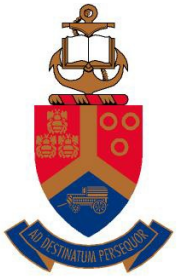

UNIVERSITEIT VAN PRETORIA
UNIVERSITY OF PRETORIA
YUNIBESITHI YA PRETORIA

NavUP

Architectural Requirements Specifications and Design

10 March 2017

Team Scala

Merissa Joubert
15062440

Avinash Singh
14043778

Joshua Cilliers
14267196

Nicolai Van Niekerk
15025269

Gershom Maluleke
13229908

Cameron Trivella
14070970

Peter Soroush
13238958

GitHub https://github.com/RobOgirl/Team-Scala.git

Contents

Table Of Contents **1**

1 Introduction **2**
1.1 Overview . . . 2
1.2 Architectural Pattern . . . 2
1.2.1 Presentation tier . . . 2
1.2.2 Application tier . . . 2
1.2.3 Data tier . . . 2

2 User Module Sub-system **3**
2.1 External Interfaces . . . 3
2.1.1 System Interfaces . . . 3
2.1.2 User Interfaces . . . 3
2.1.3 Hardware Interfaces . . . 3
2.1.4 Software Interfaces . . . 3
2.1.5 Communication Interfaces . . . 3
2.2 Performance Requirements . . . 3
2.3 Design Constraints . . . 3
2.4 Software System Attributes . . . 4
2.4.1 Availability . . . 4
2.4.2 Reliability . . . 4
2.4.3 Security . . . 4
2.4.4 Auditability . . . 4
2.5 UML . . . 5
2.5.1 Class Diagram . . . 5
2.5.2 Activity Diagram . . . 6
2.5.3 Sequence Diagram . . . 6
2.5.4 State Diagram . . . 7
2.5.5 Use Case Diagram . . . 7

1 Introduction

1.1 Overview

This document identifies the architectural design specifications that satisfy the functional requirements proposed in the System Requirements Specification. It addresses the needs of the various subsystems' non-functional requirements focusing on the quality attributes, architectural patterns as well as constraints and integration requirements of the NavUP system.

The following modules' designs are included in this document:

- Users
- GIS
- Notification
- Events

The document begins with an explanation on the chosen architectural pattern and how it will be used to modularize the NavUP system. It then outlines the architectural design of each module along with all relevant UML diagrams. Finally the chosen technologies are discussed as well as the deployment of the system.

1.2 Architectural Pattern

The NavUP system will be designed using the Three-tier architecture in which the user-interface, process logic and data storage are developed and maintained as independent modules. This will allow any of the three tiers to be upgraded or replaced without affecting any of the other layers.
The NavUP system will be divided into 3 tiers:

- Presentation tier
- Application tier
- Data tier

1.2.1 Presentation tier

This is the highest level of the NavUP application. It is the layer which users will interact with directly through the user-interface. The Presentation layer will reside on the mobile and web app and will display information pulled from the Application layer in a way that is simple and intuitive to the user of the application.

1.2.2 Application tier

This layer resides on the application server and controls the application's functionality through detailed processing. The results of this processing is passed back to the Presentation layer which will display it to the user.

1.2.3 Data tier

The Data layer includes all data persistence mechanisms and resides on the database server. The application server uses this layer to manage the stored data. It is crucial that there are no dependencies on the storage mechanisms to ensure that any updates or changes will not affect the application tier clients in any way.

2 User Module Sub-system

The Users module is responsible for the management of NavUP's users. The system will consist of three types of users namely the admin,guest and normal users.The admin, who has more privileges than other users of the application, is mainly responsible for managing other registered users and managing information about venues and activities on campus.

2.1 External Interfaces

This section gives a detailed description of the system interfaces, hardware interfaces, software interfaces as well as communication interfaces.

2.1.1 System Interfaces

- The user module will interface with any subsystem or module that wishes to access the data or functions.

2.1.2 User Interfaces

- The basic function of the user interface is to enable users to interact with the system.All the functions necessary to use the NavUP application are perfomed through the user interface. The user interface allows the user to login or register if they are first time users.

2.1.3 Hardware Interfaces

- The will be a hardware interface with routers across campus in order to triangulate the position of the user on campus and this will require the hardware interface to interact with the Users module of the system.
- The hardware interface also includes devices that users use to access the NavUP application

2.1.4 Software Interfaces

The user module will communicate with the database in order to get the details of users for registration of updating user information, and also for saving locations etc. All the I/O interaction with the database.

2.1.5 Communication Interfaces

The User module would be the core communicator in this system since every other subsystem is directly/indirectly related to user, so communication can be done through local message passing with internal events or via the abstract user management class, another means of implementation would be to consume the RESTful API within the system however this is very inefficient for an local internal event.

2.2 Performance Requirements

User module should be able to perform under high CRUD operations,and have optimised methods to allow for faster execution and faster response times.

2.3 Design Constraints

- The speed at which NavUP can perform is constrained by the processing power and the memory that is available on the device on which it runs.
- The accuracy of the results profuced by functions such as determining the user's location, which are done by the GIS module through the Users module are constrained by the strength of the WIFI connection on campus.
- This system's ability to give updates and events to users through the Users module is constrained by the external management and maintenance which is performed by the administrator of the system.

2.4 Software System Attributes

2.4.1 Availability

The user module should always be active since the majority of the application deals with the user module so having no down-time will be beneficial and required for the system to prevent crashes,

2.4.2 Reliability

The user module should make use of queuing for CRUD database operations, which most databases have and should be reliable since the database should be ACID compliant.

2.4.3 Security

Data within the user module should not have external classes accessing the information without being authenticated, user passwords and usernames should not be able to be accessed outside the scope of this module. Passwords should be encrypted using a strong encryption algorithm such as SHA-512 and should also support end-to-end encryption to guarantee more safer data transfer. The REST API should not be overexposed and have the necessary security implementations such as public/private key encryption, and should only accept authentic connections and prevent against DDOS.

2.4.4 Auditability

All CRUD operations within the user module should be logged for security and data integrity purposes.
All login attempts should also then be logged and notify an administrator if too many attempts to login was unsuccessful to ensure system security, since this could be proof for an audit trail.

2.5 UML

2.5.1 Class Diagram

The class diagram of the user sub-system makes use of the template method design pattern so that if need be one can easily construct different types of users with minimal code modifications.

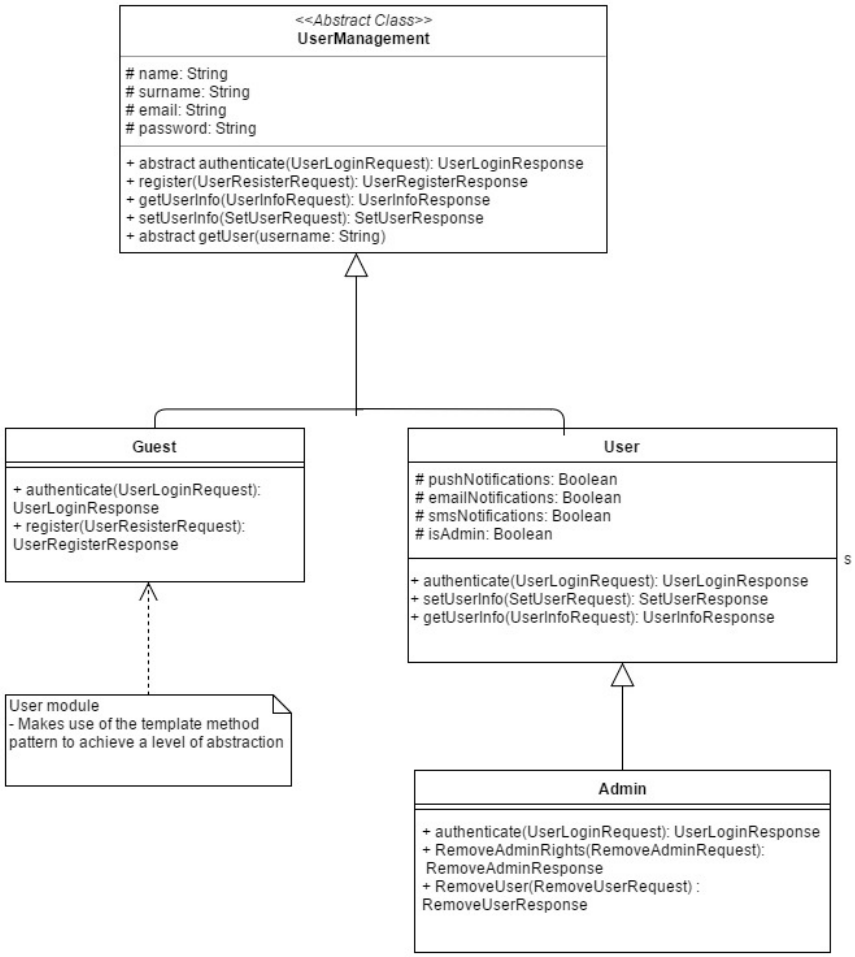

Figure 1: User Class Diagram

2.5.2 Activity Diagram

This diagram models work-flow activities of what users can do in the user module.

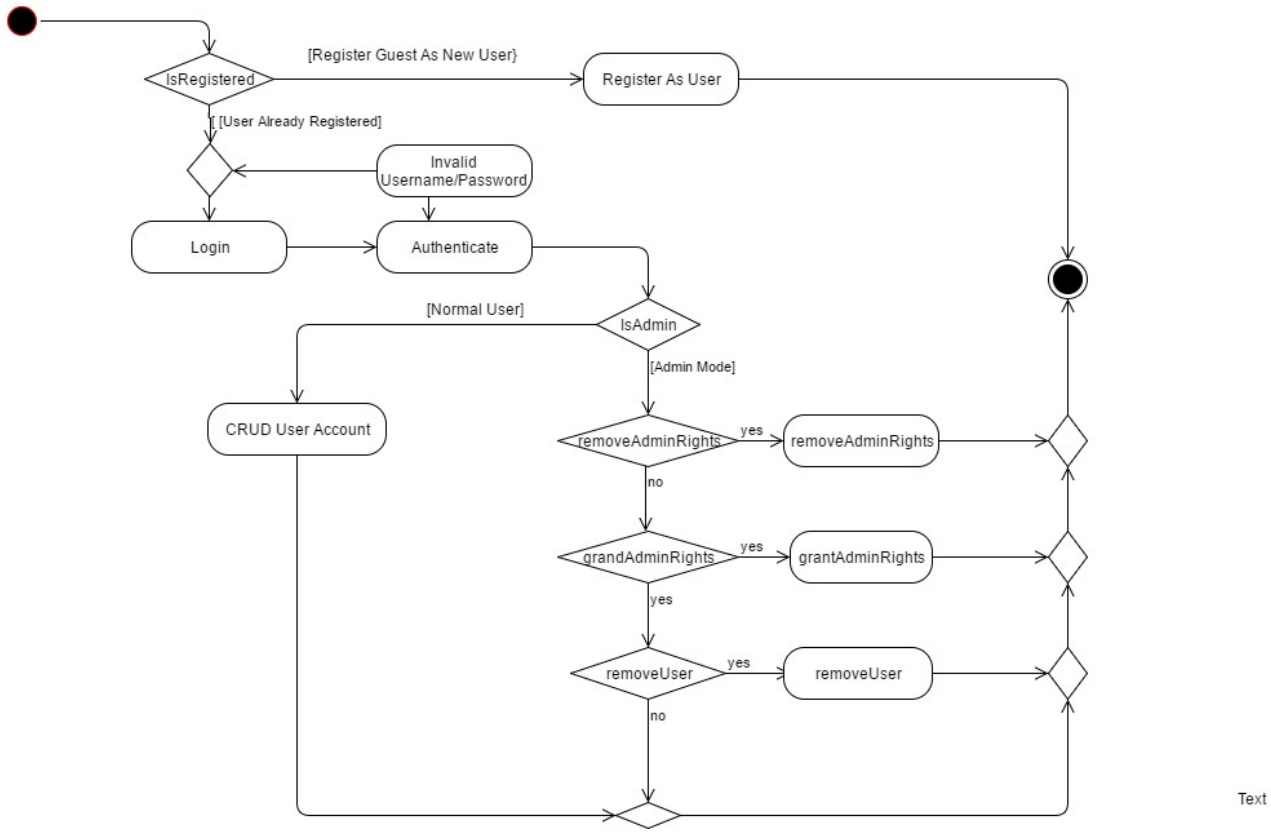

Figure 2: User Activity Diagram

2.5.3 Sequence Diagram

This diagram models time-ordered interaction behaviour between a user logging in and the interaction with the user subsystem.

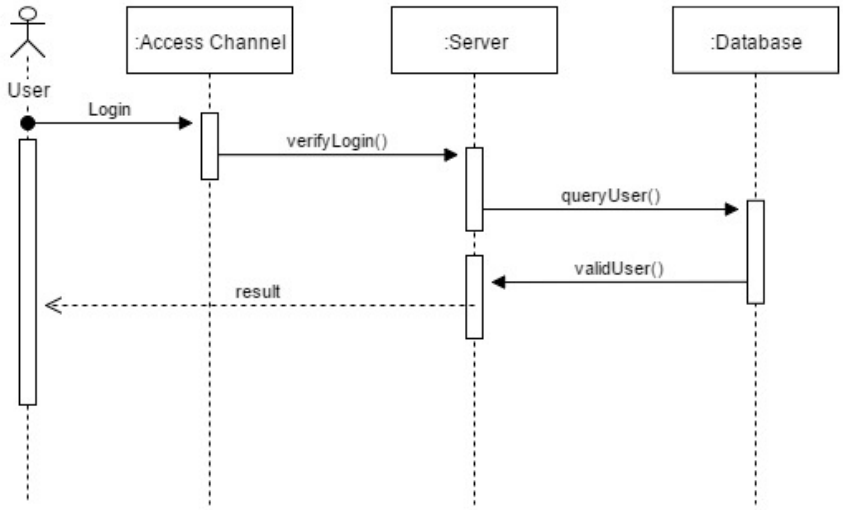

Figure 3: User Login

2.5.4 State Diagram

This diagram models state dependant behaviour of user login and user functionality.

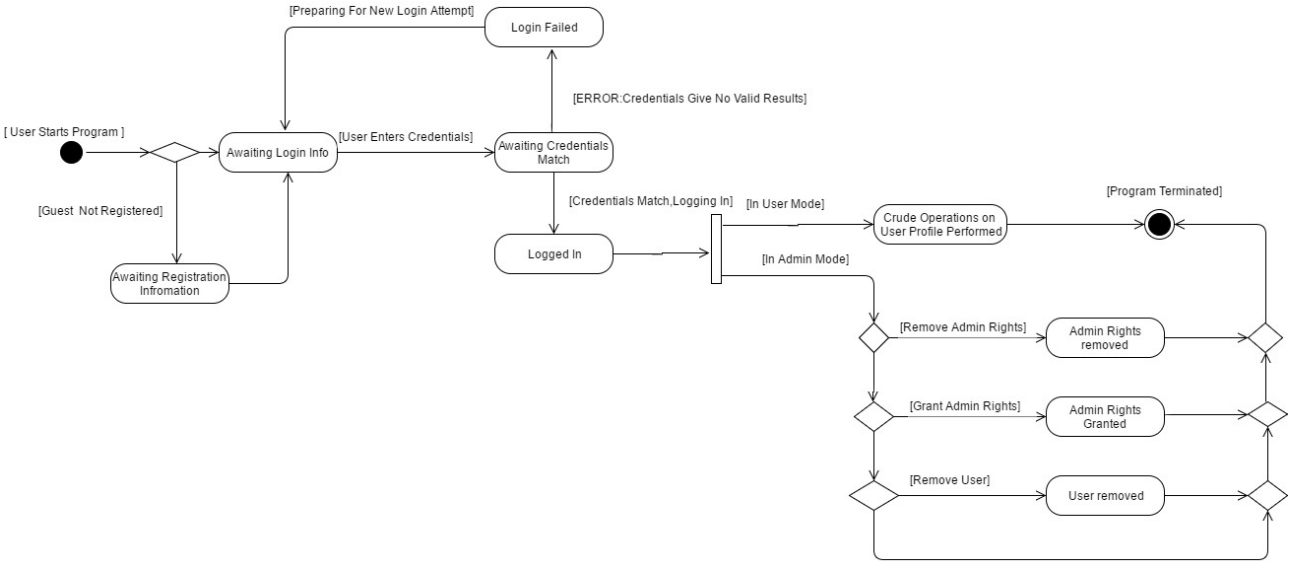

Figure 4: User Login State Diagram

2.5.5 Use Case Diagram

This is a refined version of the use case diagram given in the Requirements Specification. It shows the functions of the subsystem as well as relationships of external entities or actors. This refined version also includes a detailed flow of use cases.

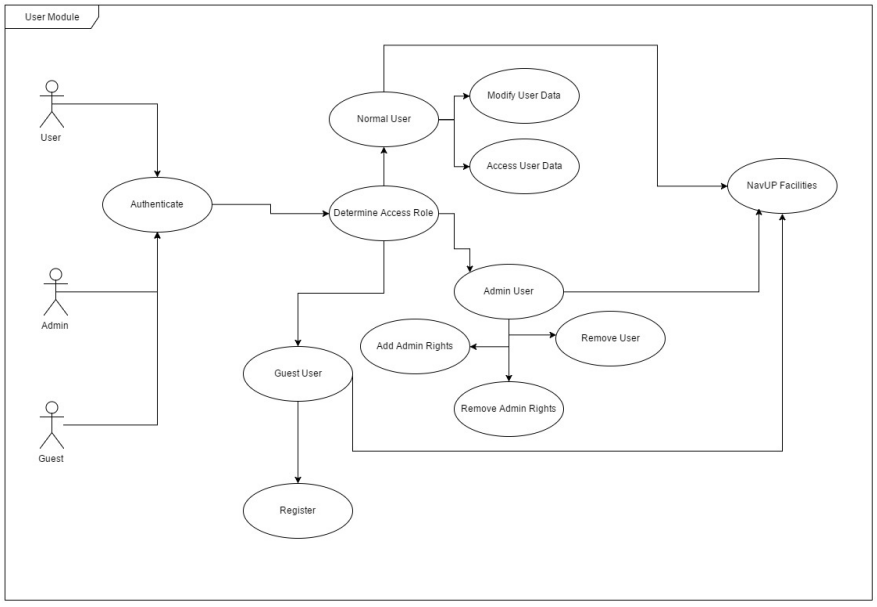

Figure 5: User Login with core functionality