UNIVERSITY OF LEICESTER

Department of Informatics, University of Leicester

# CO7501 Individual Project

**Android Application - City Tour Planner**

Author Name: Muhammad Sohaib Furqan

University email: msf7@student.le.ac.uk

University ID: msf7

**Dissertation Draft (Word Count: xxx)**

Supervisor: Dr. Rayna Dimitrova

Second Marker: Prof. Reiko Heckel

[Submission Date: xx-xxx-xxxx]

## Abstract

We live in an era where most of our daily lives are consumed by never-ending professional commitments, to the extent that people are left with next to no time to attend to their hobbies, and travelling is no different. Most people are so occupied with work that they see little value in undertaking long trips that require setting aside fixed periods of time. City Tour Planner for Android is an attempt to address this problem, the idea being to allow users to schedule single-day, single-city trips at times of their choice. This will provide value to the users in terms of flexibility – enabling them to schedule trips around their work routine as they see fit. Users will be able to retrieve the types of places they specify. The application will also keep track of Scheduled, past and deleted trips.

## Declaration

## Dedications

I dedicate this work to my parents, who showed me how to lead to lead an exemplary life. Their never-ending sacrifices, relentless hard work and unceasing encouragement and dedication towards my growth is the only reason for me to be in a position to write this dissertation today. This is a good opportunity for a special mention of my personal superhero, my dad. I could have only dreamed of achieving a Masters degree from an institute as prestigious as the University of Leicester, had it not been for his blind faith in my abilities. Thank you mum and dad for everything. I hope I made you proud (finally) and also that there’s a lot more of that to come.

# Acknowledgements

I would like to start by expressing my thankfulness and gratitude to my worthy supervisor, Dr. Rayna Dimitrova, for her sincere guidance and friendly advice throughout the course of the project. If it hadn't been for her, my journey through the last part of my MSc degree probably won't have been as smooth. Thank you Dr. Rayna for being an ideal supervisor and showing me the way to go if I ever get the chance to be at your position.

It would also be unfair of me to not mention my second marker, Dr. Reiko Heckel, whose valuable insight and comments on my progress throughout the project were a constant source of self-reflection and improvement.

All the credit for this dissertation is also equally shared by my family. My parents, wife and sisters for pushing me on and not letting me give up.

# Table of Contents

# Chapter 1: Introduction

Technology is becoming increasingly pervasive in today's world. It has changed the way people perceive things and go about their daily routine. Most things that were considered tedious and time-consuming in the past can now be done with a few keystrokes. To add to that, the tech industry itself is evolving at a lightning pace. Each day, there are new developments and milestones that were seemingly unimaginable a decade ago are achieved. The reliance on technology is now ingrained into our lives to the extent that the present order of existence would be in total disarray without it. The advent of smartphones has been the cause of yet another paradigm shift in everyday utility computing. Tasks that required being hooked up to a computer screen to achieve can now be done on the go from the comfort and portability of a personal handheld device.

It goes without saying that the boom of technology offers enormous benefits that have changed life in almost all dimensions, but this ease instilled into our lives comes at a cost – work consumes most of our daily routine and most people seldom have the time to attend to their hobbies. Over time, maintaining a balance between professional commitments and leisure time has become exceedingly difficult. It would be nice if the reliance on technology could be used to address this problem in a way that minimizes this tradeoff.

City Tour Planner attempts to utilize the value offered by smartphones to provide users with a single-day trip planning application of a city. Users are able to register with the application and login to maintain their personalized records of scheduled, past and deleted trips. The application allows users to retrieve places at their desired location which can then be added to a trip. Also, there is an option to rate trips and places. User feedback can also be collected through the application which could prove handy for future improvements. Before selecting a place to add to a trip, users also have the option to view its details. Settings have also been provided to customize user experience by allowing to specify the category of places to be retrieved. The choice can be made from a wide variety of places such as banks, hotels, airports, museums, parks and points-of-interest among others.

## Aims and Objectives

Although it builds upon other similar trip-planner apps, the purpose of the project is twofold. From a value-offering perspective, a successful implementation will relieve users of having to set aside dedicated time periods for trips, but instead will be able to organize short daily trips around their work routines. With respect to demonstration of technical ability, the project will serve as a means to access software development abilities along with the capacity to execute the complete software development life-cycle within a fixed period of time.

- The following are the objectives of the project:
- Allow users to set up a profile
- Allow users to customize trip experience
- Retrieve and present relevant information from third-party sources in an appropriate manner
- Customize retrieved information according to user interests
- Provide an estimate of the cost incurred by a specific trip
- Include pictures of venues along with user reviews of the most popular spots
- Incorporate user feedback and suggestions for improving the user experience of the application

## Challenges

- Get up to speed with the latest technologies that are being used to program android applications
- Research the best suited database technology for the project given the nature and requirements of the project
- Identify and explore appropriate APIs for retrieving data for the application
- Slow loading of images from the Google Places Web Service
- Maintaining user sessions and showing only relevant data for each logged in user
- Adopting appropriate best practices and techniques while working with the database for consistency

## Risks and Mitigation Strategies

**Risk 1:** The author has some experience in Java programming but none in building native Android applications.

**Mitigation strategy:** To enable the design of a robust product with good user experience, it is required that knowledge of the Android SDK be gained, refreshed and updated throughout the course of the project.

**Risk 2:** Possibility of backward compatibility issues.

**Mitigation strategy:** The constantly evolving nature of the Android platform often gives rise to backward compatibility problems that can be a source of frustration and inefficient utilization of resources. It is therefore necessary to be aware of API level specific changes when developing the features of the application.

**Risk 3**: Constraints enforced by the APIs or other software artifacts being used.

**Mitigation strategy:** One of the drawbacks of using a third-party API or service for development purposes comes in the form of the limitations they bring to the table in terms of functionality being offered and the security risks associated etc. It is needed that the developer continuously researches approaches to deal with, and work around, these limitations or threats.

**Risk 4:** IDE-generated files a likely source of merge conflicts in the project repository.

**Mitigation strategy:** Various configuration files are generated and modified by an IDE during the course of the development of the project. Most of these files are not controlled directly be the developer but rather are changed by the IDE as needed. These files can induce frustrating merge conflicts which become evident only when code is cloned on a different machine. If these conflicts are too many, as is the case in most projects, steps must be taken to resolve them in a manner that the project is not disturbed. If risky, it is wiser to work on a single machine. The developer has kept multiple copies of the current state of the project in an attempt to guard against this risk.

**Risk 5:** Possibility of missing deliverable deadlines or compromising project functionality

**Mitigation strategy:** To ensure timely submission of project deliverables and accurate design of project functionality, the author has maintained close contact with the project supervisor and the second marker at all times. This has enabled constant feedback and timely rectification where needed.

## Literature Review

Similar android applications including TripIt, RoadTrippers and Sygic Travels were studied as part of the literature review for this project. The following is a brief description of the main features offered by each of these applications. The reading list section indicates resources used to learn android application development. This section concludes with a look at the ways in which project design and implementation has been influenced by this background research.

### TripIt

TripIt is a travel planning application which allows for creating an itinerary for each trip. Among other features, this application allows users to forward trip confirmation emails to a dedicated email address (plans@tripit.com). It then creates an itinerary out of the forwarded emails, separately for each trip. Users also have access to a master itinerary where they can access every trip. These plans can then be added to a calendar or shared with friends.

### RoadTrippers

This is a web and mobile based application intended to help users in planning road trips. Focusing specifically on areas in the US and Canada, users of this application can choose from over 5 million locations and can use category filters to customize preferences. The application also provides users with turn-by-turn navigation of their desired location.

### Sygic Travels

A GPS navigation software that works along the lines of Google Maps but was the first to offer offline maps. Though Google has also started offering this feature now, there is still a huge difference. Sygic maps are far more memory efficient compared to Google and also allow for storing larger maps for offline usage.

**Reading List**

- Google Codelabs – android fundamentals
- Android developer guides
- Android material design guidelines

## Outcomes of the background research

A detailed study of the aforementioned applications provided the knowledge and the background necessary for designing a robust, state-of-the-art application. The best practices pertinent in industry were also easily evident. Although details of the coding practices and design patterns could not be retrieved from the high-level study of the applications, sufficient insight was gained to identify technologies being used, or in some cases, their alternatives. Specifically, the research led to getting familiar with the use of material design guidelines, which are extensively used in all modern applications and enable the design of rich UI/UX artifacts. Since the project involves fetching data from various APIs, the study also prompted a comparison of the various options that were available in this regard and helped in making a more informed decision.

## Dissertation Structure

This chapter contains seven chapters:

Chapter 1 of the dissertation will look at the motivation behind the project, its objectives and scope. It also looks at the literature survey conducted as part of the background study for the project

Chapter 2 will be directed at the foundations laid down for starting the project, including a discussion of the general approach throughout the course of the project, the software architecture, the milestones and the plans for risk management.

Chapter 3 will be devoted to detailing the requirements specifications for the project, covering high-level and detailed requirements. Non-Functional Requirements will also be included.

Chapter 4 covers the various details of the project at the technical level.

Chapter 5 provides a walkthrough of the various modules (classes, methods etc.) that constitute the project. The purpose served by each module in the broader context of the overall application will be elaborated

Chapter 6 discusses the approaches used for testing the application at length

Chapter 7 closes the dissertation with a glance at some code snippets from the application, ideas for future enhancements to improve the application, and some concluding remarks. The references for writing the dissertation will also be provided in this section.

## Summary

This chapter lays down the foundation for the rest of the dissertation. It addresses the motivation behind the project and attempts to explain to the reader why it is different and useful, the aims and objectives, the challenges faced, the potential risks that could hamper the progress of the project and the appropriate mitigation strategies adopted. It also looks at the review of literature conducted prior to the start of the development phase and discusses critically how this study of existing work enabled the developer to make better design and implementation decisions. A breakdown of the structure of this dissertation is also provided.

# Chapter 2: Planning and Design

This chapter looks at the planning done to ensure successful completion of the project. It also discusses the approaches of implementation and design adopted throughout the project lifecycle. The various milestones of the project are also documented. The chapter concludes with a thorough walkthrough of the architecture of the system and the ideas formulated to work with the underlying database technology.

## Planning

### Methodology

Work on this project is based on agile approaches in software development. This ensures that the end product is of a satisfactory quality and also that any arrangements can be accommodated along the way if necessary. In addition, best practices prevalent in industry are reinforced through application of these organization and planning techniques to the project.

The scrum framework is adopted for the most part of this project. The methodology consists of three roles each of which play dedicated roles within the SDLC. The roles namely are the Product Owner, Scrum Master and the Scrum Team. The Product Owner is responsible for the management of the Product Backlog, which is a prioritized set of requirements. The Scrum Master is in charge of the sprint backlog and is responsible for chairing and regularizing meetings. The scrum team implements requirements set out in each sprint.

In the context of the project, the supervisor acts as both the product owner and the scrum master because they set out the requirements for the project and also manage meetings and provide progress feedback. The scrum team comprises of myself as the developer and I am responsible for implementing requirements as discussed in each biweekly meeting. The time duration from one meeting to the next is considered a sprint for the project. At the end of one sprint, the developer discusses the progress made with the supervisor and notes down the feedback received. Any corrections or improvements are carried over to the next sprint where they are re-evaluated. Additionally, new requirements are set at each sprint (biweekly meeting), which collectively constitute the work plan for the next sprint.

## Approach

The commencement of the project involved researching the technologies required for successful implementation of the project. Identification of the most appropriate technology keeping in view the nature of the project was also done during this period. This involved carefully considering the advantages and limitations of each available technology and critically evaluating which of the available choices would be best suited to the project.

After the initial assessment, a tentative project schedule was developed that listed tasks along with the estimated duration for each. This schedule is meant to act as an indication of the estimated completion dates and is not set in stone.

Development of the project started by implementing login and signup through Firebase Auth UI and setting up the main layout of the application. A trip can fall in one of the following three categories: scheduled, past and trash. The main screen consists of three tabs – one for each of the previously mentioned categories. This screen also provides a handy navigation bar that allows the user to navigate to most activities (screens) within the application.

Next, functionality was added to retrieve a list of places in a city of the user's choice from the Google places web service. The type of place to be retrieved was initially set to points of interest but later customization feature was provided while developing application settings. Once the list of places were successfully retrieved, they needed to be added to a trip. This involved deciding on the appropriate database structure and storing the necessary details for each trip in a consistent format.

A trip "itinerary" is also added to the application which allowed for setting start and end times for each place in a trip.

Settings for the application are divided into two distinct categories – profile settings allow for managing details related to user credentials. Trip settings allow for making a selection of the place type to be retrieved.

## Tentative task breakdown

Using the requirements identified, a project schedule was formulated to act as a guideline for proceeding with the implementation. The following table lists the tasks along with the estimated start and end dates. It is worth mentioning here that, owing to the nature of the project, variations to this schedule might be introduced as deemed necessary by the supervisor.

| **Task** | **Start Date (DD-MM-YYYY)** | **End Date (DD-MM-YYYY)** |
|---|---|---|
| Project Allocation | 01-01-2019 | 11-01-2019 |
| Project Description | 14-01-2019 | 04-02-2019 |
| Preliminary Report | 05-02-2019 | 11-03-2019 |
| Learn Technologies | 05-02-2019 | 08-04-2019 |
| Milestone 1 | 12-03-2019 | 08-04-2019 |
| Design Interfaces and generic functionality | 09-04-2019 | 17-05-2019 |
| Interim Report | 15-04-2019 | 20-05-2019 |
| Retrieve information from online sources | 20-05-2019 | 14-06-2019 |
| Complete major functionality | 20-05-2019 | 14-06-2019 |
| Milestone 2 | 17-06-2019 | 17-06-2019 |
| Prepare dissertation draft | 18-06-2019 | 22-07-2019 |
| Finalize project | 23-07-2019 | 23-08-2019 |

| | | |
|---|---|---|
| Prepare for final viva | 23-08-2019 | 30-08-2019 |
| Project Conclusion | 02-09-2019 | 09-09-2019 |

*Table 1 - Tentative breakdown of project tasks with estimated start and end dates*

## Design

### Software Architecture

The application retrieves place details through communication with the Google Places web service. These places are shown in the application and the user can make selections from the available places to add to a trip. Upon saving, the user has to provide a name and a date, following which the trip is saved to the backend database. The application then retrieves saved trip and details and handles them in the appropriate manner to display the information to the user in conformance with the requirements of the application. The complete architecture is shown figure 1 below. It can be seen from the figure that communication between the application and the database and the application and the Google Places Web Service is two-way and performing using JavaScript Object Notation (JSON).

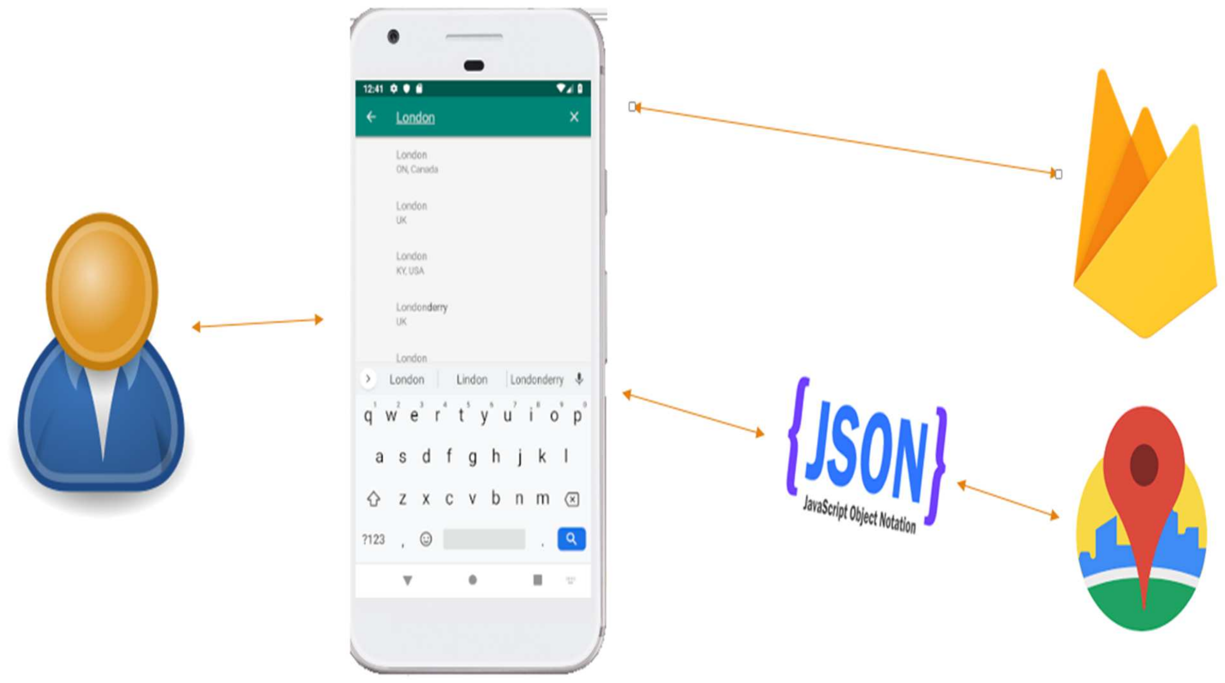
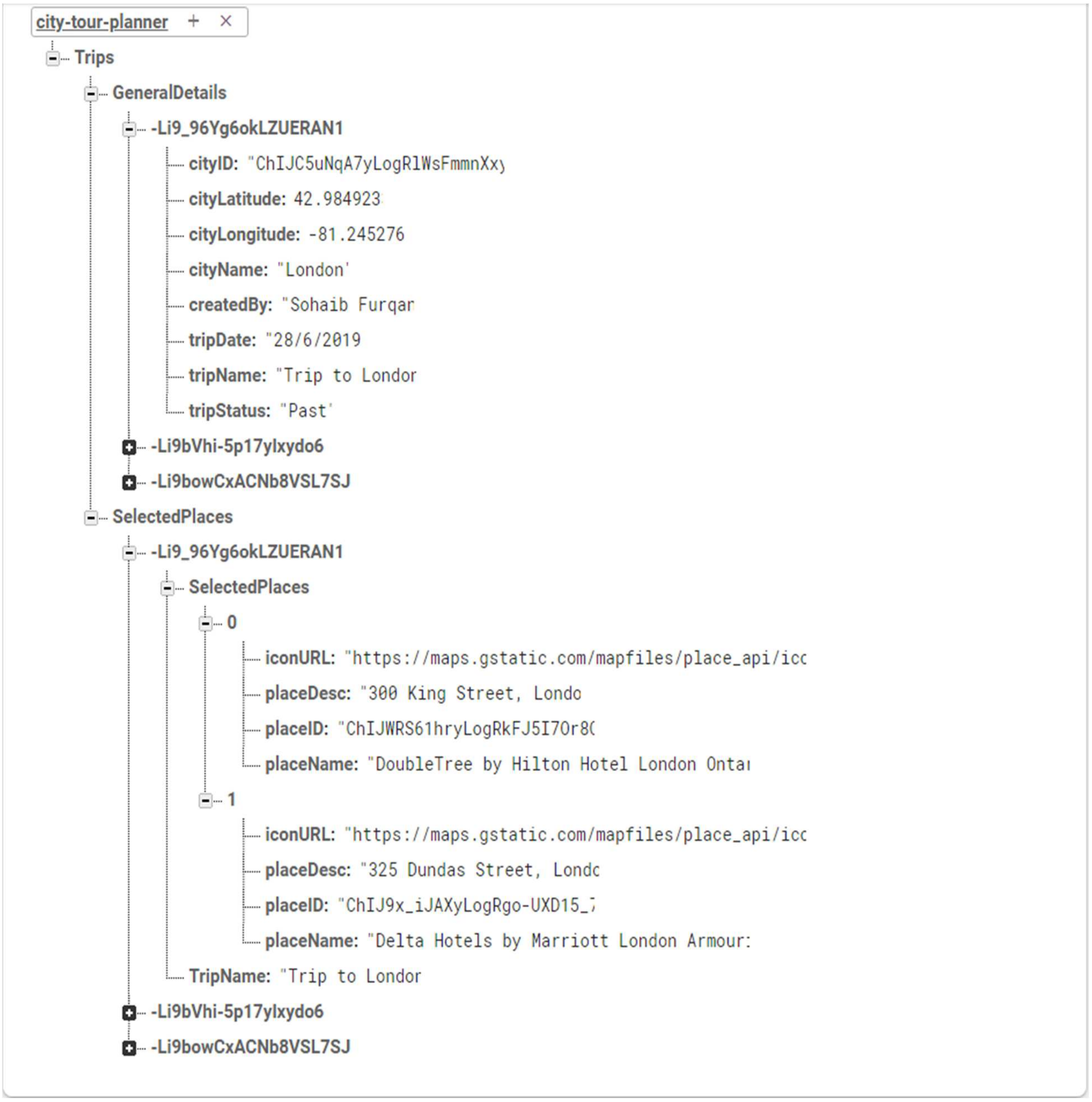


*Figure 1 - Software Architecture*

The application requires internet and location permissions, without which the main features will not work.

## Database Design

The application makes use of the Firebase real-time database for persistent data storage. It is a NoSQL database and hence needs to be structured differently than relational databases. A typical database structure for City Tour Planner is shown in figure 2 below:



```
city-tour-planner  +  ×
  Trips
    GeneralDetails
      -Li9_96Yg6okLZUERAN1
        cityID: "ChIJC5uNqA7yLogR1WsFmmnXx)
        cityLatitude: 42.984923
        cityLongitude: -81.245276
        cityName: "London'
        createdBy: "Sohaib Furqar
        tripDate: "28/6/2019
        tripName: "Trip to Londor
        tripStatus: "Past'
      -Li9bVhi-5p17ylxydo6
      -Li9bowCxACNb8VSL7SJ
    SelectedPlaces
      -Li9_96Yg6okLZUERAN1
        SelectedPlaces
          0
            iconURL: "https://maps.gstatic.com/mapfiles/place_api/icc
            placeDesc: "300 King Street, Londo
            placeID: "ChIJWRS61hryLogRkFJ5I70r8(
            placeName: "DoubleTree by Hilton Hotel London Onta
          1
            iconURL: "https://maps.gstatic.com/mapfiles/place_api/icc
            placeDesc: "325 Dundas Street, Londc
            placeID: "ChIJ9x_iJAXyLogRgo-UXD15_
            placeName: "Delta Hotels by Marriott London Armour:
        TripName: "Trip to Londor
      -Li9bVhi-5p17ylxydo6
      -Li9bowCxACNb8VSL7SJ
```

*Figure 2 - Database Structure*

Firebase stores data in the form of a JSON tree, where the entire database is a single JSON object. As shown in the above figure, the root node, "Trips" has two child nodes, namely "GeneralDetails" and "SelectedPlaces". The "GeneralDetails" node stores general information regarding the trip such as the latitude, longitude and name of the city for which the trip is being created, the name given to the trip and the scheduled date. It also has a field "tripStatus" that indicates whether a given trip is scheduled, past or deleted. The "SelectedPlaces" node is on the same level as the "GeneralDetails" node and stores information regarding the individual places added to a trip by the user, including the place name, a brief description a unique place ID.

It is notable that each child node of "GeneralDetails" as well as "SelectedPlaces" is a random sequence of characters (consider -Li9_96Yg6okLZUERAN1 in the example above). This random sequence of characters is a Push ID (a unique ID generated automatically by the Firebase real-time database whenever new data is inserted). This helps in making sure that no two entries in the database are the same. It is also worth noticing that entries with the same Push ID can be found under both "GeneralDetails" and "SelectedPlaces". It is through the same Push IDs that the application identifies details belonging to a single trip. In the above example, the "GeneralDetails" for -Li9_96Yg6okLZUERAN1 correspond to "SelectedPlaces" entries for the same Push ID.

It would seem logical (at least superficially) if all the details of a particular trip had been stored under a single child node at the same level, but since Firebase guidelines suggest that excessive nesting of data be avoided, different nodes were created for the classification. Apart from being able to avoid nested levels of data, this approach is advantageous in that only the relevant data is downloaded .i.e. if the general details of a trip are to be shown, the data for selected places will not be downloaded. Similarly, if only place details for a particular trip are required, data about the city latitude and longitude itself will not be downloaded. Of course, data can be downloaded from both nodes if needed by the application logic.

## Summary

This chapter elaborates on the planning and design strategies and approaches employed over the course of the project. It provides a biweekly task schedule, discusses the overall software architecture and the rationale behind designing the database.

# Chapter 3: System Specification

Requirements are at the very core of any successful product, and owing to the pervasive nature of software systems and their influence on every aspect of life as we know it, it is critical that the importance of successfully documenting and handling ever-changing requirements is not overlooked. Requirements gathering is one of the first steps in the Software Development Lifecycle, and if not done properly, things tend to go wrong very quickly and can prove very hard to rectify in the later stages of development. This has obvious damaging effects on the long-term quality of a product and the value it can provide to its users. It is therefore of utmost importance that client requirements for a product be gathered with great caution.

From a software engineering standpoint, requirements are divided into four major sub-categories: Business Requirements, User Requirements, Functional Requirements and Non-Functional Requirements. The remainder of this chapter discusses each of these requirements with reference to the project. It also looks at some other design aspects for the City Tour Planner application.

## Business Requirements

Software in the modern scenario inevitably finds itself linked to a business scenario in one way or the other. In the context of software engineering, business requirements document what a system should be able to do from the perspective of the end user. Collecting business requirements is quite often the first step in the software development lifecycle and involves critical assessment of the business model and domain on the part of a business analyst.

The business requirements for City Tour Planner are as defined in the following paragraph:

*"An android application is needed which should allow users to plan trips. For simplicity, the application should focus on generating single city trips only. The application should allow users to organize trips that fit around their daily work routine, thus enhancing flexibility and providing a rich user experience. Users should be able to choose from a list of available place types around which to build each customized trip."*

The above text is an illustration of the way a business analyst might set out business requirements for the application being developed. Due to the vast scope of such requirements, there is no single correct way of stating these, and business requirements for one application may be stated in a number of different ways.

## User Requirements

Where business requirement documents the needs of a business as a whole, user requirements deal with what individual stakeholders of a business want the system to be able to do. These are usually derived from business requirements.

User requirements specific to the project are as stated below:

1. User signup and login
2. Select city to show places
3. Retrieve place details and create trip
4. Display trip information in an appropriate way
5. Edit trip headers
6. Delete trips
7. Restore trips
8. Reschedule trips
9. Permanently delete trips
10. Manage trip itinerary
11. Show pictures of selected place(s)
12. Allow user to manage profile
13. Allow user to change display name
14. Allow user to change email
15. Allow user to change password
16. Allow user to permanently delete account
17. Allow user to customize trip generation criteria
18. Allow user to select type of places to retrieve
19. Show only relevant data to each user
20. Show information about application

## Functional Requirements

This category of requirements is specific to software engineering and is often derived from user requirements. Functional requirements define the desired functionality from the point of view of the system. A given requirement of this category may address all or part of the system at hand. A single user requirement may correspond to multiple functional requirements.

Following are the functional requirements for the application being developed as part of this project.

### FR01: User signup

| | |
|---|---|
| FR01-01 | System shall allow users to create an account |
| FR01-02 | System shall get email from user |
| FR01-03 | System shall get first and last name from user |
| FR01-04 | System shall get password from user |
| FR01-05 | System shall prompt user to confirm password |
| FR01-06 | System shall save account details to database when user submits credentials |

### FR02: User login

| | |
|---|---|
| FR02-01 | System shall allow users to login using preferred authentication provider |
| FR02-02 | System shall get email from user |
| FR02-03 | System shall get password from user |
| FR02-04 | System shall authenticate user when the credentials are submitted |
| FR02-05 | System shall navigate to home screen if correct credentials are entered |

## FR03: Select city to show places

| | |
|---|---|
| FR03-01 | System shall allow user to enter the name of desired city |
| FR03-02 | System shall provide autocomplete suggestions to user |
| FR03-03 | System shall allow user to select from autocomplete suggestions or type in the city name manually |

## FR04: Retrieve place details and create trip

| | |
|---|---|
| FR04-01 | System shall search details of selected city |
| FR04-02 | System shall show points of interest in selected city |
| FR04-03 | System shall allow users to choose out of the list of retrieved places by clicking adjacent checkbox |
| FR04-04 | System shall allow user to save place selections as a trip by clicking the save icon on toolbar |
| FR04-05 | System shall prompt user to enter a trip name and date |
| FR04-06 | System shall allow user to select desired date using the date-picker widget |
| FR04-07 | System shall save trip details to database in an appropriate manner when the trip name and date are submitted by user |
| FR04-08 | System shall assign trip status as “scheduled” if date of trip is after the current system date |
| FR04-09 | System shall assign trip status as “past” if date of trip is before the current system date |
| FR04-10 | System shall notify user that the new trip has been created |
| FR04-11 | System shall update database state to reflect changes |

## FR05: Display trip information in an appropriate way

| | |
|---|---|
| FR05-01 | System shall check trip status of each child node from database on loading of main activity |
| FR05-02 | System shall show trip with “scheduled” status in the “scheduled” tab on the home screen |
| FR05-03 | System shall show trip with “past” status in the “past” tab on the home screen |
| FR05-04 | System shall show trip with “deleted” status in the “trash” tab on the home screen |
| FR05-05 | System shall show each newly created trip in the appropriate tab on the home screen by reading its status from the database in real time |
| FR05-06 | System shall provide an option to perform further operations on a selected trip showing in any tab in the home screen |

## FR06: Edit trip headers

| | |
|---|---|
| FR06-01 | System shall allow to select additional operations for a trip by clicking the “more options” button next to the trip |
| FR06-02 | System shall allow user to edit trip name and date for each trip showing in the “scheduled” tab by clicking the “Edit trip Details” option |
| FR06-03 | System shall navigate to the “update trip headers” activity |
| FR06-04 | System shall popular edittexts with the current name and date of selected trip |
| FR06-05 | System shall allow user to input new name and date |
| FR06-05 | System shall update the name and date of the selected trip when user clicks the update button in the database |
| FR06-06 | System shall decide the new category of the trip according to updated date and show the trip in the appropriate tab with the new details |

## FR07: Delete trips

| | |
|---|---|
| FR07-01 | System shall allow to select additional operations for a trip by clicking the “more options” button next to the trip |
| FR07-02 | System shall allow user to delete each trip showing in the “scheduled” or “past” tab by clicking the “Delete Trip” option |
| FR07-03 | System shall prompt user for delete action confirmation |
| FR07-04 | System shall assign a trip status of “Deleted” when the user clicks the “OK” button in the confirmation dialog |
| FR07-05 | System shall update the application to remove deleted trip from the associated tab and display it in the “Trash” tab |

## FR08: Restore trips

| | |
|---|---|
| FR08-01 | System shall allow to select additional operations for a trip by clicking the “more options” button next to the trip |
| FR08-02 | System shall allow user to restore each trip showing in the “trash” tab by clicking the “Restore trip” option |
| FR08-03 | System shall check the date of the trip to decide the new category for the trip. If the date of the trip is before the current date, the system shall assign a trip status of “past”. If the date of the trip is after the current system date, the system shall assign a trip status of “scheduled” |
| FR08-04 | System shall display the restored trip in the appropriate tab according to the trip status assigned |
| FR08-05 | System shall update the application to remove deleted trip from the associated tab |

### FR09: Reschedule trips

| | |
|---|---|
| FR09-01 | System shall allow to select additional operations for a trip by clicking the “more options” button next to the trip |
| FR09-02 | System shall allow user to reschedule each trip showing in the “past” and “trash” tab by clicking the “Reschedule trip” option |
| FR09-03 | System shall check the date of the trip to decide the category for the trip. If the date of the trip is before the current date, the system shall assign a trip status of “past”. If the date of the trip is after the current system date, the system shall assign a trip status of “scheduled” |
| FR09-04 | System shall display the restored trip in the appropriate tab according to the trip status assigned |
| FR09-05 | System shall update the application to remove restored trip from the “past” or “trash” tab and show it in the “scheduled” tab |

### FR10: Permanently delete trips

| | |
|---|---|
| FR10-01 | System shall allow to select additional operations for a trip by clicking the “more options” button next to the trip |
| FR10-02 | System shall allow user to permanently remove each trip showing in the “trash” tab by clicking the “Permanently Delete Trip” option |
| FR10-03 | System shall prompt user to confirm delete operation |
| FR10-04 | System shall remove all details of selected trip from the database when the user confirms the operation |
| FR10-05 | System shall update the application to remove deleted trips from the tabs of the home screen |

## FR11: Manage trip itinerary

| | |
|---|---|
| FR11-01 | System shall open trip itinerary when user clicks on a trip in any tab on the home screen |
| FR11-02 | System shall show details of places added to selected trip |
| FR11-03 | System shall allow the user to specify start and end times for each place in the itinerary within a single day |
| FR11-04 | System shall display user-specified start and end times next to each place in the trip |
| FR11-05 | System shall provide the user with options to manage place entries for each trip |

## FR12: Show pictures of selected place(s)

| | |
|---|---|
| FR12-01 | System shall allow user to view pictures of places while selecting places to add to a trip |
| FR12-02 | System shall present the user with a list of popular places |
| FR12-03 | System shall allow user to view pictures of selected place when the user clicks on that place |
| FR12-04 | System shall allow user to swipe to view more images |
| FR12-05 | System shall present the user with further details regarding the selected place on the same screen |

## FR13: Allow user to manage profile

| | |
|---|---|
| FR13-01 | System shall allow user to manage app settings by clicking on the “settings” icon in the toolbar of the main screen or by choosing the “settings” option in the navigation drawer |
| FR13-02 | System shall allow user to change application display name |

| | |
|---|---|
| FR13-03 | System shall allow user to change email |
| FR13-04 | System shall allow user to change password |
| FR13-05 | System shall allow user to permanently delete account |

**FR14: Allow user to change display name**

| | |
|---|---|
| FR14-01 | System shall allow user to manage app settings by clicking on the “settings” icon in the toolbar of the main screen or by choosing the “settings” option in the navigation drawer and choose to manage profile settings |
| FR14-02 | System shall allow user to select the desired setting |
| FR14-03 | System shall open a dialog box populated with the current user’s name when the user selects to edit display name |
| FR14-04 | System shall allow user to enter a new name |
| FR14-05 | System shall update the display name when the user clicks the OK button |

**FR15: Allow user to change email**

| | |
|---|---|
| FR15-01 | System shall allow user to manage app settings by clicking on the “settings” icon in the toolbar of the main screen or by choosing the “settings” option in the navigation drawer and choose to manage profile settings |
| FR15-02 | System shall allow user to select the desired setting |
| FR15-03 | System shall open a dialog box populated with the current user’s email when the user selects to edit email address |
| FR15-04 | System shall allow user to enter a new email |
| FR15-05 | System shall update the email for the user when the user clicks the OK button |
| FR15-06 | System shall send an email change confirmation email to the old email address of the user |

## FR16: Allow user to change password

| | |
|---|---|
| FR17-01 | System shall allow user to manage app settings by clicking on the “settings” icon in the toolbar of the main screen or by choosing the “settings” option in the navigation drawer and choose to manage profile settings |
| FR17-02 | System shall allow user to select the desired setting |
| FR17-03 | System shall navigate to the password change activity when the user selects to change password |
| FR17-04 | System shall allow user to enter current password |
| FR17-05 | System shall allow user to enter new password |
| FR17-06 | System shall allow user to reenter new password after confirming |
| FR17-07 | System shall check if the password entered in the “current password” field is a valid one |
| FR17-08 | System shall check if the passwords entered in the “new password” and the “confirm new password” fields match |
| FR17-09 | System shall update password if password is valid and the new passwords match when the user clicks the update icon on the toolbar |

## FR17: Allow user to permanently delete account

| | |
|---|---|
| FR17-01 | System shall allow user to manage app settings by clicking on the “settings” icon in the toolbar of the main screen or by choosing the “settings” option in the navigation drawer and choose to manage profile settings |
| FR17-02 | System shall allow user to select the desired setting |
| FR17-03 | System shall prompt user to enter current password when the user chooses to permanently delete account |

| | |
|---|---|
| FR17-04 | System shall check if the supplied password is correct when the user clicks the OK button on the dialog box |
| FR17-05 | System shall prompt the user for confirmation to delete account |
| FR17-06 | System shall delete user details permanently from the database when the user confirms the action |

### FR18: Allow user to customize trip generation criteria

| | |
|---|---|
| FR18-01 | System shall allow user to manage app settings by clicking on the "settings" icon in the toolbar of the main screen or by choosing the "settings" option in the navigation drawer |
| FR18-02 | System shall allow user to specify whether or not they want to retrieve custom places by using the designated toggle switch |
| FR18-03 | System shall allow user to specify a place category if the user selects to retrieve custom places by using the listpreference |
| FR18-04 | System shall retrieve only places of the category specified in the listpreference if custom place retrieval is enabled |

### FR19: Allow user to select type of places to retrieve

| | |
|---|---|
| FR19-01 | System shall allow user to manage app settings by clicking on the "settings" icon in the toolbar of the main screen or by choosing the "settings" option in the navigation drawer and choose to manage profile settings |
| FR19-02 | System shall allow user to change trip preferences |
| FR19-03 | System shall allow user to enable/disable custom place retrieval |
| FR19-04 | System shall allow user to select the category of places to retrieve using a listpreference widget if custom place retrieval is enabled |

### FR20: Show only relevant data to each user

| | |
|---|---|
| FR20-01 | System shall display only those trips which are created by the currently logged in user |
| FR20-02 | System shall add only those newly created trips which are added by the currently logged in user |

### FR21: Show information about application

| | |
|---|---|
| FR21-01 | System shall allow user to view information about the application when the user selects the “About” item in the application navigation bar |
| FR21-02 | System shall display a dialog containing general information about the application, the developer and the project supervisor and second marker |

## Non-Functional Requirements

NFRs deal with the specifics of how the system being developed should operate as opposed to functional requirements, which document what the system should do. Additionally, while functional requirements are directly linked to user needs associated with the system, non-functional requirements are often not specified by the user and are automatically understood by the development team.

Following are the non-functional requirements for the application being developed as part of the project:

### NFR01: Performance

| | |
|---|---|
| NFR01-01 | The application must have a startup time of less than 5 seconds after the user taps the launcher icon |
| NFR01-02 | User actions must be responded to by the application in the least possible time, preferably less than 10 seconds |

| | |
|---|---|
| NFR01-03 | The application must be able to run smoothly on Android devices with API level 27 and above |

## NFR02: Security

| | |
|---|---|
| NFR02-01 | The application requires users to register and login with their credentials before they can use the application. Only properly validated users must be used to access the application |
| NFR02-02 | The application must ensure that any user does not have any level of access to other users' data. This includes not being able to view, create or modify trips created by other users |
| NFR02-03 | Once logged in, the application retains credentials of the logged in user until they log out explicitly, in which case the system shall perform the necessary steps to ensure complete removal of the previous user session |
| NFR02-04 | Once digitally signed and made available on the play store, the application must only be available to the user through designated app download platforms. The developer must make a conscious effort to ensure that unofficial or pirated APKs are not available over the internet on illegal sites |

## NFR03: Availability

| | |
|---|---|
| NFR03-01 | The application shall be available to the user from any part of the world once released on the play store. When not available, the developer may be contacted |
| NFR03-02 | The application shall be available for use all the time unless the underlying API services are down or the developer has made the application unavailable for maintenance or upgrade |

### NFR04: Disaster Recovery

| | |
|---|---|
| NFR04-01 | Appropriate disaster recovery mechanisms must be available in case an unforeseen exception arises when using the application. |
| NFR04-02 | The firebase real-time database must synchronize data across the effected client devices to ensure consistency in the event of a sudden crash |

### NFR05: Documentation

| | |
|---|---|
| NFR05-01 | The application must be accompanied by detailed documentation that describes how the entire software development lifecycle was executed during its production |
| NFR05-02 | Prompt help must be available to the user. This application uses startup tutorial slides for this purpose |

### NFR06: Usability

| | |
|---|---|
| NFR06-01 | The application should be built with a central focus on providing the best user experience possible |
| NFR06-02 | The application should be easy to understand and use for a new user |

## Assumptions and Constraints

### Development Languages and Tools

It is assumed that the end user of the application has no direct interest in the programming language that was used to build it and the approach that was taken to do so. As in most applications, users are only concerned mainly with the value it provides to them.

### Operating System

The application will be built for the Android platform and will run conveniently on devices with API level 27 or higher. Where possible, backward compatibility will be ensured.

### Browser Support

This is a native android application, and does not require support from web browsers external to the application to run.

## Summary

This chapter pens down the various constructs used in software engineering in order to specify the behavior of a system. The requirements for the application being developed as part of the project are highlighted. These include business requirements, user requirements, functional requirements and non-functional requirements. The assumptions made during the development of the application and the constraints applicable (if any) are also documented.

# Chapter 4: Technical Details

Thinking about and identifying the best-suited technologies for a project can act as a stimulus in starting development effort with the right direction. In the following text, I discuss the various technical decisions taken during the course of the project and the rationale behind them.

## Architecture Overview

The application is intended for use on the Android OS and will run on almost all android-powered devices. The target audience is any user interested in automated trip-planning. The aim of the application is for users to be able to quickly and efficiently find and customize trip experiences to their liking. Users will be able to set custom search criteria, and retrieve related information on the basis of these settings from multiple sources. Users will be asked to set preferences and these preferences will be taken into account during the search. The information will then be presented in a user-friendly way.

The user is the primary point of interaction for the application. The user may perform operations on the application (such as specifying a city name), to which a response will be returned (such as a list of places of interest within that city). The user could then use these results to create trips. Moreover, the application will also provide appropriate options for trip management (CRUD operations) and keeping history of past trips.

The application will use third-party APIs for retrieving place information. At the time of writing, the Google Places API web service and the Google Places SDK for Android is being used in conjunction to provide the necessary user experience, however, this is likely to be enhanced in future.

The project uses the Firebase database for persistent data storage, the merits of which have been laid out in significant detail in the following discussions. It is worth noting that, due to the nature of the project, changes and revisions are likely to be made throughout the course of project completion.

## Technologies

Although, the downsides of a particular technology depend largely on the nature of the project being developed, making the correct decision is crucial to timely completion of the project. An informed choice of the technologies to use can also greatly reduce the development effort involved.

## Platform and programming languages

This application targets the android platform and therefore, a significant part of it will be developed in Java using the Java Platform Standard Edition (Java SE). The database technology for the purposes of this application was initially decided to be SQLite and Object-Relational Mapping (ORM) systems such as ROOM might but as implementation progressed it become clear that the firebase real-time database will be a much more viable option. Hence the switch to that was made.

### Database selection

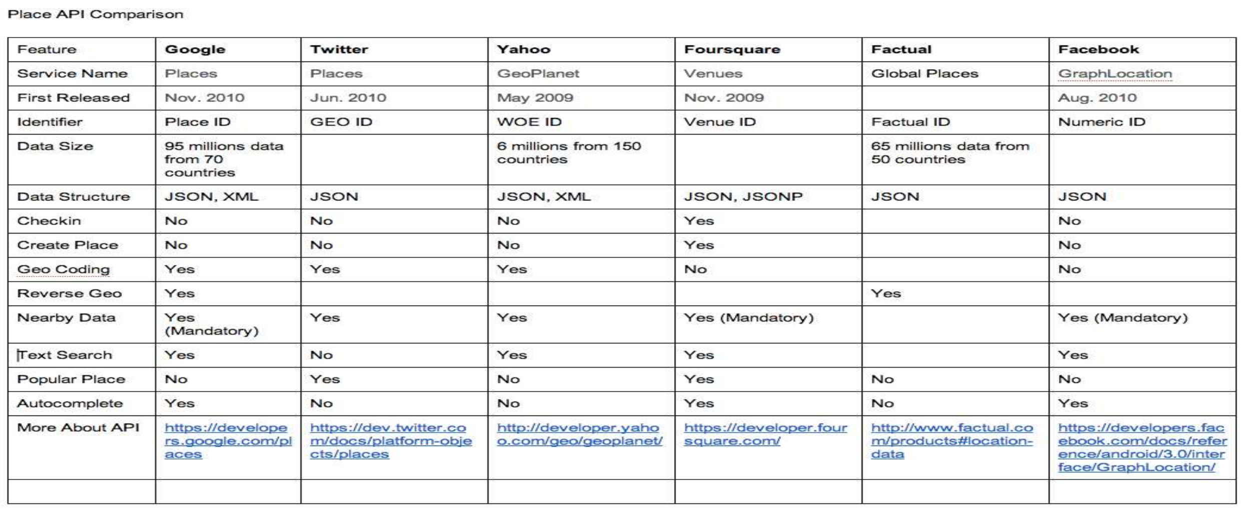
There are a few obvious advantages that Firebase offered over SQLite in the context of this project. While SQLite happens to be the traditional, and still the most popular way of persistent data storage within android applications, it comes with a fair share of downsides. One striking limitation of SQLite which prompted use of firebase for this project is that synching SQLite data across devices can prove to be difficult. A SQLite database operates on the physical device running the application. As a consequence, data that is stored is internal to each device and cannot be easily transferred from one device to the other. This can give rise to user dissatisfaction as changes made on one device, will most likely not reflect when the same user logs in to the same application from a different device. Firebase uses cloud storage, and hence makes the process automatic with instantaneous synchronization across devices.

### API selection

The application makes use of APIs to fetch the necessary data regarding places. This includes (but is not limited to) Google Maps, Trip Advisor and Foursquare. These technologies will be used as deemed appropriate at the time of development so as to enhance the user experience of the application. At the time of writing, the Google Places API is being used to fetch points of interest in a city selected by the user.

There were some choices available with regards to available APIs. One of the apps studied during the literature review, Sygic Travels, used the Sygic Travel API in their app, but for the purposes of this project, Google's offering was preferred because it has a richer places database and is generally more popular.

The following is a comparison of the various APIs as noted during the literature review.



Place API Comparison

| Feature | Google | Twitter | Yahoo | Foursquare | Factual | Facebook |
|---|---|---|---|---|---|---|
| Service Name | Places | Places | GeoPlanet | Venues | Global Places | GraphLocation |
| First Released | Nov. 2010 | Jun. 2010 | May 2009 | Nov. 2009 | | Aug. 2010 |
| Identifier | Place ID | GEO ID | WOE ID | Venue ID | Factual ID | Numeric ID |
| Data Size | 95 millions data from 70 countries | | 6 millions from 150 countries | | 65 millions data from 50 countries | |
| Data Structure | JSON, XML | JSON | JSON, XML | JSON, JSONP | JSON | JSON |
| Checkin | No | No | No | Yes | | No |
| Create Place | No | No | No | Yes | | No |
| Geo Coding | Yes | Yes | Yes | No | | No |
| Reverse Geo | Yes | | | | Yes | |
| Nearby Data | Yes (Mandatory) | Yes | Yes | Yes (Mandatory) | | Yes (Mandatory) |
| Text Search | Yes | No | Yes | Yes | | Yes |
| Popular Place | No | Yes | No | Yes | No | No |
| Autocomplete | Yes | No | No | Yes | No | Yes |
| More About API | https://developers.google.com/places | https://dev.twitter.com/docs/platform-objects/places | http://developer.yahoo.com/geo/geoplanet/ | https://developer.foursquare.com/ | http://www.factual.com/products#location-data | https://developers.facebook.com/docs/reference/android/3.0/interface/GraphLocation/ |
| | | | | | | |

*Figure 3 - Places API comparison*

## Libraries and packages

The application makes use of various libraries to perform otherwise tedious tasks. These are detailed below:

### Android dependencies

1. Android support library: implementation 'com.android.support:support-v4:28.0.0'
2. Android support compatibility library: implementation 'com.android.support:appcompat-v7:28.0.0'
3. Constraint Layout: implementation 'com.android.support.constraint:constraint-layout:1.1.3'
4. Recyclerview: implementation 'com.android.support:recyclerview-v7:28.0.0'

### Firebase dependencies

1. Firebase core: implementation 'com.google.firebase:firebase-core:16.0.8'
2. Firebase database: implementation 'com.google.firebase:firebase-database:16.1.0'

3. Firebase auth-UI: implementation 'com.firebaseui:firebase-ui-auth:4.3.1'

**Third-party libraries and dependencies**

1. Ted permissions: This library offers easy permissions handling for Android Marshmallow and above versions. It is available at https://github.com/ParkSangGwon/TedPermission
2. Volley: An HTTP library that simplifies networking tasks for android applications and makes it faster. For this project, the library is being used to parse JSON data from APIs so that it can be displayed locally within the application. It is available at https://github.com/google/volley
3. Picasso: It is a common task for android applications to load images from online sources. Picasso is an image loader and caching library for android that greatly simplifies this task. It is available at https://github.com/square/picasso

**Other specifications:**

The application will run on any device running Android 4.0.3 (IceCreamSandwich) or higher.

## Summary

The "Technical Details" chapter of the dissertation details the application architecture, the technologies involved, the database(s) selected and the third-party APIs and libraries utilized. Where applicable, pros and cons of technologies have also been discussed in the context of the project and care has been taken to bring to the attention of the reader the thought-process behind making certain decision at the expense of others.

# Chapter 5: Project Modules

The application code follows an Object Oriented paradigm and is based on the use programming constructs such as classes and interfaces among others. Additionally, many constructs specific to Android programming are also used. In this chapter, we look at the classes, interfaces, layout files etc. designed in the app. The methods contained therein are explained to aid clarity.

## Classes

### AboutDialog.java

Produces a dialog box that displays information about the application when the user clicks on the "About" menu item in the navigation bar on the home screen.

### EditTripHeaders.java

This class is responsible for the feature that allows users to change trip name and date. This is accomplished by using a TextWatcher object that collects text in variables everytime it is changed by the user. The UpdateTripHeaderDetails method uses the text collected by the TextWatcher object, stores them in a HashMap object and updates the database to reflect the modified values.

### FeedbackDialog.java

Opens an InputDialog allowing users to send their feedback about the application by entering name, email and feedback message. This feedback is saved to the database.

### GeneratedTrip.java

A Plain Old Java Object (POJO) class used to store and retrieve trip objects to and from the firebase real-time database.

### LoginActivity.java

Implements firebase signup and login functionality using the pre-built Auth UI.

### MainActivity.java

Checks application permissions, implements methods related to the navigation drawer.

**PastFragment.java**

Handles fragment display for showing past trips in the TabLayout on the home screen.

**PastTripRecyclerViewAdapter.java**

Contains code for appropriately processing trips that are shown in the "past" tab, including options to restore and delete trips, and checks on when to add a trip to the "past" tab.

**PlaceDetailActivity.java**

Handles the task of viewing more details of a selected place.

**SavePlaceDialog.java**

Opens an InputDialog that allows user to save place selections as a trip to the database after prompting the user to enter a name and date for the newly generated trip.

**ScheduledFragment.java**

Handles fragment display for showing scheduled trips in the TabLayout on the home screen.

**ScheduledTripRecyclerViewAdapter.java**

Contains code for appropriately processing trips that are shown in the "scheduled" tab, including options to edit trip header details and delete them, and checks on which trip qualifies as a scheduled one.

**SearchedPlacesAdapter.java**

**SearchedPlacesItem.java**

A Plain Old Java Object (POJO) class used to define how a place searched by the user will look like in the application.

**SettingsActivity.java**

Checks whether the settings fragment being opened is a valid fragment or not. The Android platform requires application settings to be implemented in this way.

**SettingsFragment.java**

Contains code to handle the different settings offered by the application.

### TrashFragment.java

Handles fragment display for showing deleted trips in the TabLayout on the home screen.

### TrashTripRecyclerViewAdapter.java

Contains code for appropriately processing trips that are shown in the "trash" tab, including options to restore and permanently delete trips, and checks on when to add a trip to the "trash" tab.

### UpdatePasswordActivity.java

Re-authenticates and updates the password for the current user.

## Interfaces

### FeedbackDialogListener

An interface declared in "FeedbackDialog" which passes values between the dialog and the calling class. The "MainActivity" class implements this interface.

### SaveTripDialogListener

An interface declared in "SavePlaceDialog" which passes values between the dialog and the calling class. The "PlaceSearchActivity" class implements this interface.

### OnItemClickListener

An interface declared in "SearchedPlacesAdapter" which defines methods for handling item clicks. The "PlaceSearchActivity" class implements this interface.

## Layout files

Each Java class mentioned above is has an associated layout file (.xml) that defines the user interface for that activity.

## Summary

This chapter lists the various classes, interfaces and layout files designed for the application.

# References

1. Android Developer Guides, available at http://developer.android.com/guides (Accessed: 10
2. February, 2019)
3. Material Design Guidelines, available at https://material.io/design/guidelines-overview/
4. (Accessed: 23 February, 2019)
5. Google Codelabs – Android Fundamentals, available at
6. https://codelabs.developers.google.com/android-training (Accessed: 10 February, 2019)
7. The Java Tutorials by Oracle, available at https://docs.oracle.com/javase/tutorial/ (Accessed 10
8. February, 2019)
9. Places API Comparison, available at
10. https://docs.google.com/document/d/1f_3r8Ebx94ojzeXpOb9-P6ehnSHV5Pt3WlgA0ryzYE/edit (Accessed 9 May, 2019)