UNIVERSITEIT VAN PRETORIA
UNIVERSITY OF PRETORIA
YUNIBESITHI YA PRETORIA
Faculty of Engineering, Built Environment and Information Technology

Team C# Round 2

# Architectural Requirements Specifications and Design for the NavUP System


| | |
|---|---|
| Peter Boxall | U14056136 |
| Seonin David | U15063021 |
| Mia Gerber | U15016502 |
| Oratile Motswagosele | U15306195 |
| Banele Nxumalo | U12201911 |
| Craig van Heerden | U15029779 |

# Contents

# 1 Deployment Diagram

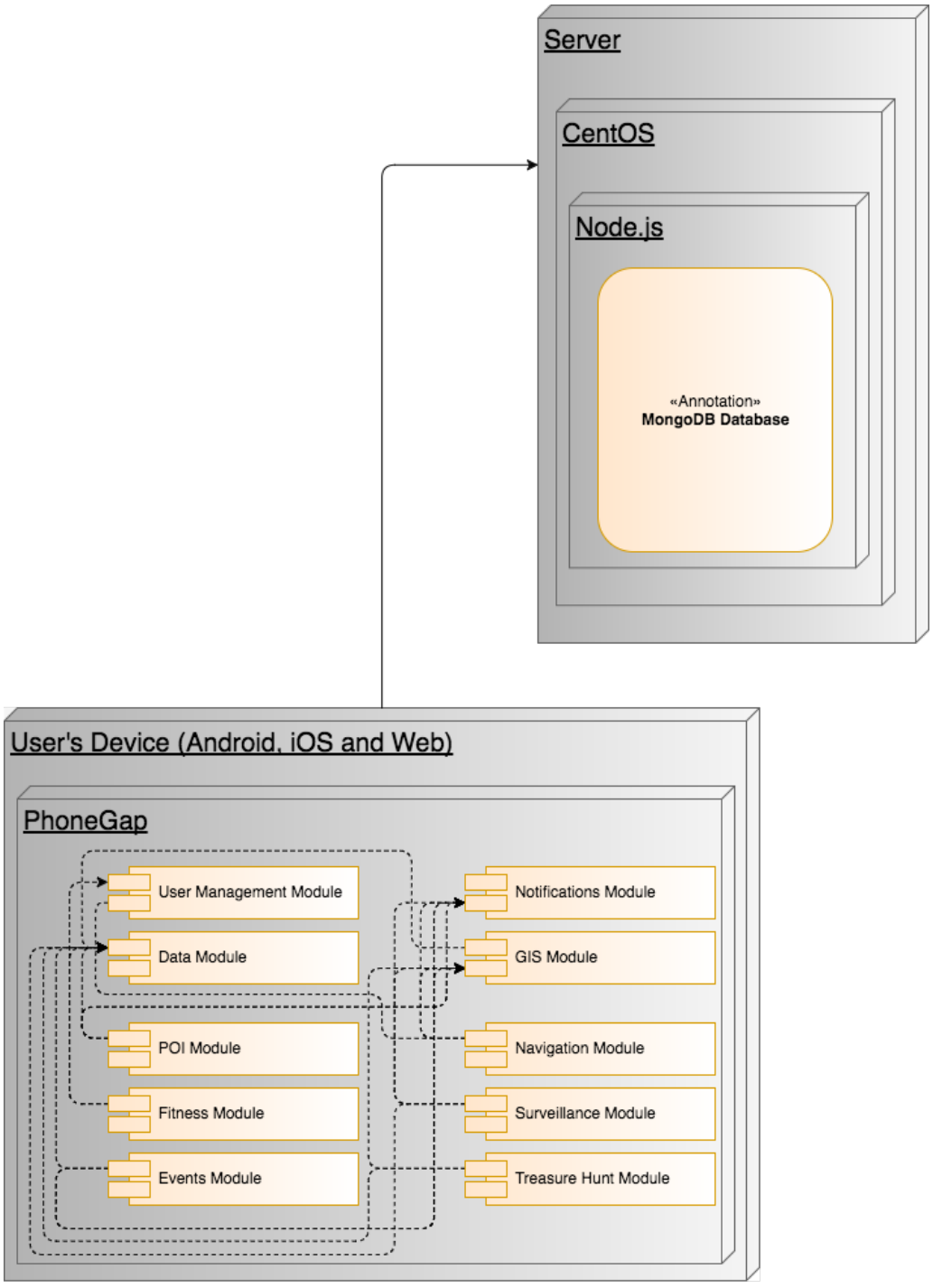


Figure 1: Deployment Diagram

# 2 Fitness Module

### 2.0.1 External Interface Requirements

Table 1: User Information

| | |
|---|---|
| Name | User information |
| Description | This is information about the user such as Age, Weight, Height and Identification,variables |
| Source & Destination | Source: Server, Destination: Device/Application |
| Range, accuracy | The range of this data : ID (0-60000), Age(0-120),Weight(20-500),Height(20-300) |
| Unit of measure | ID: Integer,,Age: Integer, Weight: kilograms, Height: centimeters |
| Timing | The information will be kept on device until such a point that the user logs out or uninstalls. The Information will be kept on server until such a point that the user deletes his/her account. |
| Relations to other | This data relates to the Fitness Information data because they are used in the calculations of the fitness statistics. |
| Data format | This data relates to the Fitness Information data because they are used in the calculations of the fitness statistics. |

Table 2: Navigational Information

| | |
|---|---|
| Name | Navigational Information |
| Description | This is the, navigational data that allows the user to navigate around campus. More, specifically with regards to fitness this data will allow us to keep track of, distances and allow fitness statistics to be drawn from the distances, travelled during navigation. |
| Source & Destination | Source:Server, Destination: Device/Application |
| Range, accuracy | The range of this information would be roughly (0-100km) as an estimate of the absolute maximum someone would walk in a single day on campus. |
| Unit of measure | The unit of,measure for this specific data would be in kilometers. |
| Timing | The information will be kept on device until such a point that the user logs out or uninstalls. The Information will be kept on server until such a point that the user deletes their account. |
| Relations to other | This information relates to fitness information because the distances travelled during navigation are to be used in calculating the number of steps made and is crucial in the calculations of calories burned and other fitness statistics that are calculated. |
| Data format | The format,of this would be stored in a float value as to allow accuracy of distances,travelled and would be transferred over JSON as to integrate with our chosen, technology of using Cordova. |

Table 3: Fitness Information

| Name | Fitness information |
|---|---|
| Description | This is information that is calculated using user information and navigational information and is presented to the user as statistical data of fitness. |
| Source & Destination | Source: Application/Device, Destination :Server |
| Range, accuracy | The range of this data is : DayStep(0-100,000) , DayCalories(0-10000) DayHeight(0-10000) as maximum that one would be able to textbackslash walk, burn or climb in a single day. |
| Unit of measure | DayStep: Integer, DayCalories: Calories, DayClimb: Meters |
| Timing | The information will be kept on device,until such a point that the user logs out or uninstalls. The Information will, be kept on server until such a point that the user deletes his/her account. |
| Relations to other | This information is related to user information as the user information is required to determine the fitness statistics. It is also related to navigational data because the distances measured in navigational data are used for the step count and the calories burned as well as determining how high the user has climbed in that day. |
| Data format | This information is related to user information as the user information is required to determine the fitness statistics. It is also related to navigational data because the distances measured in navigational data are used for the step count and the calories burned as well as determining how high the user has climbed in that day. |

### 2.0.2 Performance Requirements

1. Tracking Information Accuracy
   The tracking information that is obtained from the Navigation module needs to be as accurate as possible. This will ensure that the information calculated by the fitness module will be accurate and ensure that the user is presented with information that is not false.

2. Server Storage
   The information obtained from other modules and calculated inside the fitness module is to be stored on the server hosting the NavUP application. This means that the server will need to store certain fitness statistics and health information for roughly 60 000 users. The information will need to be able to be stored in a concurrent manner as to not create performance delays during application use.
   By storing the fitness information on the server rather than keeping it on the hand-held device, we avoid complications whereby the user might uninstall and reinstall the application. Situations whereby the user gets a new phone or performs any actions where there may be data loss can be dealt with in this manner because then information can be re-downloaded.
   By storing fitness information on the server we can also provide a future integration where health-insurance companies could link to the database and use the information to provide some form of a reward system for the user based on his/her fitness achievements.

3. The User Experience
   All calculations with regards to fitness statistics and other fitness information is to be done on the device itself rather than using the application server. This will relieve the server of traffic and avoid a congested wireless network on campus. The user experience, with regards to local device calculations needs to be perceived as

smooth and not be delayed by the calculations.

4. Communication transfer from phone to server
   Information transfer between the phone and the server is required to happen in a timely manner when the user requests to calculate fitness information. If the information were to take too long to be retrieved from the server to the phone then the user would have to wait longer than expected to see the fitness information. With these requirements of fast data transfers there becomes an inherent requirement with regards to the data being sent and retrieved. The data would have to be stored in a format of minimal size as to optimize the fore-mentioned process.

5. Reporting of fitness information
   The reporting part of the fitness module would need to be able to summarize, format and display certain information in a timely manner as to not delay the user interface thread. This would occur when the user selects the option to view his/her fitness information.

## 2.1 Design Constraints

### 2.1.1 Software Constraints

1. Device Accelerometer Support:
   Given that a device does has an accelerometer, the device manufacturer must provide a software interface to interact with the device's accelerometer.

### 2.1.2 Hardware Constraints

1. Accelerometer:
   Not all devices models are made equal. Some older devices do not have accelerometers to track steps with. The accuracy of accelerometers also vary from different device manufacturers.

2. Battery Life:
   To track steps continuously with a background process will greatly impact the battery life of devices with small battery capacities.

## 2.2 Software System Attributes

### 2.2.1 Reliability

The fitness module must be able to accurately and reliably track movement through the devices accelerometer. This accuracy and reliability ensures that the calculation of steps, calories and other health related data is accurate. The accuracy and reliability requirements are to ensure that milestones and awards are distributed accurately.

### 2.2.2 Efficiency

In order for the application to efficiently count steps and calculate other health metrics, the fitness module needs efficient algorithms to calculate the steps and other health metrics quickly from the data it receives from the accelerometer.

### 2.2.3 Portability

The fitness module should be able to work on both iOS and Android devices. There should also be no discernible differences between the fitness module on iOS and the fitness module on Android devices. They should also yield the same the same health metrics.

### 2.2.4 Coupling

The fitness module should also integrate with the user module. It should be able to request the users age and name. The fitness module should also be able to write data such as the users height and weight. Minimal health metrics should also be sent to the user module to be stored on the server.

## 2.3 Class Diagram

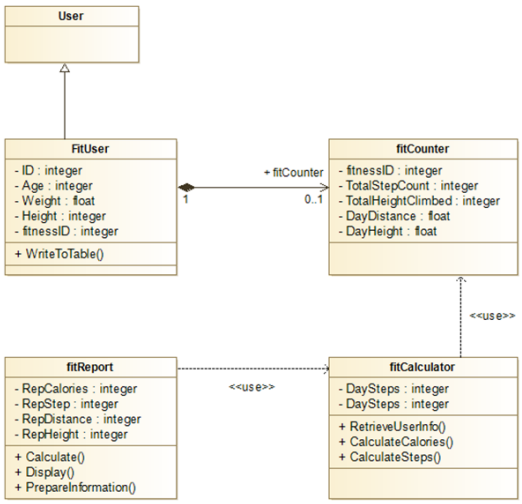


Figure 2: Fitness Class Diagram

## 2.4 Design Patterns Used

**Facade** This design pattern provides a simple interface to use the fitness module. The FitUser class inherits from the user class but also provides an interface for the rest of the Fitness Module. Without this interface the other components of the system (fitCounter, fitCalculator, and fitReport) would not be able to execute using the original user class. The fitCounter, fitCalculator, and fitReport abstract the more complex functions away from the fitUser. They are clients to the fitUser class.

## 2.5 Activity Diagram

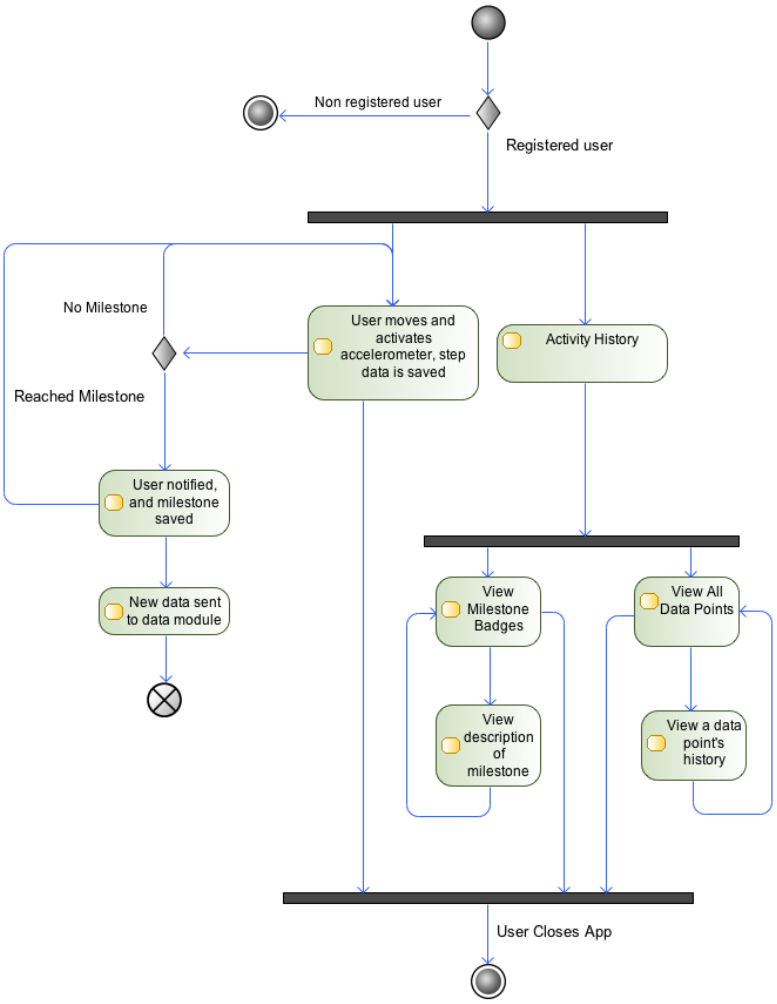


Figure 3: Fitness Activity Diagram

## 2.6 Sequence Diagram

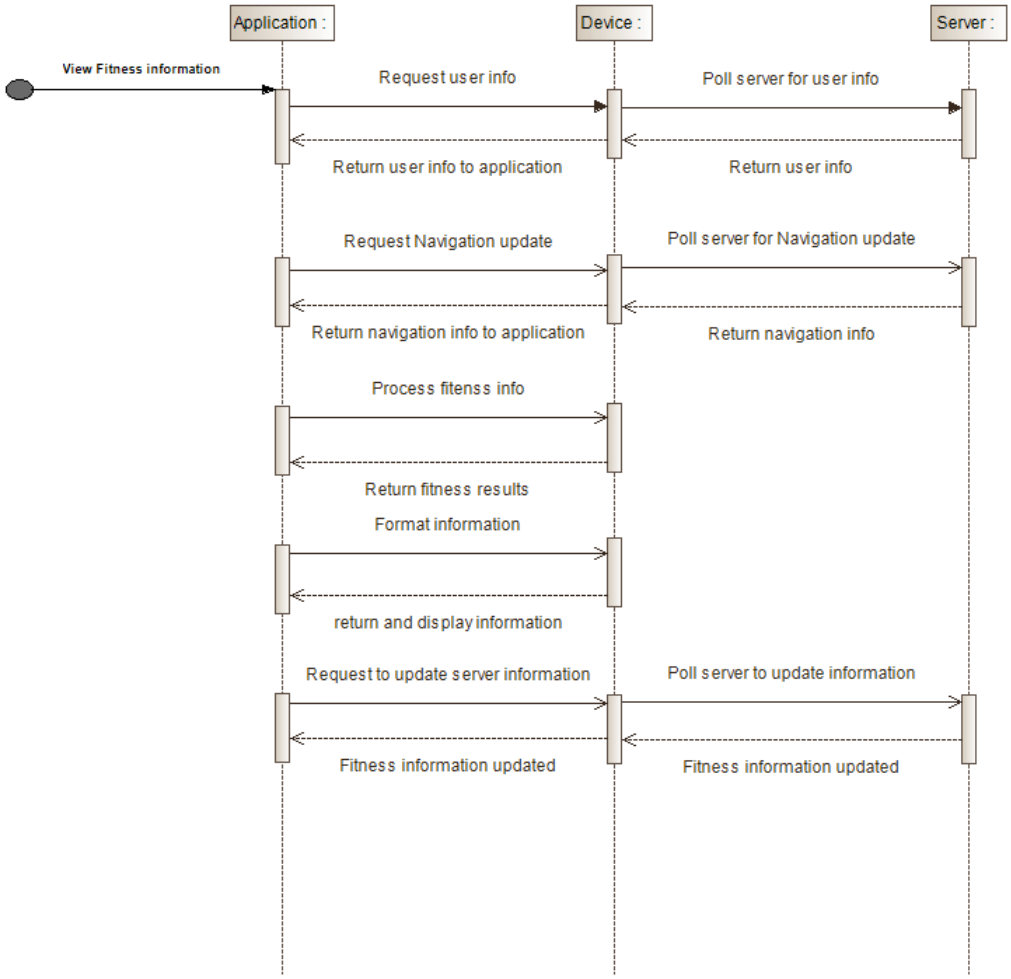


Figure 4: Fitness Sequence Diagram

## 2.7 State Diagram

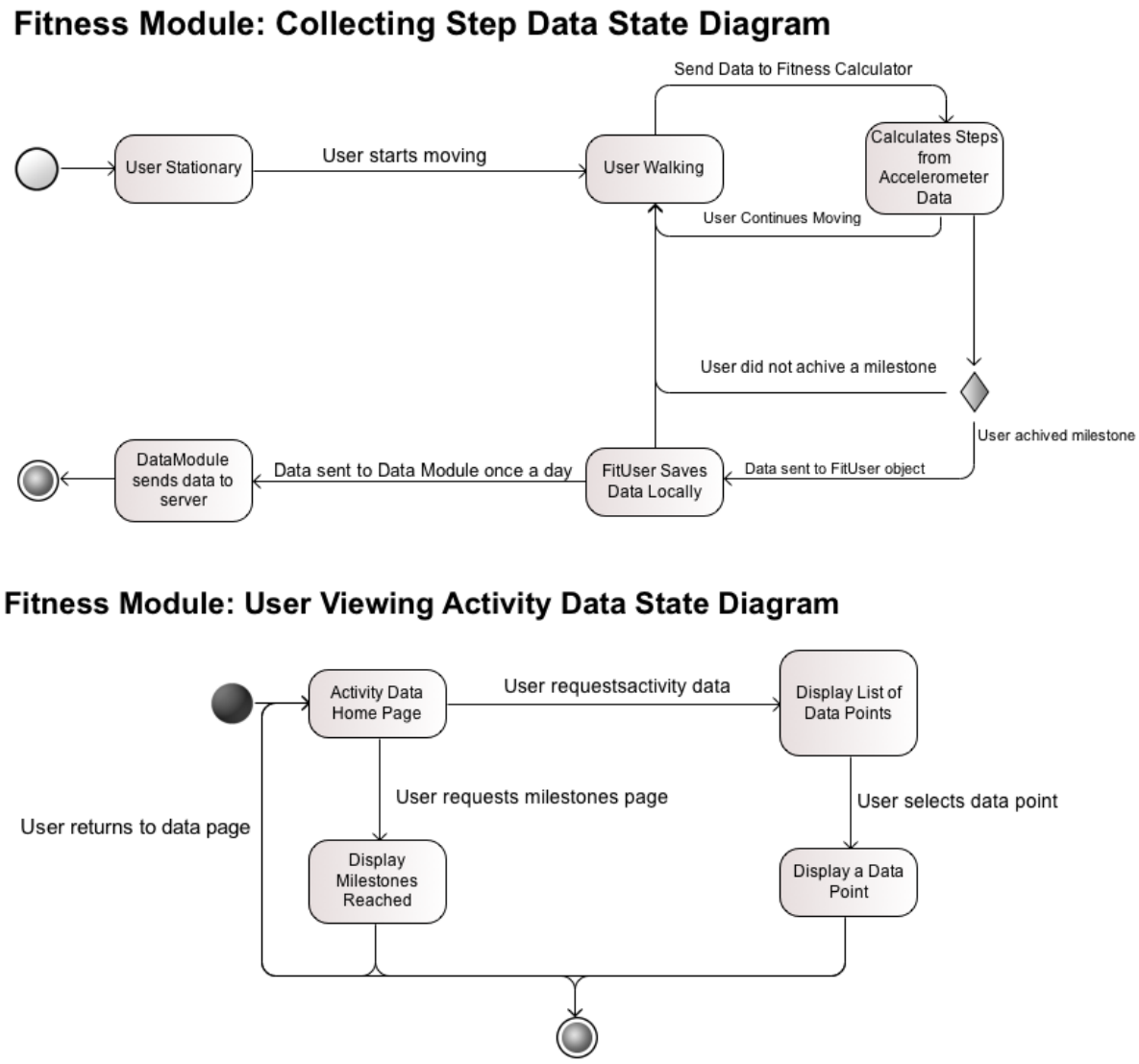


Figure 5: Fitness State Diagram

## 2.8 Use Case Diagram

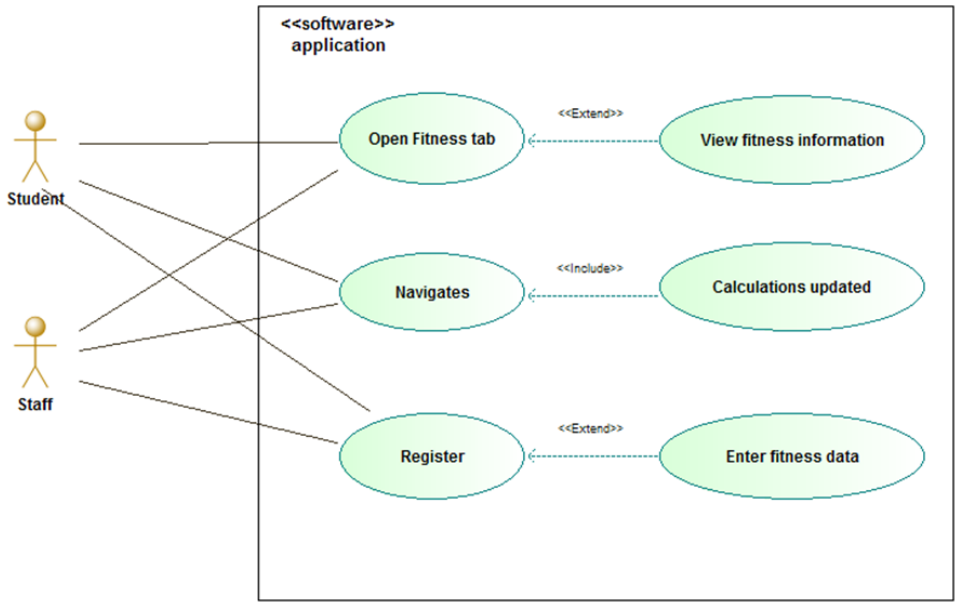


Figure 6: Fitness Use Case Diagram

# 3 Navigation Module

## 3.1 External Interface Requirements

**These requirements relate to both interfacing with other subsystems within the UP Nav system as well as interfacing with the hardware that the application will be deployed on.**

- To "Notifications" module will be used to give directions to the user during navigation in the form of push notifications.
- The "Points of interest" module will be seen by the Navigation module simply as destinations to be navigated to or as a current location.
- The GIS module will be used continually throughout navigation and as such is included in most of the UML diagrams, because of this tight coupling, implementation of the system needs to be done in a way that ensures easy communications between these two subsystems.
- Interfacing with the user's mobile device will be done with a cross platform API, explained in the "Technologies" section.

## 3.2 Performance Requirements

- Immediate response in the form of a notificaiton if user goes off route
- User can choose if application can continue navigation in the background even if the application isn't currently open. (Optional background data usage)
- "Fuzzy searching" when user is looking for a destination and misspells a word the application can still recognise the intended destination and navigate to the correct place.
- Navigation seamlessly continues when user suddenly changes from using campus wifi to using mobile data. (Interaction between GIS module and Navigation module is important here.)

## 3.3 Design Constraints

- The use of design patterns is crucial to object oriented software engineering but we are limited in the types of design patterns we are able to use within the subsystems due to the requirements imposed upon the system as a whole.
- The Navigation subsystem receives critical information from the GIS subsystem to ensure that the route calculated is correct and to recalculate the route in real time if the user goes off track. The coupling between these two modules will be high and might affect the implementation of other subsystems that also rely upon GIS information.
- Our application will be deployed across multiple platforms using different GIS implementations forcing the Navigation module to be able to perform uniformly when receiving coordinates in different formats or encodings.
- Users will want to limit the amount of data transfer occurring when using the application, the only way to implement this is to restrict communication with the server and database which could impact navigation efficiency and user experience.

## 3.4 Software System Attributes

- High cohesion with low coupling allowing for easy addition, removal or replacement of modules.
- Polymorphism will be implemented as means of specifying object behaviour when the user imposes restrictions upon the base behaviour of the object.
- All source files will be written in an Object Oriented capable language.
- We will need to interface with the user's device at a level capable of accessing push notifications and GIS data.

## 3.5 Class Diagram

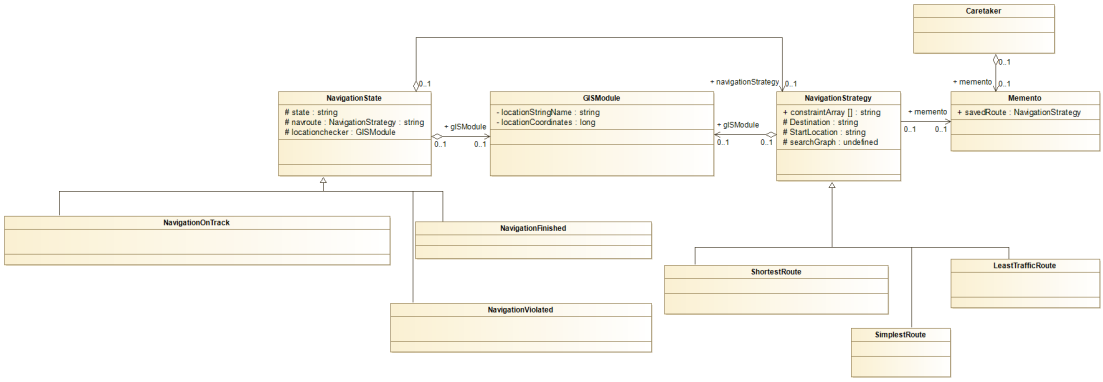


Figure 7: Navigation Class Diagram

## 3.6 Design Patterns used

**Discussion of design patterns implemented in UML diagrams:** Memento: To save a route for future use

Splitting the functions of the Navigation module essentially leaves us with two structures within the module. An internal state that is constantly changing as well as a means of navigations (different algorithms to achieve navigation to destination). Keeping this in mind I decided to use the State and Strategy design patterns, these design patterns both function on the same basic principal of modularity and separation of concerns. Additionally, because the Strategy design pattern represents what is essentially a list of directions, if the user decides to save their route so that they may return to it later, the Memento design pattern was included to ensure sensible safekeeping.

There are three possible states, namely: "Navigation in progress", "Navigation has been violated" and "Navigation exited successfully". Each one of these states interacts with the chosen strategy in a different way. which is why there is aggregation between the State and Strategy classes.

The Strategy class and its subclasses will deal with user constraints by keeping an array of said constraints and constructing a graph using information gathered from both the GIS Module as well as user input to ensure graph traversal will result in a desirable route.

## 3.7 Activity Diagram

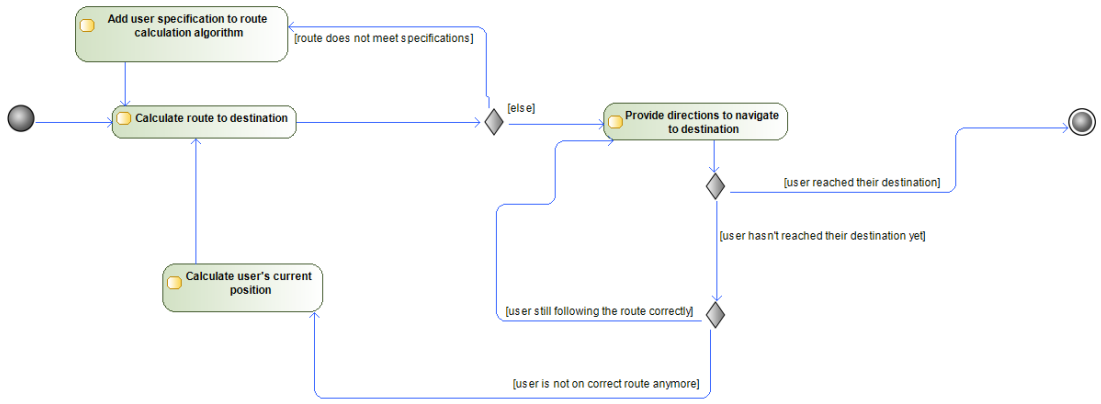


Figure 8: Navigation Activity Diagram

## 3.8 Sequence Diagram

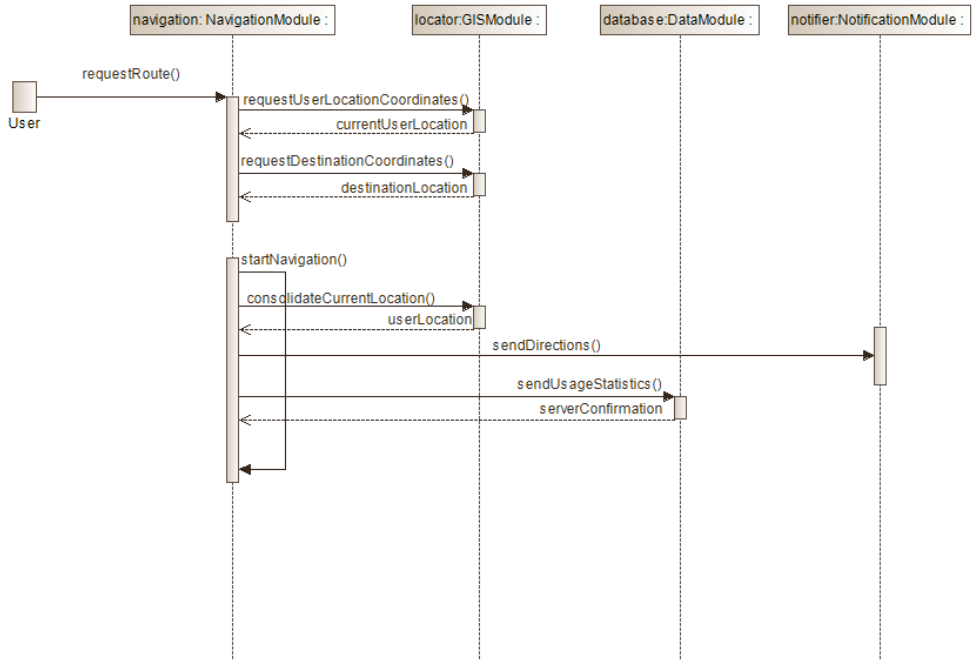


Figure 9: Navigation Sequence Diagram

## 3.9 State Diagram

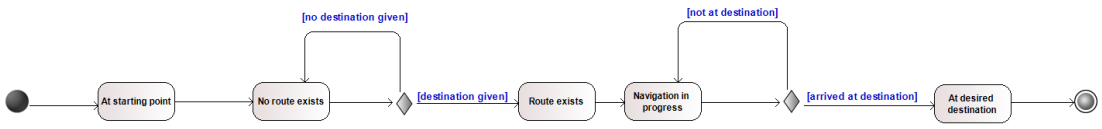


Figure 10: Navigation State Diagram

## 3.10 Use Case Diagram

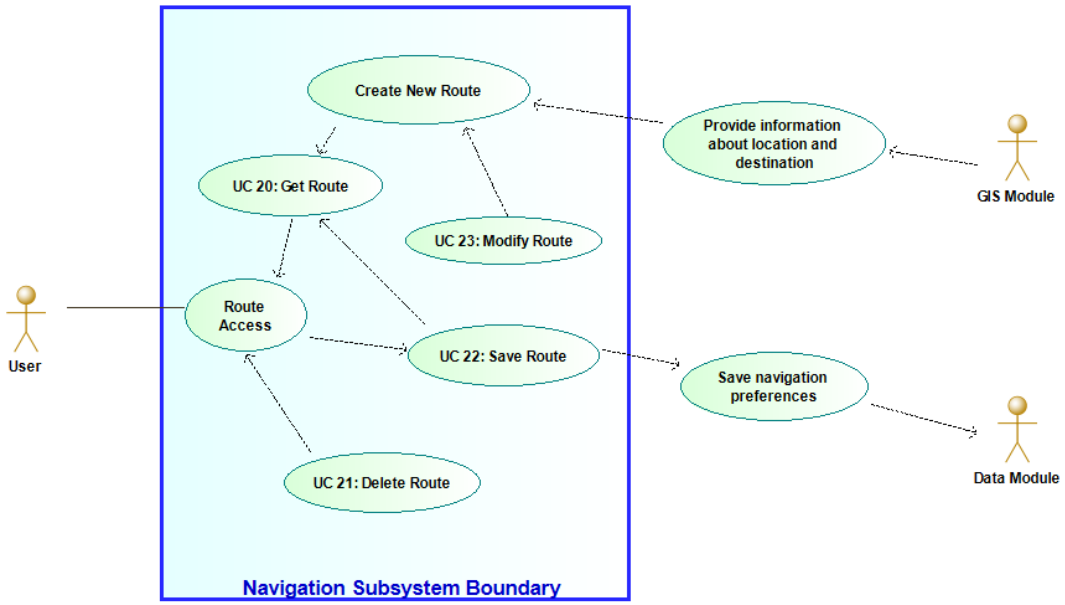


Figure 11: Navigation Use Case Diagram

# 4 Points of Interest

## 4.1 External Interface Requirements

### 4.1.1 System Interfaces

The NavUp systems interfaces include locating and retrieving site information (building names, addresses, etc.) based on the information retrieved from the Navigation module. The site information will be used for gathering and maintaining any information of interest with regard to the searched location. The acquired information about the location will be stored and pushed as a notification to the users interface where users can view and read it.

### 4.1.2 User Interfaces

The system will allow users to find locations, so the route guidelines to the destination will be displayed on the users screen interface, route guidelines include a pin-point indicating users current positon, colored route path to the destination, and the pin-point indicating the destination. So, once the information of interest with regard to the location has been acquired, the information will be sent to the user as a notification, and the notification may be in a form of SMS, E-mail, or a push notification. The user will have an option to alter how the notifications are received, that is, either as an E-mail, push notification or an SMS.

### 4.1.3 Hardware Interfaces

The Wi-Fi routers and mobile phones are the only primary hardware interfaces that may be required for the points of interest module. Mobile phones will be used for all the user interfaces and the functionality of the NavUp system and the Wi-Fi access points as a reference for detecting locations both indoors and outdoors.

### 4.1.4 Software Interfaces

The NavUp will primarily run on mobile phones. Therefore, for compatibility requirements, the NavUp system should be hybrid, that is, it has to be compatible across most, if not all ranges of mobile smart phones and mobile operating systems, that is either Android OS, iOS, or Microsoft Windows. The system can also be web-based.

### 4.1.5 Communication Interfaces

The system will frequently communicate with the campus map database, servers and the mobiles GPS to get locations and directions through Wi-Fi networking. Any acquired information of interest with regard to the desired location may be retrieved from the database through servers. The systems communication interface may also include web services.

## 4.2 Performance Requirements

### 4.2.1 Track creation/ Route update

The NavUp system should also keep track of the user's current locationso to update the route displayed on the user's screen interface, however the system should not necessarily update the route every after 1 second, the route update can be in an interval of 7 seconds per update.

### 4.2.2 User login response time

The users will be required to login, provided that the user has entered correct credentials, the NavUp system should take no longer than 6 seconds to provide full system access to the user. However, this may be dependent on the internet connection bandwith.

### 4.2.3 Positional accuracy

Accuracy is one of the vital performance required for the system. The NavUp system should be accurate enough to get the users current position, if for example the user is in the building (indoor) with multiple floors, the system should not necessarily determine the exact venue, but it should be in range and in real time.

The system should have a good ranging accuracy. The accuracy of the system is defined as the solution error, and the accuracy of the system should conform to position range and the PVT (Position, Velocity and Time). This will be useful in terms of points of interest where accuracy is very important when determining locations.

### 4.2.4 Proximity of notifications

The user will recieve any information of interest for the locations he/she visits, the amount of time it takes to recieve the notification shouldn't be long, this is to say, the system should be responsive enough to an extent that when the user gets to the desired location, it takes no longer than 25 seconds to recieve the notification.

## 4.3 Design Canstraints

Getting the current location will be difficult using WiFi because it is hard to triangulate the position of one device due to the face that the WiFi signal will not give you an accurate representation of where the device is. At best the current location might vary by 10m to 15m depending on the objects it had to pass though e.g. wall or roof. If the user cannot get WiFi signal it will be hard for the user to get the current location because then it will be using the users data and the user might be reluctant to allow that.

## 4.4 Software System Attributes

### 4.4.1 Accuracy

The data of the points of interest module needs to be accurate. The saved location of a point of interest needs to be accurate for the navigation module to navigate the user to the point of interest. The information about the point of interest also needs to be accurate to avoid misinformation.

### 4.4.2 Reusability

The points of interest module should be able to support multiple different types of locations like lecture halls, museums, restaurants, etc. The module should be able to add new types of locations without the need for extending the app.

### 4.4.3 Usability

A point of interest should be easy to find on the campus map as well as being searchable in the database. The icons on the map should also be relevant and intuitive for the different types of points of interest. The location information should also be easy to find once the user has accessed the point of interest.

### 4.4.4 Maintainability

The points of interest needs to be maintained. When there are new points of interests they need to be added to the system. If a point of interest has new information it needs to be updated in the system.

## 4.5 Class Diagram

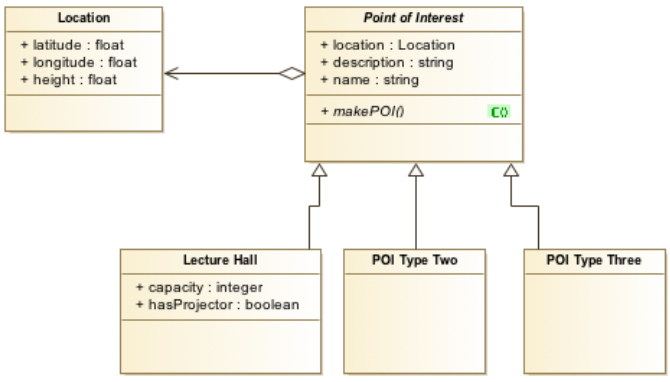


Figure 12: POI Class Diagram

## 4.6 Design Patterns Used

**Factory Method** The points of interest module needs to be able to implement different types of points of interest. The factory method defines an interface for creating an object, but lets subclasses decide which class to instantiate. Factory Method lets a class defer instantiation to subclasses. Because instantiation is deferred to subclasses it is possible to have many different types of points of interest classes which inherit from the abstract Point of Interest class.

## 4.7 Activity Diagram

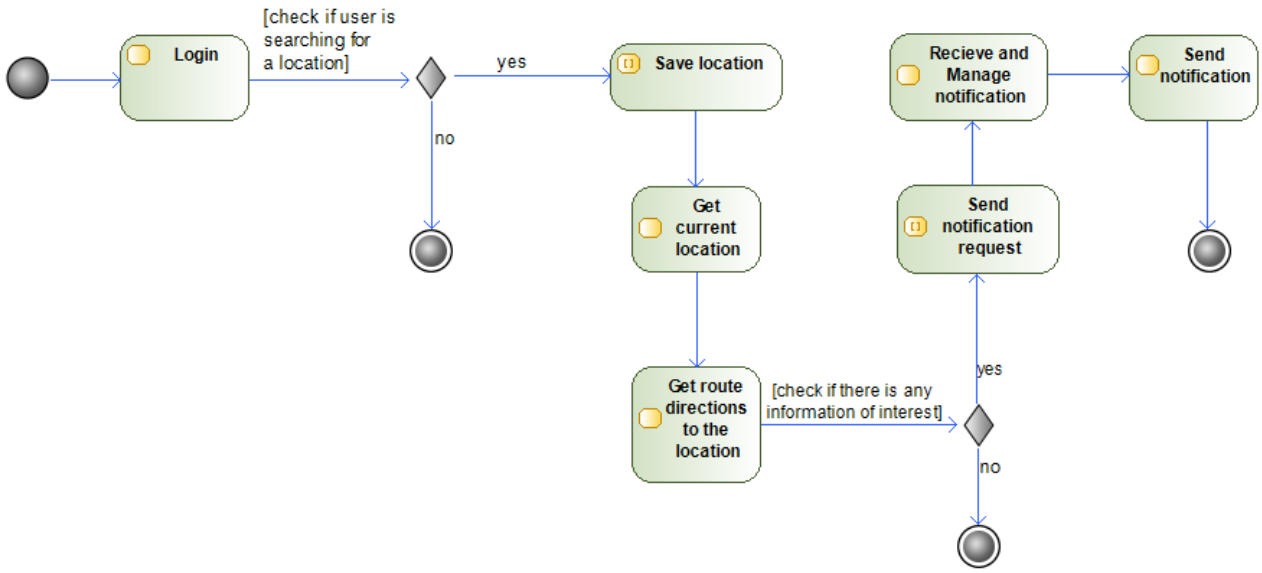


Figure 13: POI Activity Diagram

## 4.8 Sequence Diagram

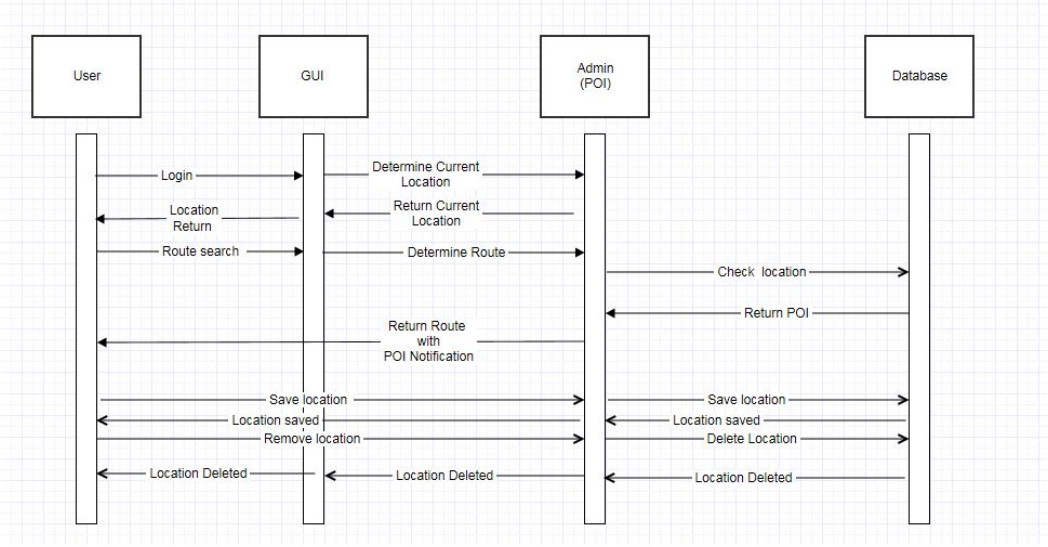


Figure 14: POI Sequenc Diagram

## 4.9 State Diagram

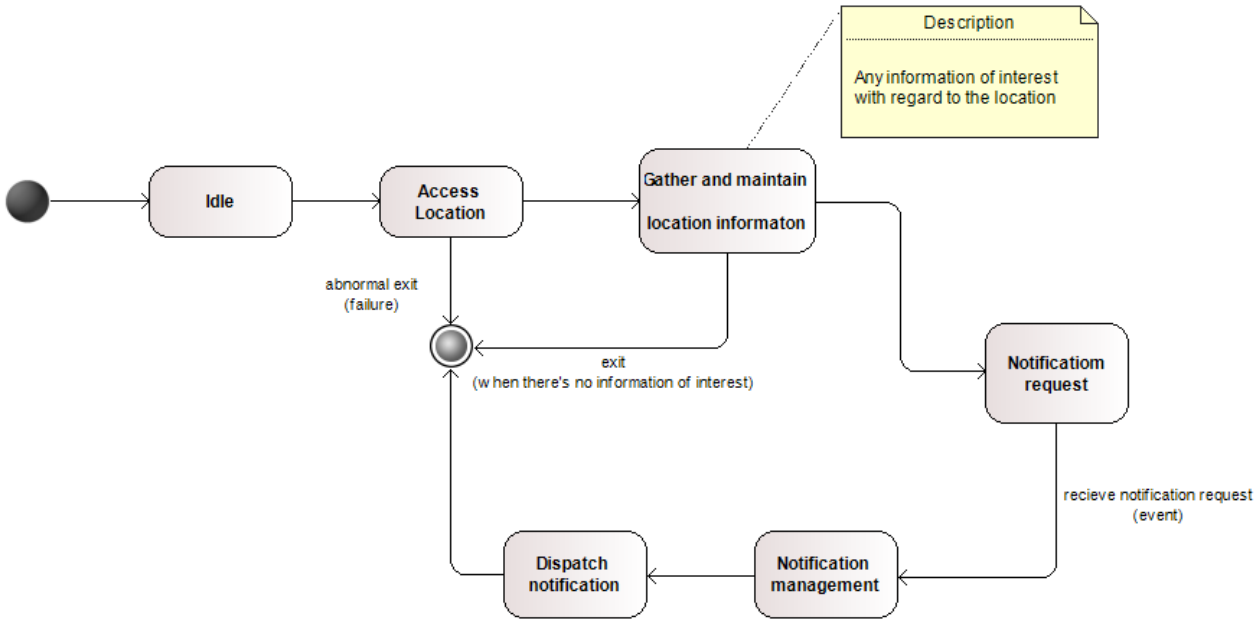


Figure 15: POI State Diagram

## 4.10 Use Case Diagram

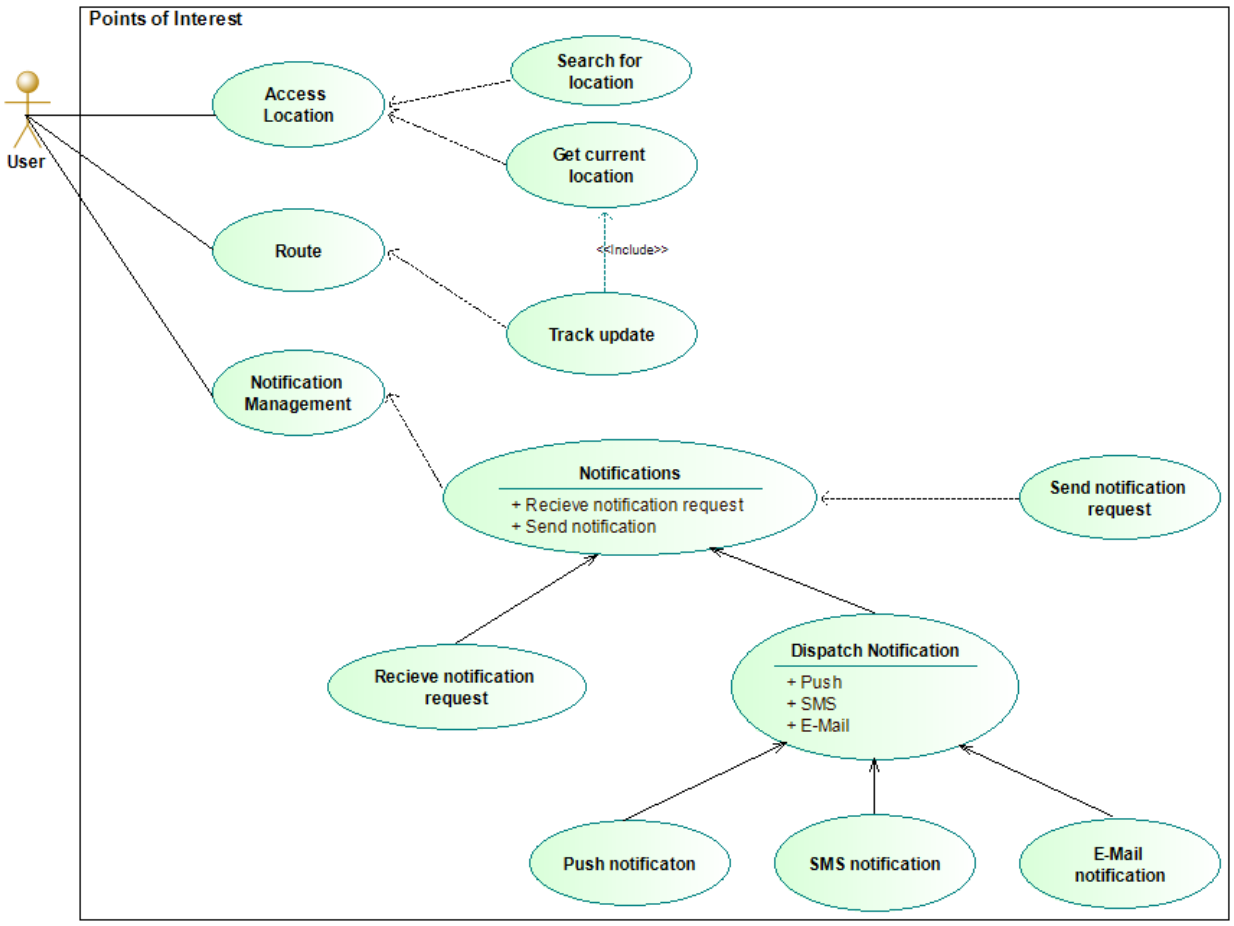


Figure 16: POI Use Case Diagram

# 5 Users Module

## 5.1 External Interface Requirements

- The user module interface will provide the user with a clean and smart design, which will make it easy for the user to access the system and logon or register with no unwanted constraints.
- The user will only be allowed to access his/her application via a smartphone or tablet. The software will work on any operating system.
- The application will be able to fit all screen resolutions with screen rotation capabilities.
- When the user logs in or register, his/her information will be sent to a server which will then connect to a database that will either validate the users credentials or add the credentials to the system.
- To meet disability requirements, users can use the accessibility function that is built into almost all smartphones and tablets.

## 5.2 Performance Requirements

- The user system will get from 20 000 to 40 000 login requests a day. This branches of into registered students logging into the system more than once a day and guests registering and logging into the system
- The user system will have at most 1000 guests using the application on the campus but this value varies in the beginning of the year because most first year have not registered within the first few weeks of the academic year.
- The user system might slow down depending on how the database is implemented and how big it will be but the design patterns used help improve database efficiency and improves server response time.

## 5.3 Design Constraints

1. Query caching
   In order to avoid traffic requests to the database and to speed up database , query caching is crucial. AdoDB provides powerful caching system which can implemented in the navUP system and help improve speed regarding interaction between front end and database.

## 5.4 Software System Attributes

### 5.4.1 Security

The navUP system will need to adhere to the highest security standards because of the personal information of users that will be stored. In order to achieve security , The

navUP system needs to store users information and the passwords should be stored in the database as encrypted to remain hidden from the unauthorised users .

### 5.4.2 Performance

The system should ensure that basic operations like user registration , user login should be of maximum speed.

### 5.4.3 Availability

The navUP system will need to be connected to the internet to ensure that the system can communicate with the database.

### 5.4.4 Reliability

The navUP system will need to work roburstly without any failures including failure to retrieve information from the database.

## 5.5 Class diagram

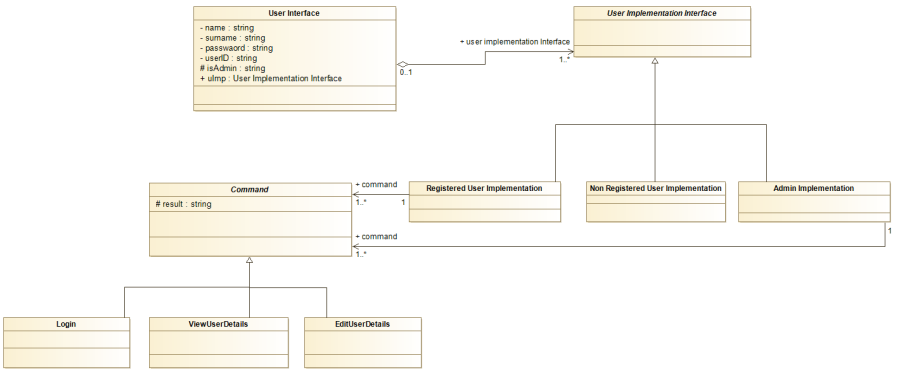


Figure 17: Class diagram for user management module

## 5.6 Design Patterns Used

In the user class diagram, I used the bridge design pattern combined with the command pattern. I used the bridge pattern because it maintains a stable client interface while allowing the implementations change. Thus, when a Non-Registered user registers then the implementation can change to registered user implementation and the client will not be affected by the changes in implementation. The client does not know which concrete implementation is being used which is good in terms of security so the client cannot damage the system. The command pattern is used to allow the for requests to be encapsulated which reduces the code size and it allows the operations to be executed in different ways. The requests can be queued and executed later which is useful in

increasing the performance of operations. By decoupling the operations is reduces the amount of request that it will have with the database thus, making it more efficient and making the other database operations run faster.

## 5.7 Activity diagram

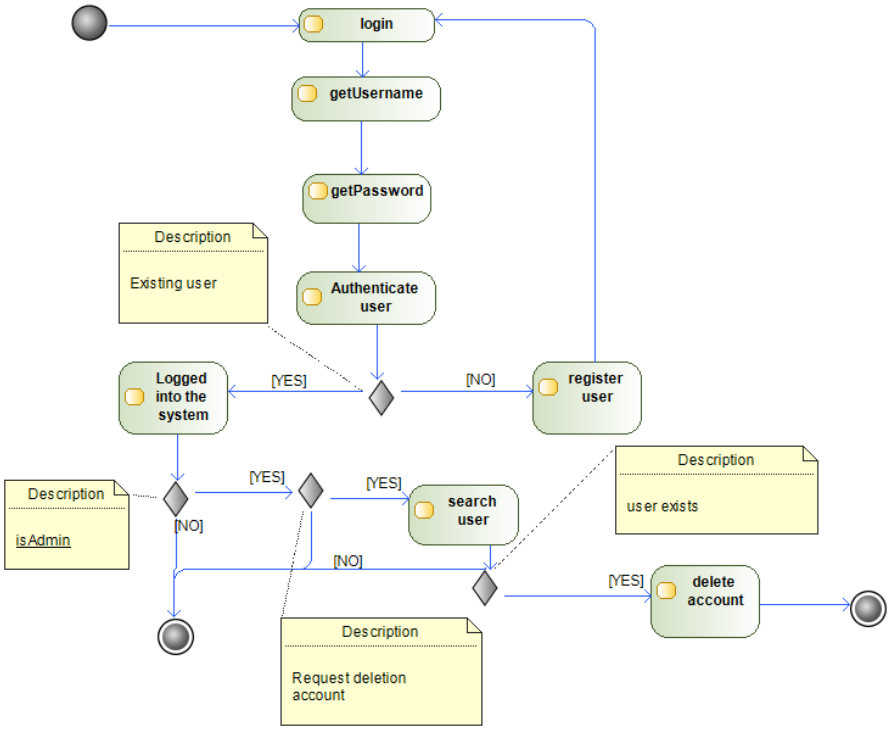


Figure 18: Activity diagram for user management module

## 5.8 Sequence diagram

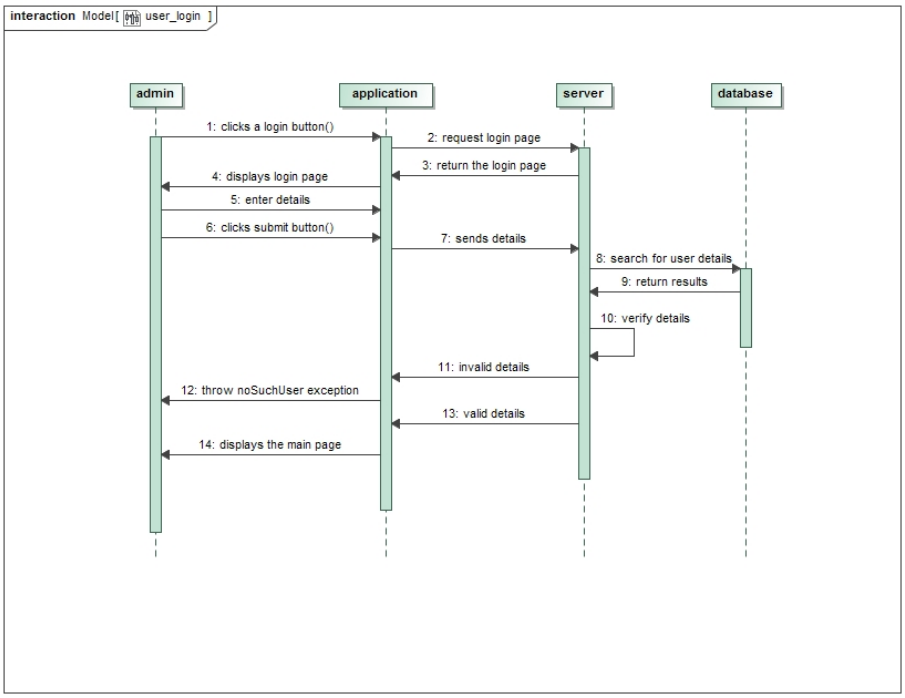


Figure 19: Sequence diagram for user management module

## 5.9 State diagram

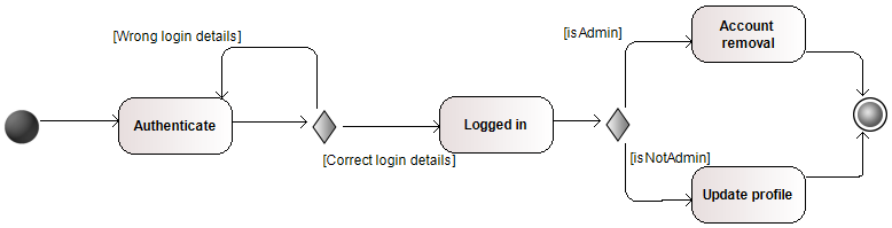


Figure 20: State diagram for user management module

## 5.10 Use Case diagram

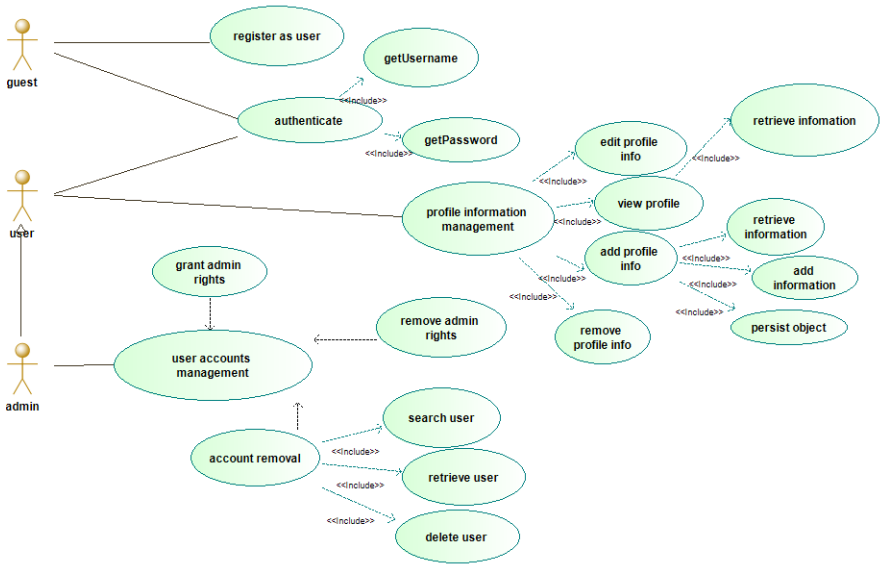


Figure 21: Use case diagram for user management module

# 6 Technologies

For the NavUP system to be available to as many users as possible the system needs to be available on the two major mobile operating systems, iOS and Android as well as having a web interface. For the system to work on all three of these platforms we need a framework that works on all of these platforms. The main advantage of having only one framework that works on all three platforms is that it minimizes the amount of code required to make the app compatible with these three platforms. The latest distribution of Cordava, PhoneGap, fits this requirement well.

## 6.1 User Application

PhoneGap uses web technologies to create a front-end application for iOS and Android. Because web technologies are used the front-end code can be easily ported to be able to run on a browser. To improve the interface Angular.js and Express.js will be used. Cordova also have many open source plugins available on their website, these plugins make integration with the device's hardware simpler and also provide plugins that will make the implementation of some modules simpler as they have many of the functions these modules require.

## 6.2 Server

The server's operating system will be CentOS. CentOS is a very stable lightweight release of Linux and is primarily used for web servers. Most web hosting companies run their web servers on CentOS because of these reasons. The web server that will be used is Node.js. Node.js works best in real-time applications, because they work so well in real-time applications Node.js is perfect for the NavUP system which needs to provide real time data to the front-end application.

## 6.3 Database

We will use MongoDB as the database because it works well with NodeJS and is more secure than SQL.

### 6.4 Advantages and Disadvantages of Chosen Technologies

| Advantages | Disadvantages |
|---|---|
| PhoneGap works on all three required platforms. | PhoneGap is not native to any of the three platforms and performance can sometimes be subdued. |
| PhoneGap is free. | PhoneGap does not support all functionality that might be needed. |
| Node.js is great for real time applications | Node.js isnt consistent. Its several, development firms feel that the API keeps enhancing at frequent intervals. |
| CentOS is perfect for web servers. | CentOS is not frequently updated and therefore new software is not always available. |
| JavaScript will be the main language used, which will simplify the implementation. | JavaScript can sometimes be slow to execute. |