Bilkent University

Department of Computer Engineering

# Senior Design Project

Project Final Report

*Consigliere*

Selin Erdem
Furkan Küçükbay
Orhun Çağlayan
İrem Yüksel
Umut Mücahit Köksaldı

| | |
|---|---|
| Supervisor: | Assoc. Prof. Dr. Mehmet Koyutürk |
| Jury Members: | Prof. Dr. Uğur Güdükbay |
| | Prof. Dr. Cevdet Aykanat |

May 3, 2018

This report is submitted to the Department of Computer Engineering of Bilkent University in partial fulfillment of the requirements of the Senior Design Project course CS491/2.

# 1. Introduction

In the last few years, people's daily lives have changed in many different ways. Simple things became more complicated and we started to take more responsibilities and with more responsibilities, came higher expectations. Consequently, concepts such as making plans, using our time efficiently have gained more importance than ever before. Yet, it is a known fact that nowadays people struggle when it comes to planning a day and running errands. With the current technologies, it is possible to make a plan by entering the details such as time and place, and having reminder alerts for these tasks, but these applications do not help the user in terms of time efficiency and organization of their tasks.

Consigliere is both an iOS and Android application that will work as a daily organizer and task manager. With Consigliere, we aim to help people use their time more efficiently and regain the time wasted on traffic. The user simply needs to enter whatever errands they have to run for that day and the application will provide an optimal plan for the user to complete these errands. This plan is designed by taking into account the roads the user will have to take to get to the locations of their tasks, and the traffic situation in the road. The application periodically checks the traffic status of relevant roads and the crowd level of the locations in order to update the daily plan dynamically and send the user push notifications informing them of the opportunities to run their errands in a timely manner. The user can distribute his/her errands between the tasks that have specified starting time and the errands between those tasks will be optimized.

In this report, the final version of Consigliere will be described. Firstly, we will give a brief summary of Consigliere's functionalities. Then, the algorithms that we use will be described. Afterwards the project's final architecture and design will be shown. Afterwards, the global, economic and social impacts of Consigliere and the engineering solutions and contemporary issues will be briefly discussed . The tools and technologies we used will be listed. Lastly, the user manual and class diagrams of Consigliere are included.

## 2. System Overview

Imagine the day-to-day routine for a person. It would not be uncommon for them to have to travel to different locations in order to check off every item in their to-do-lists. Using the current apps on the market, they would first need to keep their daily tasks in a to-do-list app and then manually enter their locations into a navigation application in order to get where they want to go. Furthermore, they do not know in which order they should complete their errands in order to spend less time in traffic and finish all of their errands as quickly as possible.

Consigliere combines all these steps into one, by building the bridge between a task and its location, and using traffic data to suggest the best route for a user to complete their errands and arrive at their intended destinations in an efficient manner.
In Consigliere, each user has an account that they can use across different devices and access their tasks at all times. If they do not have an account, they can also create one from the login screen.

After logging in, each user will be presented with a list of their tasks and errands that they have added for that day. If they wish, they can also view other days for better organization and planning. They can order their tasks and errands to specify the time interval that their errands will fall in. They can also delete their tasks or mark them as complete.
In order to create new tasks, the user needs to switch to the map screen and look for the location of their errand / task on the map using the search functionality. After doing so, they can add a task / errand for that specific location which will be added to their task lists for future reference and route planning.

Finally, the users can use the 'generate route' functionality to get the most optimized route for the errands that fall into the current interval. They can view this route inside Consigliere's map screen, or they can transfer the route over to Google Maps for active navigation.

## 3. Consigliere Algorithmic Design

The application fetches currently added tasks and errands from database in order to show them in a descriptive manner. Then, user can arrange the tasks and errands by drag-dropping the errands between the tasks, so that app can generate an optimized route considering the preferences of user. When user presses the calculate route button, app will send a query to Google Maps for the route between user current location and first next task of the user. If there are errands to be done before the first next task, we consider these errands in our route also. Then, app will send the user to the Google Maps application for the navigation.

## 4. Final Architecture and Design

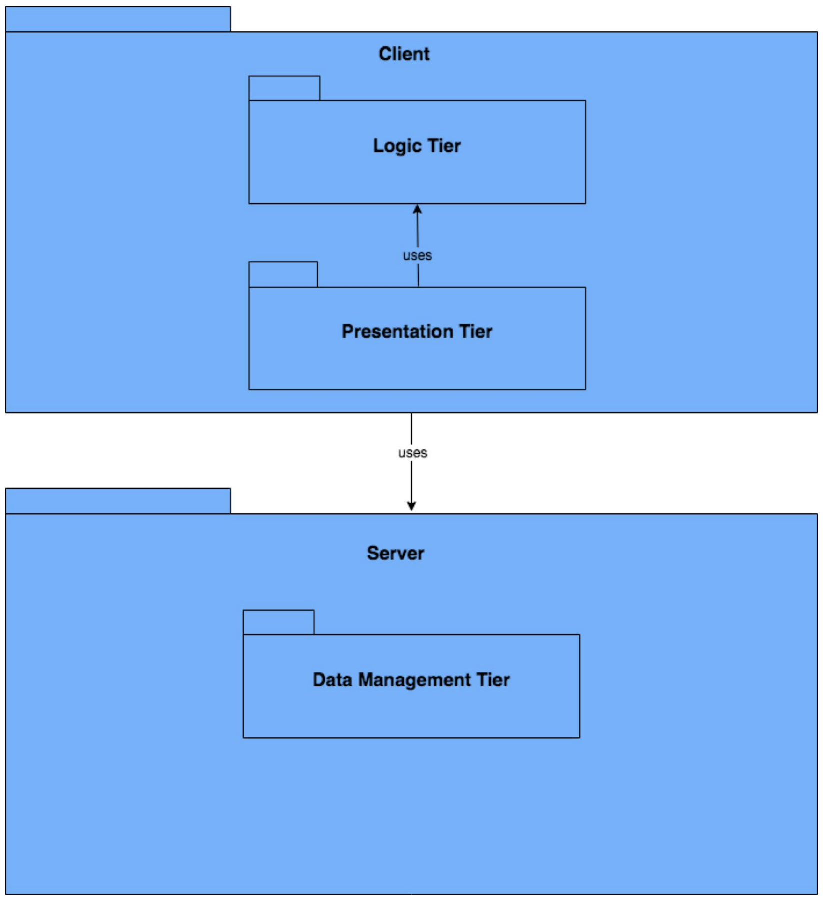
### 4.1. Overview

In the subsystem decomposition, the subsystem structure of our system is described in detail. Firstly, we provide an overview of the main components that make up our application, the client and the server. Then we provide the details of how different software components inside the client layer interact with each other and exactly which parts of the client establish the communication between server and device. Furthermore, we specify the features of the server layer which is mostly responsible for persistent data storage and user authorization.

### 4.2. Subsystem Decomposition

Our application follows a client/server model, with the server side being responsible for data management, and the client handling the logic and the presentation tier. The data management tier of the application handles the addition, deletion and editing of user created tasks, as well as any authorization regarding the user's account credentials. The logic tier of the application is responsible for interacting with the server to gather the location and time data of the user's tasks, as well as the specified distribution of errands between the tasks. Then the logic tier will provide the user with an optimized route to complete their errands in the most time efficient manner, taking into consideration the traffic and distance between task/errand locations. The presentation tier of the application communicates with the server to retrieve user and task related data. It lets the user view their recurring tasks for easy editing and deletion operations, as well as providing a calendar for the user to both view and manage their tasks / errands.

Consigliere uses its server to store user account information such as email and password, as well as storing all data related to user created tasks such as date, time and location. The route optimization and the general user interaction is handled by the client which is the mobile application side of Consigliere. We went with this specific distribution of tasks for server and client, since we do not need to scale up the number of tasks our application needs to handle, considering that most users will have a relatively small set of tasks for each day. Rather, we would want portability of data from one client to another, should the user want to change their mobile device, and scale up the number of users if needs be. In addition, this model will also allow us to port the application to other mediums such as smart-watches or desktop, since most people may want to plan their tasks on a desktop and access the provided route from a mobile device.



Subsystem Decomposition

## 5. Impact of Engineering Solutions

Consigliere has a lot of impacts on personal time management. Firstly, there are a lot of map applications and separate agenda applications. These applications either provide map service or work as daily planner. However, there are no conventionally used application which mixes agenda and map like Consigliere. On the other hand, Apple Calendar can label a task with a location, but in Consigliere you can see your tasks and errands on a map with one simple tap. Therefore, we put the map and agenda close to each other in order to give user better experience than map or agenda application.

Also, there are two type of jobs in our application which improves user experience since user can both add less important daily errands and time specific tasks to do. Therefore, our aim and vision when creating consigliere is giving users the ability to plan their day by visualizing their to do list on a map.

Finally, as user presses the route calculation button, we send the user to google maps and start the navigation so that user does not lose any time by entering his tasks to google maps by himself.

## 6. Engineering Solutions and Contemporary Issues

Today, time is one of the most important resources in our daily lives. Personal efficiency is quite important and time management is what people seek. In our application, one can plan his/her day on the map and manage it. Therefore consigliere tries to solve one of the modern problems of contemporary daily life: time management.

Furthermore, we tried to implement cross platform mobile application since we aim to address bigger user base. Currently, mobile applications are quite popular and most people user mobile phones as their primary computer. By doing so, we are trying to catch the current trend with Consigliere.

We also store the user data in a firebase database so that user can login to his account by using any arbitrary phone in which consigliere is installed and manage their tasks. By using current availability of internet, we gave the user flexibility of  online account so that he/she does not need to store the data locally.

## 7. Tools and Technologies Used

**React Native** is a framework for building native apps using only JavaScript without bothering too much with either Objective-C or Java [1]. We used React Native to build both Android and iOS versions of our app. Developed by Facebook, with its thriving third party library resources, React Native eased the coding process.

**Atom** is a free and open-source text and source code editor that can be extended and configured with plug-ins [2]. We used Atom with ESLinter plug-in to code the React Native project.

**VSCode** is a source-code editor developed by Microsoft [3]. Together with Atom, we used VSCode as our IDE.

**GitHub** is a Git repository [4]. We utilized Git as the version control system and used GitHub as our repository. Team members worked in different Git branches to be later merged into the master branch.

**Firebase** is a NoSQL cloud database [5]. We stored all our data including users for authentication and date-based task details.

**Android Studio** is an IDE for Android Development [6]. We used the emulators Android Studio provides to test our code. We also used logcat functionality of Android Studio to debug the code running in Android.

## 7.1. Library Resources

**axios** is a promise based HTTP client for the browser and node.js [7]. We used this library to make API calls.

**lodash** is a Javascript utility library. We used lodash to ease the hassle of working with arrays, strings and JS objects.

**redux-thunk** is a middleware that allows to write action creators that return a function instead of an action [9]. We used redux-thunk to be able to use component states across different components.

**react-native-router-flux** is a library to handle navigation inside app.

**react-native-background-task** is a library to realize periodic background tasks even if the app is closed [10]. We used this library to determine the remaining time for the first next task and send notification to the user when the time comes.

**react-native-google-places** provides Google Places Widgets and API services for React Native Apps [11]. We use this library to handle Google Places search facility and getting current location.

**react-native-sortable-list** is a library that provides drag and drop wrapper for Lists in react-native. We used this to sort our tasks and errands in our opening page when the user wants to sort his/her errands.

**react-native-push notification** is a library to handle local and remote notifications in both iOS and Android [12].

### 7.2. APIs Used

**Google Maps API** is used to display maps, search places, find the optimized routes for errands and navigate the user when route button is pressed.

## 8. References

[1] https://facebook.github.io/react-native/
[2] https://atom.io/
[3] https://code.visualstudio.com/
[4] https://github.com/
[5] https://firebase.google.com/docs/database/
[6] https://developer.android.com/studio/intro/
[8] https://github.com/axios/axios
[9] https://github.com/gaearon/redux-thunk
[10] https://github.com/jamesisaac/react-native-background-task
[11] https://github.com/tolu360/react-native-google-places
[12] https://github.com/zo0r/react-native-push-notification