app? Any bugs?

4 responses

The ability to remove friend requests I've already sent would be nice, also sharing photos taken during trips

Navigating to the place on the map by clicking the trip plan

Changing username to the existing one allows me to modify other user's trips and phrases

Additional features: I'd like to receive notifications which remind me about the upcoming trips.

Bugs: I need to refresh the social section (trips) every time to see friend's trips. One of the trips is displayed twice.