Arpit Desai, Brody Williams, Michael Acosta, Vasilis Vloutis, Sakshyam Silwal, Yong Wu

12 April 2018

SOFTWARE REQUIREMENTS SPECIFICATION FOR RaiderNAV

Table of Contents

Table of Contents 2

1. INTRODUCTION 3

1.1 Purpose 3

1.2 Scope 3

1.3 Definitions, acronyms, and abbreviations 3

1.4 References 3

1.5 Overview 3

2. OVERALL DESCRIPTION 4

2.1 Product perspective 4

2.1.1 System interfaces 4

2.1.2 User interfaces 4

2.1.3 Software interfaces 11

2.1.4 Communications interfaces 11

2.2 Product Functions 11

2.3 User characteristics 11

2.4 Constraints 11

2.5 Assumptions and dependencies 12

3. SPECIFIC REQUIREMENTS 13

3.1 Functions 13

3.2 Use Cases 27

3.3 Design constraints 38

3.4 Software system attributes 38

3.4.1 Security 38

3.4.2 Portability 38

1. INTRODUCTION

1.1 Purpose

The purpose of this Software Requirements Specification (SRS) document is to provide a detailed description of the user and system requirements for the *RaiderNAV* software application (henceforth variably referred to as "the app," "the software," "the system," or "RaiderNAV"). This document is intended for all individuals with a stake in the development of the *RaiderNAV* application, including system customers, managers, developers, system engineers, system test engineers, and system maintenance engineers.

1.2 Scope

The *RaiderNAV* software application is designed to provide college students at Texas Tech University with a better way to find their classes, and schedule their day, and to provide both students and visitors with a better way to navigate through campus.

1.3 Definitions, acronyms, and abbreviations

The following terms, abbreviations, and acronyms are used in this document:

- API - Application Programming Interface
- App - Mobile application
- CSV - Comma Separated Value
- GPS - Global Positioning System
- HTTP - Hypertext Transfer Protocol
- HTTPS - HTTP over TLS
- JSON - JavaScript Object Notation
- SRS - Software Requirements Specification
- TLS - Transport Layer Security

1.4 References

The following APIs and technologies are referenced in this document:

- Android: https://www.android.com/
- Google Maps Android API: https://developers.google.com/maps/android/
- Java: https://docs.oracle.com/javase/9/

1.5 Overview

The rest of this document describes the software requirements for *RaiderNAV* and is organized as follows: Section 2 provides the overall description of the software, including product perspective, interface requirements, product functions, user characteristics, constraints, and assumptions. Section 3 provides specific requirements, including functions, constraints, and software system attributes pertaining to security and portability.

2. OVERALL DESCRIPTION

2.1 Product perspective

The software will exist in a single capacity as an Android application. The app will rely on Google Maps via API calls for major functionality. Other functions, such as scheduling, will be accomplished solely by the app.

2.1.1 System interfaces

The following system interfaces will be adhered to:

- The app must function on Android hardware.
- The app must communicate with Google Maps.

2.1.2 User interfaces

Prototype/mockup images of the user interface follow.

Figure 2.1.2.1 Main Screen.

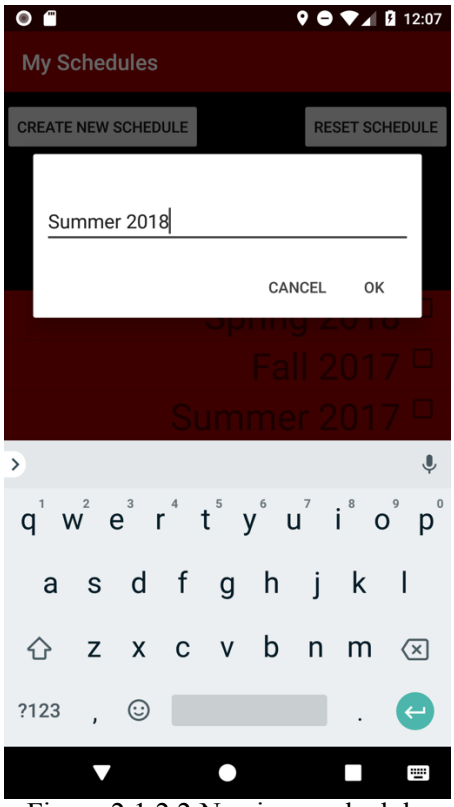


Figure 2.1.2.2 Naming a schedule

Figure 2.1.2.3 Listing schedules.

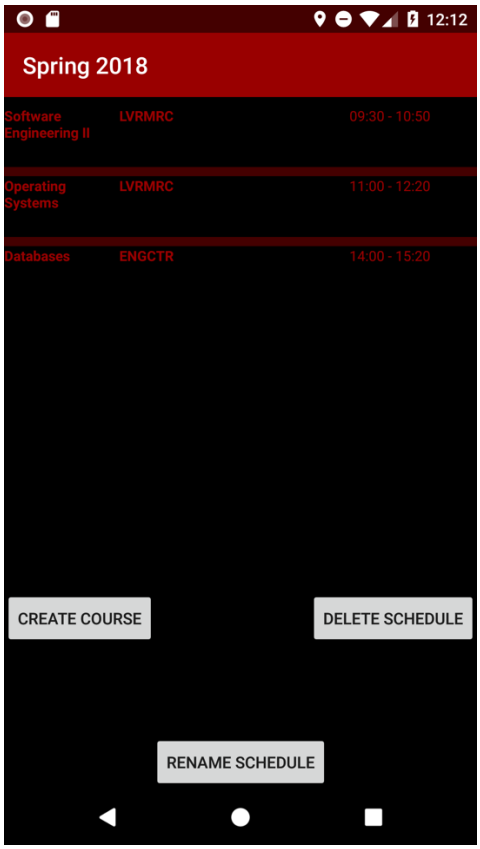


Figure 2.1.2.4 Viewing a specific schedule.

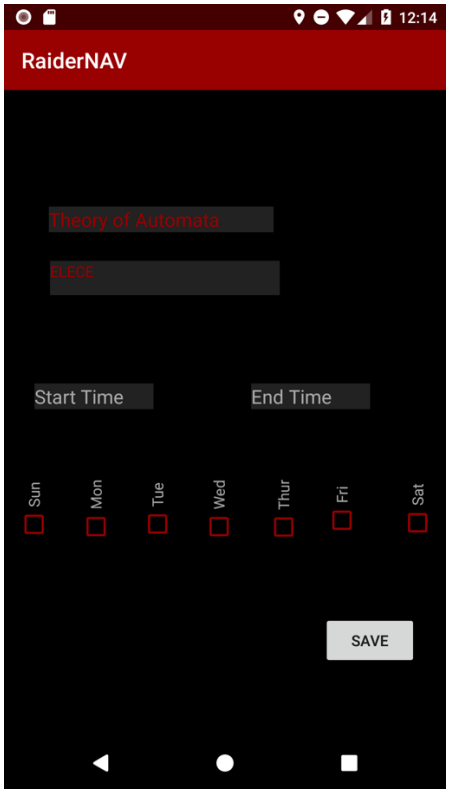


Figure 2.1.2.5 Adding a course.

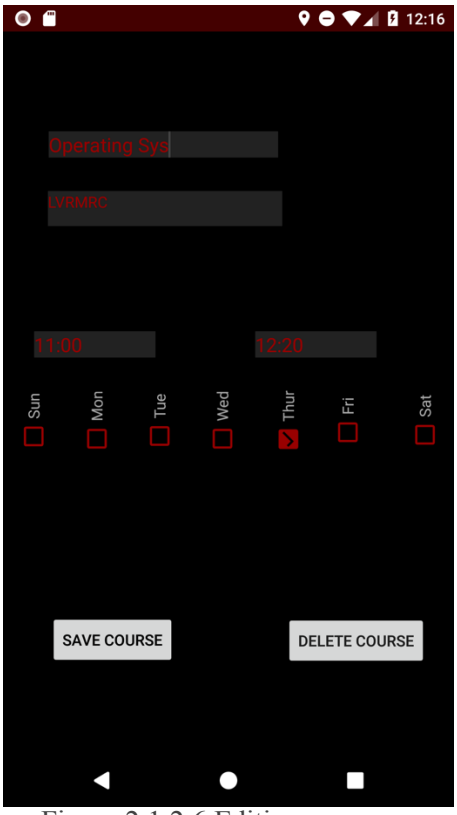


Figure 2.1.2.6 Editing a course.

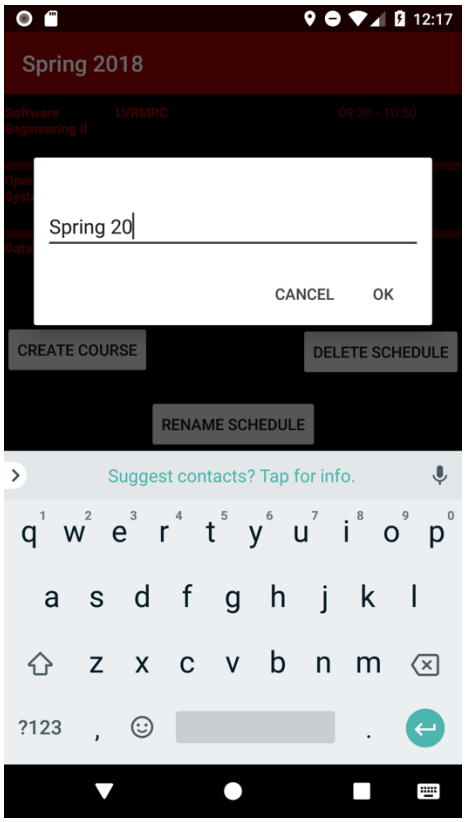


Figure 2.1.2.7 Renaming a schedule.

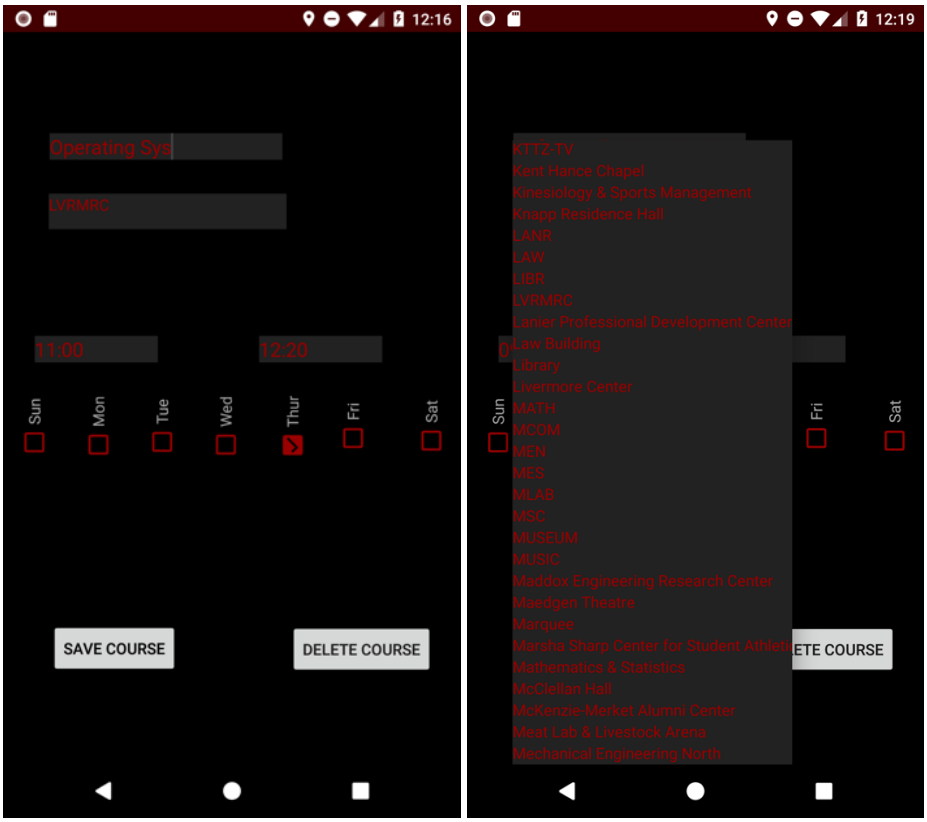


Figure 2.1.2.8 Editing a class (with dropdowns).

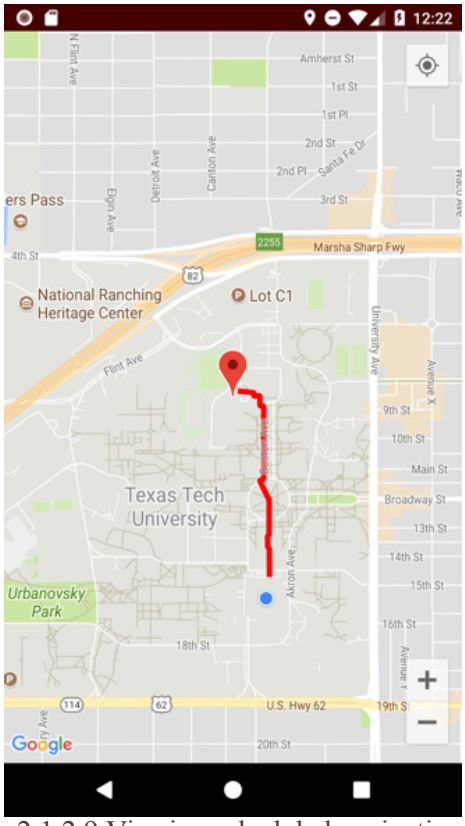
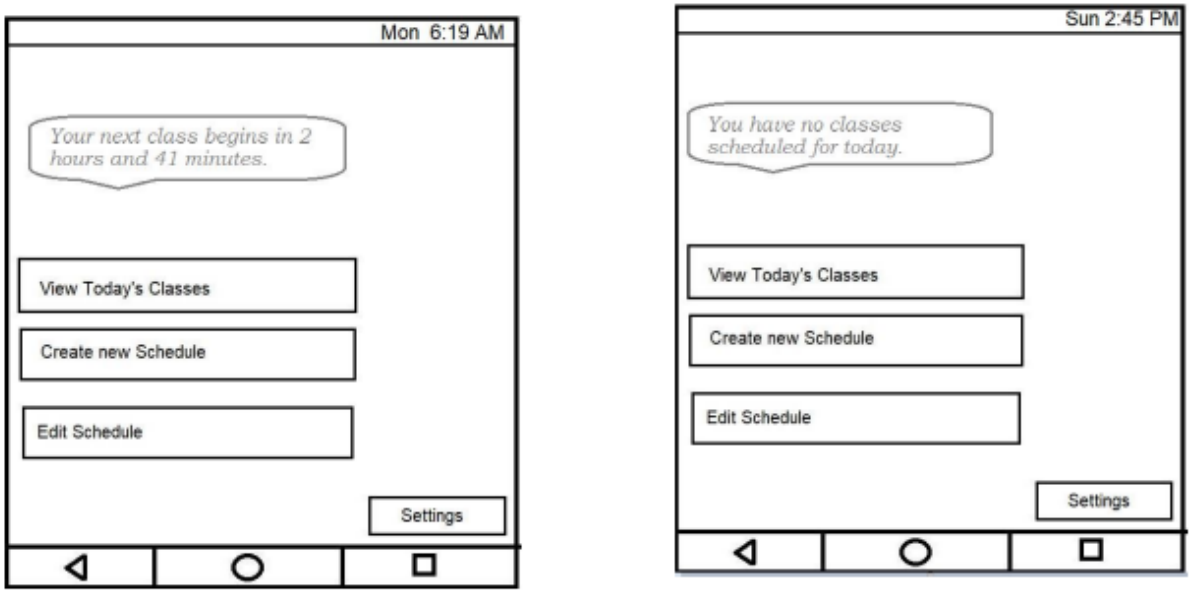


Figure 2.1.2.9 Viewing scheduled navigation map.

Recalculate button during Navigation

Figure 2.1.2.10 Recalculate button.



Home Screen after creating a schedule

Home Screen on a day with no classes

Figure 2.1.2.11 Home screen after schedules activated.

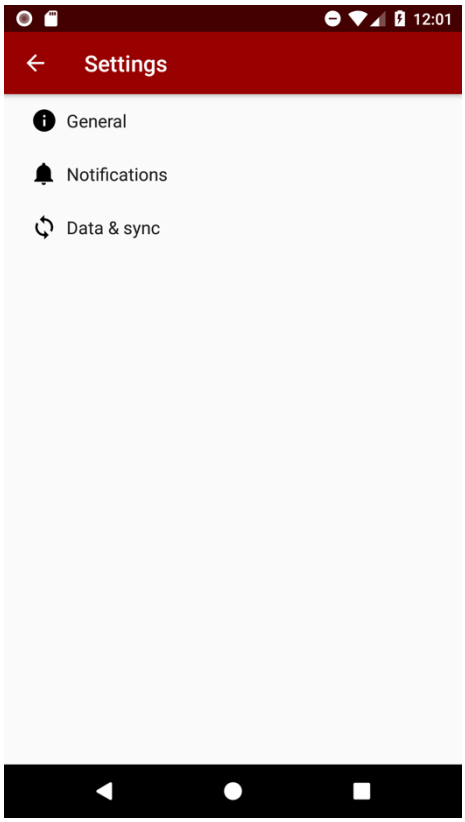


Figure 2.1.2.12 Settings menu.

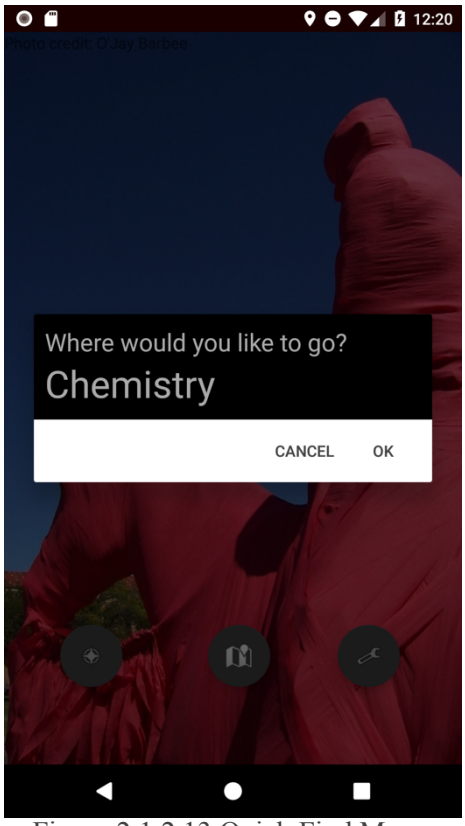


Figure 2.1.2.13 Quick Find Menu

2.1.3 Software interfaces

The app must run on Android version 4.0.3 or greater.

The app must utilize the Google Maps API version 9.2.0 or earlier.

The following data formats will be used:

- CSV
- JSON

2.1.4 Communications interfaces

The following communication interfaces will be used:

- The Android app will communicate with the Google Maps servers over HTTPS.

2.2 Product Functions

The app will implement the following primary functions:

- Schedule builder for students
- Build a unique schedule for each day
- Map that shows walking routes in between classes/destinations
- Direction calculation for one-time unscheduled locations

2.3 User characteristics

This application is intended to be used by college students that are enrolled in classes at Texas Tech University and visitors to the campus. Students and visitors do not need to know the address of the buildings or the full names of the buildings they wish to navigate to.

All users must be able to read and enter text on a smart phone.

2.4 Constraints

Constraints to all aspects of the system must adhere to:

- Hardware Constraints:
  - Minimal local device storage usage
  - Minimal memory usage
  - Limited processing capability
  - Small screen size
- The general safety and security guidelines which are specified in section 3.4.1

2.5 Assumptions and dependencies

The following are the assumptions made for the application:

- Users have the necessary peripherals to interact with the system: a smart phone running Android and an internet connection.

3. SPECIFIC REQUIREMENTS

3.1 Functions

The following stories and functions will be implemented:

Story 1. As a user, I want to be able to add my class schedule (course, building, time) for the day to the app, so that I can receive complete and correct walking directions to each building in order from the app.

1.1 Functional User Requirements:

1.1.1 The software shall allow the user to add new schedules consisting of the course number, name of the building, days of the week, and the time when the class starts.

1.2 Non-Functional User Requirements:

1.2.1 The software shall allow users to expertly add schedules after 15 minutes of tinkering to become familiar with the software.

1.3 Functional System Requirements:

1.3.1 The main screen of the software shall display a button to view schedules.
1.3.2 The software shall internally have the names of valid Texas Tech University buildings in an array.

Story 2. As a user, I want to be able to save my schedule for each of my classes and give the schedule a name so that I can recall it quickly in the future. (Story Points: 2)

2.1 Functional User Requirements:

2.1.1 The software shall allow the user to save any created schedule with a user-given name.
2.1.1 The software shall maintain a list of schedule events (e.g., individual course schedules) within each schedule.

2.2 Non-Functional User Requirements:

2.2.1 The software shall not have idle time more than 2 seconds when saving a schedule.
2.2.2 The software shall allow students to expertly save a schedule after 5 minutes of tinkering to become familiar with the software.

2.3 Functional System Requirements:

2.3.1 The software shall automatically store a schedule on the device when the schedule is created.
2.3.2 The software shall store schedules to the mobile device's storage in JSON format.
2.3.3 The software shall display a save button when a user is editing a course.
2.3.3 The software shall store changes associated with an existing schedule when the user presses a 'save course' button.

Story 3. As a user, I want to be able to save and recall *multiple* schedules, because I may not have the same schedule every week.

3.1 Functional User Requirements:

3.1.1 The software shall allow the user to create multiple schedules.
3.1.2 The software shall allow the user to activate or deactivate multiple schedules.

3.2 Non-Functional User Requirements:

3.2.1 The software shall allow users to expertly save or recall any schedule in a list of schedules after 5 minutes of tinkering to become familiar with the software.

3.3 Functional System Requirements:

3.3.1 The software shall provide a scrollable list of selectable schedules to the user.
3.3.2 The software shall distinguish to the user schedules which are active and schedules which are inactive.
3.3.3 The software shall store schedules to the mobile device's storage in JSON format.

3.3.4 The software shall allow a user to enable or disable a schedule by utilizing a toggle button or checkbox tied to that schedule.

Story 4. As a user, I want to be able to easily delete saved schedules, in case my schedule changes or I have progressed to a new term with different classes, so that I am not encumbered by schedules that are no longer relevant.

4.1 Functional User Requirements:

4.1.1 The system shall allow the user to delete any saved schedule.

4.2 Functional System Requirements:

4.2.1 The system shall provide the user with a "Delete" option when a schedule is being displayed.

4.3 Non-Functional Requirements:

4.3.1 The system shall store the user's list of saved schedules in a single file not exceeding 25MB.
4.3.2 The system shall respond to a request to display or edit the saved schedules file within 2 seconds 99% of the time.

Story 5. As a user, I want the app to automatically determine the geographical location of buildings that my classes are in based only on the name of the building as provided by the schedule in the TTU registration system, so that I can easily have my schedule generated without research or prior knowledge of where the buildings are.

5.1 Functional User Requirements:

5.1.1 The system shall intelligently determine the on-campus location of each of the user's classes based only on the building prefix and room number entered by the user during schedule input.

5.2 Functional System Requirements:

5.2.1 The system shall utilize a data structure/database that maps building prefixes and room numbers with physical addresses on campus.
5.2.2 The system shall provide mapped physical addresses to the mapping engine for use during route generation.
5.2.3 The system shall generate and display an error message for any invalid schedule input that does not map to a physical address.
5.2.4 The system shall generate and display an error message in the event the mapping engine service cannot be reached.

5.3. Non-Functional Requirements:

5.3.1 The system shall correctly process all building number to physical address mappings for a given schedule within 3 seconds 99% of the time.

Story 6. As a user, I want the app to intelligently determine when classes for the day are already over and provide the reduced set of directions only to classes which are not yet over, so that I am not encumbered by excess, unnecessary information.

6.1 Functional User Requirements:

6.1.1 The system shall dynamically generate the user's route based on the selected schedule and time of day.

6.2 Functional System Requirements:

6.2.1 The system shall be able to retrieve the current time of day from the operating device's onboard system time.
6.2.2 The system shall generate a local copy of the selected schedule when the user chooses to generate their route.
6.2.3 The system shall cross-reference the retrieved time of day against the class times on the schedule.
6.2.4 The system shall remove from the local schedule copy any classes which have already ended.
6.2.5 The system shall pass the validated schedule copy to the mapping engine for route generation.

6.3 Non-Functional Requirements:

6.3.1 The system shall process a time validated schedule to the mapping within 3 seconds of the user requesting a route, 95% of the time.
6.3.2 The system shall not require any further input from the user from the time the user requests a route until the mapping engine returns its results.

Story 7. As a user, I also want the app to give me the option to toggle showing the routes to ALL classes for the day regardless of whether they are already over, so that I can show my schedule to others in person.

7.1 Functional Requirements:

7.1.1 The system shall allow the user shall be provided a button to show the user all his class routes for the day in one screen.
7.1.2 The system shall be able retrieve the stored information of his classes from that day.
7.1.3 The system shall send the information to the GPS and get the entire routes that he will complete that day.

7.2 Non-Functional Requirements:

7.2.1 The system must be capable of changing the screen from showing one route to showing All routes in the day in less than 4 seconds, 90% of the times.

7.3 Functional System Requirements:

7.3.1 The system should provide an ALL view button before the user starts their route and while the user is going through their route.
7.3.2 The system shall send every destination to GPS, in the order that the user will be walking through it on the map.
7.3.3 The system shall put all the routes on one map and show them to the user after getting the locations of all destinations.

Story 8. As a user, I want the app to show me the shortest walking route between classes, so that I can arrive as quickly as possible in the event that I am running late.

8.1 Functional Requirements:

8.1.1 The system shall allow users to choose arbitrary points on the map as starting point and destination.
8.1.2 The system should calculate the shortest route from two locations depending on distance.

8.2 Non-Functional Requirements:

8.2.1 The system must be capable of connecting with Google Maps Directions, input the two destinations, and respond to the users with the shortest distance to class in less than 5 seconds, 95% of the time.

8.3 Functional System Requirements:

8.3.1 The system will get location of point when the user long click the screen.
8.3.2 The system will print the map destination to the screen to show the shortest path for the user to get to their destination.

Story 9. As a user, I want the app to regularly update and display my current location on the navigation map, so that I can more easily follow the directions provided by the app.

9.1 Functional Requirements:

9.1.1 The system shall be able to call the GPS function and get the users location
9.1.2 The system shall be able to use a timer and constantly call the GPS function to get the location as accurate as possible.

9.2 Non-functional Requirements:

9.2.1 The system must be capable of connecting to the GPS and getting the users exact position each time after recalculating in less than 2 seconds, 98% of the time.

9.3 Functional System Requirements:

9.3.1 The system should calculate the current location of the user using the GPS function provided by Google GPS.
9.3.2 The system shall also have a timer that is set to a small count, such as 2 seconds, to recalculate the location of the user every time.
9.3.3 The system shall show the user their new location on the map every time the location of the user is recalculated.

Story 10. As a user, I want the app to show me an estimated amount of time to travel from my current location to the next destination in my schedule, so that I can react with more urgency if I am in danger of being late.

10.1 Functional User Requirements:

10.1.1 After "Start" button is selected by the user, the screen should show a time. This time is calculated based on the average speed and the route between current location and next destination.

10.2 Non-Functional User Requirements:

10.2.1 System should respond in 10 second

10.3 Functional System Requirements:

10.3.1 System reads current location
10.3.2 System has average speed data
10.3.3 System has the "next destination" information
10.3.4 System calculate time needed time=route length/average speed
10.3.5 System show the result

Story 11. As a user, I want the app to tell me how much time is available until my next class begins, so that I can more easily fit unscheduled activities into my day without calculating myself the amount of time available until class begins.

11.1 Functional User Requirements:

11.1.1 Screen should show the time remaining for the next class.

11.2 Non-Functional User Requirements:

11.2.1 System should respond in one second

11.3 Functional System Requirements:

11.3.1 System reads the time of next class
11.3.2 System calculate result: "next class time"-"current time"
11.3.3 System shows the result

Story 12. As a user, I want the app to notify me with special messages when I am in danger of being late to class, based on the time the class begins and the travel time necessary to reach the building, so that I can afford to be distracted and do not have to constantly check the app.

12.1 Functional User Requirements:

12.1.1 The screen should show a warning with the message “Going to be Late!”.

12.2 Non-Functional User Requirements:

12.2.1 The system shall automatically show the warning.

12.3 Functional System Requirements:

12.3.1 The system shall read the result from 11.3.2
12.3.2 The system shall have a pre-set time
12.3.3 The system shall pop up a warning if “pre-set time” is less than the time from 11.3.2

Story 13. As a user, I want the reminder feature to honor silent mode and notification settings on the mobile device, so that I am not inconvenienced by additional settings.

13.1 Functional user requirement:

13.1.1 The system shall honor device silent mode setting.

13.2 Non-functional user requirement:

13.2.1 The system shall not mention this feature to the user.
13.2.2 The system shall provide no functionality to alter this behavior.

13.3 Functional system requirements:

13.3.1 The system shall register notifications with the device and allow the operating system to properly handle the functionality.
13.3.2 If necessary, all reminder functions of the system will check device settings and will not generate a reminder if the device is set to not notify.

Story 14. As a user, I want to be able to completely disable all reminder features, regardless of phone notification settings on the mobile device, so that I am not bothered by unnecessary reminders when I am confident about my schedule.

14.1 Functional user requirement:

14.1.1 The system shall allow disabling notification settings.

14.2 Non-functional user requirement:

14.2.1 The setting will be prominent within a settings menu as one of few settings. Toggling the setting should be simple.
14.2.2 The setting should take effect within 2 seconds of the user making their setting selection.

14.3 Functional system requirements:

14.3.1 When the notification reminder setting is set to disabled, no reminder functions will be activated. Each reminder function will check for the setting to determine whether a reminder may occur.
14.3.2 In the settings view, toggling (both on and off) occurs by pressing the disable reminders setting. The setting will visibly indicate the current setting with a toggle box, check box, or textual explanation.
14.3.3 When a setting selection is made, the settings view should still be displayed but the visual indicator of the particular setting should be updated.

Story 15. As a user, I want the app to recalculate the shortest path to the next destination in my schedule with the push of a ‘recalculate’ button, in the event that I veer from the original route and require new directions.

15.1 Functional user requirement:

15.1.1 The system shall recalculate the path to the next destination when a ‘recalculate’ button is pressed.

15.2 Non-functional user requirement:

15.2.1 The system shall allow users in 99% of cases to expertly use the feature on their first attempt without training.
15.2.2 The system shall perform the recalculation within 10 seconds of the recalculate button being pressed when the device has GPS satellite visibility and a non-congested data connection.

15.3 Functional system requirements:

15.3.1 When the recalculate button is pressed, a function call will use the Google Maps API to refresh the map.
15.3.2 The button shall be fairly prominent (take up at least 5% of available screen space) without crowding out the map from being usefully visible (the button shall take up no more than 20% of available screen space)

Story 16. As a user, I want to be able to arbitrarily add some *other* locations to the app and to my schedules, so that I can plan in regular lunch destinations or walks for exercise between classes.

16.1 Functional User Requirements:

16.1.1 The system shall allow the user to add destinations to their schedule that do not correspond to classes.

16.2 Functional System Requirements:

16.2.1 The system shall allow the user to add destinations to their saved schedules by using either the building number or physical address.
16.2.2 The system shall allow the user to label these destinations in their saved schedules.

16.3 Non-Functional Requirements:

16.3.1 The system shall be implemented such that destinations not related to classes are easily added, modified, or removed by the user.

Story 17. As a user, I want to be able to add arbitrary locations which will only be considered on that day only, and not saved to the schedule for use on other days, so that I can benefit from the features of the app even with short-term plans without inconveniencing myself by having to remove destinations from the schedule again later.

17.1 Functional User Requirements:

17.1.1 The system shall allow the user to add locations to an existing schedule before route generation that will not be permanently saved to that schedule.

17.2 Functional System Requirements:

17.2.1 The system shall allow the user to add locations to their route request based on either a building number or physical address.
17.2.2 The system shall not write temporary destinations to a saved schedule.
17.2.3 The system shall pass a local schedule copy, including both saved and temporary destinations, to the mapping engine after a user requests a route.

17.3 Non-Functional Requirements:

17.3.1 The system shall send a route request to the mapping engine within 3 seconds of the user selecting “Request Route”, 95% of the time.

Story 18. As a user, I want to be able to mark a class as temporarily canceled for the day, so that I do not receive unnecessary notifications, reminders, or directions for classes I will not be attending that day.

18.1 Functional User Requirements

18.1.1 The system shall allow user to temporarily disable events on their active schedules for the day without affecting the stored version of the schedule.

18.2 Functional System Requirements

18.2.1 The system shall give the user an option to temporarily disable an event/course when it is being viewed by the user.
18.2.2 A temporary deactivation will not affect the general schedule
18.2.3 A temporarily changed schedule has higher priority than the general schedule for routing and notification consideration.

Story 19. As a user, I want to be able to edit existing/saved schedules to change the buildings and times of each class, so that I can correct for schedule changes and prior input mistakes without creating entirely new schedules.

19.1 Functional User Requirements

19.1.1 The system shall allow existing/saved schedules to be edited

19.2 Non-Functional User Requirements

19.2.1 The system shall respond to an attempt to edit a schedule within one second
19.2.2 The system shall respond to an attempt to save an edited schedule within one second

19.3 Functional System Requirements

19.3.1 The system shall display an edit button on schedules
19.3.2 The system shall show the schedule selected by the user
19.3.3 The system shall allow the user to edit the schedule
19.3.4 The system shall allow the user to save the schedule

Story 20. As a user, I want to be able to export schedules to other common digital calendar formats, so that I can integrate some benefits from the app with other apps that I use every day.

20.1 Functional User Requirements

20.1.1 The system shall be able to generate Excel and CSV files based on the schedule.

20.2 Non-Functional User Requirements

20.2.1 The system shall keep the schedule confidential.

20.3 Functional System Requirements

20.3.1 The system shall be able to convert schedules into Excel and CSV files.
20.3.2 The system shall be able to send Excel and CSV files via email

Story 21. As a user, I want to be presented with a message the first time I load the app that informs me of the privacy implications of using the app, so that I am aware of how my information is being handled.

21.1 Functional User Requirements

21.1.1 The system shall show the privacy policy only when the application is opened for the first time.

21.2 Non-Functional User Requirements

21.2.1 The system shall load the home screen within two seconds when the okay button is pressed.

21.3 Functional System Requirements

21.3.1 The system shall show the privacy policy dialog box only once when the application is loaded on a device for the first time.
21.3.2 The shall allow a button that takes you to the home screen when pressed, on the bottom of the dialog box.

Story 24. As a user, I want a quick-find feature so that I can find the nearest path to a single destination without inputting a schedule.

24.1 Functional User Requirements

24.1.1 The system shall, from the main screen, allow one-click access to a destination prompt.
24.1.2 The system shall, upon receiving a destination from the user, generate a map in which the destination is marked.
24.1.3 If the user's device has GPS or network location capabilities, the system shall show upon the map—with colored lines—the walking directions from the user's present location to the user's selected destination.

24.2 Non-Functional User Requirements

24.2.1 The system shall respond to the initial button press on the main screen within 2 seconds 99% of the time.
24.2.2 The system shall generate a map upon the user's selection of a destination within 15 seconds under normal conditions and if the user has an internet connection 95% of the time.

24.3 Functional System Requirements

24.3.1 The system shall provide a button on the main screen which, when pressed, prompts for a destination.
24.3.2 The system shall provide the destinations available to the user as a dropdown selection list.

24.3.3 The system shall provide a button to confirm the destination selection when prompting for a destination.
24.3.4 The system shall provide a button to cancel the destination selection and return to the main screen when prompting for a destination.
24.3.5 The system shall allow the user to, with a finger swipe on the map screen, explore the map for adjacent locations.
24.3.6 The system shall provide a button on the map screen which, when clicked, refocuses the user's view on the user's location.
24.3.7 The system shall provide a button to zoom the map screen out.
24.3.8 The system shall provide a button to zoom the map screen in.
24.3.9 The system shall print the name of the destination on the screen over the destination marker when the destination marker is pressed by the user.
24.3.10 The system shall default on centering on the user's location when maps are generated.

Story 29. As a user, I want to be able to name each schedule myself, so that I am able to more easily distinguish between them.

29.1 Functional User Requirements

29.1.1 The system shall allow the user to independently name each of his/her saved schedules
29.1.2 The system shall re-label the saved schedule shown in the schedule selection menu using the user specified name.

29.2 Non-Functional User Requirements

29.2.1 The system shall be designed in such a way that users are able to name their schedules as intended, 99% of the time, after 5 minutes of training.
29.2.2 The system shall respond to name/rename schedule requests within 5 seconds, 95% of the time.

29.3 Functional System Requirements

29.2.1 The system shall provide a text box for user input when naming a schedule.

Story 30. As a user, I want to be asked to verify before a schedule or course is deleted, so that I do not accidentally delete a schedule.

30.1 Functional User Requirements

30.1.1 The system shall provide the user with a notification that allows the user to confirm the deletion of the schedule.

30.2 Non-Functional User Requirements

30.2.1 The system shall respond to the delete button and pop up a notification to confirm the deletion, in less than 2 seconds, 99% of the time.

30.2.2 The system shall be able to respond to the cancel button within the notification and take the user back to the Schedules screen in under 2 seconds, 98% of the time.
30.2.3 The system shall be able to respond to the delete button within the notification, delete the entire schedule, and take the user back to the schedules screen within 8 seconds, 90% of the time.

30.3 Functional System Requirements

30.3.1 The system shall provide the user with a notification prior to the deletion of the schedule.
30.3.2 The system shall allow the user to confirm the deletion of the schedule by clicking the Delete button in the notification window.
30.3.3 The system shall allow the user to cancel the deletion of the schedule by clicking the Cancel button.
30.3.4 The system shall take the user back to the schedule screen once the process is completed.

3.2 Use Cases

Figure 3.2.1 Unified use case diagram.

Story 1
Use Case: Add Schedule
Summary: App User creates a new class schedule
Actors: User
Dependency:
Precondition: The application should be displaying the My Schedules screen
Description:
1. User presses Create New Schedule button on the My Schedules screen.
2. The application prompts the user to enter the schedule name.
3. User inputs schedule name and presses OK.
4. The system adds the schedule to a visible list on the My Schedules screen.
Alternatives:
3a. The user presses Cancel
3a.1 The system displays the My Schedules screen.
Postcondition: The user has created a schedule.

Story 2
Use Case: Save Class schedule
Summary: App User saves a created class schedule
Actors: User
Dependency: View Saved Schedules
Precondition: The user has entered the course's details (name, building, days of week, and the starting and ending times) in the course edit screen.
Description:
1. User presses the Save button.
Alternatives:
1a. If the schedule saved for a specific day overlaps with regards to time with another schedule on the same day, the application displays an error message.
Postcondition: The class schedule is saved.

Story 3
Use Case: Edit Class schedule
Summary: App User edits a previously saved class schedule.
Actors: User
Dependency:
Precondition: The application should be in the My Schedules screen and the list of schedules must have at least one element.
Description:
1. The user presses desired schedule from list on My Schedules screen.
2. The application lists the courses saved to that schedule along with Create Course, Delete Schedule, and Rename Schedule buttons.
3. The user presses a desired course from the displayed list.
4. The application displays modifiable fields for the course name, building, start and end times, and days of the week, along with Save and Delete buttons.
5. The user fills in fields as desired and presses Save button.
Postcondition: The user has edited a previously saved class schedule.

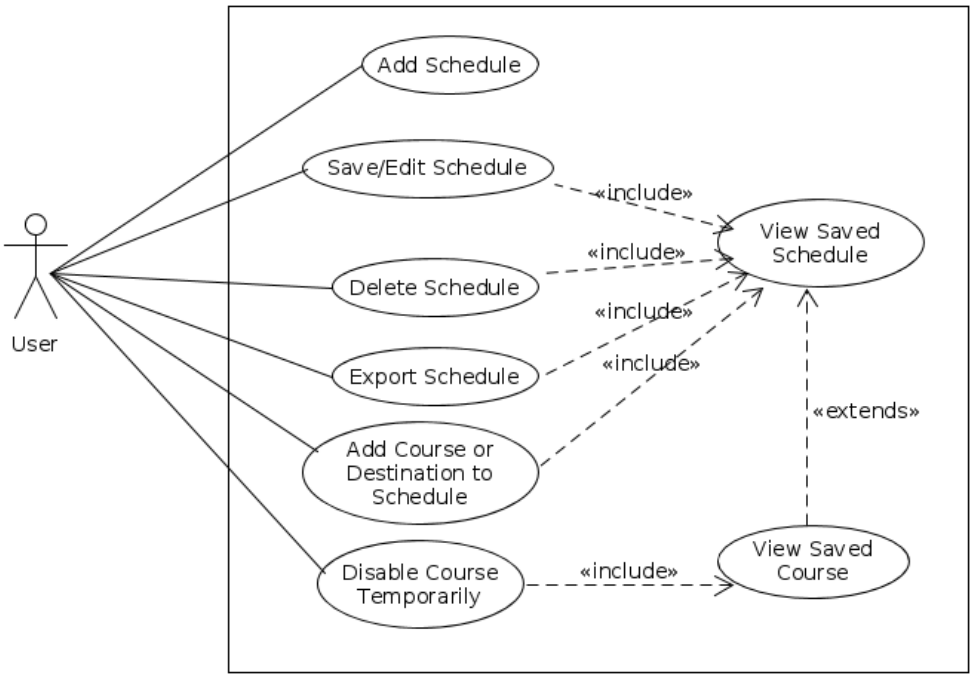


Figure 3.2.2 Use cases 1-4, 16, 18-20, and 29.

Story 4

Use Case Name: Delete Class Schedule
Summary: This user case will allow user to delete a previously created class schedule.
Actor: User
Dependency: View Saved Schedule
Precondition: The application should be in the My Schedules screen and the list of schedules must have at least one element.
Description:
1. The user presses desired schedule from list on My Schedules screen.
2. The application lists the courses saved to that schedule along with Create, Delete, and Rename buttons.
3. The user presses the Delete Schedule button.
4. The application deletes the schedule from the schedule list and returns the user to the My Schedules screen.
Postcondition: User has deleted the class schedule.

Story 5
Use Case Name: Navigate to Campus Destinations
Summary: This use case allows users to select a building name/acronym from a drop-down menu which the system will then determine a geographic location for and add to the daily route.
Actor: User and Mapping Engine
Dependency: The name (or acronym) of the building must be known by the user.
Precondition: The user has selected the "Add course" or "Create Schedule" option.
Description:
1. The user selects the desired building name/acronym from the drop-down menu.
2. The user enters other pertinent details for the course/schedule and selects "Save".
3. The system passes the option selected to the CoordinateMap class.
4. The CoordinateMap class returns the location mapped to the desired selection.
Postcondition: Coordinates representing the location of the user's selection have been received by the system.

Story 6
Use Case Name: Cull Routing Destinations
Summary: After a class ends, the map and schedule will automatically remove that class from their lists for the day.
Actor: User and Mapping Engine
Dependency:
Precondition: Schedules with times for events are active.
Description:
1. A daily schedule will keep the time of when each class starts and ends
2. A function will be implemented to show that the class time has finished
3. Once the class time is finished, the map will change since the finished classes will disappear from the routes on the map.
Postcondition: The updated map should only show the classes that are still going to happen throughout the rest of the day.

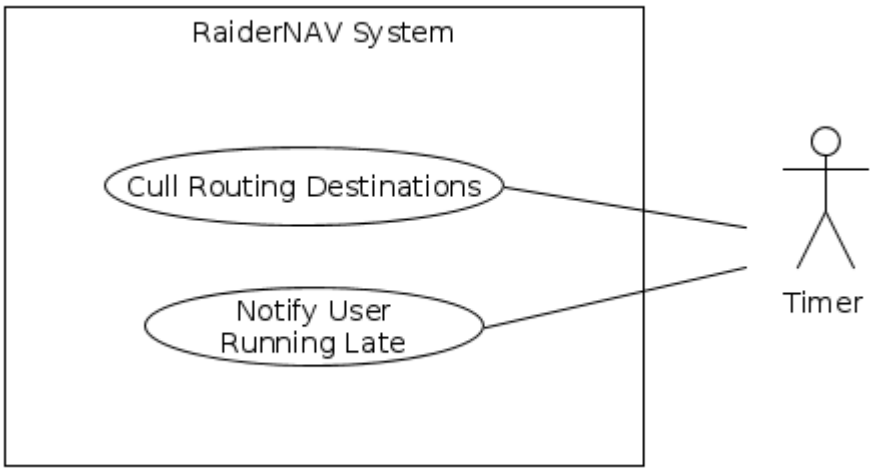


Figure 3.2.3 Use cases 6 and 12.

Story 7
Use Case: Display All Routes
Actor: User
Dependency: None
Precondition: None
Description:
1. User presses a button to get all the routes
2. The GPS gets all the routes for that day
3. All the routes are displayed on one map
Postcondition: All available routes are displayed to the user.

Story 8
Use Case: Calculate Shortest Route
Actor: User
Dependency: None
Precondition: User is in map screen.
Description:
1. The user long presses to select a starting point on the map.
2. The system displays a marker over the user's starting point on the map.
3. The user long presses to select a destination on the map.
4. The system displays a marker over the user's destination on the map, along with a shortest-path between the two points.
Postcondition: The shortest route between starting point and destination has been calculated.

Story 9
Use Case: Recalculate Path
Actor: User
Dependency: None
Precondition: User has already calculated a route
Description:
1. User starts their route towards the destination
2. Current location is retrieved from the GPS
3. Every 3-5 seconds, the time sends an input to the GPS to recalculate location
4. Current location is constantly displayed to the user on the map
Postcondition: Route from current position to destination has been recalculated.

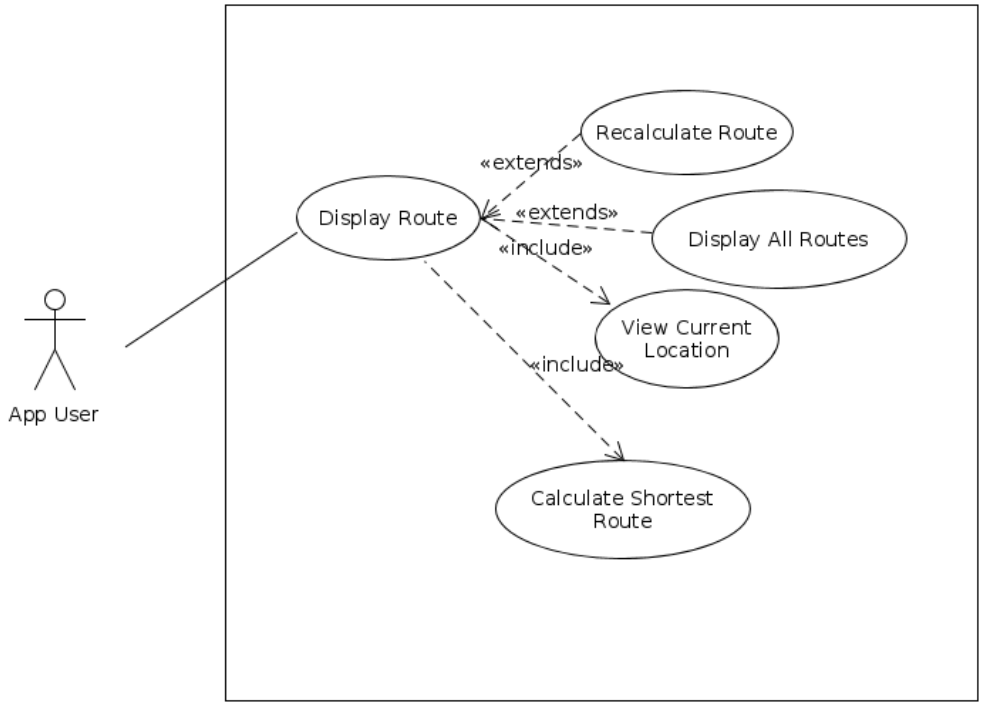


Figure 3.2.4 Use cases 5, 7-9, 15, 24.

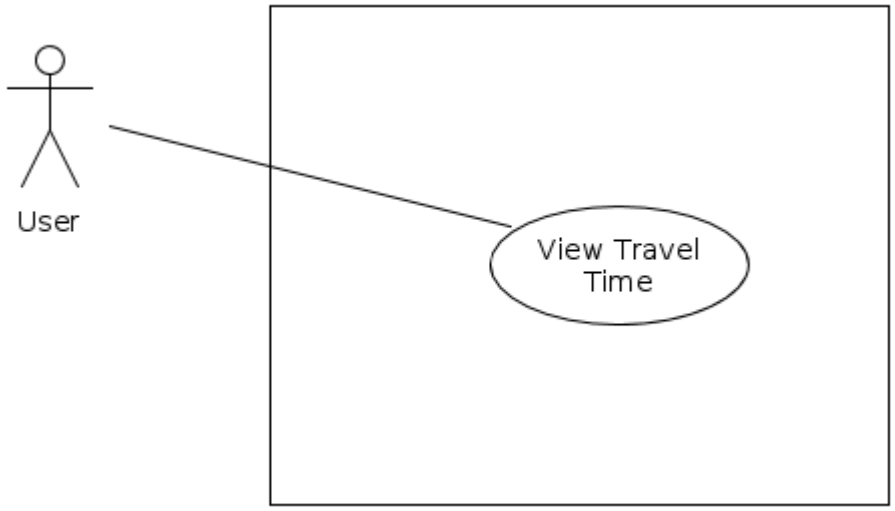


Figure 3.2.5 Use cases 10-11.

Story 10
Use Case: View Time needed
Summary: The app shows the time to travel from user's current location to the next destination.
Actor: User
Precondition: The route from current location to the next location, course's beginning time are ready to use; User's average velocity is pre-set
Description:
1. The app read route's length from Google API function
2. The app does calculation to get time needed C2: S/V (S is route length, V is average velocity)
3. The app prints/shows the result on the screen.
Postcondition: The app has displayed the estimated travel time to the user.

Story 11
Use Case: Edit schedule
Summary: The app should allow users to edit schedule.
Actor: User
Precondition: The schedule exists.
Description:
1. User click the "Edit" button on the existing schedule.
2. User input destination location, class time into the schedule.
3. User click the "save" button to finish editing
Postcondition: The user has saved changes to the schedule.

Story 12
Use Case: View time remaining
Summary: The app should show the time remaining to next class.
Actor: User
Precondition: course's beginning time is in the schedule
Description:
1. The app does the calculation C1: T1-T2 (T1 is the next course's beginning time, T2 is the current time)
2. The app shows the result: T1-T2
Postcondition: The app has displayed the remaining time to class to the user.

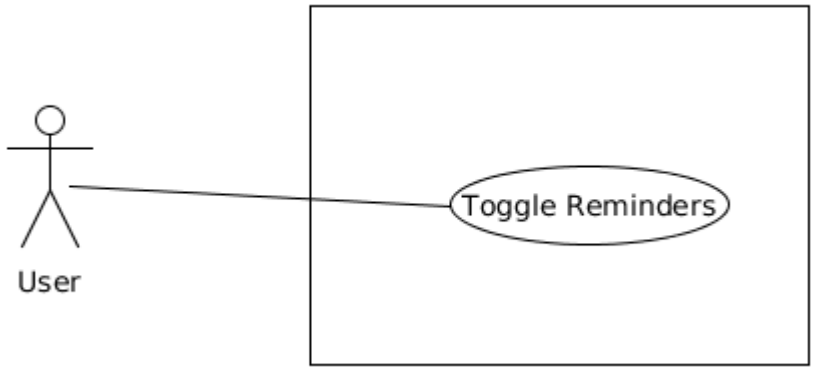


Figure 3.2.5 Use cases 13-14

Story 13
Use Case Name: Toggle Reminders
Summary: App user enables or disables reminder/notification feature
Actor: User
Dependency: None
Precondition: None
Description:
1. User presses Settings button
2. Settings menu is displayed to user.
3. User presses Reminder Toggle button.
4. App disables reminders.
Alternatives:
4a. If reminders were already disabled, reminders are enabled.
Postcondition: App user has toggled reminder setting.

Story 15
Use Case Name: Recalculate Path
Summary: App user updates navigation path
Actor: User
Dependency:
Precondition: Map is displaying
Description:
1. User presses Recalculate button
2. Map and path are updated and displayed to user
Postcondition: App user has current map displaying

Story 16
Use Case Name: Add Daily Location to your schedule.
Summary: This will add a specific location to the schedule to repeat daily or to multiple days considering the users preference.
Actor: User and Mapping Engine
Dependency: none
Precondition:
Description:
1. User selects add a location to my schedule option.
2. Application prompts the selection , is it temporary ?
3. User selects no button.
4. The application prompts the user to enter the location address, time when the user wants to start moving towards the location.
5. User enters the location address and time.
6. User selects the save location to my schedule.
7. The set days screen loads, User selects the days he wants to add the location navigation to.
Alternatives:
7a. There is a time conflict with a class in the day the location navigation is set to. The system shows an error message.
Postcondition: User has added a location navigation with navigation time to the schedule.

Story 17
Use Case Name: Add Temporary Location for one day
Summary: Add a temporary location that will only affect today's schedule and not future schedules.
Actor: User and Mapping Engine
Dependency: none
Precondition:
Description:
1. User selects add a location to my schedule option.
2. Application prompts the selection, is it temporary ?
3. User selects Yes button.
4. The application prompts the user to enter the location address, time when the user wants to start moving towards the location.
5. User enters the location address and time.
6. User selects the save location to my schedule.
7. The set days screen loads, User selects the days he wants to add the location navigation to.
Alternatives:
7a. There is a time conflict with a class in the day the location navigation is set to. The system shows an error message.
Postcondition: User has added a location navigation with navigation time to the schedule.

Story 18
Use Case: Edit temporary schedule
Summary: The app should allow user to change the schedule temporarily.
Actor: User
Precondition: The general schedule has been created.
Description:
1. The schedule could be changed with two options: temporarily, generally
2. The temporarily change will not affect general schedule
3. The temporary change has higher priority than general schedule.
Postcondition: A temporary schedule has been edited.

Story 19
Use Case: Notify late
Summary: The app should pop out a notification, saying "you are going to be late" when the user is in danger of being late to class.
Actor: User
Precondition: User is on the route to the next class
Description:
1. If C2>C1, notification pop out, else notification will be dismissed
2. If user arrives at the class location, notification procedure will be dismissed
Postcondition: The user has been notified that they are running late.

Story 20
Use Case: Export schedule
Summary: The app should allow users to export schedule.
Actor: User
Precondition: The schedule exists.
Description:
1. User click the “export” bottom on the existing schedule.
2. The app will send schedule via email in two formats: Excel and CSV.
Postcondition: The selected schedule is exported.

Story 21
Use Case: Display privacy policy
Summary: The app displays the privacy policy when the application is loaded the first time.
Actor: User
Precondition: User has opened the app for the first time.
Description:
1. The privacy implications of using the application is displayed to the user.
2. The user is prompted to select OK if he agrees to the terms.
Postcondition: The application home screen is loaded.

Story 24
Use Case: Navigate to Unscheduled Location
Summary: The app shall allow the user to generate a map to a location which is not within any schedule and which will not be saved to any schedule.
Actor: User
Precondition: User is currently on the main screen.
Description:
1. The user presses the unscheduled location button.
2. The system displays the list of locations to choose from.
3. The user selects a location from the list and confirms the decision.
4. The system displays a map to the destination.
Alternatives:
3a. The user selects the “Cancel” button.
3a.1. The system returns the user to the main screen.
Postcondition: The user has generated a map to an unscheduled destination.

Story 29
Use Case: Rename Schedule
Summary: The app shall allow the user to rename a saved schedule.
Actor: User
Precondition: User previously saved a schedule in the app. User is currently on “My Schedules” screen.
Description:
1. The user selects a saved schedule.
2. The system displays the schedule details.
3. The user selects the “Rename Schedule” button.
4. The system displays a prompt showing the current schedule name.

5. The user enters a replacement name for the selected schedule using the Android keyboard and selects “Ok”.
6. The system returns the user to the “My Schedules” screen.
Alternatives:
5a. The user selects the “Cancel” button.
Postcondition: The user has successfully renamed a saved schedule.

Story 30
Use Case: Delete Notification
Summary: Before the user deletes a schedule, the user must be given a notification to ensure that the user wants to delete the schedule, because all data will be lost.
Actor: User
Precondition: The user is at the schedule screen.
Description:
1. The user presses the Delete button.
2. A notification is given to the user to confirm the deletion, along with an option to cancel and not delete the schedule and a Delete button to delete the schedule.
3. If the user presses Delete, then the schedule will permanently be deleted and the user is then taken back to the schedules page.
Alternatives:
3a. If the user presses cancel, then they will be taken back to the main schedule page.
Postcondition: The user is returned to the schedule screen if they pressed Cancel or to the My Schedules screen if the user confirmed schedule deletion.

3.3 Design constraints

The software is constrained to design requirements specified by the platforms it is being deployed onto in accordance with the following list:

Constraints to all aspects of the system must adhere to:

- The general safety and security guidelines which are specified in section 3.3.1
- Hardware Constraints:
    - Minimal local device storage usage
        - The app will take up no more than 100 megabytes of storage.
    - Minimal memory usage
        - The app will consume no more than 100 megabytes of memory.
    - Limited processing capability
        - The app will be responsive under mobile processors.
    - Screen size
        - The app's display will fit within mobile phone screens.
    - Limited battery
        - The app must be conservative with processing demands so as not to needlessly drain battery power. Specifically, the app should not utilize GPS functionality except when that functionality is necessary.
- Language Constraints:
    - The app will need to be programmed in Java.

3.4 Software system attributes

3.4.1 Security

The system will conform to the following security requirements:

- All communications between the app and Google servers will be conducted over HTTPS.
- Personally identifiable information will not be stored on the local storage of the mobile device running the app.
- A privacy policy will be presented to users on first run to inform them that their GPS location will be used by the app.
- Scheduling data must be stored only on the mobile device.
- Non-exported scheduling data must be stored in a secure manner such that other applications are unable to retrieve the data.

3.4.2 Portability

The following portability considerations must be adhered to:

- The app must work on Android version >= 4.0.3
- The app must be programmed in Java
- The app must support use of all Unicode characters.