



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/ROB0GIRL/TEAM-SCALA.GIT](https://github.com/Rob0girl/team-scala.git)

Contents

Table Of Contents	1
1 User Module Sub-system	2
1.1 External Interfaces	2
1.1.1 System Interfaces	2
1.1.2 User Interfaces	2
1.1.3 Hardware Interfaces	2
1.1.4 Software Interfaces	2
1.1.5 Communication Interfaces	2
1.2 Performance Requirements	2
1.3 Design Constraints	2
1.4 Software System Attributes	3
1.4.1 Availability	3
1.4.2 Reliability	3
1.4.3 Security	3
1.4.4 Auditability	3
1.5 UML	3
1.5.1 Class Diagram	3
1.5.2 Activity Diagram	5
1.5.3 Sequence Diagram	5
1.5.4 State Diagram	6
1.5.5 Use Case Diagram	6

1 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.

1.1 External Interfaces

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

1.1.1 System Interfaces

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

1.1.2 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 performed through the user interface. The user interface allows the user to login or register if they are first time users.

1.1.3 Hardware Interfaces

- There 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

1.1.4 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.

1.1.5 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.

1.2 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.

1.3 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 produced 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.

1.4 Software System Attributes

1.4.1 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,

1.4.2 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.

1.4.3 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.

1.4.4 Auditability

All CRUD operations within the user module should be logged for security and data integrity purposes.

1.5 UML Diagrams

1.5.1 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.

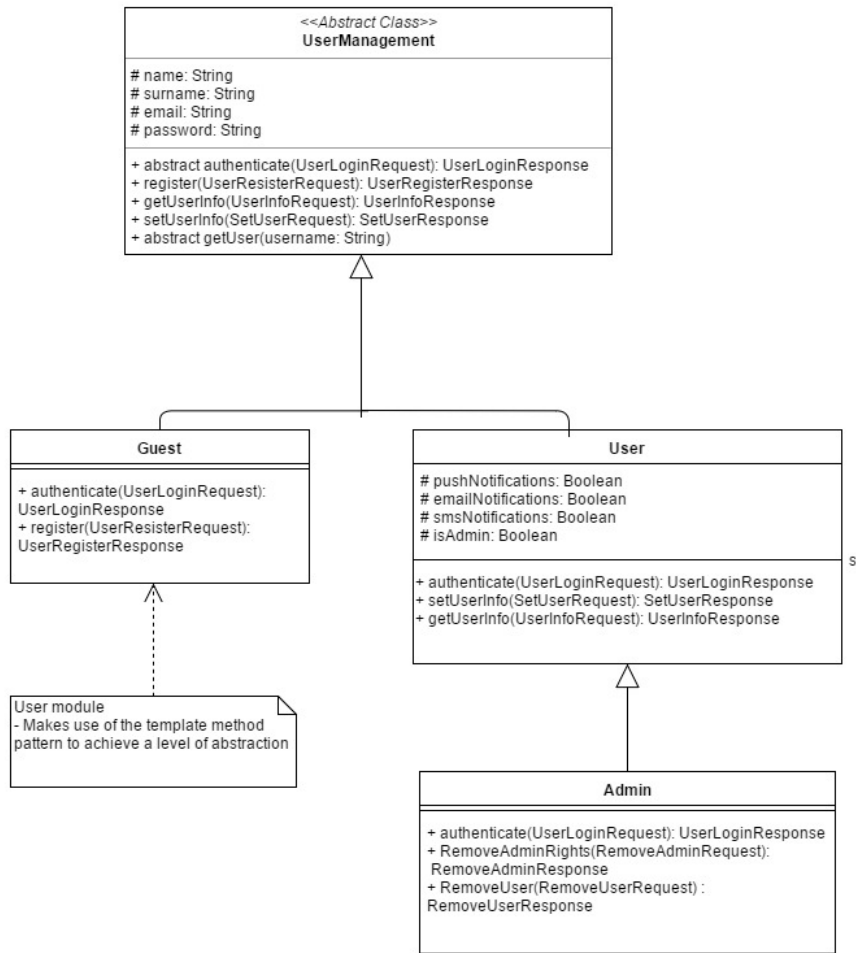


Figure 1: User Class Diagram

1.5.2 Activity Diagram

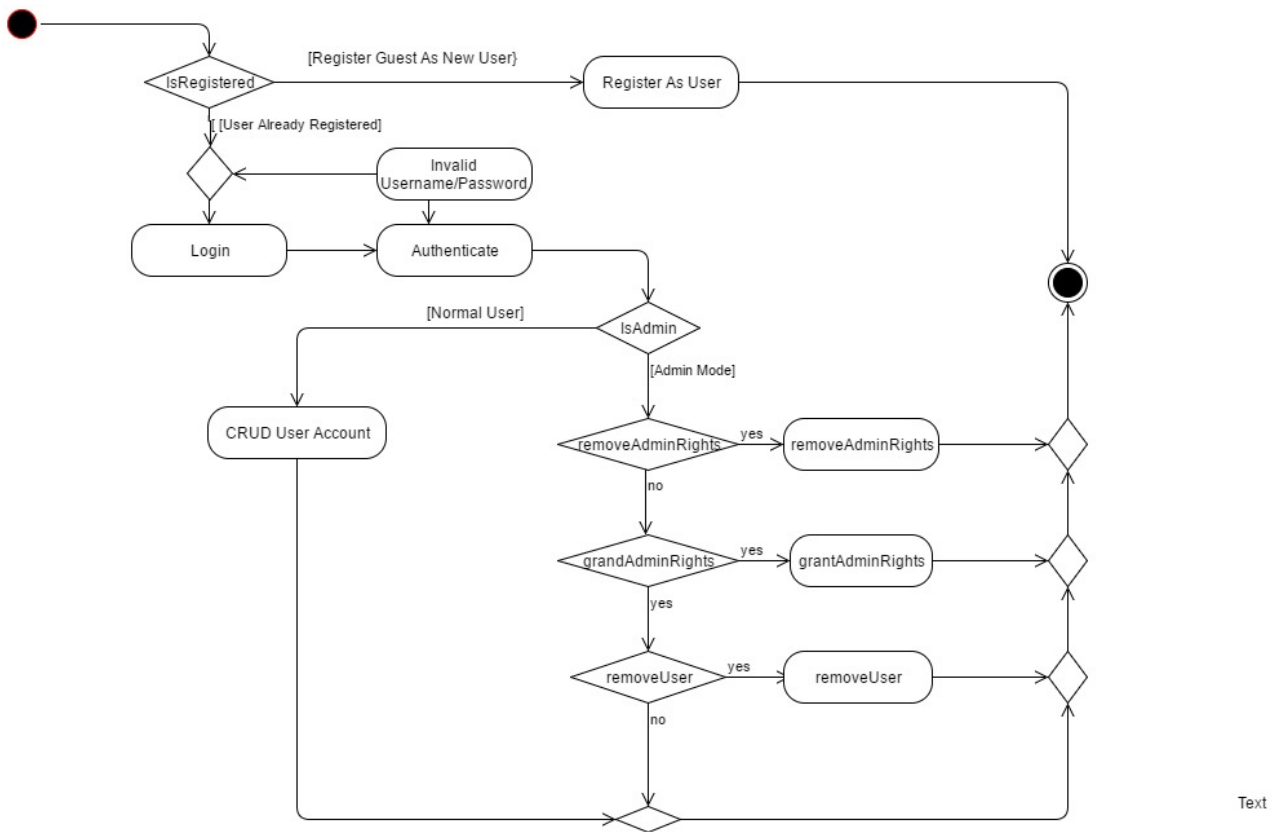


Figure 2: User Activity Diagram

1.5.3 Sequence Diagram

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

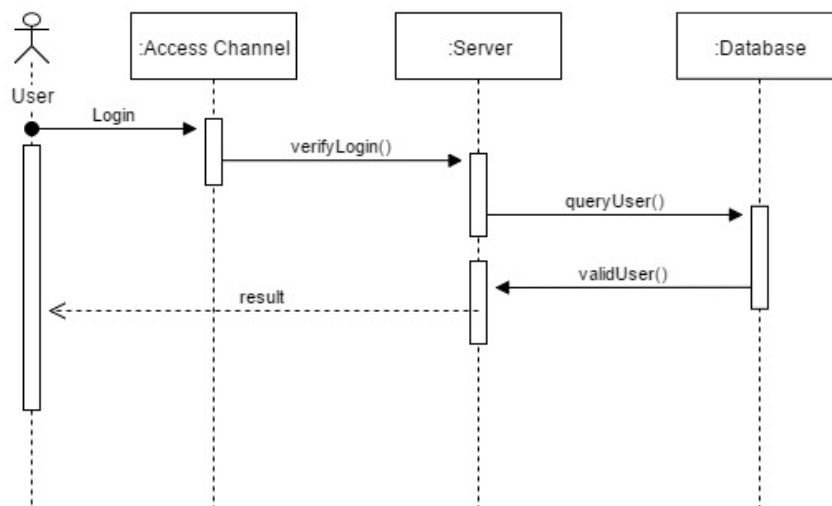


Figure 3: User Login

1.5.4 State Diagram

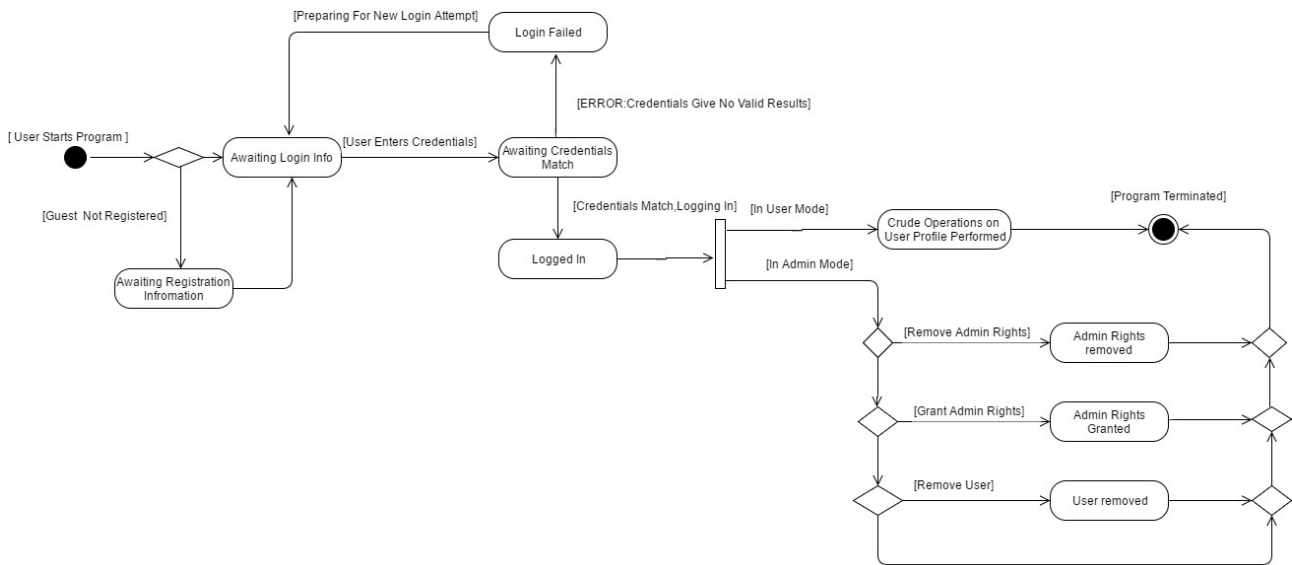


Figure 4: User Login State Diagram

1.5.5 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.

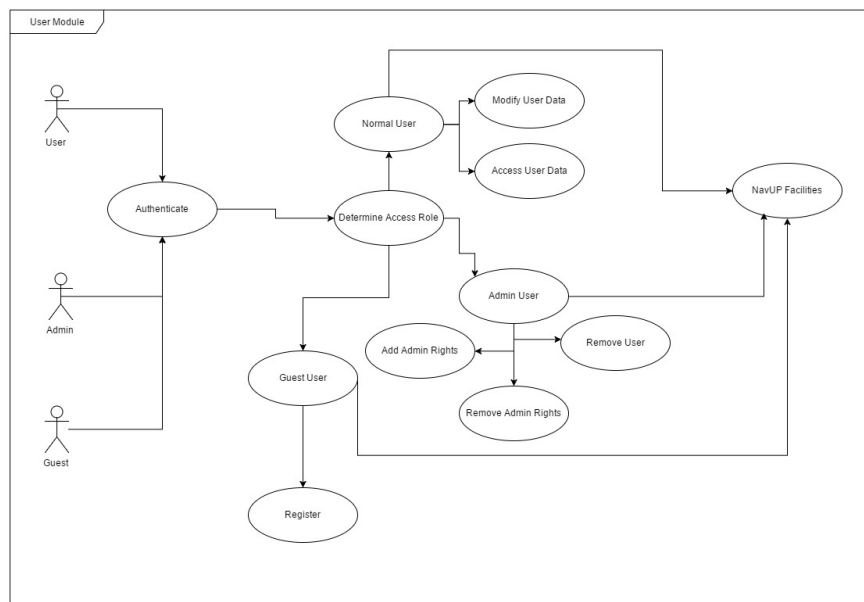


Figure 5: User Login with core functionality