Software Design Document

Friday April 6, 2017

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

Introduction:

This report describes Team 3's Software Design for our project

1. Introduction (generally similar to the introduction in your architecture document)
   1.1. Purpose

The software will be using the individual pages of the app as building blocks for the entire app. This is useful for two reasons: 1) It allows us to build up a base and utilize the prototyping model by building a little bit on each page and then testing that everything works together, and 2) It allows us to work on multiple pages concurrently without fear of breaking something else at the same level (i.e. changing something on the profile page won't affect the trip selection page since they do not interact).

The primary communications will be the pages of the app with the server. There are three primary user-visible pages (Profile, Trip Creation, Trip Selection) with the possibility of others being added later. The Trip Creation and Trip Selection pages need to be able to access information on the profile page as well as communicate with the server. This will allow auto-filling of information during trip creation and basic filtering during Trip Selection. The Trip pages need to be able to access the server to check what trips exist.

The server never needs access to the app itself. It should only ever be queried by the app never query the app.

1.2. Scope

What is covered by this document. E.g., does it cover the entire project, or just the changes from some other product?

This document will cover how the two main pieces of our app, the server and the UI, fit together. It will also briefly discuss the way that the different pages of our app interact with each-other. But, due to the fact that the different pages exist for different purposes, this discussion will not be a major point of this document.

Within each piece it will describe, at a high level, the primary objects and uses of those objects. For the main app, this will include the different pages and purposes of those pages. For the server this will include the Trip and User objects.

1.3. Definitions, Acronyms, and Abbreviations

Swift - Apple's programming language which we will be using.

XCode - An apple produced IDE which is optimized for iOS app production

Page - Our app will consist of multiple screens that accomplish different tasks, these screens will be referred to as the pages of our app.

Travel Buddies - The name of our app. Abbreviated as TB.

Trip Page - The Trip Selection and Trip Creation pages.

Trip - A trip will be a custom object which will include information such as date range, type of trip (camping, skiing, vacation, etc.), expected cost, users committed, and number of users desired.

User - A User will be a custom object representing an account that each user creates and can update through their profile.

AWS Mobile Services - Amazon Web Services Mobile. Amazon provides a SDK specifically for iOS development to configure and integrate cloud-based backend servers for mobile applications.

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

Give a very high level description of the project. A couple of paragraphs will do. Serves to give a context for the next section Overall Description of the specification

This project can be broken down into two main components: the server and the user interface. The server will store information that is input into the UI. Information can be accessed from the server by the UI in order to accomplish a user's particular goal. The

application will be split into several pages that are linked to one another via buttons and tabs on the UI. On the front end, a user will be able to traverse across the entire application interface and interact with any page. The entire front will be built-out in a way where none of the pages directly interact with one another. The user interface is comprised of a login, create account, and trip (selection and creation) pages. Reliance on another page is not necessary for one page to perform its task, as actions on a singular page will be interacting directly with the server.

2. Software Design Description

2.1. App / UI

2.1.1. Design Overview

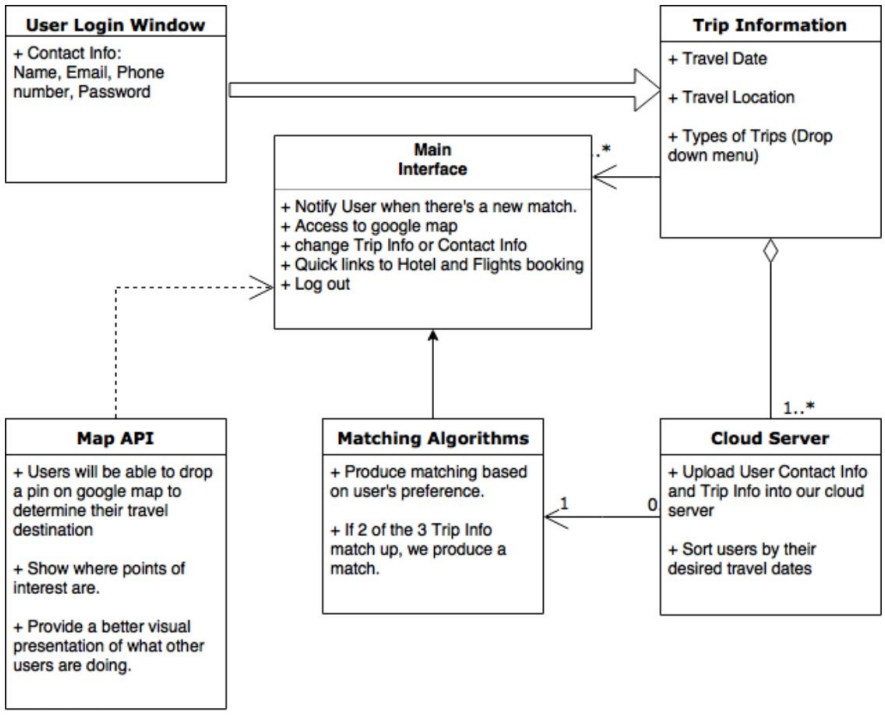
Are you performing a functional design or an object-oriented design? What is the overall function of this component or subsystem?

Our app will be built-out and perform the appropriate tasks through the manipulation of pages. The overall function of this subsystem (the UI) is to serve as a medium for the user to interact with the server. The data provided by the user will be input through the UI in order to reach and be stored in the server. Examples of actions that are performed through the UI: creating new accounts, logging into existing accounts, and creating new trips.

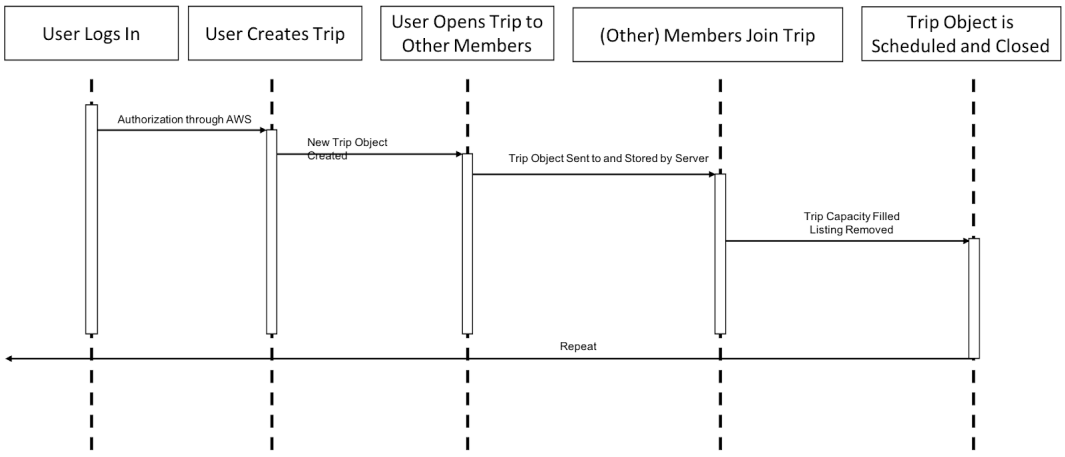
2.1.2. Language and Infrastructure

The Travel Buddies UI will be written in Swift since it is an iOS application. Our application will be divided into several pages that each perform a different task. Apple's IDE, Xcode, will be used to manage and run the files that make up this software project.

2.1.3. Class diagram or data structure diagram



2.1.4. Sequence diagram for key use cases



2.1.5. Detailed Design

2.1.5.1. Logic/Algorithm Design

As has been stated in previous reports, the most logical way to divide our software into modules is by pages. These do not require any real algorithmic design, as they will not be overly complex in this regard, the focus being on their aesthetic and interaction with one another.

An important note is that we will employ matching algorithms for user login and trip creation. When information is required from the server, either for

authorization or updating, it will need to be matched or checked against current information to make sure the update is valid. This will be the basis of our server use as information will be stored in hierarchical JSON format and requested by our application by user actions.

2.1.5.2. State Diagram (if applicable)

For the purposes of our application, a state diagram is not applicable.

2.2. Server

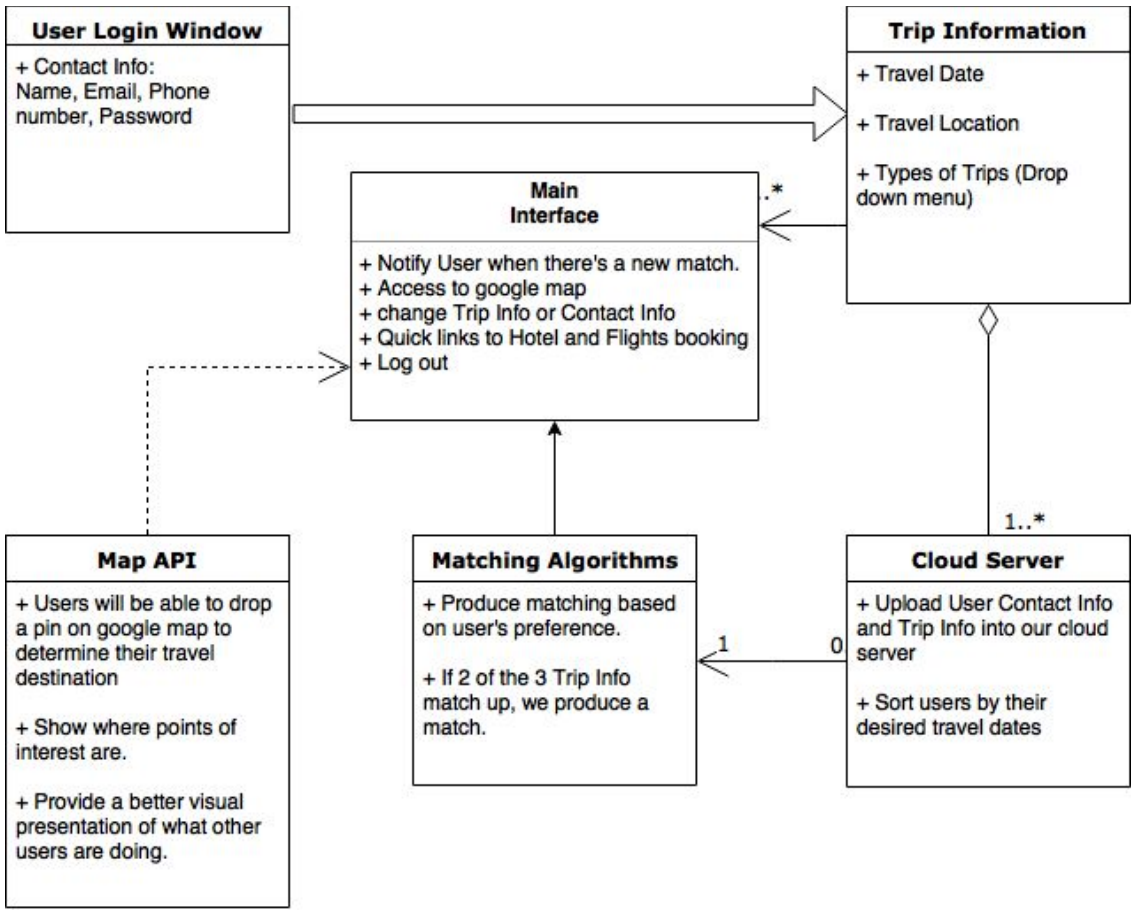
2.2.1. Design Overview

The server will hold a JSON file that has two types of objects: Trip and User/Profile. It will be queryable and respond in useful way. So at a high level, the server exists solely to hold and process the information that the user or app may possibly want access to while hiding the information that the user shouldn't have access to, like other people's passwords.

2.2.2. Language and Infrastructure

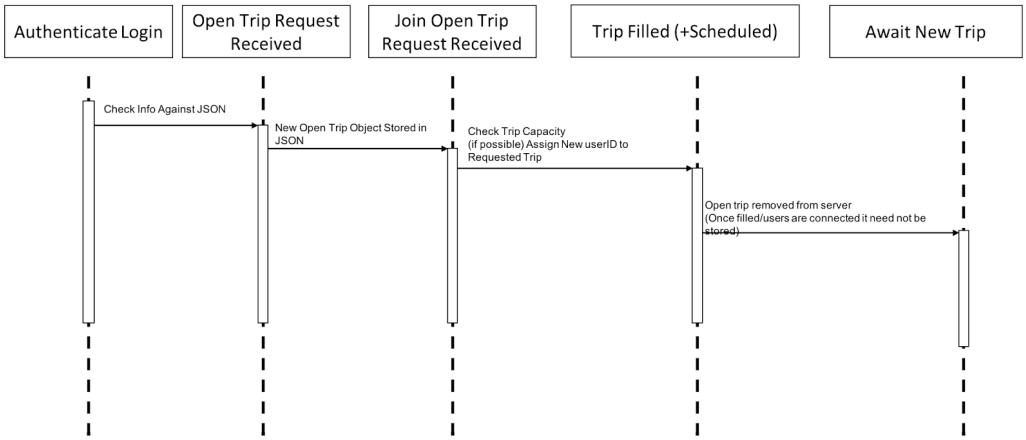
As we will be using AWS Mobile Services iOS development SDK, the server integration will occur seamlessly within our Xcode project and be written in Swift using the required AWS packages.

2.2.3. Class diagram or data structure diagram



2.2.4. Sequence diagram for key use cases

Below is a sequence diagram for the main server use case, posting a new trip object.



2.2.5. Detailed Design

2.2.5.1. Logic/Algorithm Design

The internals of our server will make use, for the most part, of functions provided by the AWS iOS SDK that we will be using. Although triggered by some sort of request or refresh on the user end, in terms of the actual code we will utilize built-in functions to push new information to the server, authorize user login, and check user requests to join open trips. These functions will simply open and close connection to our server that will then add, remove, or cross-reference information with our central JSON file.

2.2.5.2. State Diagram (if applicable)

For the purposes of our application, a state diagram is not applicable.

3. Inter-component or inter-subsystem communications

3.1. Login page

3.1.1. Interface description. Include maximum data rate, message format (if applicable), response format (if applicable), error handling

The interface is comprised of multiple pages that have been previously enumerated.

3.2. Account Creation

The account creation page can be accessed via a hyperlink on the main login page. When an account has been created, the user will be redirected to their profile page. Upon creating an account, the application will query the server as to whether such an account already exists. If it does not, then the server will store this new account.

3.3. Trip Selection Page

The trip selection page can be accessed from the profile page and the trip creation via hyperlinked tabs (XCode provides "buttons" to perform such operations). Upon selecting a trip in this page, the trip selection page will query the server in regards to whether such profile has already selected this trip. If not, the server will store this trip as a "trip selected" by the user.

3.4. Trip Creation page

The trip creation page can be accessed from the user profile page. It will be indicated on the profile page via a hyperlinked tab. This tab is classified as a "button" in XCode and can be dragged onto the interface directly in the IDE. When the user attempts to create a trip, the trip creation page will query the server as to whether such a trip has been created. If it has not, the new trip will be stored in the server.

4. Metrics

4.1. Size information

4.1.1. Number of Modules

There are 5 modules, three in the UI and two in the server.

4.2. Complexity

4.2.1. Weighted Methods per Class

We have two high level classes, the UI and the Server. Within the UI each page constitutes a different method with each specific action within the page being roughly equivalent to a helper function. So since there are three pages of relatively low complexity let's give each page a complexity value of 1 and call the UI a complexity of 3.

The Server has two main parts, a database and an app. The database has two types of objects in it, so let's give it complexity 2. The app will handle most of the heavy lifting of our app. It has to do two main things: Account Authentication/Creation and Trip Searches. The account handling is not that complex, check if it exists, if it does then sign them in or don't let them make the new account; Complexity 1. The trip searches

are where it is really interesting. This could by far be the most complex piece of the entire software if we were to attempt to heavily optimize the searching algorithm. But since we are not attempting to sell this app, we will likely just use a simple brute force search. So again Complexity 1.

This means that the app has complexity 5 overall which means it is a very basic application. This makes sense considering that we picked the app to explore iOS development not to have a super challenging coding problem.

4.2.2 Information Flow Metrics