
NavUP
Architectural Requirements, Specifications and
Design

Compiled By

Boikanyo Modiko - u15227678
Linda Potgieter - u14070091
Marthinus Richter - u15160671
Nkosenhle Ncube - u13247914
Quinton Swanepoel - u15245510
Ruan Klinkert - u14022282

201
TEAM Tlc

1 Introduction

1.1 Deliverable

We are required to use the high level functional requirements of the NavUp system given in a separate document to identify the architectural design specifications that satisfy the identified functional requirements from our last assignment, focusing on the subsystems architectural designs.

1.2 Scope

The final objective is to create a NavUP mobile application as well as a web-based interface. The NavUP mobile app should be available on both android and ios platforms while the web-based interface will be used for administration and maintenance. The primary objective of NavUP is to allow students, visitors and staff to successfully navigate around campus in a efficient manner.

1.3 Definitions, Acronyms and Abbreviations

Term	Definition
App	Mobile application
CRUD	Create, Read, Update and Delete
GPS	Global Positioning System
TUCBW	This Use Case Begins With
TUCEW	This Use Case Ends With
UP	University of Pretoria

2 Overall Architectural Design

2.1 Overview

The development of the NavUP system takes a very modular approach, hence requiring effective communication between several modules all developed separately. This involves the implementation of several design patterns and the linking of them with one another.

2.1.1 Performance Requirements

- The system will make use of campus wifi aswell network coverage.
- Users will be able to register on the system, but also be able to use the system as a guest.
- Admin users will have control of system tasks and user management.
- Admin users should be able to set up an event which users may attend.
- Notifications for events should be pushed through to the users when requested by the admin.
- The admin user/s should have easy access to the systems GIS module.
- Admin should be able to easily add points of interest to the system.
- The points of interest module should be used by the events module when selecting a destination for the event, and also by the navigation module for when providing directions.

2.1.2 Design Constraints

- The application should be within certain size constraints as to allow the application to install on even entry level smartphones without causing storage problems.
- The application must not be memory intensive causing entry level smartphones to become slow when running the application.
- The application requires access to network/wifi coverage aswell as location data for the device running it in order to provide accurate shortest and least congested routes.
- Logging into the system and loading relevant user specific data should take place within a reasonable amount of time. This requires accessing and retrieving database information in a efficient and effective manner.

2.2 Deployment Diagram

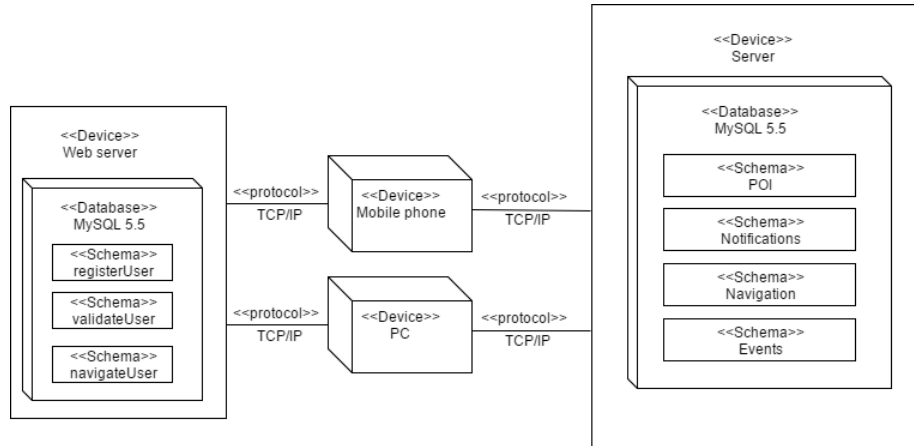


Figure 1: Deployment diagram

3 User Management Module

3.1 UML Class Diagram

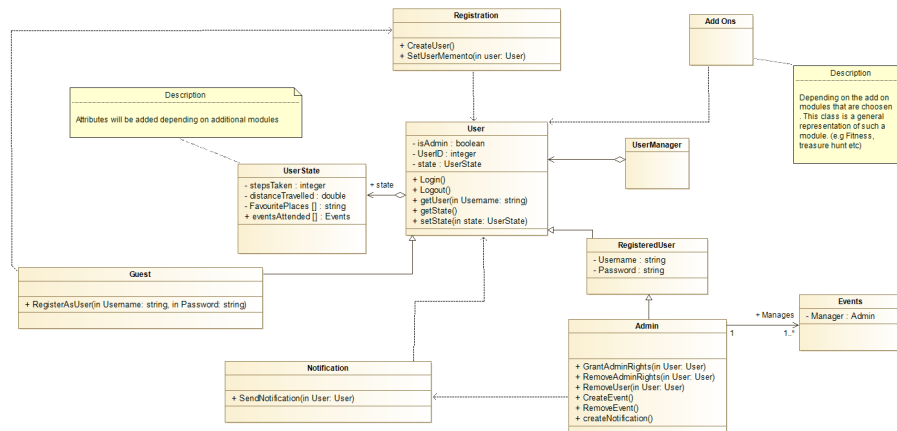


Figure 2: Class Diagram for User Management System

3.2 Use Case Diagram

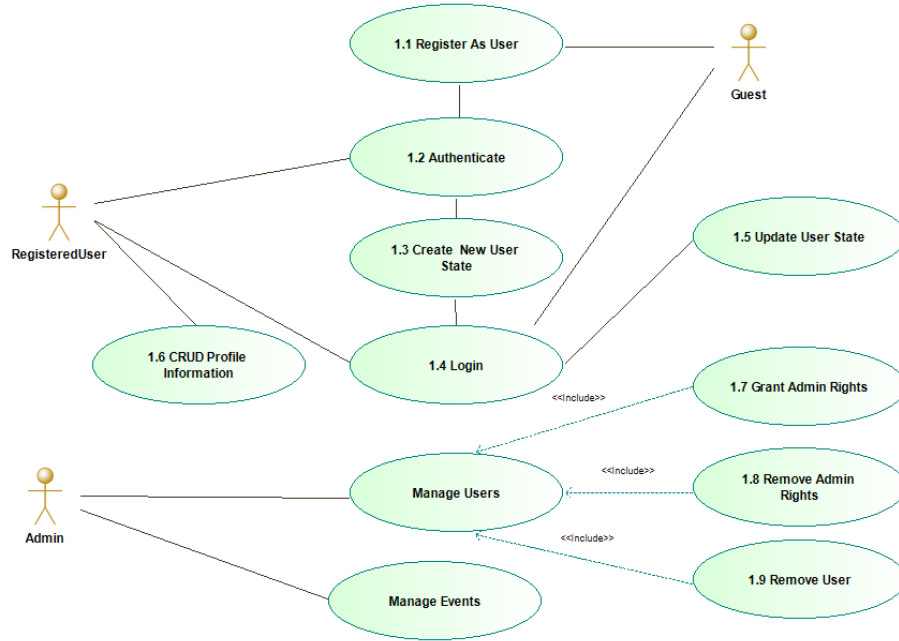


Figure 3: Use Case diagram for User Management System

3.3 Architectural Design of Module

The user management module will handle the login and registration of various users, namely Admin Users, Registered Users, and Guest Users. Guest users will still be able login into and make use of the NavUP system without registering but no data will be saved on the system for them. The system stores all registered users' details including those from other modules.

The user management system makes use of two major design patterns, Template and Memento. The inheritance between user and the types of users demonstrates the template design pattern. The registration class acts as the originator to create objects of type user in the system; the user acts as the memento storing a state object for each class; the user manager is the caretaker for the memento. This shows how the memento pattern is implemented. The UserState class is used to define the objects that will hold state data that is unique to each user, such as, steps taken, places visited, events attended and other attributes that will be brought forward by extra modules that are added to the system.

This module connects to the notifications module, events modules and other add-on modules. Notifications are dependant on the user/s to which they are to be sent and hence there is a dependency present between these modules. The events module requires an admin user to instantiate and manage the event. One admin user may manage many events, but an event may only be managed by one admin user.

3.4 Technologies Used

A MySQL database will be used to manage basic user activities such as registration, authentication, and user privileges. The database will store information regarding all users and how each user should be able to interact with the system. This database should interlink with the ios, android, and web platforms.

The admin user should be able to perform basic CRUD operations such as adding and removing users, creating events, and updating details regarding points of interest.

Server scripting such as PHP will be required to facilitate interaction between the front end web interface and the database.

3.5 Non-Trivial Implementation Tasks

4 Notifications Module

4.1 UML Class Diagram

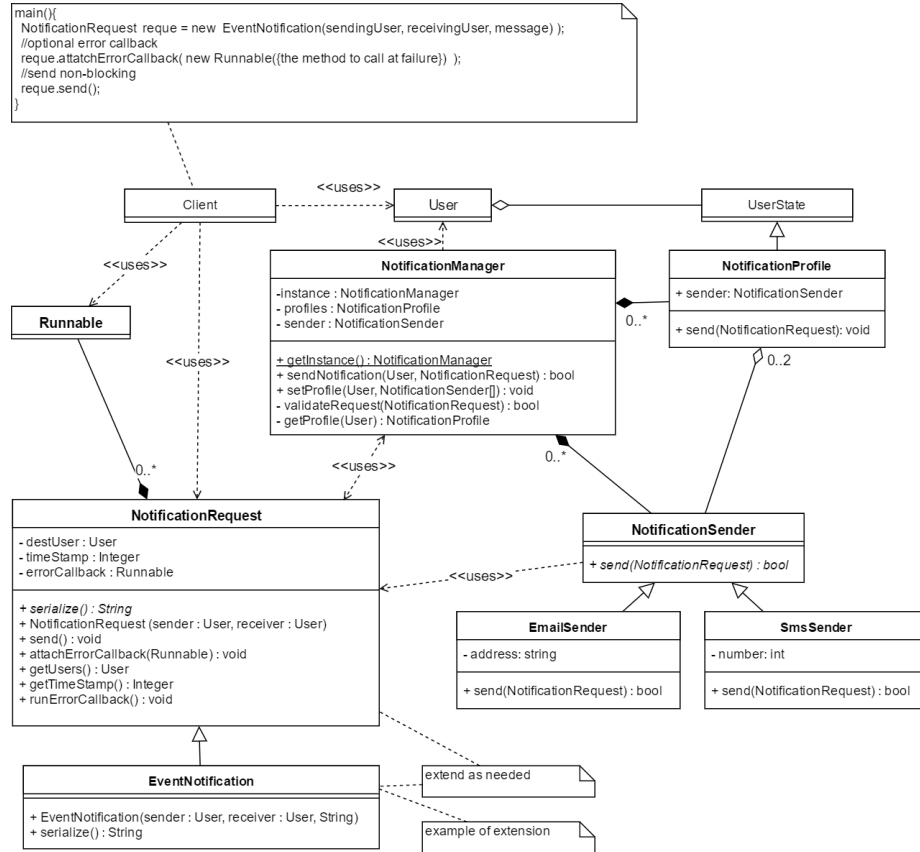


Figure 4: Class Diagram for Notifications Module

4.2 Use Case Diagram

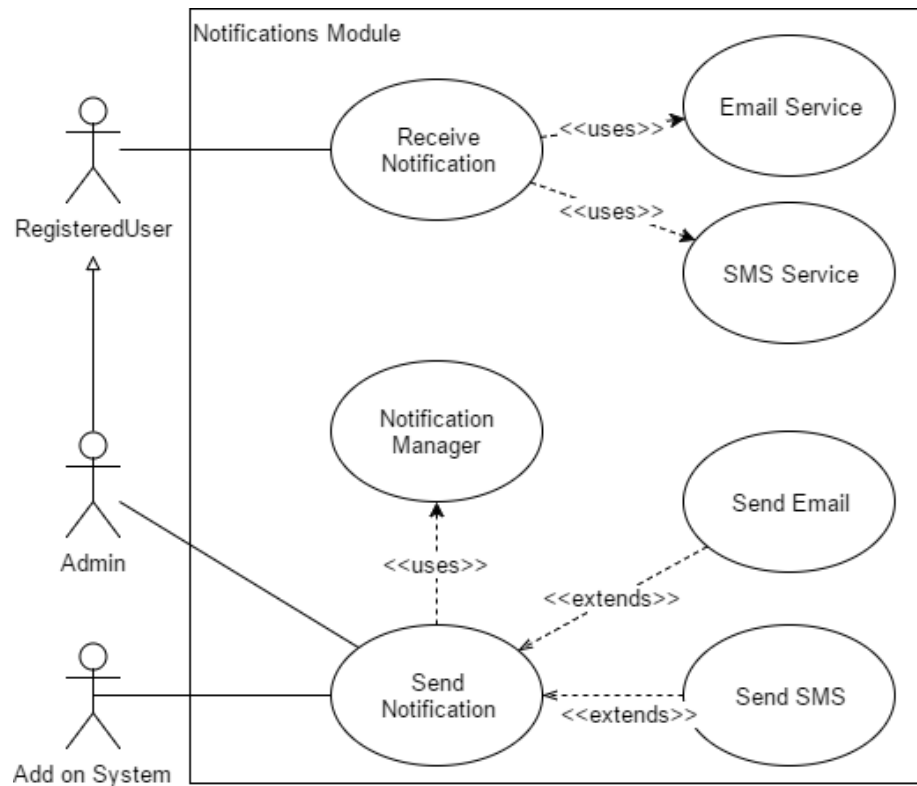


Figure 5: Notifications Request and Send Use Case

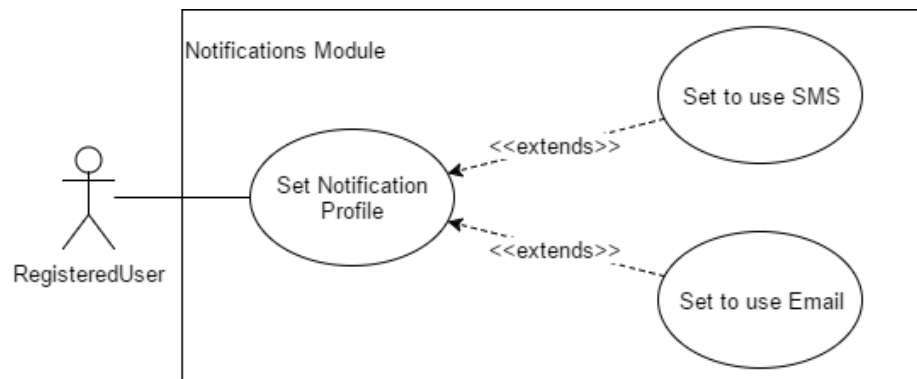


Figure 6: Notification Profile Use Case

4.3 Architectural Design of Module

4.4 Technologies Used

Email Authenticated SMTP must be used to send messages to the email server that NavUp will use. This is to prevent the email server from relaying messages that are not properly validated. Internet Message Access Protocol (IMAP) will be used to store emails, as this is most prevalent system and thus support ought to be sufficient. POP3 might be an option as emails need not be stored for extended periods after the email has been delivered. TCP/IP will be used implicitly.

SMS A Direct-to-SMS gateway will be the most appropriate service to use, encoding all messages in plain ASCII text to promote device compatibility. The infrastructure such as GSM is determined by the service provider.

4.5 Non-Trivial Implementation Tasks

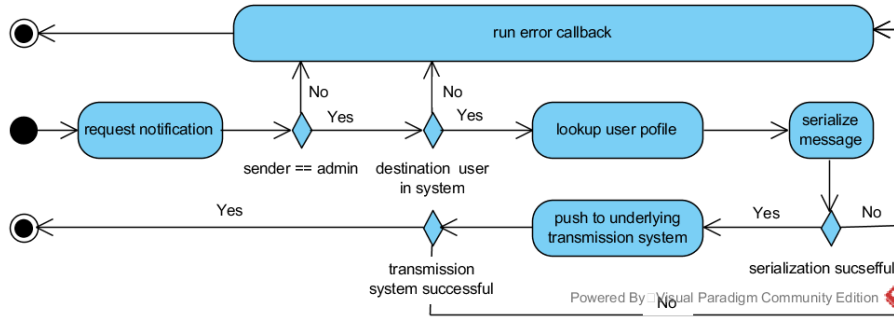


Figure 7: Activity Diagram for Notifications Module, Request Notification

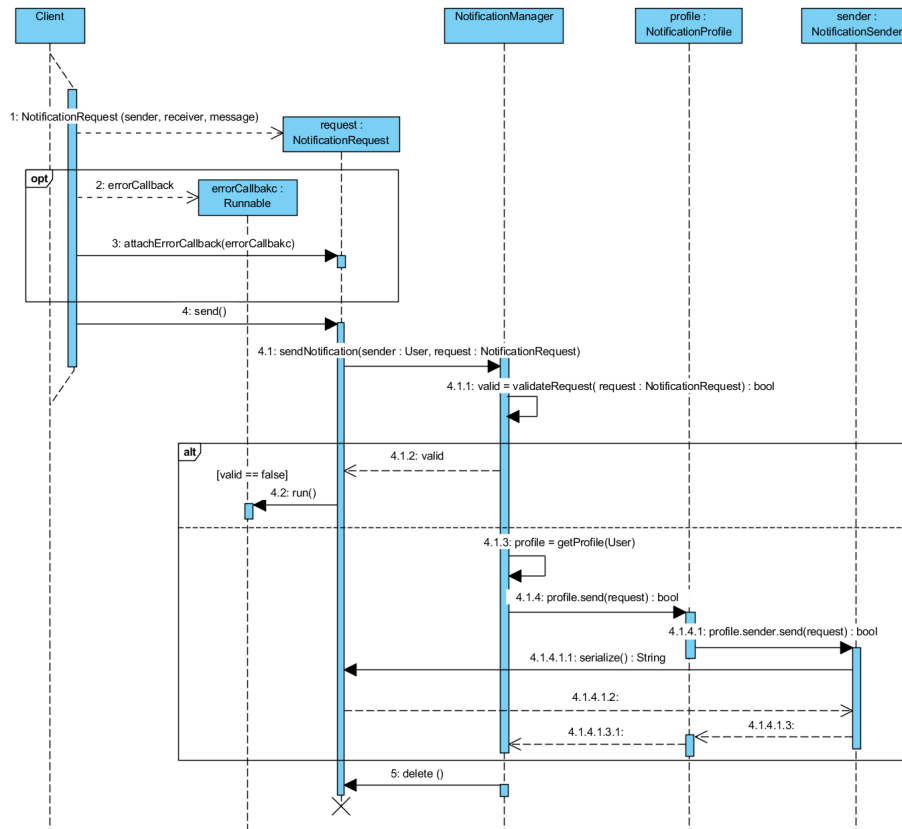


Figure 8: Notification Module - Client Interaction

5 Points of Interest Module

5.1 UML Class Diagram

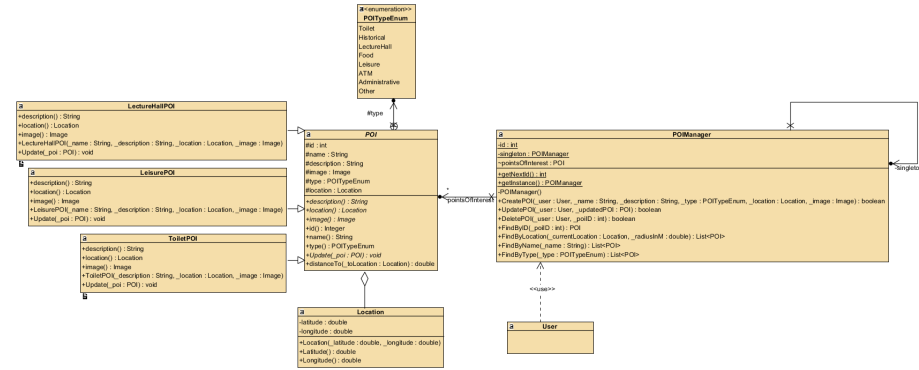


Figure 9: Class Diagram for the User management module

5.2 Use Case Diagram

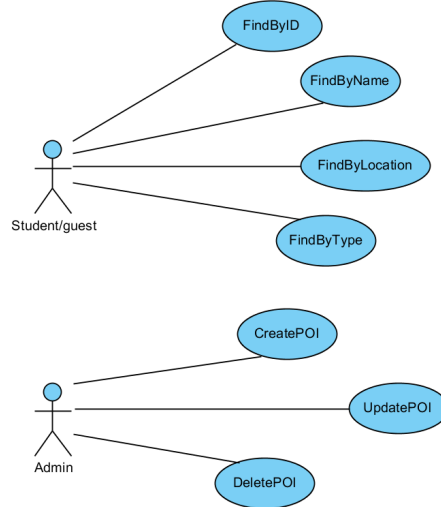


Figure 10: Use Case Diagram for the User management module

5.3 Architectural Design of Module

The Points of Interest module will be used to persist and deliver data regarding locations that a user might find of interest. It will interact with the Events

module, as well as the GIS module in order to retrieve and save its geographical location.

One can assume that different types of Points of Interest (POI) will have different fields, constraints and requirements, therefore the POI module will make use of the factory design pattern, in order to create objects without exposing the creation logic to the client and allow us to refer to the newly created object using a common interface.

The Points of Interest Manager class will make use of the Singleton design pattern to ensure that only one instance of the class exists, it will also act as the factory class to the points of interest class.

At the core a Point of Interest object will have a location, description, unique id, type, name and image. The concrete object will then be able to impose additional attributes and constraints, such as Gender for bathrooms for example.

Finally, the abstract point of interest class will be equipped with a stringify function to get a JSON representation of the object.

5.4 Technologies Used

It can be assumed that different types of Points of Interest will have different fields, constraints and requirements, therefore a traditional relational database will not be suitable for this type of data. Hence we will rather go for a document-oriented database like MongoDB, as the database server.

The Points of Interest CRUD functions will be exposed as Simple Object Access Protocol (SOAP) web services, using ASP.NET core and Entity framework carrying a JSON payload.

The logic of the module will be written in .NET.

5.5 Non-Trivial Implementation Tasks

The point of interest module will be called in two ways. Firstly, by administrative user who would be able to add, remove and update points of interest around campus. Secondly, students and guests will be able to use it, by searching for specific locations by name, type of location, specific areas etc. Lastly the points of interest classes SearchByLocation function will be called by the navigation module, as the user navigates campus.

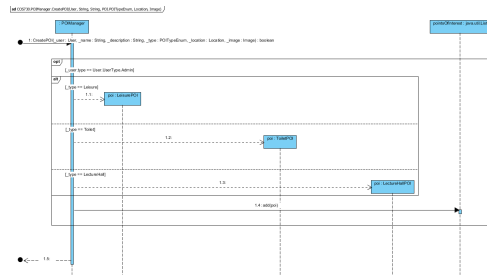


Figure 11: Sequence Diagram for the CreatePOI function

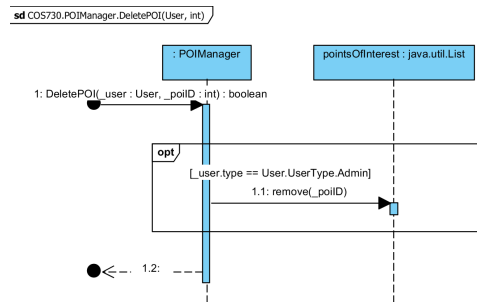


Figure 12: Sequence Diagram for the DeletePOI function

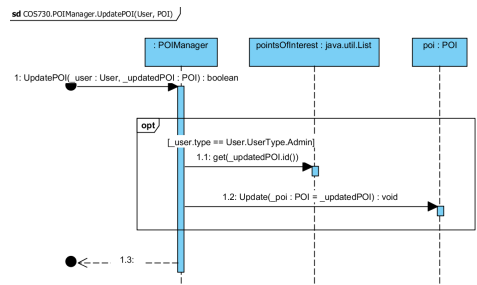


Figure 13: Sequence Diagram for the UpdatePOI function

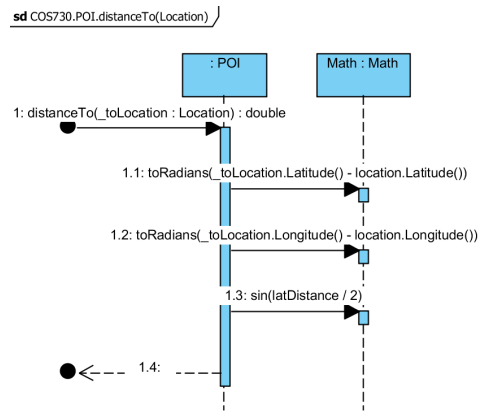


Figure 14: Sequence Diagram for the DistanceTo function

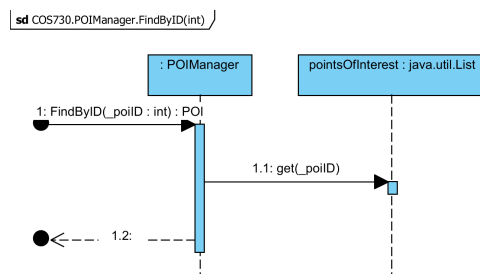


Figure 15: Sequence Diagram for the FindByID function

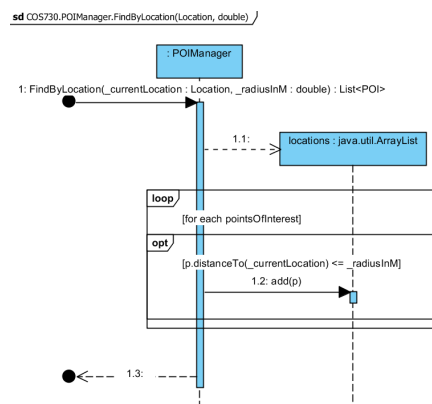


Figure 16: Sequence Diagram for the FindByLocation function

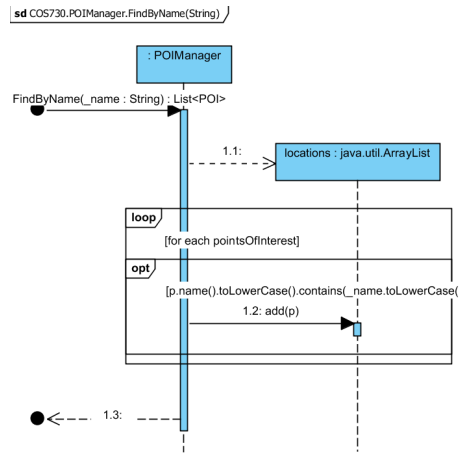


Figure 17: Sequence Diagram for the FindByName function

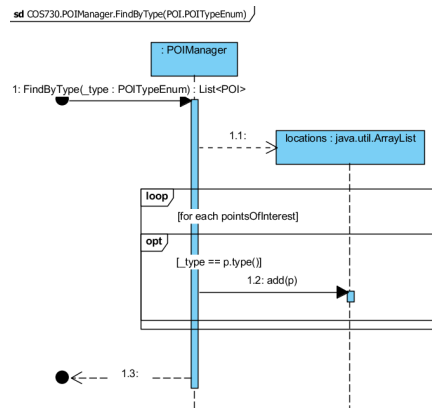


Figure 18: Sequence Diagram for the FindByType function

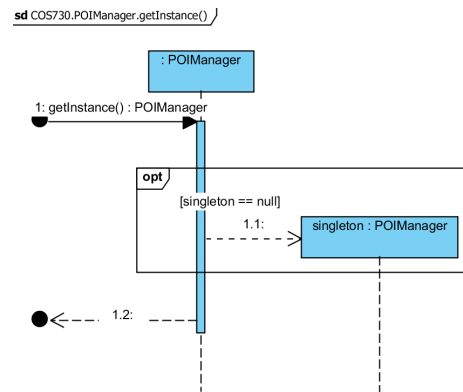


Figure 19: Sequence Diagram for the `getInstance` function

6 Events Module

6.1 UML Class Diagram

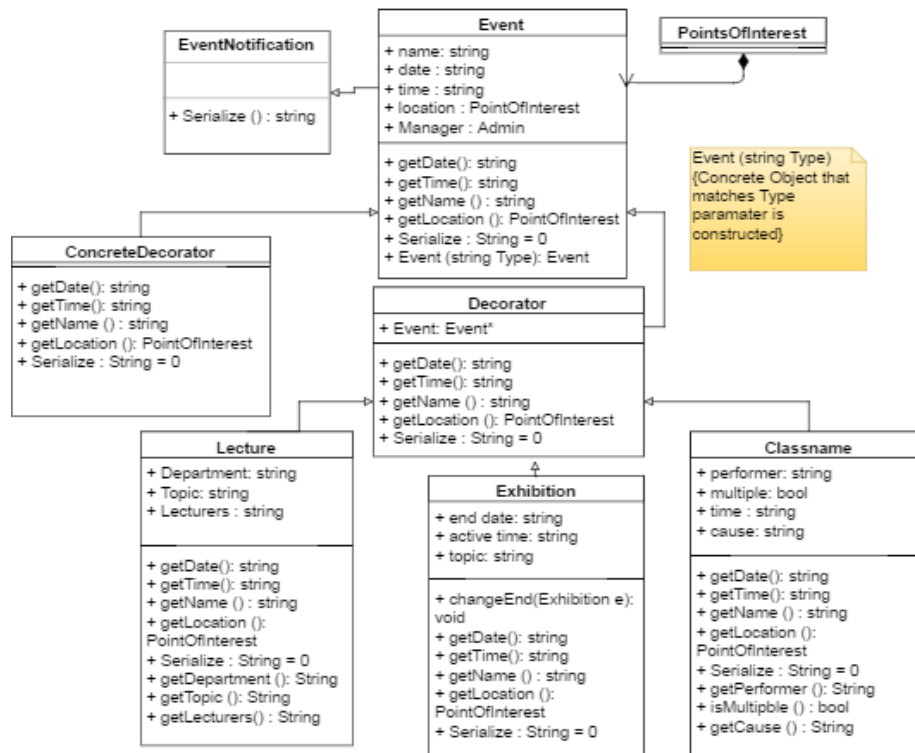


Figure 20: Use Case Diagram for the Events module

6.2 Use Case Diagram

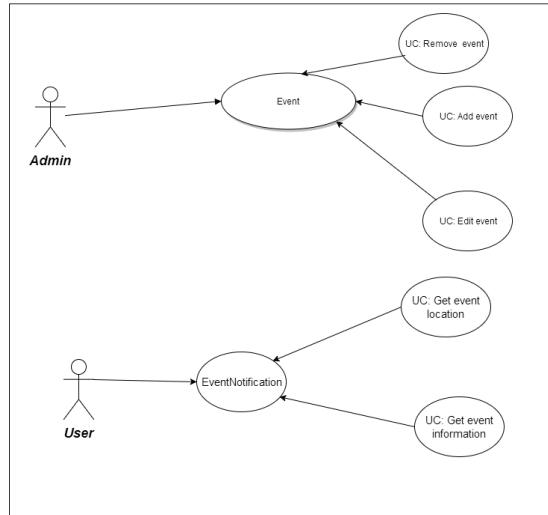


Figure 21: Use Case Diagram for the Events module

6.3 Architectural Design of Module

The events module will be able to inform different users about events on campus. The module will make use of the location services of the navigation system, thus when a user passes a particular point of interest which has public events, the user will receive a notification via sms or email about these events.

The NavUP system will make use of a variety of design patterns. In the events module there will be an events superclass which abstracts the core functionality and members that an event object will provide. This means it will be possible to specify different events based on the type, i.e. a special lecture, exhibition, performance etc. In order to add functionality to these different types of events, the decorator design pattern will be used. A concrete decorator will be defined to implement core functionality defined by the abstract event class, and the decorator class will hold a reference to an event object to which functionality can be added dynamically.

A single event object will have 4 different attributes namely a name for the event, a date, a time, and a location which is a Point of Interest object. Different subclasses will have additional attributes, such as a special lecture having a lecturer's attribute, performances having performers information, and exhibitions having a topic description. The decorator pattern will allow us to add additional functionalities to each subclass based on the additional attributes which enhance specificity.

Each concrete event class will also be equipped with a Serialization function which will return the string containing information about the event which will be used in the notification that is sent to the user, which in the end is the goal of the module.

6.4 Technologies Used

For the CRUD operations, on the system, by the administrator, the Mongo database would be used. This is to ensure that events will appear instantly and be viewed fast enough.

The information entered by the administrator when giving other users administrative privileges will be stored on a MySQL database. This information should be easily accessible to the administrator.

6.5 Non-Trivial Implementation Tasks

The events module will be accessed when the user is using the NavUP system to navigate the UP campus. The system will continuously get the current location of the student and when a student passes a point of interest the system will check for events at the particular location. These events will then be compiled into a notification that the user will receive either as an SMS, email, or both. Once a notification has been sent the system will return to tracking the users location if the user is still using the system for navigation.

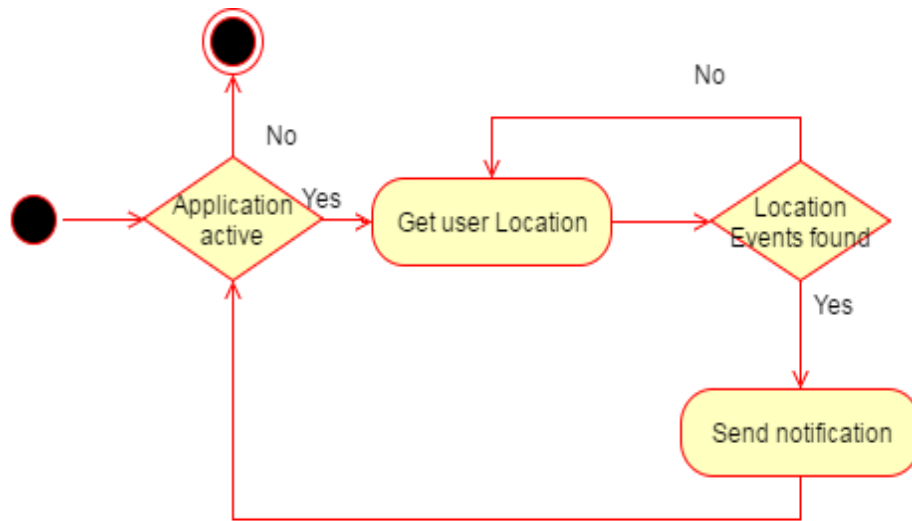


Figure 22: A