Department of Computer Science
University of Pretoria

Software Engineering
COS301 MiniProject

Team Maroon

NavUP

Bondjobo, Jocelyn (J) 13232852
Mweshi, George (G) 16394713
Letsoalo, Joseph (J) 15043844
Setoaba, Phuti (P) 13032616
Trivella, Camron (C) 14070970
Coetzer, Albert (A) 15244882

Contents

1 Introduction

1.1 Purpose

The purpose of the System Requirements Specification is to give an overview and understanding of the system we are planning to implement. It is created to give the client an overview of what we plan to implement and what resources and requirements it will need.

1.2 Scope

1. Name:
   NavUP
2. Purpose:
   A navigation system that will help students, faculty and guests navigate their way around the University of Pretoria's Hatfield campus in the most efficient way possible.
3. Objectives, goals and benefits:
   The objective of NavUP is to create a functioning navigation system that can easily take users to exact lecture halls and other locations around the Hatfield campus.
   The goal of this system is to make it easier for new students and guests to find their way around the campus as well as make it easier to avoid human traffic when travelling to lectures or other locations.
   The benefits of NavUP will be efficiency when it comes to new students and guests travelling to new lecture halls, helping users get to locations on time and making it a less stressful experience for new students trying to find their way around campus for the first time.

2 Overall Description

2.1 Product Perspective

2.1.1 System Interfaces

Each component and system/subsystem in the overall program will need its own interface between its own software and the bigger program. For instance, the crowd sourcing routines must be able to record and report a density map to the main branch of the program that must then be able to be accessed and utilized by an analyser which must then return it to the main branch to be retrieved as needed for navigation by the user.

2.1.2 User Interfaces

The user interface must consist of an easily understandable and navigable GUI, simple enough for even the most inexperienced to use. The different options and account information must be visible and responsive and the navigation component must be clear and accurate. The user should intuitively know how to get from any point to any other point in the program.

2.1.3 Hardware Interfaces

The hardware interface must allow the program to access and utilize systems on the device on which the program is running, specifically the GPS system if the program uses it, the wireless network adapter to check for Wi-Fi routers and signal strength, the screen to display information and the device used to make selections. The hardware interface will be the backbone of the program, allowing directions and instructions to be communicated to the user.

2.1.4 Software Interfaces

The routines by which the program will be able to call, run and interpret other software on the device being used to run the program. The routines will, for example, utilize the software responsible for managing the connection between the device and the Wi-Fi routers. This will co-exist with the hardware interface to make use of the hardware on the device for the program.

2.1.5 Communications Interfaces

The communications interface will most likely be used to connect devices being used to run the program to other devices being used to run the program. One of the uses for this will be in crowd sourcing density data of the users to help effectively and intelligently create the most efficient instructions for a specific user or to record and utilize the most common trends in user movements to predict behaviour and increase efficiency.

2.1.6 Memory

Information such as maps, router positions and user identities will most likely need to be stored on a server by the program administrators. Information gathered about the users, such as common routes, time spent using the program, user information and any other data collected by the program should also be stored in the same way.However, an alternative would be to store the information on the user's device and report it to the server when needed.

2.1.7 Operations

The primary operation of the program will be directing students around the University of Pretoria's main campus, secondary operations may include crowd sourcing utilities and a rewards program. It should be possible to store and retrieve information about a user's movements in order to back trace if necessary.

2.1.8 Site Adaptation Requirements

The program should be available in a variety of language options.

2.2 Product Functions

1. Basic Functionality
   NavUP has basic functionality that allows the user's current location to be determined indoors and outdoors. The user can further search for a location, save a location, navigate to a location and see the route in 3D. Zoom in or out capability is also available. Location markers are there to help the user differentiate their starting point, destination as well as other locations. The application can also generate an time estimate that the user may take to get to their destination.

2. Surveillance
   Through surveillance, NavUP is able to provide and visualise pedestrian traffic information on campus and further provide an alternative route if one is available to avoid pedestrian traffic. Another capability that can be achieved is to record the user's number of steps which can be used by the user to track their movement for fitness or leisure or by a third party to run competitions.

3. Extra features
   The application can give recommendations of locations on campus in accordance to the user's interests and preferences. And Rate and review places on campus.

2.3 User Characteristics

The application NavUP can expect to have three types of users; the registered student or staff member, the guest and the administrator.

1. The registered student or staff member and guest user are expected to be able to provide a destination location in terms of name of the building or faculty and room.

2. The registered student or staff member will be able to save a track record of movements, favourite locations.

3. The guest will have limited functionality; hence will only be able to navigate from place to place.

4. The administrator must be able to set goals and provide information regarding competitions to be facilitated through NavUP. This entails the award giving process either for attendance of an event or achieving the set number of steps.

2.4 Constraints

As with any big project, there are a lot of constraints that need to be taken into account when developing the application.

1. Programmer Experience
   As third year students, one constraint this project faces is the lack of experience in the development team. The size and complexity of the program could prove to be more challenging than we originally anticipated.
2. WiFi Coverage
   Even though there has been a lot of upgrades in the WiFi coverage inside of buildings on Hatfield Campus, there are still a number of ”dead spots” that could prove difficult to navigate through. With WiFi being the current plan to navigate users to specific rooms inside of buildings, a building without proper WiFi coverage will be challenging to deal with.
3. Time
   For a project of this magnitude, the time allocated to develop and actually implement the software system could prove to be challenging considering the amount of other work the majority of the students have during third year.

2.5 Assumptions and Dependencies

There are a certain amount of assumptions we need to make in order for this project to work as well as dependencies that we need for it to be a success.

1. Smartphones
   We need to assume that all users will have smartphones with the technologies we require them to have. Without this the application will not work for those users.
2. WiFi Coverage
   As we mentioned in the constraints above, WiFi coverage is crucial for indoor navigation. Without it the entire app falls apart.
3. Access to Tuks Network
   Without sufficient access to the wireless network we will be unable to implement the WiFi navigation system which is crucial to indoor navigation.

3 Specific Acquirements

This section gives a detailed description of the system requirements. It describes all the functional as well as the quality requirements of the system.

3.1 External Interface Requirements

This section provides a detailed description of the system interfaces,user interfaces, hardware interfaces, software interfaces and communications interfaces. It also provides a description of the inputs and outputs for each of the interfaces.

3.1.1 User Interfaces

The system will consist of both an Android-based and iOS-based(Xcode) interface which will allow system users to interact with the system. These interfaces will be used to enter different types of information into the sytem regarding venues,points of interest, events and activities.
There will also be provisions for text inputs and push buttons which will allow the users to search for locations and save locations. Graphics will be used to provide a visual representation of the directions to a location as well as information on events taking place on a particular date.

1. **User Inputs** The application will receive user inputs through the user interface. The interface will provide both the keywords to use for locations searching and the information on venues, events and activities to be stored in the database.

2. **Navigator** The application will determine your current location and show a map depicting the optimal route to the point of interest by accessing the GPS system.
3. **Calendar** The application will include a calendar which will show all the events on the corresponding dates.

3.1.2 Hardware Interfaces

The application will run on an Android mobile device as well as on an iOS mobile device.All hardware interfacing and all connections to the database server will be handled by the operating systems on the mobile device and web server. In case of using the wireless network adapter or GPS, this will be managed by the respective applications in the mobile device.

3.1.3 Software Interfaces

The application will run on the Android operating system, specifically version 4.0. and upwards.It will also run on iOS operating system version 7.3 and above.

3.1.4 Communications Interfaces

The application will communicate with the different parts of the system via API function calls.The exact formats and protocols for incoming and outgoing messages should be abstracted by the APIs.

3.2 Functional Requirements

3.2.1 Use case prioritization

1. **Critical**
   - Add Event
   - View/Edit Event
   - Add Venue
   - View/Edit Venue
2. **Important**
   - History of events and venues searched
   - System Log
3. **Nice to have**
   - Visual and Voice guiding
   - Android Interface
4. **Diagrams**
   - Please refer to the three last pages for diagrams.

3.3 Performance Requirements

This section describes all the performance related capabilities of the system.

3.3.1 Real-Time Information

The application will provide information that is up-to-date at all times. The system user will be notified should any delays occur.

3.3.2 System Resource Management

The application should not consume system resources such that the mobile device becomes unusable. There should be provision for the application to work in the background should the user wish to use other applications.

3.4 Design Constraints

1. User Movement
   Because user location changes, to account for the location of the user in a building with one/ weak
2. Failure connection
   Wi-Fi connection will lead to failure in updating to user location. Since the app uses Wi-Fi to find location and navigate if the user fails to connect to Tuks Wi-Fi at that moment the app will not work regardless of whether the user uses cellular network data.
3. Location in buildings
   Finding the current location of the user and displaying it on the app inside the buildings will be restricted since we don't have the lay-out/map of all buildings.

3.5 Software System Attributes

3.6 Other Requirements

3.6.1 Technology and Android clients requirements

1. **Programming Languages**
   - Java
   - eXtensible Markup Language (XML)
2. **Frameworks**
   - Android Studio
3. **Libraries**
   - Android Butterknife
4. **Database System**

   Couchbase Mobile which consists of:
   - Couchbase Lite
   - Couchbase Sync Gateway
   - Couchbase Server
5. **Operating System**
   - Android 4.0. Devices and upwards
6. **Dependency Management and Build Tools**
   - Gradle

3.6.2 Technology and IOS clients requirements

1. **Programming Languages**
   - Objective C
2. **Frameworks**
   - Apple's Swift
3. **Libraries**

- CocoaPods
- Carthage

4. **Database System**
    - AFNetworking
    - JSONModel
    - MagicalRecord
    - SDWebImage
    - ReactiveCocoa
5. **Operating System**
    - IOS version 7.3 and above
6. **Dependency Management and Build Tools**
    - Swift Package Manager (SPM)

3.6.3 Quality requirements

- **Performance**
    1. **Description**
       Application performance can be defined as the amount of work that can be accomplished by an application in question in a measured time interval. The time interval is normally measured in seconds, where the amount of work can be defined as the throughput, latency or data transmission time.
        - **Throughput**
          The number of requests and responses which can be processed by the system in a given time interval for example the time the application takes to locate its user on the map using wifi access points and gps.
        - **Latency and Data transmission time**
          A time interval measured as the time it takes to service a request such as to find the best route or path to guide the user to a certain destination.

       The aim for this system, is to increase the throughput and decrease the latency. As the developer has no control over the network medium used, he/she must aim for a minimal request and response payload as to decrease the data transmission time.
    2. **Justification**
       Our aim for this system is to increase throughput, decrease latency and data transmission time. This will ensure we have a system that is responsive at all times, including peak times and delivers an excellent user experience.
    3. **Requirements**
        - Function calls must be timed and benchmarked and this data should be logged.
        - Network responses should be cached on server side to lighten the load on the device.
- **Reliability**
    1. **Description**
       The designed system needs to be accessible from both inside and outside builinds at the University of Pretoria campuses as well as from other networks, especially on other campuses on the TENET network. The system should be reliable in its accessibility.
    2. **Justification**
       The system must not be unreliable in that it crashes under large workloads for example when many users are actually using the application and some users get denied service during work critical times when workloads are high. This causes a detriment in user productivity. To ensure that the system can be used confidently as a tool to increase productivity and ease the work process, the development aim must be for the system to be as reliable and accessible as possible.
    3. **Requirements**

  - To enable access of data from the Android app even when offline such as preferences, events, etc..
  - Hot swapping of system modules should not affect system service reliability.

- **Scalability**

  1. **Description**
     Scalability refers to the application in question's ability to handle an above normal workload for exemple when many users are currently using the app as it must provide and visualise information related to pedestrian traffic on campus for example in the form of heat maps of user locations.
  2. **Justification**
     The system needs Scalability because it needs to support many users at the same time.
  3. **Requirements**
     - The system should be able to handle the growing amount of data or number of users using the app at the same time.

- **Security**

  1. **Description**
     Security in application software refers to authentication, authorization, data security and accounting. Authentication refers to the systems' ability to provide a way of identifying a user, normally with our system it will identify its users anonymously from the device name.
  2. **Justification**
     Security is a important aspect of any software product. In terms of information security, we are concerned about integrity, availability, confidentiality and non-repudiation in that order.
  3. **Requirements**
     - System should be resistant hacking as some people could use it to steal some personal information on the device about the user.

- **Flexibility**

  1. **Description**
     Flexibility refers to the ability of the system to be changed dynamically either by hot swapping certain components in a live system or by extending the system with some kind of plugin for example if there are some new building added on campus it should able to be updated or extended on the data the device has.
  2. **Justification**
     Flexibility is important for any system. A non-flexible system is restricted to using technologies that were hard coded into it, and this necessitates, at best, large scale refactoring every time an upgrade is available since new technologies need to be reintegrated, makes adding new features tedious, and risks the system becoming archaic. A flexible system requires minimum effort to upgrade and expand, allowing for the system to easily grow in usefulness and function beyond the original vision.
  3. **Requirements**
     - Modules should be decoupled from one another, allowing the system to be extensible without a break in service which is achieved by integrating new modules and swapping out existing ones.

- **Maintainability**

  1. **Description**
     The system is to be designed in such a way that it is easily updated, modified or extended by the client in the future. In order to achieve these requirements, design patterns and best practices such as coding style guides are normally used to ensure uniformity and modularity across the system.
  2. **Justification**
     Many systems require regular changes, not because they were poorly designed or implemented, but because of changes in external factors.
  3. **Requirements**
     - All code should be documented in the applicable language documentation framework, such as JavaDocs for a Java based system, etc.

    - A coding style guide/manual should be set up and associated with the project, such that all developers use similar coding styles and conventions, to allow for more readable code that is easier to maintain.
    - System should be separated in distinct, concise and independent modules relating to separate concerns, to allow for easier maintenance.

- **Auditability/Monitorability**

  1. **Description**
     The system is to be designed to be verbose and transparent in its workings, and to ensure maximum data security, to allow role players to have insights into how the system is used and how it may be improved. These requirements are achieved by making the maximum amount of relevant data available to its users and by enforcing strict constraints on the data that is stored.
  2. **Justification**
     This is an important process in Software Engineering, where all the informations must be correct so requiring all the developers to see who made changes and when so that consistency must be kept in order to keep the system's data accurate and reliable.
  3. **Requirements**
     - Data in the app should always be consistent. This implies that all data should adhere to constraints placed on the data by the data model, such as regex patterns, minimum and maximum length, non nullable fields, etc.

- **Integrability**

  1. **Description**
     The system should allow for future external integration with other platforms such as security authentication providers, different map database to be loaded, etc.
  2. **Justification**
     The system necessitates integrability to allow for maximum usability, as the integrability of the system is directly related to how usable it is. To be usable, the system must allow for easy migration, not just from previous systems, but also to future systems and future data storage mediums. The usability is also largely determined by how well the back-end system integrates with front end clients, and which clients are supported.
  3. **Requirements**
     - The system should allow technology neutral importing and exporting of data.
     - The back-end system should integrate with an Android mobile app clients.
     - The system should be able to integrate with different back-end authentication services.

- **Cost**

  1. **Description**
     The cost of the system entails any initial expenses as well as any ongoing expenses which the client may incur at some point. Such expenses arise from software licenses, external computing resources required as well as future maintenance of the system in terms of time.
  2. **Justification**
     The expenses made and software licenses cost must be taken into account in order to set the price of the software.
  3. **Requirements**
     - System should be cheap to operate, maintain and extend. If the quality and maturity of technologies available allow it then the technologies used must be freely available/usable.
     - As far as possible, open source compatible, mature technologies should be used, to ensure system stability and deployment on different OS as far as possible.

- **Usability**

  1. **Description**
     Usability refers to ease with which humans, and to a lesser extent, servers interact with the system in question. Usability can be measured in various ways such as using quantifiable scientific measures or more subjective measures with a key question point being if the API follows conventions and so on.

2. **Justification**
   It is important that the new system is usable as it is a user-centric system. Ensuring that the system is usable will ensure that users are provided accurate and correct information from the system which will for better performance and departmental strategic planning. As this system will also be used by parties outside of the University of Pretoria, it is important that the system conveys a professional image, as this will reflect on the image of the University of Pretoria.
3. **Requirements**
   - Mobile devices running Android 4.0. and upwards should be fully supported.
   - Material design UI guidelines prescribed by Google must be used, to ensure that the clients feel modern and familiar.
   - Android app should support the full back-end API specifications.
   - The user must be allowed to elect whether they would like their data to be available offline.
   - Mobile client users should be able choose between working on Wi-Fi or network data.

4 User Case Diagram, Actor-System Diagrams and Traceability Matrix

1. **User Case Diagram**

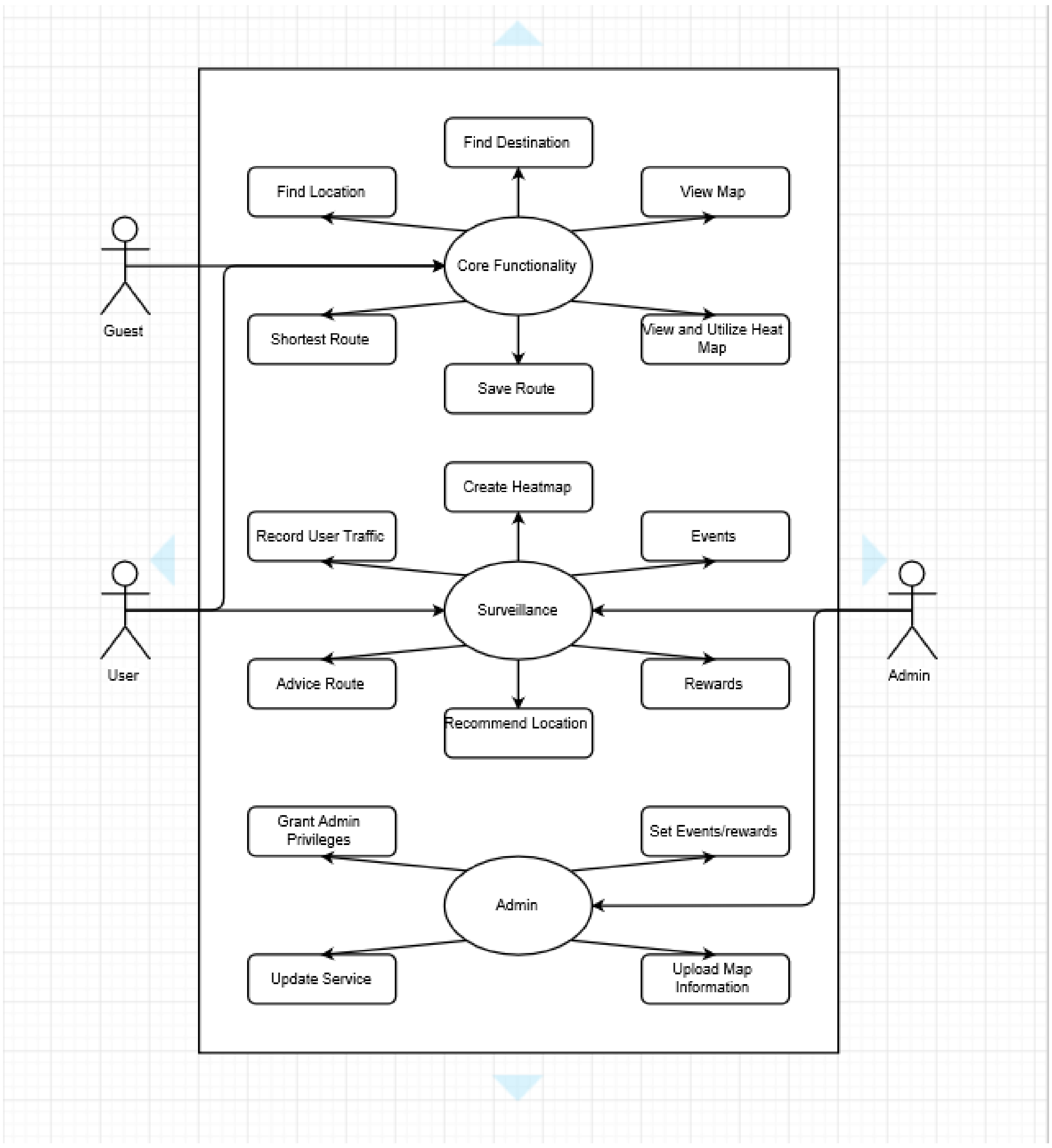


Figure 1: User Case Diagram

2. **Actor-System Diagrams**
   The actor-system diagrams are diagrams that model how the system interacts with the actors to carry out the use cases.

3. **Traceability Matrix**

USE CASE 1: Find Location	
Actor: Guest	System: NavUP
	0. The NavUP displays the main menu
1. The Guest clicks the Find Location button on the main menu	2. The system determines and displays the Guests current location information as well as an Exit button
3. The Guest clicks the Exit button	4.The system closes the Guest location information page and displays the main menu

Table 1: Actor-System Interaction for Use case: Find Location

USE CASE 2: Find Destination	
Actor: Guest	System: NavUP
	0. The NavUP displays the main menu
1. The Guest clicks the Find Destination button on the main menu	2. The system displays an input text box for entering the destination
3. The Guest enters the destination he wishes to find in the input textbox and clicks Submit button	4. The system displays all destination information as well as an Exit button
5. The Guest clicks the Exit button	6. The system closes the page showing all destination information and displays the main menu

Table 2: Actor-System Interaction for Use case: Find Destination

USE CASE 3: View Map	
Actor: Guest	System: NavUP
	0. The NavUP displays the main menu
1. The Guest clicks the View Map button on the main menu	2. The system displays an input text box for entering the point of interest
3. The Guest enters the point of interest he wishes to view in the input text box and clicks Submit button	4. The system displays a map showing the location of the point of interest as well as an Exit Button
5. The Guest clicks the Exit button	6. The system closes the page showing the point of interest map and displays the main menu

Table 3: Actor-System Interaction for Use case: View Map

USE CASE 4: Find Shortest Route	
Actor: Guest	System: NavUP
	0. The NavUP displays the main menu
1. The Guest clicks the Shortest Route button on the main menu	2. The system determines Guests current location and displays an input text box for entering the point of interest
3. The Guest enters the point of interest he wishes to navigate to and clicks Submit button	4. The system displays a map showing the shortest route to the point of interest from his current location as well as an Exit Button
5. The Guest clicks the Exit button	6. The system closes the page showing the shortest route map and displays the main menu

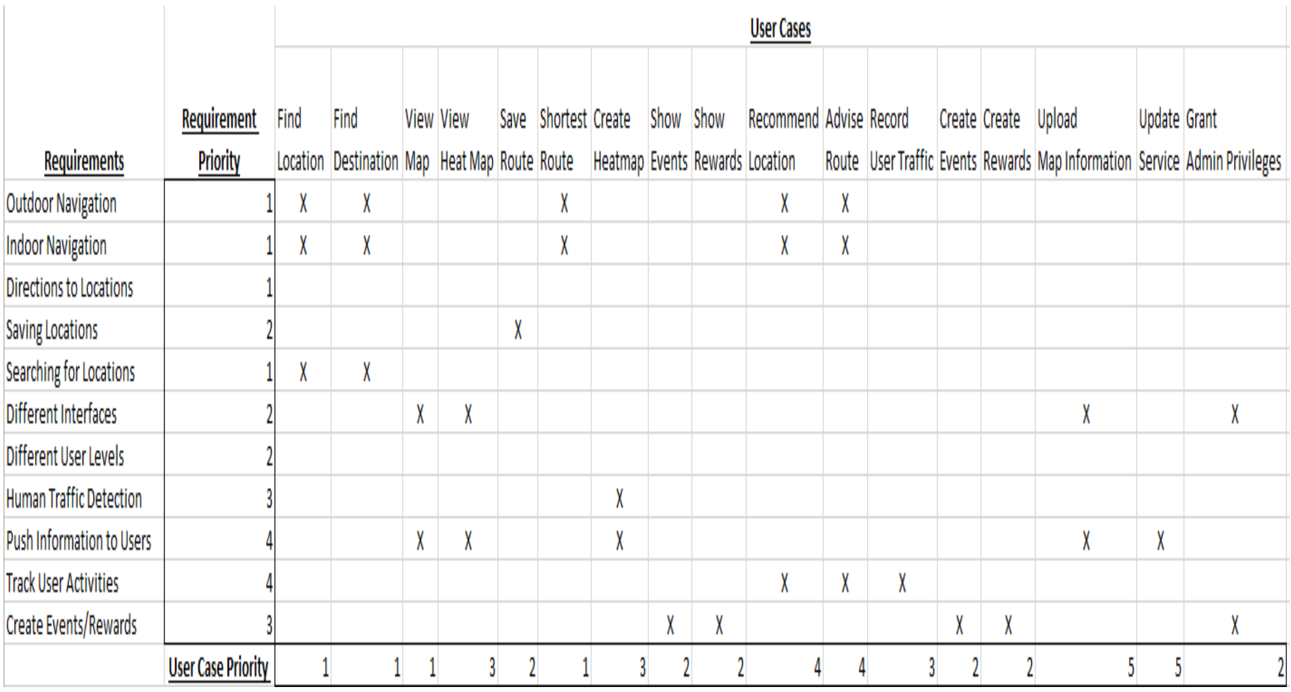
Table 4: Actor-System Interaction for Use case: Find Shortest Route

USE CASE 5: Save Route	
Actor: Guest	System: NavUP
	0. The NavUP displays the main menu
1. The Guestclicks the Save Route button on the main menu	2. The system determines Guests current location and displays an input text box for entering the point of interest
3. The Guest enters the point of interest he wishes to navigate to and clicks Submit button	4. The system displays a map showing the route to the point of interest from his current location as well as an Save Route Button
5. The Guest clicks the Save Route button	6. The system requests the Guest to provide a name for the file containing route information
7. The Guest enters file name and clicks Submit button	8. The system saves the file in a default location and displays the Exit button
9. The Guest clicks the Exit button	10. The system displays the main menu

Table 5: Actor-System Interaction for Use case: Save Route

USE CASE 6: View and Utilize Heat Map	
Actor: Guest	System: NavUP
	0. The NavUP displays the main menu
1. The Guest clicks the View and Utilize Heat Map button on the main menu	2. The system determines the Guests current location and displays a heat map of his current location as well as an Exit button
3. The Guest clicks the Exit button	4. The system closes the page showing the heat map and displays the main menu

Table 6: Actor-System Interaction for Use case: View and Utilize Heat Map



		User Cases																
Requirements	Requirement Priority	Find Location	Find Destination	View Map	View Heat Map	Save Route	Shortest Route	Create Heatmap	Show Events	Show Rewards	Recommend Location	Advise Route	Record User Traffic	Create Events	Create Rewards	Upload Map Information	Update Service	Grant Admin Privileges
Outdoor Navigation	1	X	X				X				X	X						
Indoor Navigation	1	X	X				X				X	X						
Directions to Locations	1																	
Saving Locations	2					X												
Searching for Locations	1	X	X															
Different Interfaces	2			X	X											X		X
Different User Levels	2																	
Human Traffic Detection	3							X										
Push Information to Users	4			X	X			X								X	X	
Track User Activities	4										X	X	X					
Create Events/Rewards	3								X	X				X	X			X
	User Case Priority	1	1	1	3	2	1	3	2	2	4	4	3	2	2	5	5	2

Figure 2: Traceability Matrix

5 Open Issues

5.1 GitHub Repository

Team Maroon Repository: `https://github.com/josephbondjobo/COS301-Maroon-Team/tree/develop/Task%201`

This repository contains:

1. All work done by team members.
2. CONTRIBUTORS.md file outlining which members where involved in which phases of the project.