

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

12 February 2018

SOFTWARE REQUIREMENTS SPECIFICATION FOR RaiderNAV

Table of Contents

1. INTRODUCTION3

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 DESCRIPTION4

2.1 PRODUCT PERSPECTIVE 4

2.1.1 *System interfaces*..... 4

2.1.2 *User interfaces*..... 4

2.1.3 *Software interfaces*..... 7

2.1.4 *Communications interfaces*..... 7

2.2 PRODUCT FUNCTIONS8

2.3 USER CHARACTERISTICS8

2.4 CONSTRAINTS..... 8

2.5 ASSUMPTIONS AND DEPENDENCIES 9

3. SPECIFIC REQUIREMENTS.....9

3.1 FUNCTIONS 9

3.2 USE CASES21

3.3 DESIGN CONSTRAINTS.....30

3.4 SOFTWARE SYSTEM ATTRIBUTES30

3.4.1 *Security*.....30

3.4.2 *Portability*.....31

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’). 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, navigate through campus, and schedule their day.

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
- 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.

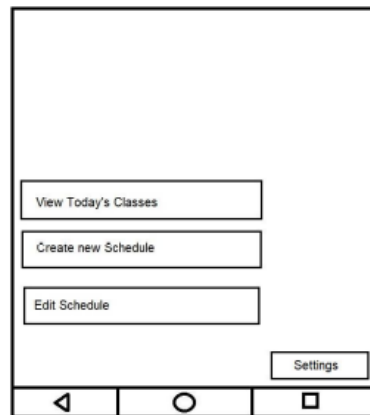
2.1.1 System interfaces

The following system interfaces will be adhered to:

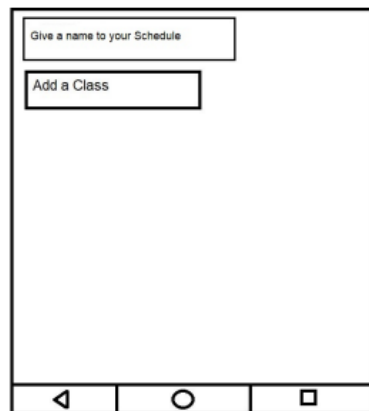
- The app must function on Android hardware.
- The app must communicate with Google Maps.

2.1.2 User interfaces

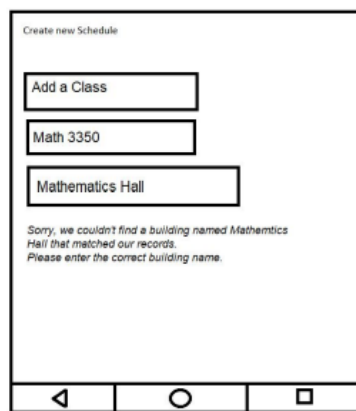
Prototype/mockup images of the user interface follow.



Home screen



Naming your Schedule



Building Name Error

Adding a Class (continued)

John's Monday Schedule

Enter Schedule name here

Add a Class

Enter Course Number

Enter Building Name

Start Time

Finish Time

Add Another Class to this schedule.

I am finished creating this schedule.

John's Monday Schedule

Course Name	Location	Time
Math 3350	Mathematics	8am - 9:55am
English 1301	Holden Hall	10am - 10:50am
ECE 2350	Livemore Centre	12:30pm - 1:50pm

Set this schedule for:

Mon	Tue	Wed	Thu	Fri	Sat	Sun
●	○	●	○	●	○	○

Create and Save Schedule

Completing creating a schedule and Saving it

Edit Schedule

John's Monday Schedule

Schedule for Tuesday/Thursday

Saturday Coffee and Softball

Select Schedule you would like to edit

Edit Schedule

MWF Classes

	Location	Time
English	English	8:00-9:00
Calculus	Mcom	9:00-10:00
Chem		1:00-2:00

AGR

Biology

Biology LH

Chem

More

Edit Schedule

MWF Classes

	Location	Time
English	English	8:00-9:00
Calculus	Mcom	9:00-10:00
Chemistry	Biol	1:00-2:00

Start Time

1:00

End Time

2:00

2:00

3:00

4:00

5:00

SAVE

Edit Schedule

MWF Classes

		Location	Time
✕	English ▼	English ▼	8:00-9:00 ▼
✕	Calculus ▼	Mcom ▼	9:00-10:00 ▼
✕	Chemistry ▼	Biol ▼	1:00-2:00 ▼

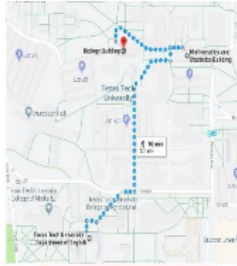
Edit Schedule

MWF Classes

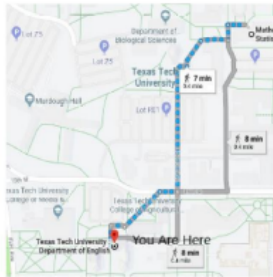
		Location	Time
✕	English ▼	English ▼	8:00-9:00 ▼
✕	Calculus ▼	Mcom ▼	9:00-10:00 ▼
✕	Chemistry ▼	Biol ▼	1:00-2:00 ▼

SAVE

Today's Schedule



Start Route



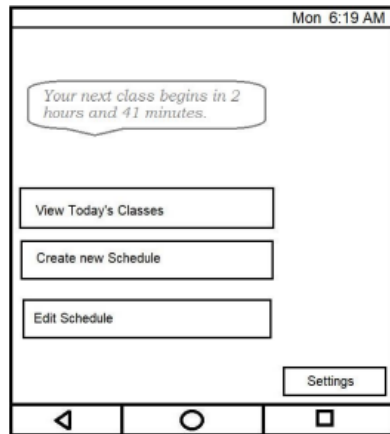
Walk Straight for 0.2 miles



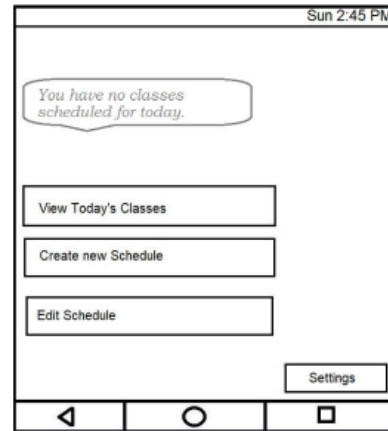
Walk Straight for 0.2 miles

Recalculate

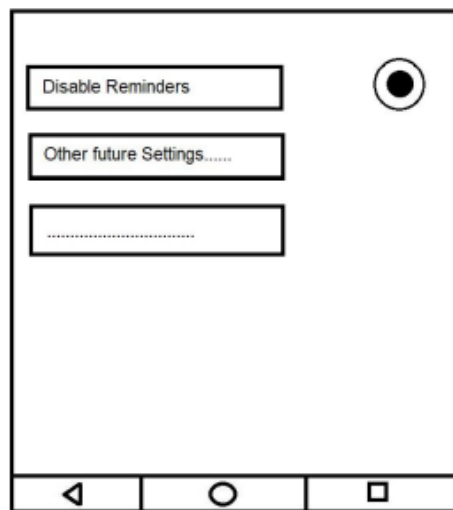
Recalculate button during Navigation



Home Screen after creating a schedule



Home Screen on a day with no classes



Settings menu (selected from Homescreen)

2.1.3 Software interfaces

The app must run on Android version ≥ 4.0 .

The app must utilize the Google Maps API.

The following data formats will be used:

- CSV

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

2.3 User characteristics

This application is intended to be used by college students that are enrolled in classes at Texas Tech University. The student does not need to know the address of the building of their classes or the full name of the building.

All users must be able to read and write.

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 smartphone 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 and the time when the class starts.

1.1.2 The software shall show the route to the next class in order of their starting times.

1.1.3 The software shall have a list of building names and the user will only be able to input a building name if it matches a building name on the list.

1.2 Non-Functional User Requirements:

1.2.1 The software shall be convenient and user-friendly.

1.2.2 Students should be able to expertly add the schedule as required after 15 minutes of tinkering with it.

1.3 Functional System Requirements:

1.3.1 The main screen of the software will contain a create schedule button.

1.3.2 The software saves the name of valid Texas tech University buildings in an array.

1.3.3 The software creates an error message if the user enters a building name that doesn't match with name of any building in the array.

Story 2. As a user, I want to be able to save my schedule for the day and give it 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 and give it a name.

2.1.2 The software shall allow the user to load a previously saved schedule from the main screen.

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 Students should be able to expertly save a schedule in the app after 5 minutes of tinkering with it.

2.3 Functional System Requirements:

2.3.1 The software shall have a save button appear on the screen once the user has entered in the course name, building name and the time.

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 day.

3.1 Functional User Requirements:

3.1.1 The software shall allow the user to set created schedules for a specific day of the week.

3.1.2 The software shall allow the user to set a single schedule for multiple days of the week.

3.2 Non-Functional User Requirements:

3.2.1 The user must be pleased with the GUI appearance.

3.2.2 The software shall have all menus in a consistent format.

3.3 Functional System Requirements:

3.3.1 After clicking the save schedule button, the software will prompt the user to set it for a day or multiple days of the week.

3.3.2 The software shows a time conflict error if the schedule set for a specific day overlaps with a different previously saved schedule for the same day.

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 recall a list of stored schedules by pressing an appropriate menu button.
- 4.1.2 The system shall allow the user to edit or delete any recalled scheduled.

4.2 Functional System Requirements:

- 4.2.1 The system shall be designed such that the option to display stored schedules is easily accessible to the user from either a drop-down menu or side panel.
- 4.2.2 The system shall provide the user with “View”, “Edit”, and “Delete” options once a schedule has been selected.

4.3 Non-Functional Requirements:

- 4.3.1 The system shall store the user’s list of saved scheduled 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 After getting the location of each destination, the system shall put all the routes on one map and show them to the user.

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 be able to connect to the GPS consistently to constantly get the information needed.

8.1.2 The system should calculate the shortest route from classes depending on distance and time of travel.

8.1.3 The system should use a timer to understand which destination the user is going to.

8.2 Non-Functional Requirements:

8.2.1 The system must be capable of connecting with google GPS, 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 shall get the current location of the user using the GPS function, which will utilize the Google GPS helper function.

8.3.2 The system will get the location of the user and the location of the destination and calculate the shortest path between them.

8.3.3 The system will print the map destination to the screen to show the shortest location and time 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 Every time the location of the user is recalculated, the app must show the user their new location on the map.

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 be selected by 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 $\text{time} = \text{route length} / \text{average speed}$

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 Screen should show a warning with message "Going Late!"

12.2 Non-Functional User Requirements:

12.2.1 System automatically shows the warning.

12.3 Functional System Requirements:

12.3.1 System read result from 11.3.2

12.3.2 System has pre-set time

12.3.3 System pop warning if "pre-set time" < time "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 Software must honor device silent mode setting.

13.2 Non-functional user requirement:

13.2.1 Easy to use: There should be no mention of this feature in the app.

13.2.2 Easy to use: There should be no functionality to altar this behavior.

13.3 Functional system requirements:

13.3.1 The software will register notifications with the device and allow the operating system to properly handle the functionality.

13.3.2 If necessary, all reminder functions 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 software shall allow disabling notification settings.

14.2 Non-functional user requirement:

14.2.1 Easy to use: The setting will be prominent within a settings menu as one of few settings. Toggling the setting should be simple.

14.2.2 Speed: 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 software shall recalculate the path to the next destination when a 'recalculate' button is pressed.

15.2 Non-functional user requirement:

15.2.1 Ease to use: The feature will require no training. Users in 99% of cases must be able to use the feature on their first attempt.

15.2.2 Speed: The recalculation should be completed within 10 seconds of the recalculate button being pressed.

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 cancelled for the day and have directions and scheduling in the app updated accordingly, but have the

regular schedule resumed the next 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 should allow user to change schedule temporarily, the general schedule should not be affected.

18.2 Functional System Requirements

18.2.1 System should give user options to choose: temporarily change, general schedule change

18.2.2 Temporary change will not affect general schedule

18.2.3 Temporary changed schedule has higher priority than general schedule.

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 Existing/saved schedules could be edited

19.2 Non-Functional User Requirements

19.2.1 System should respond in one second

19.3 Functional System Requirements

19.3.1 There should be an edit bottom on schedules

19.3.2 System should show the schedule selected by user

19.3.3 System should allow user to edit the schedule

19.3.4 System should allow 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 System should be able to generate excel and CSV file base on the schedule.

20.2 Non-Functional User Requirements

20.2.1 System should keep schedule confidential.

20.3 Functional System Requirements

- 20.3.1 System should be able to convert schedules into Excel and CSV files.
- 20.3.2 System should be able to send Excel and CSV files via email

3.2 Use Cases

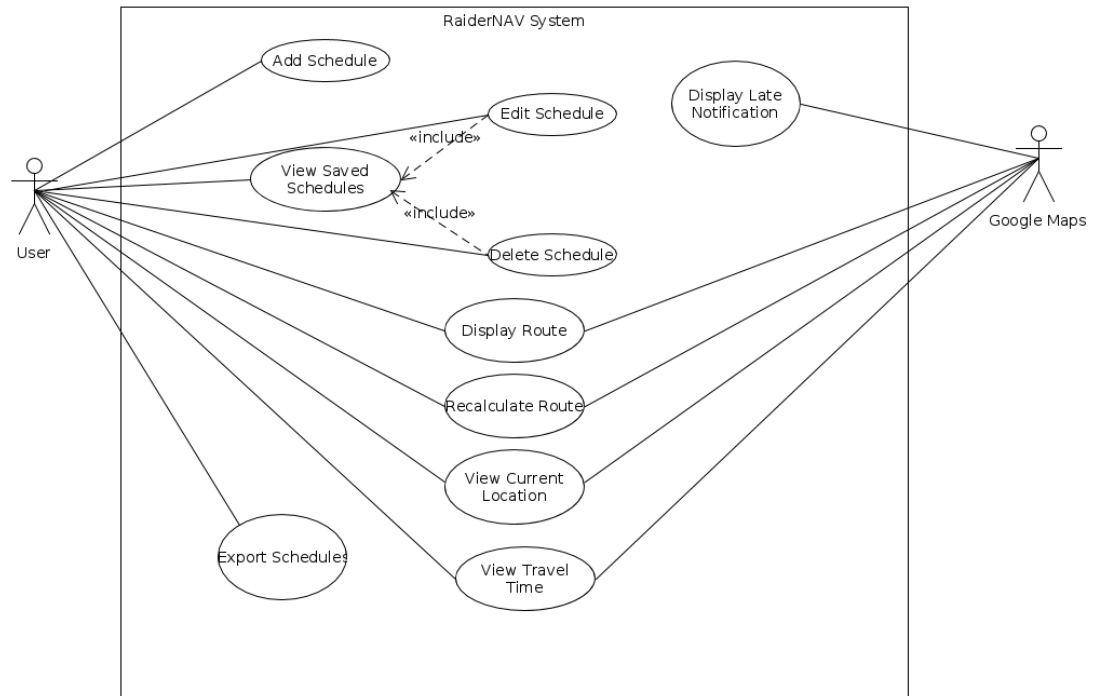


Figure 3.2.1. Unified use case diagram.

Story 1

Use Case: Add Class schedule

Summary: App User creates a new class schedule

Actors: Application User

Dependency: none

Pre-condition: the application should be in the Home-screen

Description:

1. User presses add schedule button on the home-screen.
2. The application prompts the user to enter the course name, enter the building name and the starting and ending time of the class.
3. The application allows the user to add more courses in the same schedule.

Alternatives:

2a. If the user enters a building name which doesn't match with the name of building in a set array, application will show error message saying no building of such name found.

Post-Condition: The user has entered all the courses, their building names and start and end times for each course.

Story 2

Use Case: Save Class schedule

Summary: App User saves a created class schedule

Actors: Application User

Dependency: none

Pre-condition: the user has entered the course name, building and the starting and ending times of all classes in a schedule.

Description:

1. User presses the save schedule button.
2. The Application prompts the user to set the schedule to a specific day or multiple days of the week.

Alternatives:

2a. 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.

Post-Condition: The class schedule is saved.

Story 3

Use Case: Edit Class schedule

Summary: App User edits a previously saved class schedule.

Actors: Application User

Dependency: Add Class Schedule Use Case

Pre-condition: the schedule list in the application is not empty.

Description:

1. The user presses edit schedule button on the home screen.
2. The application prompts the user to select a schedule from the list of previously saved schedules.
3. The user selects the schedule that needs to be edited.
4. The user will be able to edit one or more of the class; the course name, building name and the start and end time.

Alternatives:

4a. If the user enters a building name which doesn't match with the name of building in a set array, application will show error message saying no building of such name found.

Post-Condition: The user has edited a previously save class schedule.

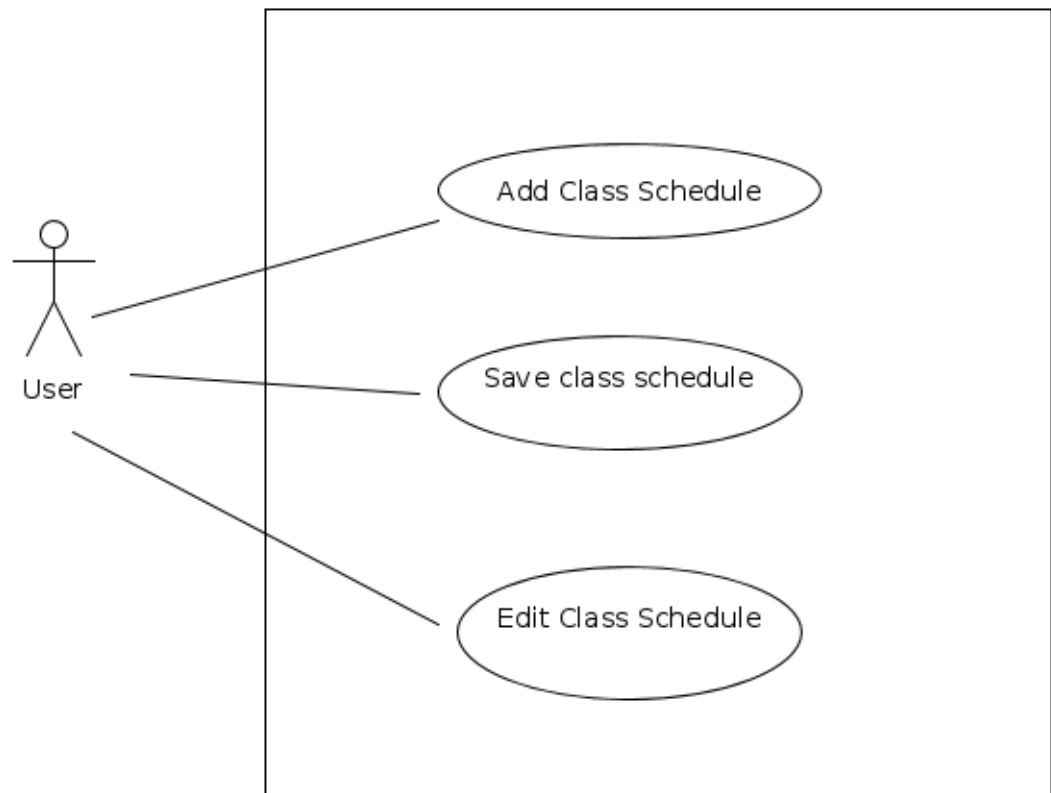


Figure 3.2.2. Use cases 1-3.

Story 4

Use Case Name: Delete Schedule

Summary:

Actor:

Dependency:

Precondition:

Description:

Postcondition:

Story 5

Use Case Name:

Summary:

Actor:

Dependency:

Precondition:

Description:

Postcondition:

Story 6

Use Case Name:

Summary:

Actor:

Dependency:

Precondition:

Description:

Postcondition:

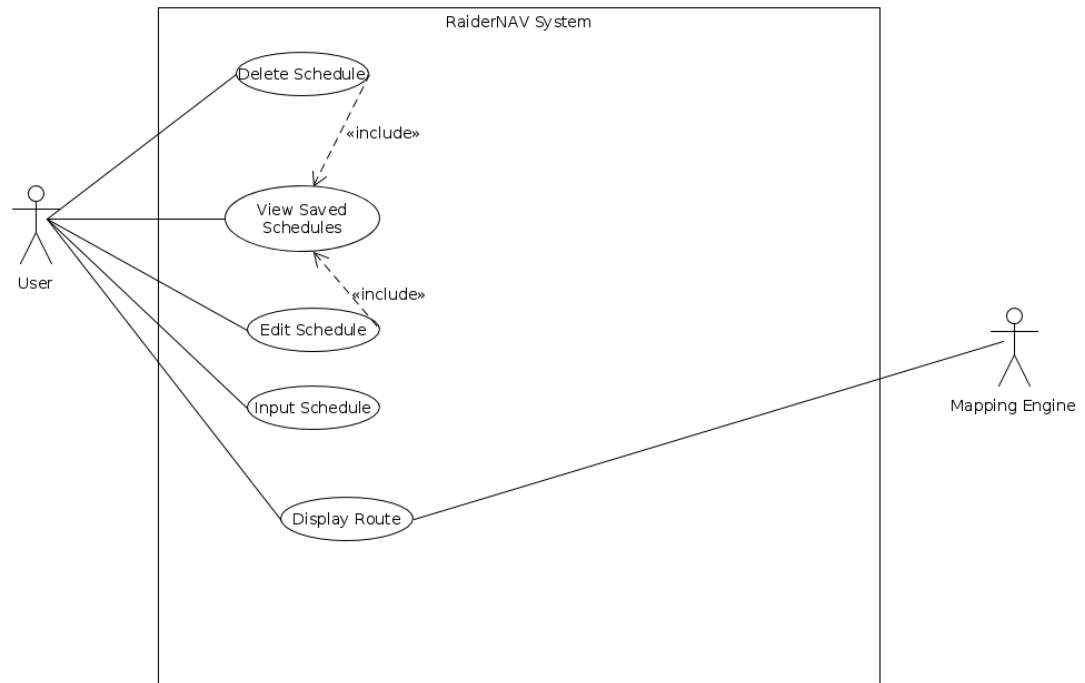


Figure 3.2.3. Use cases 4-6.

Story 7

Use Case: All Routes

Actor: App User

Dependency: Which daily schedule is displaying

Precondition:

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

Story 8

Use Case: Shortest Route

Actor: App User

Dependency: Location of the user and destination

Precondition:

Description:

- 1) The user determines which destination they are going to
- 2) GPS takes the destination and finds the shortest route from the user

Story 9

Use Case: Recalculate Location

Actor: App User

Dependency: Timer

Precondition: Timer recalls the user location automatically

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

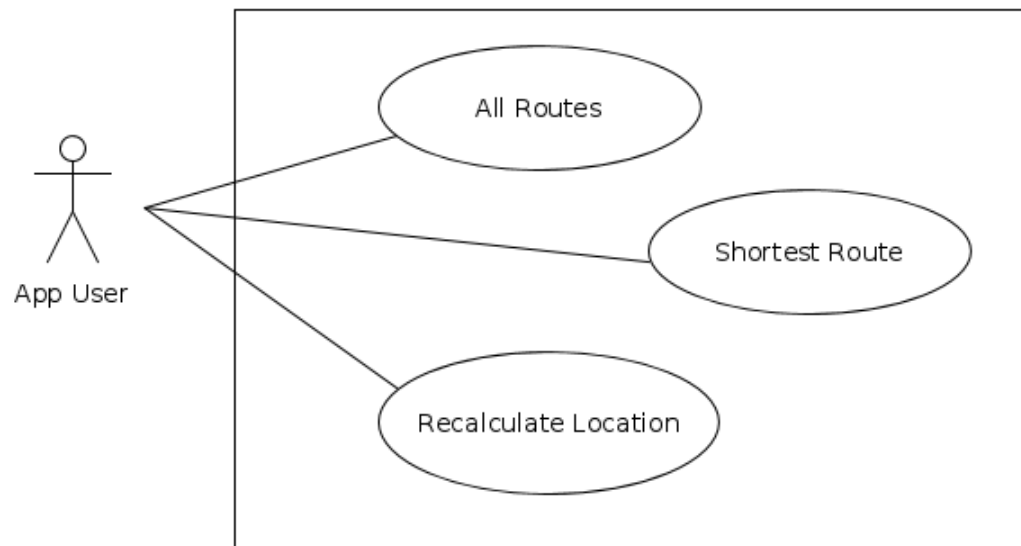


Figure 3.2.4. Use cases 7-9.

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:

The app read route's length from google api function

The app does calculation to get time needed $C2: S/V$ (S is route length, V is average velocity)

The app prints/shows the result on the screen.

Story 11

Use Case: Edit schedule

Summary: The app should allow users to edit schedule.

Actor: User

Precondition: The schedule is existed

Description:

User click the "Edit" button on the existing schedule.

User input destination location, class time into the schedule.

User click the "save" button to finish editing

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:

The app does the calculation $C1: T1-T2$ ($T1$ is the next course's beginning time, $T2$ is the current time)

The app shows the result: $T1-T2$

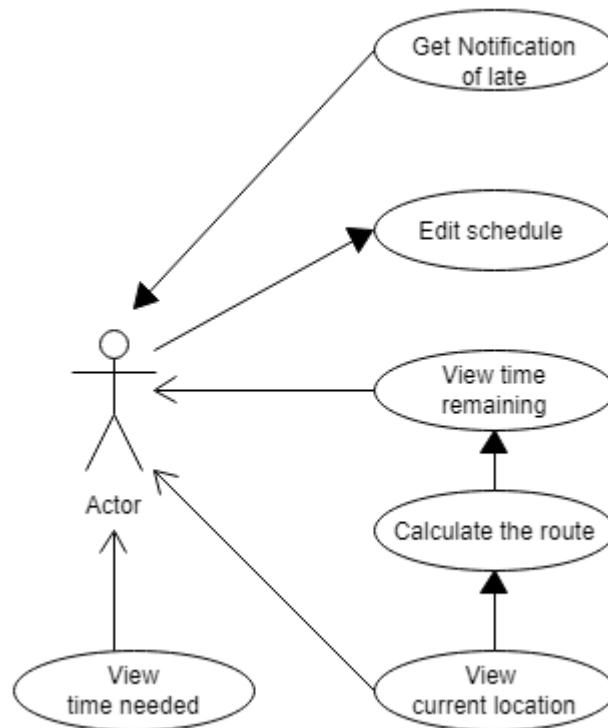


Figure 3.2.5. Use cases 10-12 and 19.

Story 13

Use Case Name: Toggle Reminders

Summary: App user enables or disables reminder/notification feature

Actor: App 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: App 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

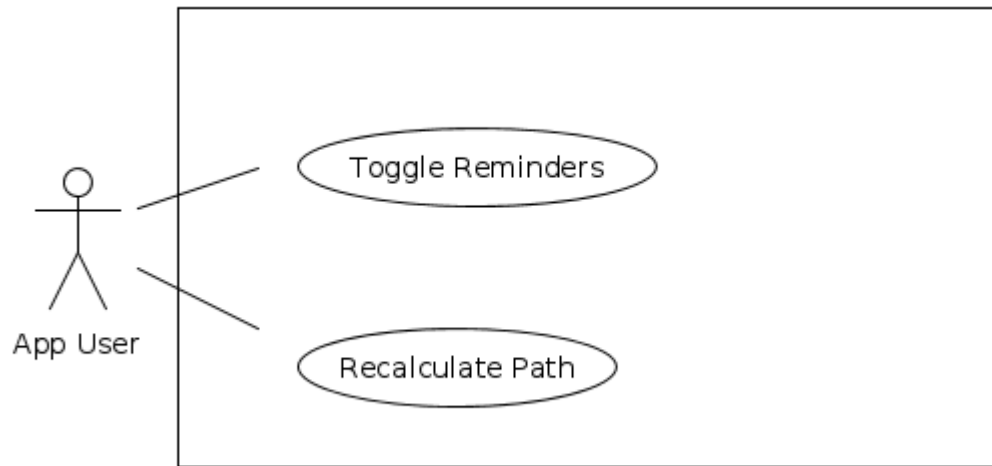


Figure 3.2.6. Use cases 13 and 15.

Story 16

Use Case Name:

Summary:

Actor:

Dependency:

Precondition:

Description:

Postcondition:

Story 17

Use Case Name:

Summary:

Actor:

Dependency:

Precondition:

Description:

Postcondition:

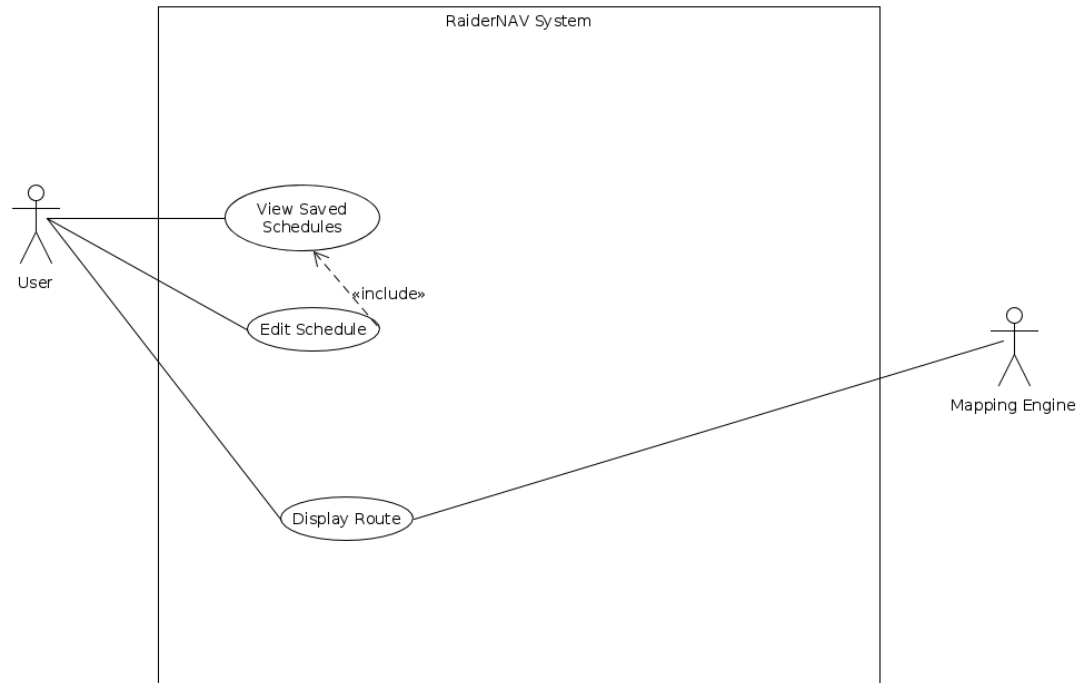


Figure 3.2.7. Use cases 16 and 17.

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:

The schedule could be changed with two options: temporarily, generally

The temporarily change will not affect general schedule

The temporary change has higher priority than general schedule.

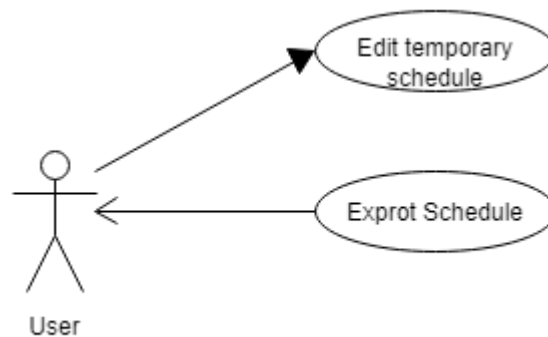


Figure 3.2.8. Use cases 18 and 20.

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: The user

Precondition: user is on the route to the next class

Description:

If $C2 > C1$, notification pop out, else notification will be dismissed

If user arrives at the class location, notification procedure will be dismissed

Story 20

Use Case: Export schedule

Summary: The app should allow users to export schedule.

Actor: User

Precondition: The schedule is existed

Description:

User click the “export” bottom on the existing schedule.

The app will send schedule via email in two formats: excel and CSV.

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
- The app must be programmed in Java