Bilkent University

Department of Computer Engineering

Senior Design Project

Consigliere

Project Analysis Report

Selin Erdem, Irem Yüksel, Orhun Çağlayan, Furkan Küçükbay, Umut Mücahit Köksaldı

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

Project Analysis Report
Nov 6, 2017

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.

Contents

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 will be an iOS 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 will simply need 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. In addition, the application will estimate how much time the user will spend on an errand by analyzing the busyness of the location and the nature of the user's task. The application will also periodically check 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.

In this report, first the existing systems, their qualities, and the missing features of the current systems are described. Then a description of Consigliere is provided. Functional, non-functional, and pseudo requirements are presented. Afterwards, the system models of our system are included. Use case descriptions and the use case diagram is given. The object model and the dynamic models of the system are also provided and explained. Finally, the screen mock-ups and the navigational paths are included.

2. Current System

Current systems can be examined under two main functionalities: map applications and daily planner applications. Map applications generally gives user a time or distance optimized route between two or more points which user enters to the application. The map applications generally lack the optimization when it comes to calculate the route between more than two points. For example, Google Maps calculates the route between three nodes with given order (go to first point, then second point, then third point) even if going to the second point first shortens the distance.
Secondly, Daily planners lack the functionality of maps when it comes to planning the day. Generally, daily planner applications work as simple agendas and do not optimize the time or distance. Although there are several applications for calculating the route and planning the day separately, there are no application for planning the day on the map in a time-efficient manner.

3. Proposed System

3.1 Overview

Consigliere is basically an application that enables users to plan and organize their errands in a time-efficient manner by suggesting dynamic daily plans. The applications main functionality will be mapping out a route that enables users to run their errands unerringly and efficiently. Main goal of our application is to help the users to finish his daily tasks in timely manner. It will also offer other several useful functionalities such as sending reminder notifications and saving parking spot to find the car easily.

What makes Consigliere different from classic task managers or map applications such as Google Maps or a regular daily planner is its incorporation of daily routine and task planning. Although most map applications provide route planning by selecting multiple points of interest, they fall short at optimizing the efficient route in terms of time and distance. Similarly, daily planners generally work just like agendas and only notifies the user about the upcoming events or tasks but they do not offer time management in terms of route optimization.

Consigliere will offer number of features such as authorization, task managing, optimization of daily plans, route planning according to several crucial constraints such as traffic intensity. Main challenge of this application is finding an optimized, relevant route that includes location of each errand that the user should run, which is basically an algorithmic and innovative approach to the travelling salesman problem [1]. Furthermore, the application will optimize the way that users run their daily errands in the interest of saving time using different factors such as traffic intensity, road conditions and crowdedness. It will also offer several useful functionalities such as reminders.

3.2 Functional Requirements

3.2.1 User Profile

- Anyone with a valid email address will be able to sign up to use Consigliere.
- Their data and entered tasks will be bound to their user account and synchronized across other devices.

3.2.2 Daily Planner

- The user will be presented with a daily plan on how to complete their errands in the most efficient way possible. The plan will be devised in a similar way to the Traveling Salesman Problem [1].
- The daily plan will consider the traffic data of the roads in order to not let the user get stuck in traffic while going to a certain task's location.
- The application will even reroute the user to do other tasks if the roads will open up in the meantime.
- The application will try to predict how much time the user will spend while doing an errand, and adjust the daily plan accordingly; for instance, if the user's task at the bank will take a considerable amount of time and the traffic will clear up during the time he/she spends at the bank; then the application will first direct the user to the bank even if it may be further away from the user's location.
- The application will consider the busyness of the locations at different times of day and guide the user to complete their tasks in the most appropriate order as possible, combined with the aforementioned optimization approaches.

- The application tries to look at the user's task and constraints in a holistic manner and tries to offer the most time-efficient way that is possible to complete all of the tasks from start to finish.
- The application will try to analyze the patterns of the user in order to determine their preference regarding the tasks and their respective locations.
- The application will periodically check the traffic and the crowdedness of locations and update the daily plan as the day progresses.
- The application will save the location of the parking spot that the user parked his/her car.
- The application will send the user push notifications if the traffic has cleared up or the busyness of a location has gone down either expectedly or unexpectedly, prompting the user to complete their tasks in a timely manner. The application will provide the information of how much time the prompted task will take in terms of travel and the actual errand time.

3.2.3 Task Management

- The application will try to extract the location and the task type from the task input. This is to minimize the burden of specifying the location for each task individually.
- Should the application not be able to extract the location or extracts it with some error, the user will be able to go into more detail and specifically set the location for each of their tasks, if they wish to do so.

3.3 Nonfunctional Requirements

3.3.1 Usability

- The user interface must exhibit conceptual integrity and simplicity.
- The user interface should be user-friendly.
- Novice users must be able to install the application and operate its major use cases with little or no training.

3.3.2 Supportability

- Dependencies in the design of the system should be minimized to allow quick updates in the future.

3.3.3 Reliability

- The traffic data used by the application must be up-to-date and accurate.
- The suggested routes must be accurate and route suggestions must consider temporary factors such as roads under maintenance and construction, or road closures.

3.3.4 Efficiency

- The system should be able to complete the planning task under 10 seconds.
- The response time of the system should be less then 100 milliseconds.

3.3.5 Security

- The system should ensure security of personal data of the users.

3.3.6 Scalability

- The system should support up to 15 task/stops per day

3.4 Pseudo Requirements

- The application will be a mobile application.
- IOS mobile application will be developed.

3.5 System Models

3.5.1 Scenarios

3.5.1.1 Create Account Use Case

Use Case Name: CreateNewAccount

Participating Actors: User

Entry Conditions:

- User is on Login page

Exit Conditions:

- User is on Main page

Main Flow of Events:

1. User presses 'Sign Up' button
2. Application displays Create Account page
3. User enters her username as 'x'
4. User enters her email address as 'x@gmail.com'
5. User enters her password
6. User presses 'Create Account' button
7. Application checks the information
8. Application creates X's new account
9. Application displays the new homepage of User

Alternative Flow of Events:

A. User entries are incomplete /inaccurate

7A. Application warns user about the missing or inaccurate entries.

10A. User completes the missing entries.

3.5.1.2 Login Use Case

Use Case Name: LoginUser

Participating Actors: User

Entry Conditions:

- User is on Login screen

Exit Conditions:

- User is logged in OR
- User authentication fails

Main Flow of Events:

1. User enters his mail address
2. User enters his password
3. User presses Login button
4. Application verifies the username and password
5. User is navigated to the main page

Alternative Flow of Events:

A. User enters wrong username and/or password

4A. Application fails to verify the username and password

5A. Application displays a warning message

6A. Application returns to the login screen

3.5.1.3 Add an Task with Specifications Use Case

Use Case Name: AddSpecificTask

Participating Actors: User

Entry Conditions:

- User is on Planner page

Exit Conditions:

- User is on Homepage page

Main Flow of Events:

1. User presses “add task” button
2. Application displays Add Task page
3. User selects” specific task “option
4. User enters the identifier of the task
5. User enters the location of the event
6. User enters the starting time of the task
7. User presses “Save Task” button
8. Application checks and adds the task to User’s schedule
9. Application updates the schedule if necessary
10. User returns to Planner page

Alternative Flow of Events:

A. User leaves Add Task page without adding any new task

B. User adds an task with time conflict

7B. Application warns user about the time conflict

8B. User changes or deletes one of the conflicting activities.

3.5.1.4 Delete/Modify an Task Use Case

Use Case Name: ModifyTask

Participating Actors: User

Entry Conditions:

- User is on Planner page

Exit Conditions:

- User is on Planner page

Main Flow of Events:

1. User presses on the task that he wants to modify
2. Application displays task page
3. User selects" modify task "option
    a. User changes the time or location of the task
4. User presses delete task button
    a. Application deletes the task from User's schedule
5. User presses "Save" button
6. Application checks the eligibility of the modifications
7. Application updates the schedule if necessary
8. User returns to Planner page

Alternative Flow of Events:

A. User leaves Task page without any change

B. User changes an task's time and causes time conflict

7B. Application warns user about the time conflict

9. User changes or deletes one of the conflicting activities.

3.5.1.5 Add an Errand Use Case

Use Case Name: AddErrand

Participating Actors: User

Entry Conditions:

- User is on Planner page

Exit Conditions:

- User is on Homepage page

Main Flow of Events:

1. User presses "Add task" button
2. Application displays Add Task page

3. User selects" Add Errand "option
4. User selects the errand to add
5. User enters the identifier of the task
6. User specifies the location of the errand
7. User enters the starting time of the task
8. User presses "Save Task" button
9. Application checks and adds the task to User's schedule
10. Application updates the schedule if necessary
11. User returns to Planner page

Alternative Flow of Events:

A. User leaves Add Task page without adding any new task

B. User does not specify the location of the errand

9B. Application finds an optimal location and assigns it to the task

3.5.1.6 Add a Recurring Task

Use Case Name: AddReccuringTask

Participating Actors: User

Entry Conditions:

- User is on Planner page

Exit Conditions:

- User is on Homepage page

Main Flow of Events:

1. User presses "Add Task" button
2. Application displays Add Task page
3. User selects" Recurring Task "option
4. User enters the starting time of the task

5. User enters the repeating days for the task
6. User presses “Save Task” button
7. Application checks and adds the task to User’s schedule
8. Application updates the schedule if necessary
9. User returns to Planner page

Alternative Flow of Events:

A. User leaves Add Task page without adding any new task

B. User adds an task with time conflict

6B. Application warns user about the time conflict

8B. User changes or deletes one of the conflicting activities.

3.5.1.7 Delete/modify a Recurring Task

Use Case Name: ModifyReccurringTask

Participating Actors: User

Entry Conditions:

- User is on Reccurring Activities page

Exit Conditions:

- User is on Planner page

Main Flow of Events:

1. Application displays Recurring Activities page
2. User presses on the task that he wants to modify
3. Application displays Task page
4. User selects” modify task “option in Task page
    a. User changes the recurring time or location of the task
5. User presses delete task button
    a. Application deletes the recurring task from User’s schedule

6. User presses “Save” button
7. Application checks the eligibility of the modifications
8. Application updates the schedule if necessary
9. User returns to Planner page

Alternative Flow of Events:

A.User leaves Recurring Task page without any change

B.User changes an task’s time and causes time conflict

7B. Application warns user about the time conflict

9. User changes or deletes one of the conflicting activities.

3.5.1.8 Add Permanent Address Use Case

Use Case Name: AddPermanentAddress

Participating Actors: User

Entry Conditions:

- User is on Map Page

Exit Conditions:

- User adds a permanent Address OR
- User types invalid address

Main Flow of Events:

1. User types an address to address bar on the map page
2. Application finds the address
3. User Flags the Address as Home, Work etc.
4. Address is added

Alternative Flow of Events:

A. Address is invalid

3A. Application warns the user

4A. Application returns to the Map Page

3.5.1.9 Mark Errand as Completed Use Case

Use Case Name: MarkErrandComplete

Participating Actors: User

Entry Conditions:

- User is on Tasks Page AND
- User finished an Errand

Exit Conditions:

- User marks the Errand as Complete

Main Flow of Events:

1. User Selects the Errand he finished by pressing on it.
2. Pop up containing the information of event is shown to the user
3. User marks the errand as “Completed”
4. Pop up closes
5. Application rearranges the remaining errands

3.5.1.10 Change Permanent Address Use Case

Use Case Name: ChangePermanentAddress

Participating Actors: User

Entry Conditions:

- User is on Maps Page AND
- User has an existing permanent address

Exit Conditions:

- User changes the address OR
- User fails to change the address

Main Flow of Events:

1. User types an address to address bar on the map page
2. Application finds the address
3. User Labels the address with one of the labels he/she previously used for another address
4. The application creates a pop up saying “Change the [Label Name] Address?”
5. User selects yes
6. Address is changed

Alternative Flow of Events:

A. User does not change the address

5A. User presses no button

6A. Address is not changed

3.5.1.11 Calculate Route Use Case

Use Case Name: CalculateRoute

Participating Actors: User

Entry Conditions:

- User has entered at least one errand AND
- User is on the task screen

Exit Conditions:

- Optimal route is calculated and shown to the user

Main Flow of Events:

1. User presses “calculate route” button after entering at least one errand to the app
2. Application calculates the optimal route given tasks, user location and live road data
3. Optimal route is shown to the user by switching to the map screen.

3.5.2 Use Case Model

3.5.2.1 Use Case Diagram

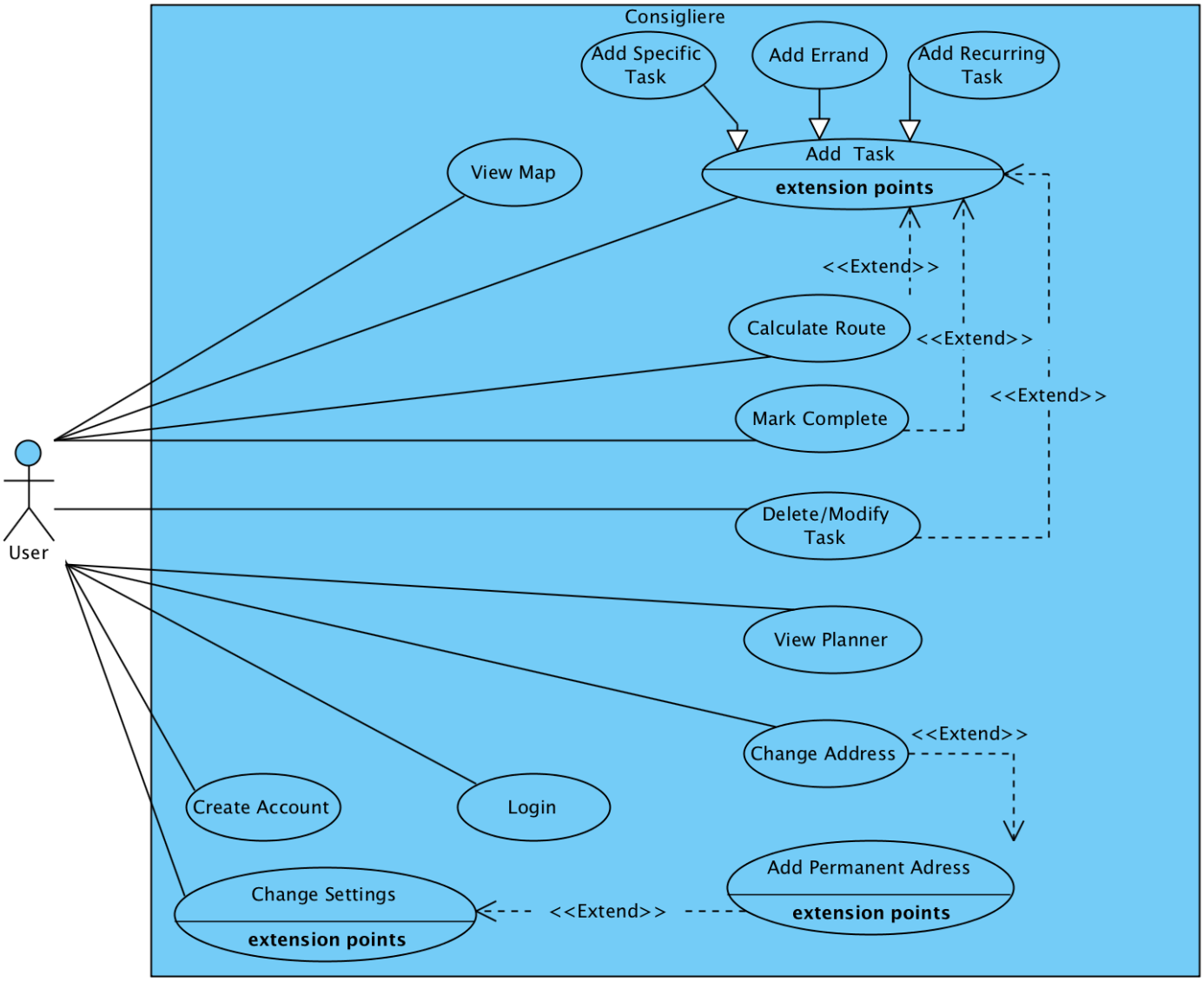

Figure 1. Use-Case Diagram

Note: "Login" is done as part of all use cases

3.5.2.2 Use Case Descriptions:

- **Create Account:** Create a new account
- **Login:** Login to the app
- **Add task:** Add a task to the application (general case)
- **Add Specific Task:** Add a task which has a specific time and location

- **Add Errand:** Add an errand which has no specific time or location (i.e flexible activity)
- **Add Recurring Task:** Add a task which will recur at several times in a week or month
- **View Map:** View the map that shows the route
- **Change Address:** Change one of the saved addresses
- **Change Setting:** Change the general application settings
- **View Planner:** View the planner page that displays the daily schedule of the user.

3.5.3 Object and Class Model

3.5.3.1 Class Diagram

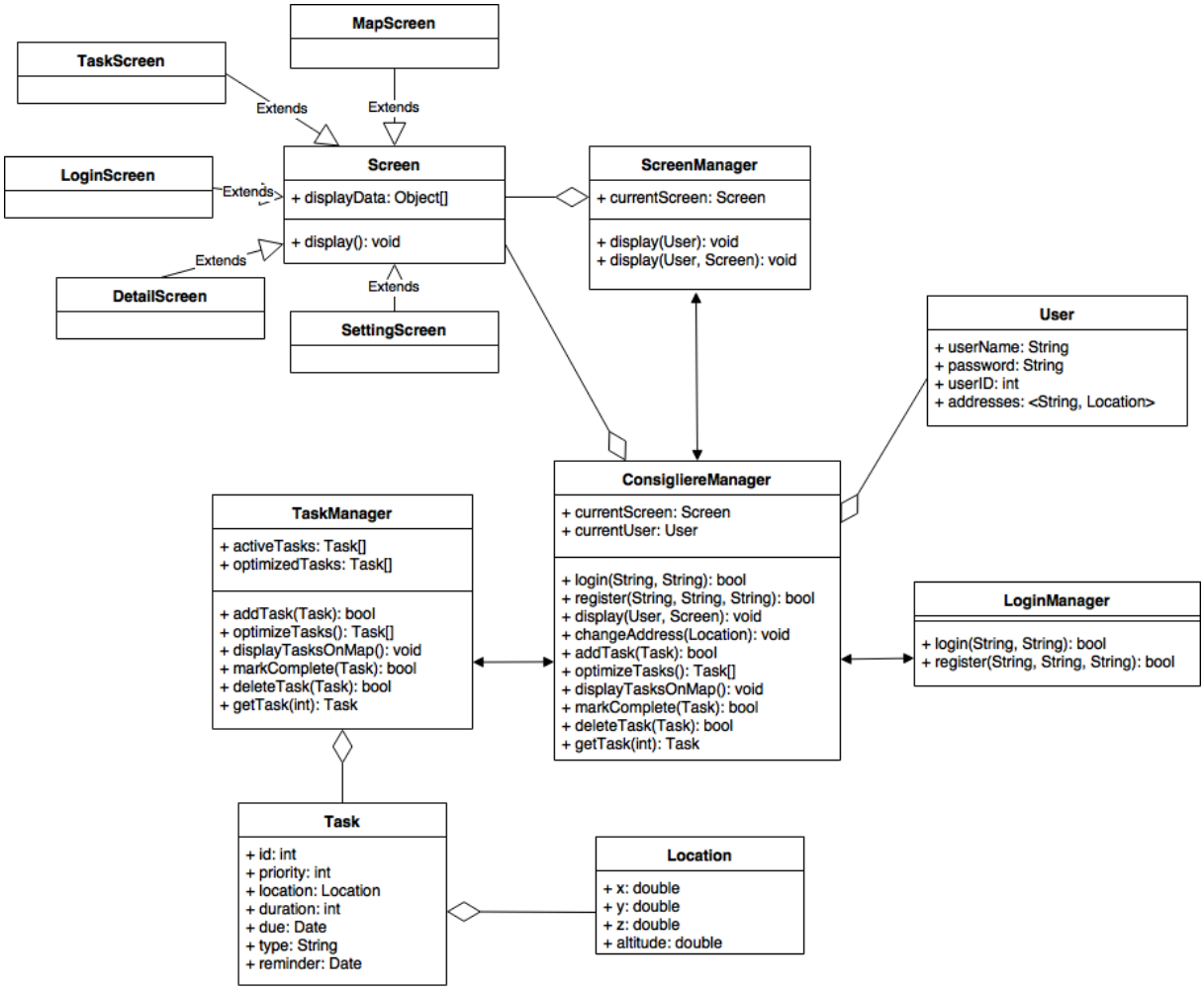

Figure 2: Class Diagram

3.5.3.2 Explanation of the Class Diagram

- **ConsiglieреManager**

ConsiglieреManager acts as a wrapper class that coordinates and governs the whole execution of the application. It instantiates the objects needed and makes the necessary procedural calls to the other manager modules located in the system. In addition, it keeps track of the current screen and the current user's session, and manages what information should be displayed and which screen should be displayed by communicating with the ScreenManager. It also manages the login system by working together with the LoginManager class.

- **TaskManager**

TaskManager contains two lists of Task objects, one of them has the currently active tasks and the other has these tasks in the most optimal order. This class deals with the addition, deletion and editing of Task objects, as well as communicating with the ScreenManager through ConsiglieреManager to display the optimal route for the tasks.

- **LoginManager**

This class deals with the signing in and registration of the users to the application.

- **ScreenManager**

Keeps track of the currently displayed screen and updates the display according to the instructions received from ConsiglieреManager.

- **Screen**

Parent class for all screens that the application will display. MapScreen, TaskScreen, LoginScreen, DetailScreen and SettingScreen all inherit from this class.

- **User**

Keeps track of the data for the user such as the user name and password. Each user has an ID associated with them to further ensure uniqueness. In addition, each user has a

HashMap of addresses with the description of the address associated with the location of said address.

- **Location**

Keeps the data for a location in 3D space.

- **Task**

Manages the data for a specific task. An id and priority is given to a task, as well as a location and a due date. The type of the task is held in a string (activity, errand, recurring event). Also, the user can set a reminder alert of their choosing for each task.

3.5.4 Dynamic Models

3.5.4.1. Sequence Diagrams

3.5.4.1.1 Add a Task after logging in

Scenario: User Atiba opens the app and logs in using his email and password. Then, his information is shown on the screen. Secondly, Atiba adds a task to his schedule and starts his day. His tasks are optimized using the app's algorithm and displayed on the map screen. Atiba then completes his task and marks it as completed. At the end, there are no tasks left for app to optimize and show, his information is shown on the screen again.

Description: Having opened the app and entered the user details, the user is authorized and brought to task screen where s/he enters his task as a text. In case the location and time data cannot be understood by the app, the user as an alternative flow can enter his/her task using the specific location and time bars on the app's GUI. A Task object is created and added into TaskList. When the user hits the button to signal the app to optimize the routes, task is optimized, added to optimizedTaskList and Maps screen is shown to the user during the

continuation of the task's route. When the user thinks that the task is complete he marks the task as complete by using the screen buttons and that same task is added to the completedTaskList.

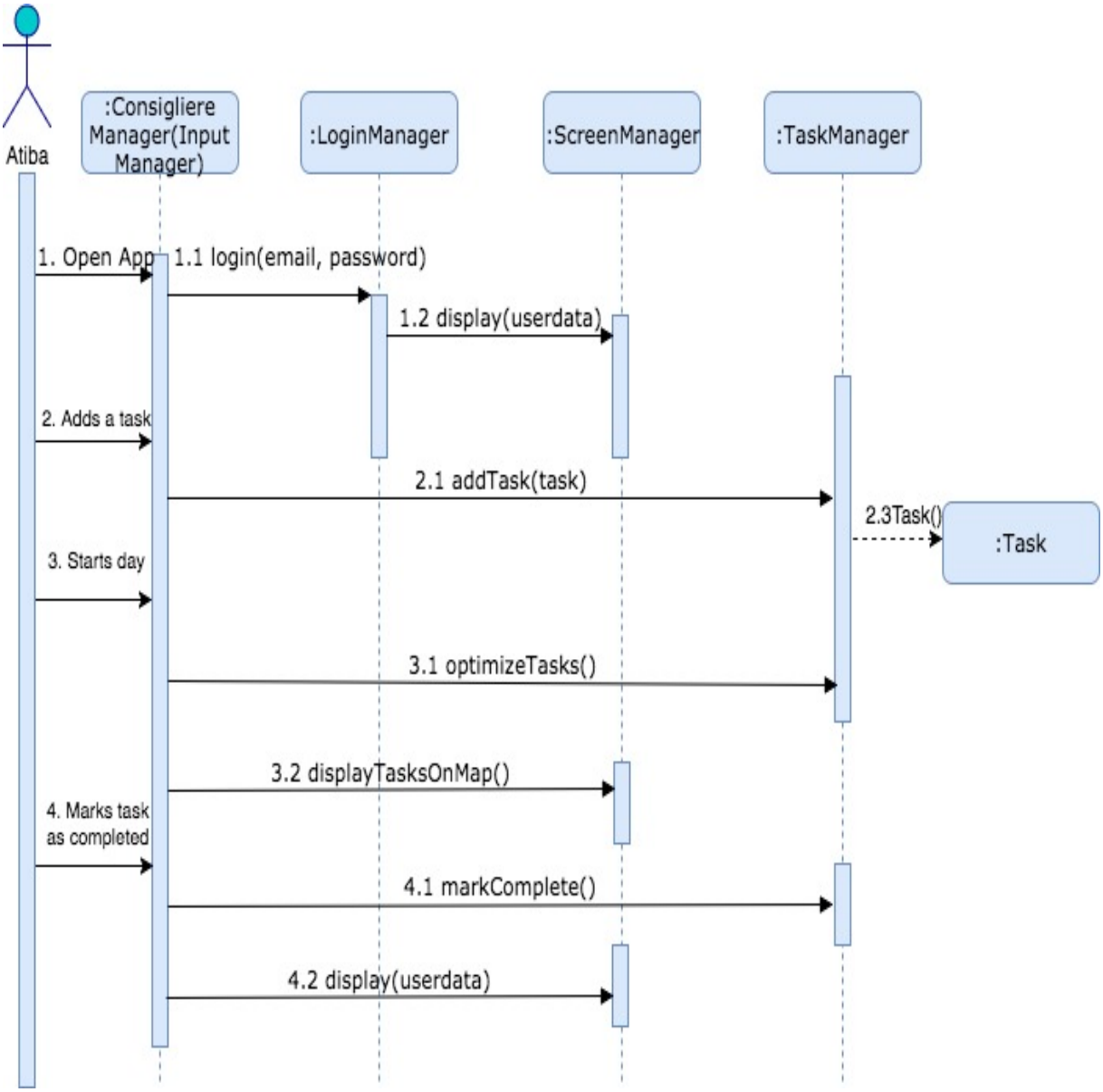

Figure 3. Sequence Diagram for Scenario 1

3.5.4.1.2 Add tasks and delete one

Scenario: User Ezhel opens up the app and is shown his tasks on screen. He decides to add two tasks and sees that one of the task was wrongly added. He removes the wrong task and without adding a new task starts optimization by clicking the button on a screen. He then follows his route on Maps.

Description: Without logging in, the user is directly brought to task screen where s/he adds his/her tasks. Two Task objects are created and added into TaskList. Still on the task screen, he cancels one of the entered task by clicking delete button. When the user hits the button to signal the app to optimize the routes, task is optimized, added to optimizedTaskList and Maps screen is shown to the user during the continuation of the task's route. When the user thinks that the task is complete he marks the task as complete by using the screen buttons and that same task is added to the completedTaskList.

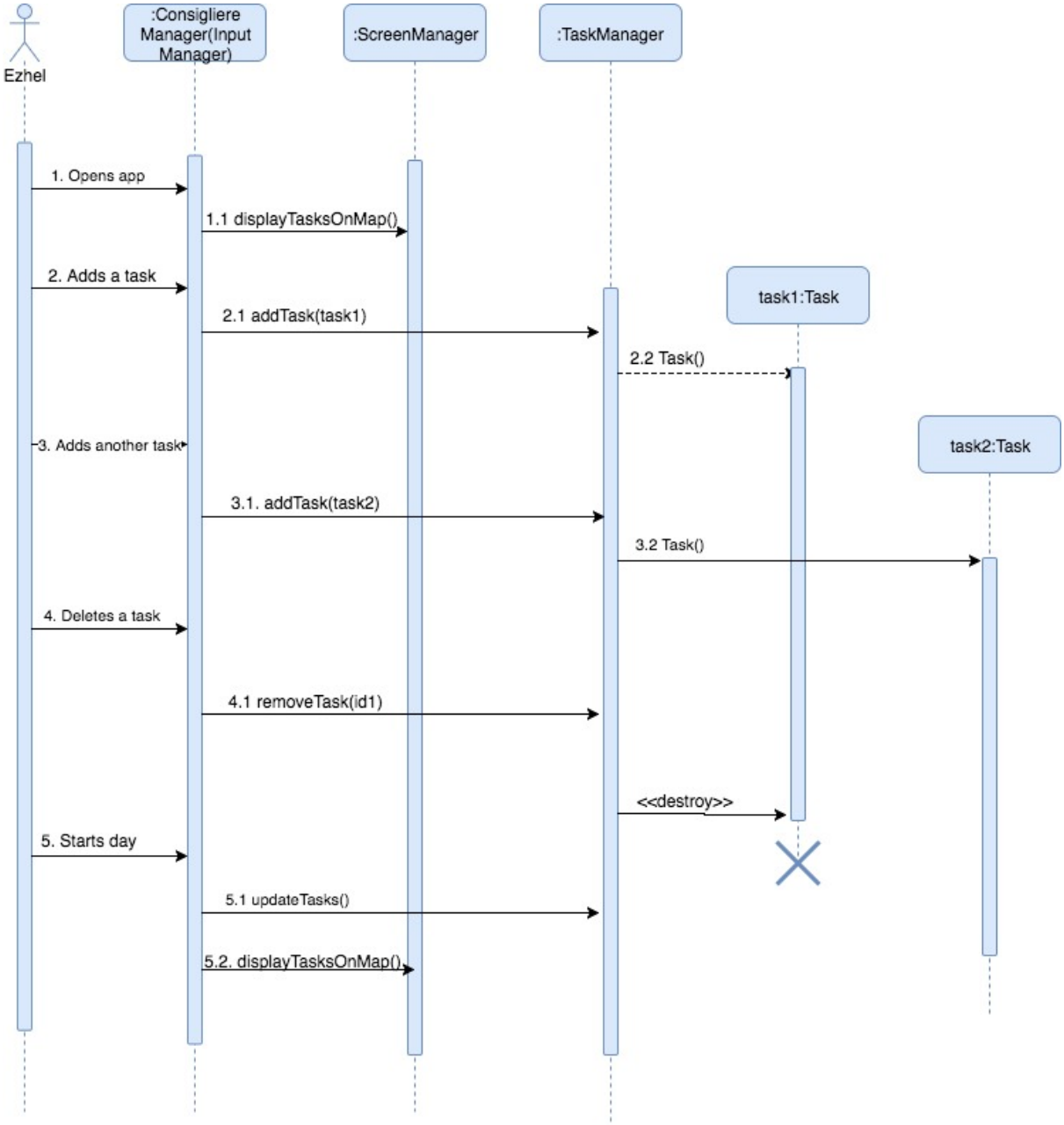

Figure 4. Sequence Diagram for Scenario 2

3.5.4.1.3 Change address

Scenario: User Salah opens the app and logs in using his email and password. Then, his information is shown on the screen. On the main page he clicks on Settings button and switches to Settings page. He enters and saves a permanent address to his user details.

Description: Having opened the app and entered the user details, the user is authorized and brought to task screen. He clicks on Settings button and through interacting with InputManager, s/he writes his address in text area. As soon as he clicks on save button, the address is saved into the database.

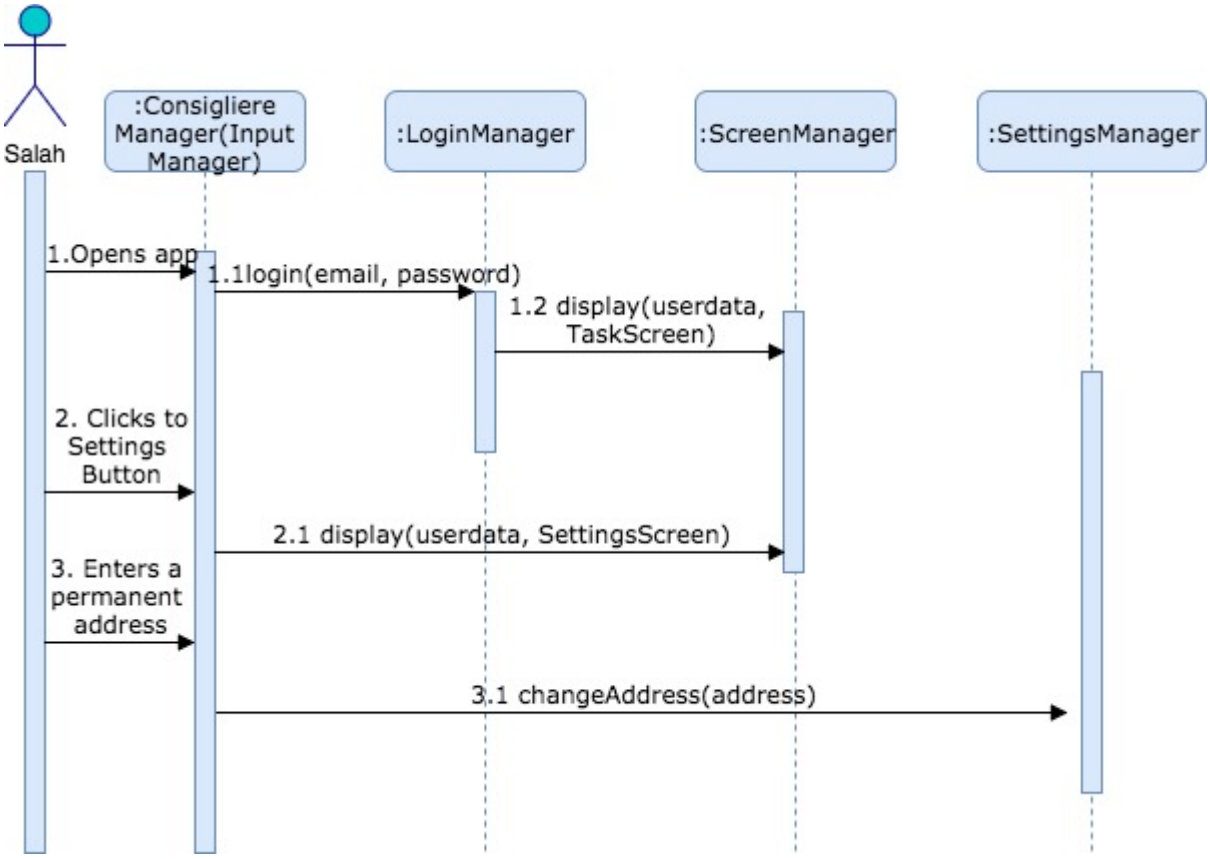

Figure 5. Sequence Diagram for Scenario 3

3.5.4.2 Activity Diagram

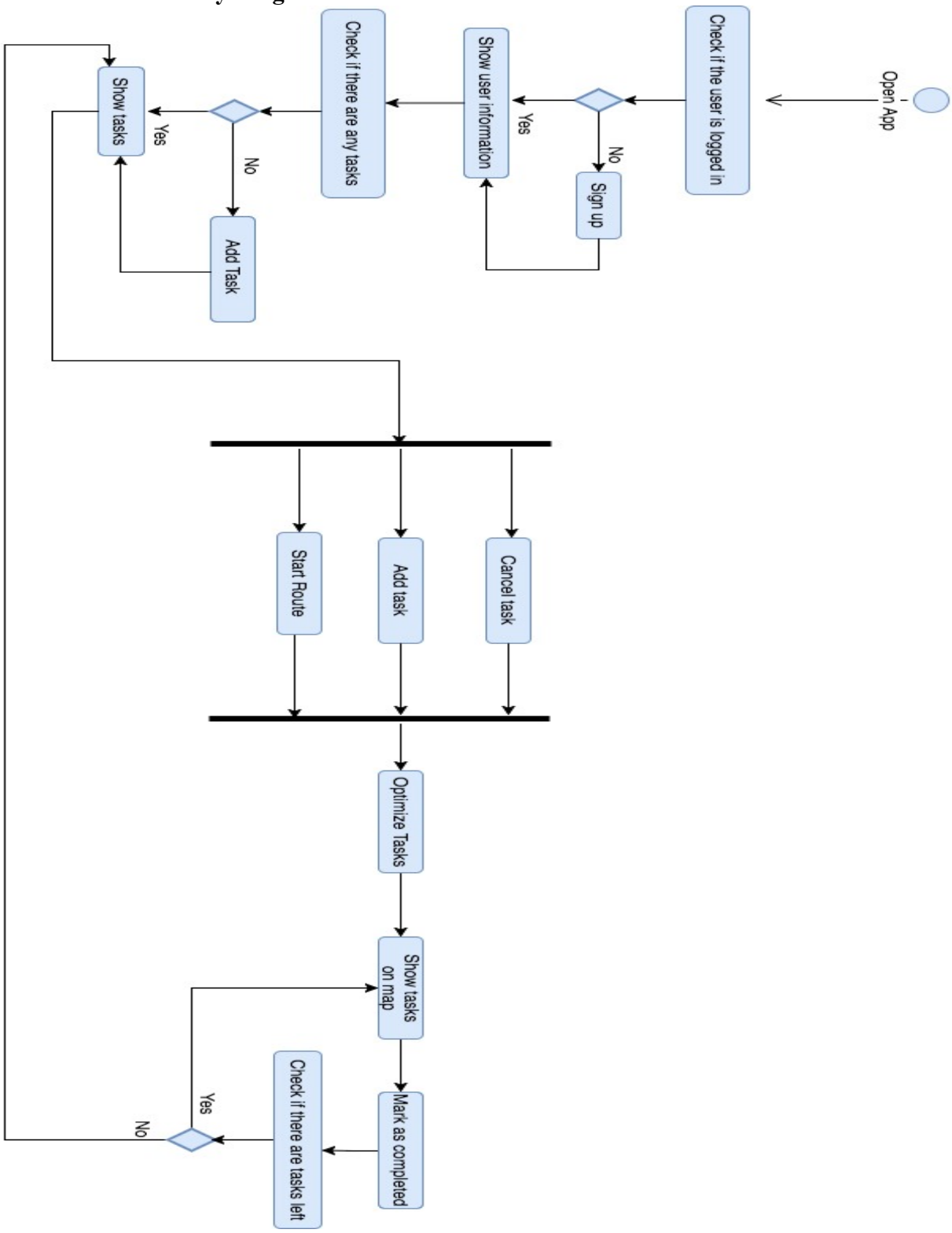

Figure 6: Activity diagram

3.5.5 User Interface

3.5.5.1 Login Screen

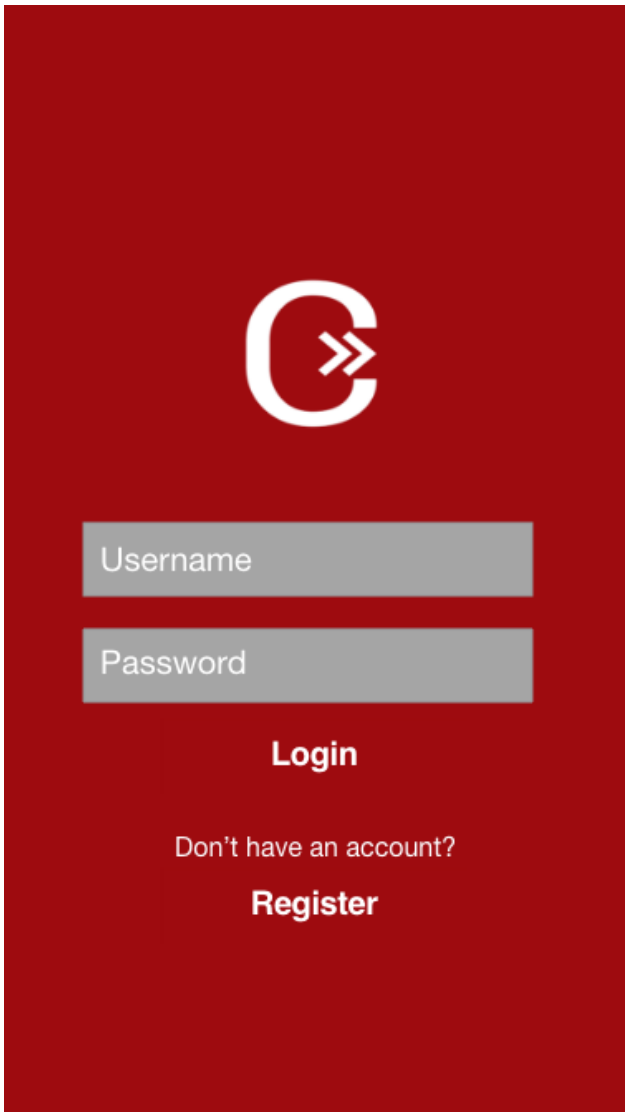

Figure 7: Login Screen

The Login Screen is the first screen that the user will be greeted with when they launch Consigliere. From this screen, they can enter their credentials and press the Login button to move on to the Tasks Screen, or if they do not already have an account and wish to create one, they can first click the Register button to create their Consigliere account.

3.5.5.2 Tasks Screen

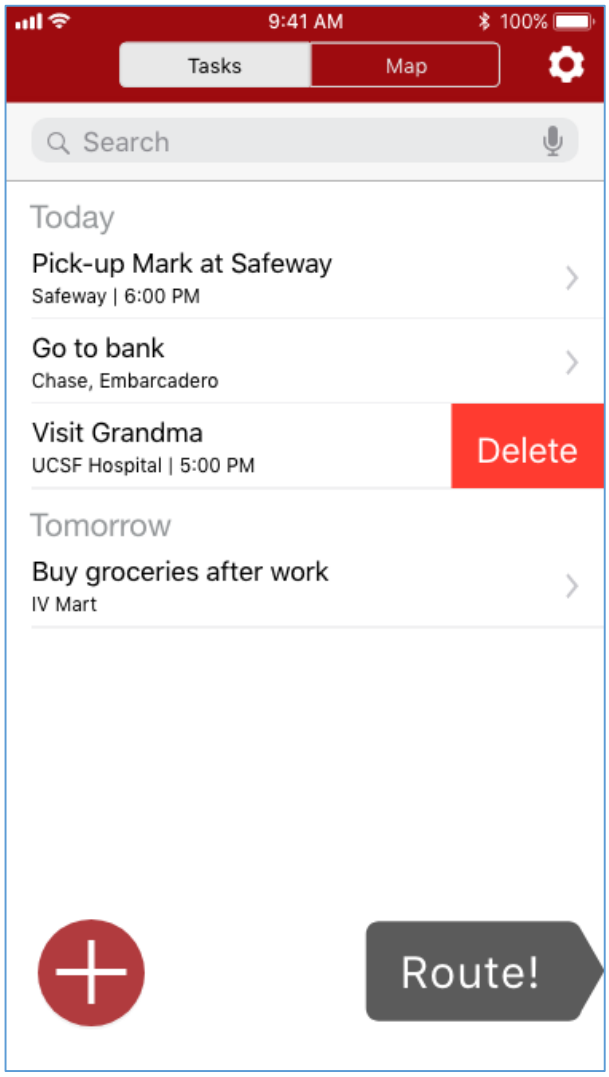

Figure 8: Task Screen

The Tasks Screen is one of the two main components of Consigliere, and it is the primary way for the user to manage their tasks. The screen gives an overview of the user's daily errands as a table of the summary of the tasks of today and the following days. Each entry in the table consists of the description for the task, followed by the location and the time of said task. The user can swipe left on any of the tasks and the option to delete that task will appear before them. If they choose to delete a task, their daily plan will be recalculated according to the changes they have made. Moreover, if they click on any of the tasks, the Task Details screen will be brought up for that specific task.

The user can click on the plus button to the bottom left corner of the screen to add a new task, which will prompt them with the Add Task screen. In addition, the user can search for a specific task using the search bar at the top of the task list.

The settings icon to the top right of the screen will bring a pop-up menu consisting of the global user preferences.

The Route button featured in the bottom right portion of the screen will move the user to the Map Screen and start guiding them through the optimized route for their tasks. In addition, they can also use the navigation tab at the top of the screen to move to the Map Screen.

3.5.5.3 Map Screen

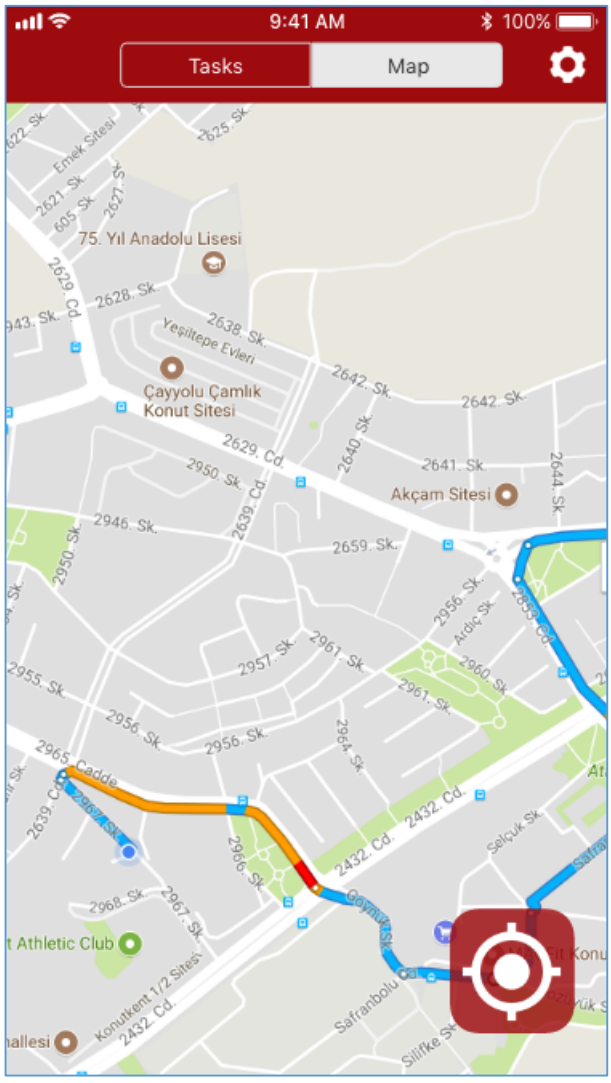

Figure 9: Map Screen

The map screen is another main component of Consigliere, and it shows the optimized route that the user should take to complete their tasks in the most time efficient way. The target button to the bottom right of the screen will center the map around the user's current location, and the map can be navigated by pinching and scrolling through the page, similar to the traditional interface of Google Maps.

The global (application-wide) navigation bar can be found at the top, exhibiting identical behavior to the one described above, in Task Screen.

3.5.5.4 Task Details

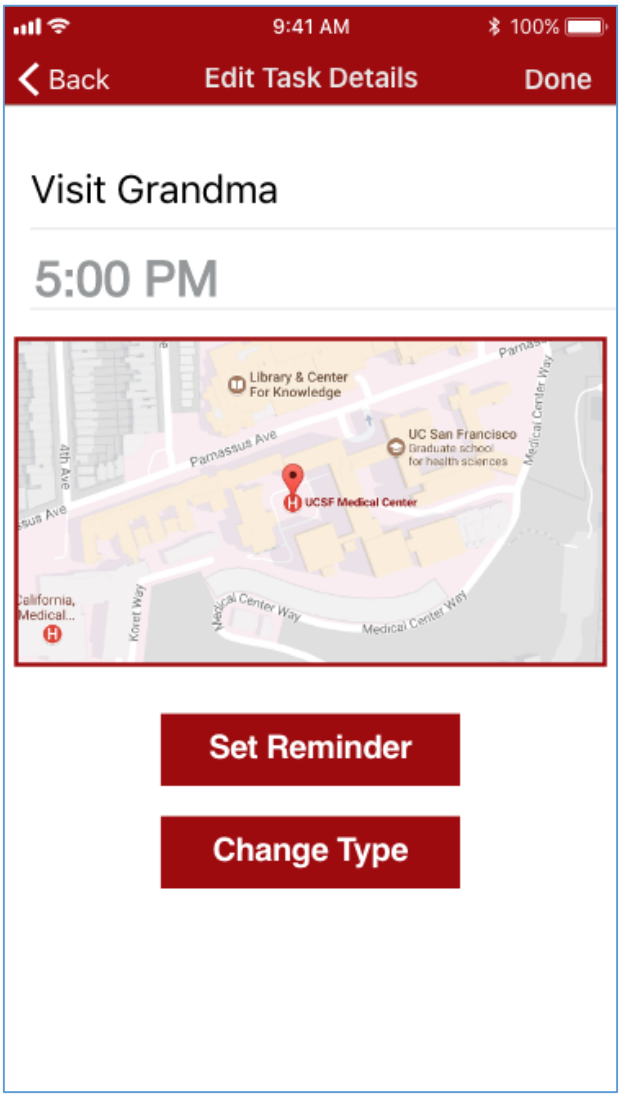

Figure 10: Task Details screen

Upon clicking on a task in the Tasks Screen, the user will be brought to the Task Details screen. Here, they can change the time and the description of their task. They are also presented with a mini-map, indicating the location of the task. Upon clicking on the map, they will be presented with a larger map and they will be able to specify a new place for their task on the map if they wish to do so.

The Set Reminder button allows the user to set a reminder alert for a specific time in order to be notified of the task.

The Change Type button allows the user to change the type of the task and choose among the three different types: Task, Errand and Recurring Event.

The navigation bar at the top of the screen provides a back option, which cancels the changes made by the user and returns them to the Tasks Screen, and the Done option, which saves the changes that the user has made and returns them to the Task Screen.

3.5.5.5 Add Task Screen

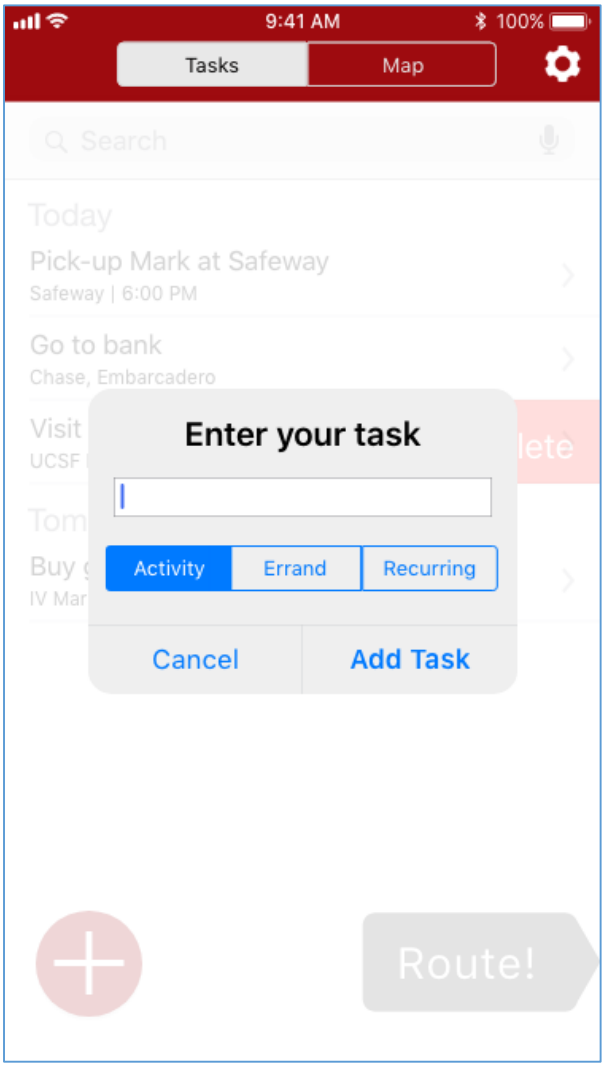

Figure 11: Add task screen

When the user clicks on the big red plus button on the Task Screen, they will be greeted with a pop-up Add Task Screen. Here, the user can type in a description of the task, and then choose from the three different task types: Task, Errand and Recurring Event. After they have done so, they will click the Add Task button and the application will return back to the Tasks Screen, with the newly created task added to the list of tasks.

If the user hits the Cancel button, the action will be aborted and the user will be sent back to the Task Screen with its old list of tasks still in display.

3.5.5.6 Settings Screen

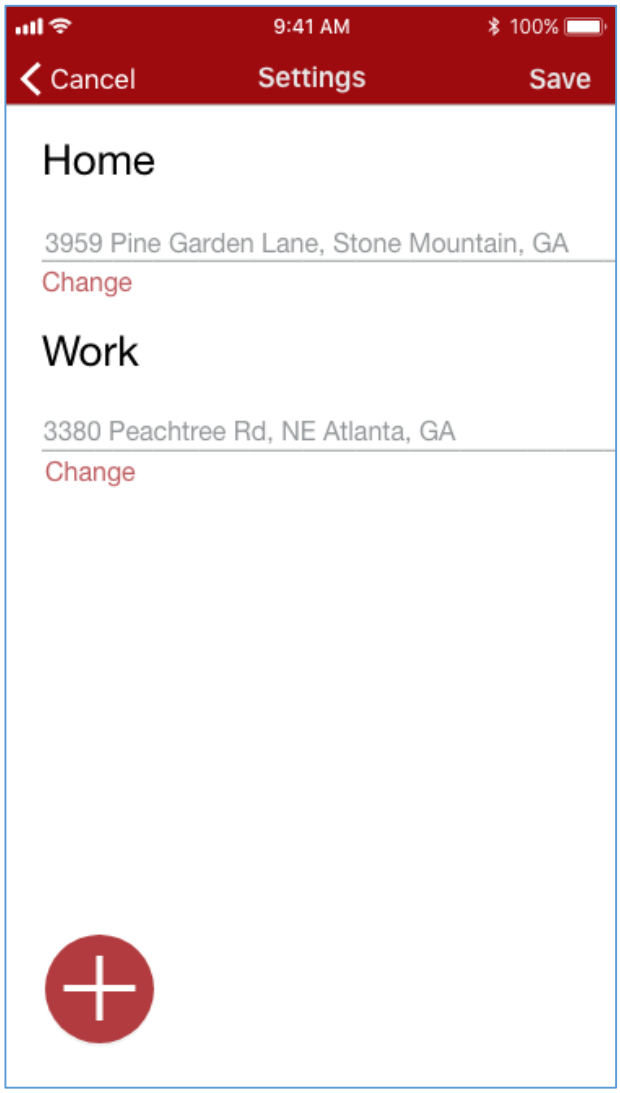

Figure 11: Settings screen

The Settings screen can be reached at any time by clicking on the gear icon displayed in the application-wide navigation bar. In the Settings screen, the user can manage their permanent addresses, such as work, home and school. Their current addresses are displayed as a list and they can change the address by clicking on the Change button featured directly beneath the text description of the addresses. Upon clicking the Change button, a map will pop-up and the user will be able to explicitly point out their address using the map.

In addition, the user can define more permanent addresses if they wish to do by clicking on the big red plus button at the bottom left corner of the Settings screen.

4. References

[1] "Traveling Salesman Problem", *Math.uwaterloo.ca*, 2017. [Online]. Available: http://www.math.uwaterloo.ca/tsp/. [Accessed: 15- Oct- 2017].