TED University

Department of Computer Engineering

CMPE 491 Senior Project I

Plan My Day

High-Level Design Report

Project Group Members: Gökalp Fırat
Nehir Yiğit Karahan

Supervisor: Tayfun Küçükyılmaz

January 02, 2019

Contents

High-Level Design Report

Plan My Day

1 Introduction

Plan My Day is being developed to help people with managing their daily plans. People cannot manage their time efficiently due to some reasons like not having enough knowledge about the city they are living in or locations of the places where they want to be.

We are going to develop a mobile application which is PMD (Plan My Day) as a solution to these problems mentioned above. Users tell their plans to the application by writing either speaking and the most efficient path for them is decided by our software.

1.1 Purpose of the System

The purpose of the system is to have an autonomous and robust system to achieve efficiency in planning daily life. To do this, we need to achieve these main goals:

- Providing a useful software solution to make planning daily life easier,
- Reducing the effort of end-users
- Providing a quicker and easier method to customers on finding places they want to go
- Reducing the mistakes which manual operations may cause. For example, the user may pass a place when going to another one. Then, s/he would have to come back to the place that s/he missed before. S/he would walk in vain.
- Keeping track of the users and their plans.

1.2 Design Goals

We consider the comfort of the end users who will use our software. In order to make the software reliable and functional, we will try to make it fault-tolerant and we will add easy-to-use interfaces. Besides these, our first goal will be the satisfying both functional and nonfunctional requirements as developers.

The design goals of the system are:

1. **Security:** This goal provides that the system keeps and protect data and resources.

2. **Usability:** The system meets user’s requests.

3. **Portability:** Converting one platform to another platform can be easy for the system.

4. **Reliability:** The system can maintain performance for all over times.

5. **Usability:** The system should aim understandable, useable by its intended users.

1.3 Technologies Used

Java: is used as main programming language to implement our program.

Android SDK: This development kit includes all the tools and libraries we need for an Android application.

DBMS: Our application needs a local database system. Considering performance and cost DBMS is the perfect choice for our application.

Python: It is our server side language for applications server requests. It will be used for dividing sentences into words and extract locations inside these words. These will be done with server because of the performance cost issues.

Android Studio: Android Studio (Integrated Development Environment) will used with Java programming language as our main development environment.

Firebase: An online fast and easy to use database. It is needed to store login information of users.

1.4 Definitions, Acronyms, and Abbreviations

Android Studio: Integrated Development Environment (IDE) for Android App Development

DBMS: Database Management System

Python: Interpreted, object-oriented, high-level programming language

Java: Compiled, object-oriented programming language

1.5 Overview

We want to create a mobile app which makes people's life easier to plan. The app will listen and read user's plans out loud like a friend or assistant. The app uses phone's GPS to find locations and it prepares the best efficient path for a user to follow up. App expects users to enter their plan's detailed information such as location and time. With this information, the app can organize the day for them as best it can, so the user can schedule his/her day efficiently.

It may not very useful for people who know the best paths for their plans. However, it can be very handy for people who don't know the city or paths. Users can tell their plans with voice or write as a text. We expect users to use sentences such as "I want to go a shopping mall then I want to eat a hamburger and I need to withdraw some money", this is for converting the sentence into small plans and find best locations for them. There will be also voice notifications to warn the user about their plans. This app can be very useful to minimize the time used for the plans and maximize the free time.

A user can't see other user's plans, and this provides privacy for our users.

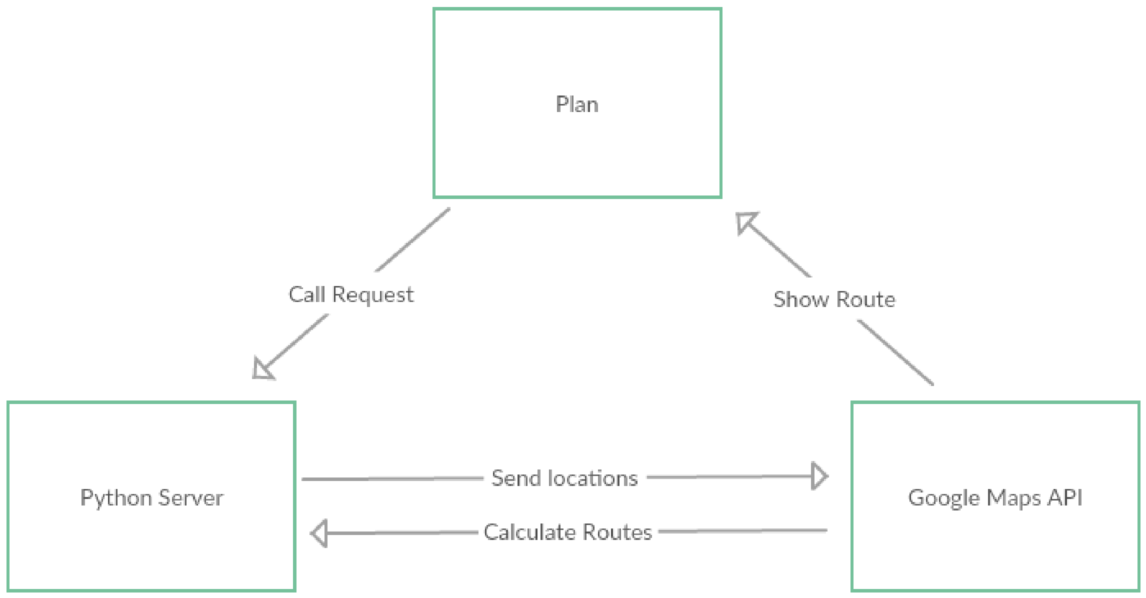
2 Software Architecture

There will be two databases one of them will be the firebase database for us to see our users and handle some operations on them when it is necessary. And the other database will be the SQLite which will run locally on user’s devices in order to provide the option of saving the plans and reuse them as they wish. Our interface will be designed for this, and there will be easy to use interfaces for users to have all the provided functionality of our application.

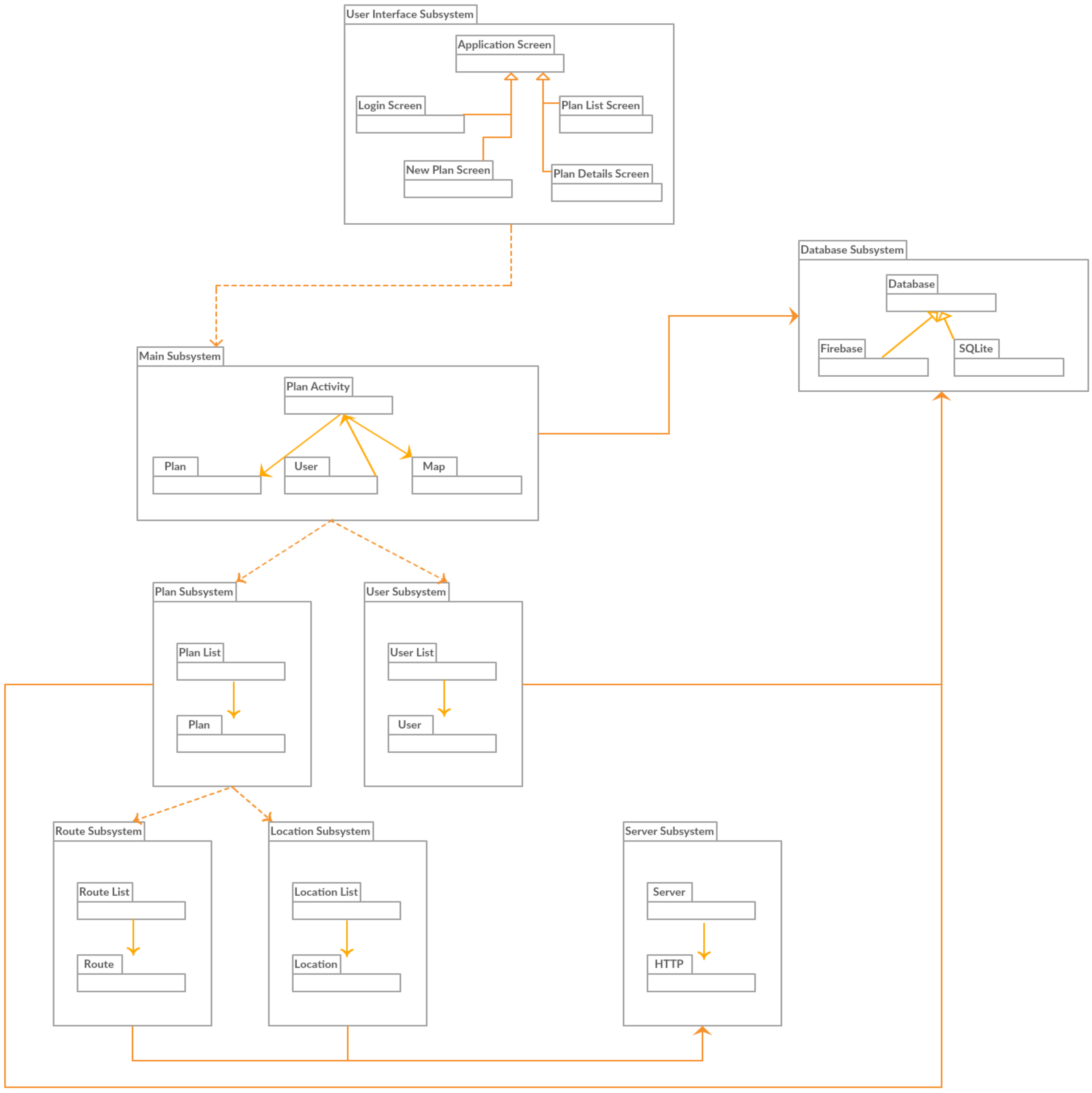
Our program will use the Google maps API in order to create the plans and show those plans to the end users. GPS will be used to get the end user’s location and plan him/her accordingly to this information.

In the interface, users will be able to enter their plan by speaking or writing to the relative fields and after they finish, our program will create a task list in order and show it to the user by asking if the user approves or rejects the plan. We are doing it in case we may fail in the parsing the user’s input plan into wrong tasks. If the user approves his/her plan, our app will create the user’s map accordingly to the plan and put it on the user’s device and users can choose to be navigated after this point.

We will create a dataset of ours in order to understand the places on the Google maps can be used for the user’s plan or not. For example, a user told his/her plan and mentioned that s/he will buy flower but didn’t mention which flower shop she/he will be going. So, we need to find flower shops for her/his. To achieve this, we need to create a dictionary first. This dictionary will be used to understand the relationship between the flowers and flower shops. So, we will be sending the user to the correct places where selling the flower. We are planning to use python for this.



2.1 Subsystem Decomposition



2.1.1 Subsystem Services

The User Interface Subsystem is responsible for all interaction between the users and relatives. User interface decomposed into a set of pages loaded and displayed in device. All entry data, modifications will be handled directly through the user interface.

The Main Subsystem is responsible for all interaction between UI system and the Plan Subsystem, User Subsystem. The Main Subsystem is also responsible for interactions between the Plan Activity class and the Plan, User, Map classes that make up the Main Subsystem. The Main Subsystem is also responsible for the interactions between the User and Plan classes to the Database Subsystem. Each class is responsible for utilizing the appropriate and necessary methods within the Database class to crud (create, read, update, delete) actions to all corresponding data.

The Plan Subsystem is responsible for all interactions involving Plans. All interactions with any Plan will be handled through and by the Plan List class, except for Database access, from within the Plan Subsystem. The Plan Subsystem is also responsible for the interactions between the Plan List and Plan classes/objects to the Database Subsystem. Both the Plan List and Plan class will each be responsible for utilizing the necessary methods within the Database class to crud actions.

The User Subsystem is responsible for all interactions involving Users. All interactions with any User will be handled through and by the User List class, except for Database access, from within the User Subsystem. The User Subsystem is also responsible for the interactions between the User List and User classes/objects to the Database Subsystem. Both the User List and User class will each be responsible for utilizing the necessary methods within the Database class to crud actions.

The Route Subsystem is responsible for all interactions involving Routes. All interactions with any Route will be handled through and by the Route List class, except for Server access, from within the Route Subsystem. The Route Subsystem is also responsible for the interactions between the Route List and Route classes/objects to the Server Subsystem. Both the Route List and Route class will each be responsible for utilizing the necessary methods within the Server class to requests and response actions.

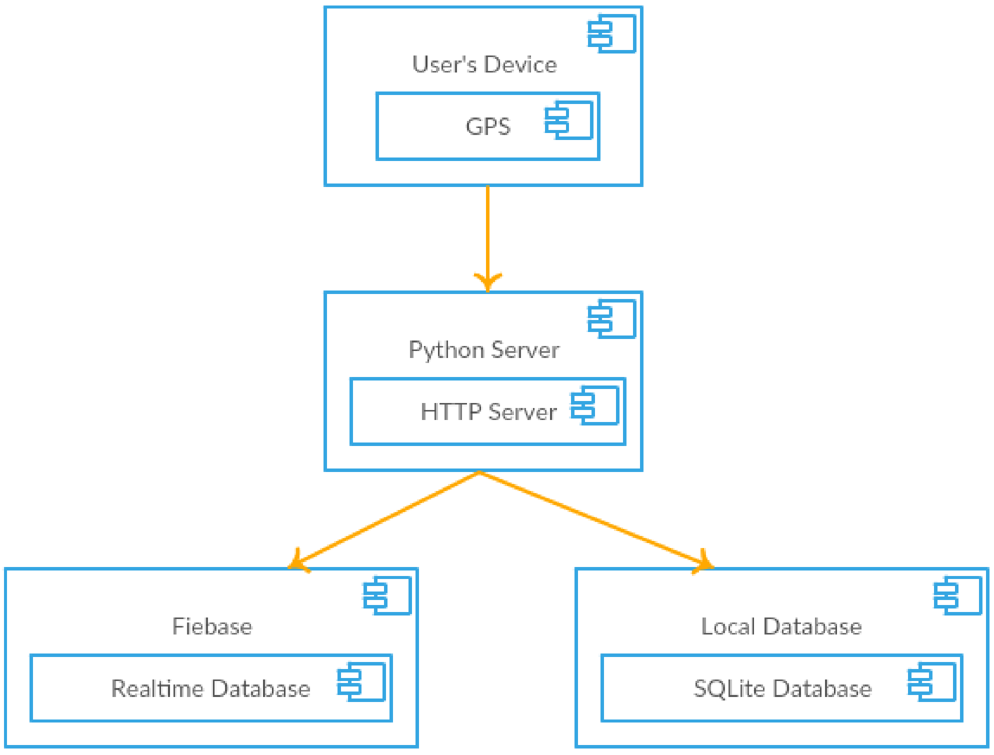
The Location Subsystem is responsible for all interactions involving Locations. All interactions with any Location will be handled through and by the Location List class, except for Server access, from within the Location Subsystem. The Location Subsystem is also responsible for the interactions between the Location List and Location classes/objects to the Server Subsystem. Both the Location List and Location class will each be responsible for utilizing the necessary methods within the Server class to requests and response actions.

2.2 Hardware/Software Mapping

The following components make up the Plan Application:

- User’s Mobile Device
  - GPS
- Firebase
  - Realtime Database
- Python Server
  - HTTP Server
- Local Database Server
  - SQLite Database

The User’s Device activates GPS and send request to Python Server via HTTP/HTTPS protocol using the internet service in their device. The Python server has a HTTP Server which contains responses and functionalities. Python Server communicates with Firebase or Local database according to request using SQLite class or API for connecting and performing actions.



2.3 Persistent Data Management

Application will largely depend on a relational database to perform modifications from users and storing these data. User data will be stored at Firebase Server and can be manipulated through the Database Subsystem, which will ensure data integrity and consistency. There will be local database too. It is for storing plans for latterly use. Database Subsystem will contain all necessary commands & queries that will be accessible by the rest of the Subsystems.

The data stored in the Firebase and local database will include:

- Users
- Plans
- Routes
- Locations
- Feedbacks

2.4 Access Control and Security

Our system has an authentication system and it is based on google authenticator since we are using Firebase in our server side. Even the 2-step verification process may be activated if users are using their Google accounts to log in. Authenticated users will be able to use only creating and deleting plans and this database write operations will be held on the local database since we use SQLite to keep the plans locally on their device. Besides, our user credentials will be held on the Google servers itself because we are using firebase and its owner is Google.

There will be no security vulnerability in our application as long as there is no security vulnerability in Google's servers. The only way of an attacker can harm our users is stealing their phones physically when they logged in and loading their plans and observe to have some ideas about their life. Unless someone steals user's phone, nobody can see the plans.

	Administrator	**User**	**Guest**
Logging In	Yes	Yes	Yes
User Settings	Yes	Yes	No
Create Plan	Yes	Yes	Yes
Plan Info	Yes	Yes	No
Edit Plan	Yes	Yes	No
Ban User	Yes	No	No
Logging Out	Yes	Yes	No
Save Plan	Yes	Yes	No

2.5 Global Software Control

When a new user registers with e-mail and password we are recording him/her to our firebase real-time database and all users can be controlled or observed from the firebase console. The authentication system is consistent because it is based on a single structure which is firebase login.

When a user creates a plan and chooses the save it, it is saved to the local database (SQLite) and as long as the user doesn't delete it, or the maximum number of saved plans is not reached, the plan will be there and will be ready to reuse.

The Plan application architecture is to have an explicit decentralized software control. The system control is distributed among different activities such that each object delegates some responsibility to other objects. The requests are event-driven:

- The participant initiates the User Interface Subsystem by logging in.
- The Main Subsystem initiates the Plan Subsystem by creating a plan.
- The Main Subsystem initiates the User Subsystem by login.
- The Plan Subsystem initiates the Route Subsystem by checking the plan info.
- The Plan Subsystem initiates the Location Subsystem by searching a place.
- The Database Subsystem can be initiated at any point by Plan, User and Main subsystems. Any crud request (add/update/delete/retrieve) to the repository initiates the Database Subsystem. The system stores its contents in the database between executions. When the application is run again, it retrieves the contents from the previous execution. Any change in the contents during this execution updates the database.

2.6 Boundary Conditions

Startup: Go to application and login
Shut Down: End the process of application from tasks
Error Conditions:

- **Logging in:**
  - Username or password field are blank.
  - Password is not minimum 6 characters long.
  - Password and username don't match.
  - Username is wrong or does not exist.
  - Page don't redirect to index after login
- **User settings:**
  - The system crashes while editing and updating.
  - User is unable to change certain settings without permissions.
- **Plan Creation:**
  - User is not confirmed mail address.
  - Plan text is not clearly indicated.
  - Already a plan exists in same name
  - Couldn't connect to database
  -

- **Search Route:**
  - Can’t connect to Google API.
  - No route exists.
- **Find Locations:**
  - Too many locations.
  - Can’t find the location from name.
- **Check Plan Info:**
  - Plan is passed.
  - Plan is closed.

3 Glossary

Plan: A detailed proposal for doing something

Place: A particular position in space

GPS: Global positioning system, accurate worldwide navigational system

Navigate: Plan and direct the route of a person from his/her current location

SQLite: SQLite is a relational database management system. It is a software library. This library has some features which are self-contained, serverless, zero-configuration, and transactional SQL database engine [1].

SQL (Structured Query Language): SQL is a standard language for relational database system. It is a language to operate databases; it includes database creation, deletion, fetching rows, modifying rows, etc. [1]

Database: Data storage system

UI: User Interface

4 References

[1] https://www.tutorialspoint.com

[2] Object-Oriented Software Engineering, Using UML, Patterns, and Java, 2nd Edition, by Bernd Bruegge and Allen H. Dutoit, Prentice-Hall, 2004, ISBN: 0-13-047110-0.

[3]http://www.cs.fsu.edu/~myers/cop3331/notes/requirements.html

[4]https://balsamiq.cloud