Software Requirements Specification

March 2, 2018

Team 3 (Jack Taylor, Mitchel Smith, Elan Kainen, Jason Wen)

Introduction:

This report describes Team 3's management plan for its project to develop an iOS application to connect travel buddies based on interests and trip scheduling.

1. Introduction

1.1. Purpose

The purpose of this project is to develop an iOS app that allows users to plan and schedule trips with other travelers with similar desired destinations. Functionally, this app will let users make trip propositions and match them with others based on both travel and personal interests. The app hopes to take advantage of the rise in popularity of the sharing economy and success of services like Uber, Tinder, Airbnb, and other such apps.

1.2. Scope

This document will cover the requirements of the entire project. The project is planned to follow the prototyping model, and therefore it implies that the scope of each iteration will be based of a previous prototype.

1.3. Definitions, Acronyms, and Abbreviations

Swift: a programming language developed by Apple Inc. and utilized for the development of iOS, macOS, watchOS, tvOS, and Linux- based products.

Xcode: an integrated development environment for macOS containing a suite of software development tools by Apple used to create software for macOS, iOS, watchOS, and tvOS.

iOS: a mobile operating system developed by Apple, that powers many of the company's mobile devices.

Google Maps API for iOS: Google Maps API specifically for iOS developers. Available to developers via CocoaPods and allows Maps and its features to be easily integrated into projects of all sizes.

1.4. References

- Lynda.com, CodeAcademy, Youtube Tutorials
- Start Developing iOS Apps
- Google Maps SDK for iOS
- Stack Overflow Q&A's
- The Swift Programming Language (Swift 4.0.3)
- A Concise Introduction to Software Engineering - Jalote

1.5. Overview

Our app is called Travel Buddy. As the name suggested, the purpose of our app is to help people find travel buddies who have similar desired destinations and travel dates. For instance, a user may be interested in visiting Shanghai, China for spring break 2018. They will login to our app and enter some basic information:

Destination: Shanghai, China

Travel Date: March.2 - March 11

Trip Types: Sightseeing (other options could be fishing, kayaking, hiking, etc.)

Our app will store the user inputs in a built-in server, and then produce matches using our sorting algorithms. The application will allow the user to create a trip object with traits based on date, location, and type of trip. A user will be able to input the trip's attributes and subsequently be connected to another user who has created a trip proposal with matching attributes so that they can fulfill their mutual goals. Users that have been matched will have experienced such because they created a trip object with a matching destination. However, it is also possible that users have been matched due to a common geographic vicinity or set time of travel.

The application will utilize the Google Maps API to geolocate a user and then find other users in a set area with common trip objects. When a user is submitting a trip proposal they will be able to utilize features of Google Maps, namely Google Places, to highlights points of interest and drop pins to be linked to their proposal. Based on both the user's personal attributes (account characteristics) and trip requirements (trip characteristics), the application will search for a match among other currently "open" proposals. If a match is found, users will be notified and free to connect with each other for their next Travel Buddy-planned excursion.

2. Overall Description

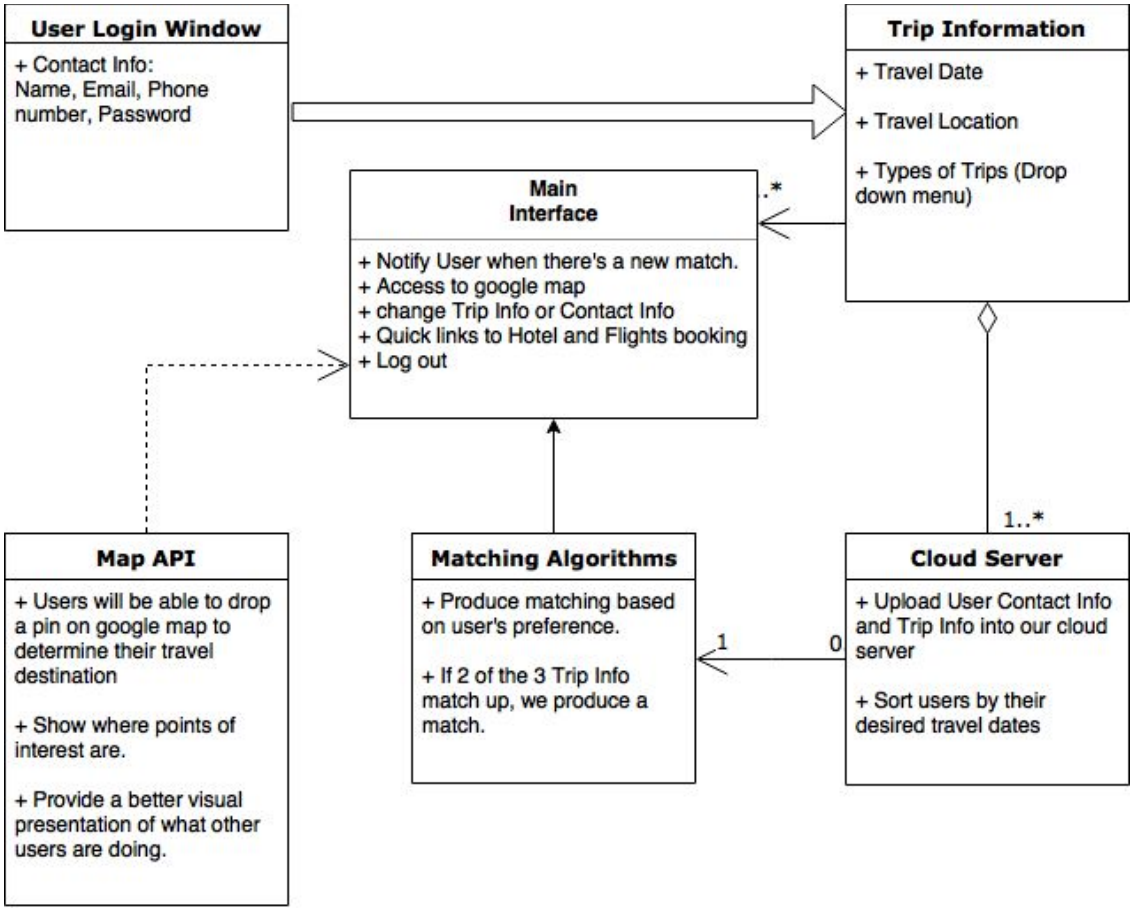
2.1. Product Perspective

Because the product will be an iOS app written in Swift, it will, by default, be run on Apple hardware and with Apple's own operating system for mobile applications, iOS.

These interconnections are made possible by writing the app in Swift on XCode. Xcode is an IDE that allows developers to create software on several of Apple's operating systems, including iOS. XCode development tools include a "live" user interface that, when code is run, displays the result on a model of a select Apple device (every generation of iPhone, iPad, and etc.). This feature gives the programmer the ability to see a simulation of what their application looks like at each phase of its development without deployment to a physical device. Hence, the IDE and language we will be utilizing are specifically tailored to development for Apple products.

In regards to the infrastructure deployed by this application, most of what we will be building will have had no prior design as a reference point for its development. We plan on taking note of the design and architecture of existing apps that capitalize on the "sharing economy" or bring users together for a mutual purpose, such as Tinder, and Uber. Since we are building a new app, anything we inherit from the infrastructure of previous apps will merely be design inspiration. In this sense, the hierarchy of our application's pages and other user interface related components will be the major force driving the choices we make.

2.2. Product Functions



2.3. User Characteristics

Targeted users for this app are not limited to any specific age range, gender, or demographic. This app is intended to bring together those who enjoy travel and encourage meeting new people. Ideally, a user of this application should understand how to use an iPhone, including its main features, as well as how to use mobile applications, including their functional nuances. As long as the user has experience navigating the likes of Facebook, Uber, or Airbnb, they will have no issue utilizing and following our user interface and accessing all of its capabilities.

In regards to traits, users should be honest, open individuals who are not using the application to cause harm unto others. Discretion is always advised, as individuals who have matched and decided to meet up should be aware of the fact that they will be meeting a stranger. Regardless, such caution has already become an unknown standard for other applications as the intersections of social media and personal utility become ever more pertinent.

2.4. General Constraints

Our application will be dependent and constrained by the battery life and computational power of the iPhone (varying models). In this sense, we must be careful to write well-planned code, as the tools we will use allow code to be optimized to run comfortably on iOS devices. It will be up to us to use best practices when developing the app and avoid certain use cases resulting in long loading screens and battery drain (such as trip scheduling). However, we do not expect our app to be all that computationally intensive.

The major constraint we expect to experience is the number of users and open trip requests possible at a given time without sacrificing the functionality of the app. We do not expect that the built-in server method we plan to use will be robust enough to handle traffic surpassing that of a prototype filled with toy users and functionality examples. This of course, is all we require of the application during this project, but it is worth noting in this report that architectural changes would most likely be required in order to scale.

2.5. Assumptions and Dependencies

We assume that our built-in server will function as we necessary and be able to properly keep information stored. Additionally, the assumption that the application will have robust trip matching capabilities is important to note. Our application “intends” on geolocating a user and then scanning a set radius for other users with similar trip credentials. Building out this software with the Google Maps API is vital, but our ability to create a robust matching algorithm is a real assumption, as we are just four students whose days are numbered and must juggle other classwork.

Our team has also assumed that we will be able to add additional functionalities to the application in order to enhance the user experience, such as a chat tool. This of course will only be completed if the rest of the application can be built-out with extra time remaining.

3. Detailed Requirements

3.1. External Interface Requirements

3.1.1. User Interfaces

3.1.1.1. Any given screen should have no more than 4 commands on it due to screen size limitations

3.1.2. Hardware Interfaces

3.1.2.1. iOS specifically iPhone

3.1.3. Software Interfaces

3.1.3.1. Must interface fluidly with the Google Maps API, i.e. never take more than 4 seconds assuming a good connection.

3.1.4. Communication Interfaces

3.1.4.1. chat, if done, should be nearly instant.

3.2. Functional Requirements

3.2.1. Account Management

3.2.1.1. The User shall be able to create a username and password (editable)

3.2.1.2. The User shall be able to create and edit their profile, including biography, location preferences, and travel buddy preferences

3.2.1.3. The User shall be able to set up account recovery information

3.2.2. Login Screen

3.2.2.1. The User shall be able to login to a pre-existing account

3.2.2.2. If the User types in information which does not correspond to an existing account, the system shall output a failed login message and return to the login screen

3.2.2.3. The system may place a cap on attempted logins

3.2.2.4. The User shall be able to recover lost credentials

3.2.3. Trip Creation Screen

3.2.3.1. The User shall be able to create a trip

3.2.3.2. If the User attempts to create a trip with one of the following: invalid date range, lack of location, lack of type; the system shall highlight the missing information and request that it be input before continuing

3.2.4. Trip Selection Screen

3.2.4.1. The User shall be able to sign up for a public trip

3.2.4.2. If the User attempts to sign up for multiple competing trips, the system shall output a warning message and force a choice.

3.2.4.3. The User may be able to save a trip for later

3.2.5. System Settings Screen

3.2.5.1. The User shall be able to sign out

3.2.5.2. The User shall be able to mute sounds

3.3. Performance Requirements

At this point in the project it is challenging to select quantitative measures for response times for certain tasks, as we are not yet fully aware of the backend actions that will be necessary for certain requirements. However, qualitatively we will require that our application runs smoothly, as this is absolutely crucial for user-centric software such as Travel Buddies. Although our project itself is not going to be deployed to the masses, we will approach the project as if it will be to produce a smooth and enjoyable product that exhibits clear effort put into UI. Switching between pages, signing in, and creating both user profiles and trip proposals should be a fairly quick process and we aim to avoid events which lag for time periods much larger than a few seconds.

From personal use of apps, we can put a reasonable maximum on some of these statuses. Changing between screens should not take more than 3 seconds; above three seconds and it feels like we are waiting for a long time. Loading the app shouldn't take more than 6 seconds, again because any more than that and the experience is no longer enjoyable. Creating a trip can take up to 8 seconds but there must be some form of moving object which indicates that the app hasn't frozen.

3.4. Design Constraint

At this point, we believe we have outlined the main sets of constraints in the preceding requirements. It will be fairly straightforward to prevent the ballooning of memory usage in our application as we build it out and continue along our prototyping process. The capabilities of Xcode will allow us to maintain confidence that our project will conform to the standards necessary to run on iOS devices through constant testing on virtual machines as well as Apple's own documentation for iOS development.

3.5. Attributes

No such attributes exist.

3.6. Other Requirements

No such requirements exist.