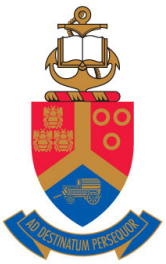

UNIVERSITEIT VAN PRETORIA
UNIVERSITY OF PRETORIA
YUNIBESITHI YA PRETORIA

NavUP
Implementation Documentation - Users Module

Compiled By

Takalani Sigama - u14166365
Melvin Zitha - u12138747
Mathapelo Matabane - u12206522

2017
Longsword Users TEAM

Abstract

This document stipulates the implementation phase of the NavUP system - User subsystem. The documentation highlights the definition of the user management system built, the technologies implemented, as well as describing the architecture of the User subsystem.

Contents

1 Introduction **4**
1.1 Scope 4

2 User Management Functions **5**
2.1 getUser 5
2.2 authenticate 5
2.3 isAuthenticated 5
2.4 registerAsUser 5
2.5 grantAdminRight 5
2.6 deleteUser 5

3 Helper Functions **5**
3.1 persistUser 5
3.2 check 6
3.3 getUserFromDb 6
3.4 updateUser 6

4 Technologies **6**
4.1 API Restful 6
4.2 Java Persistence API 6
4.3 SQL 6

1 Introduction

1.1 Scope

The User management module is responsible for maintaining information about the registered users of the system, including the authority levels of each user. Administrators can manage information about venues and activities whilst users who have signed up may request services from the various modules and persist private information related to particular services.

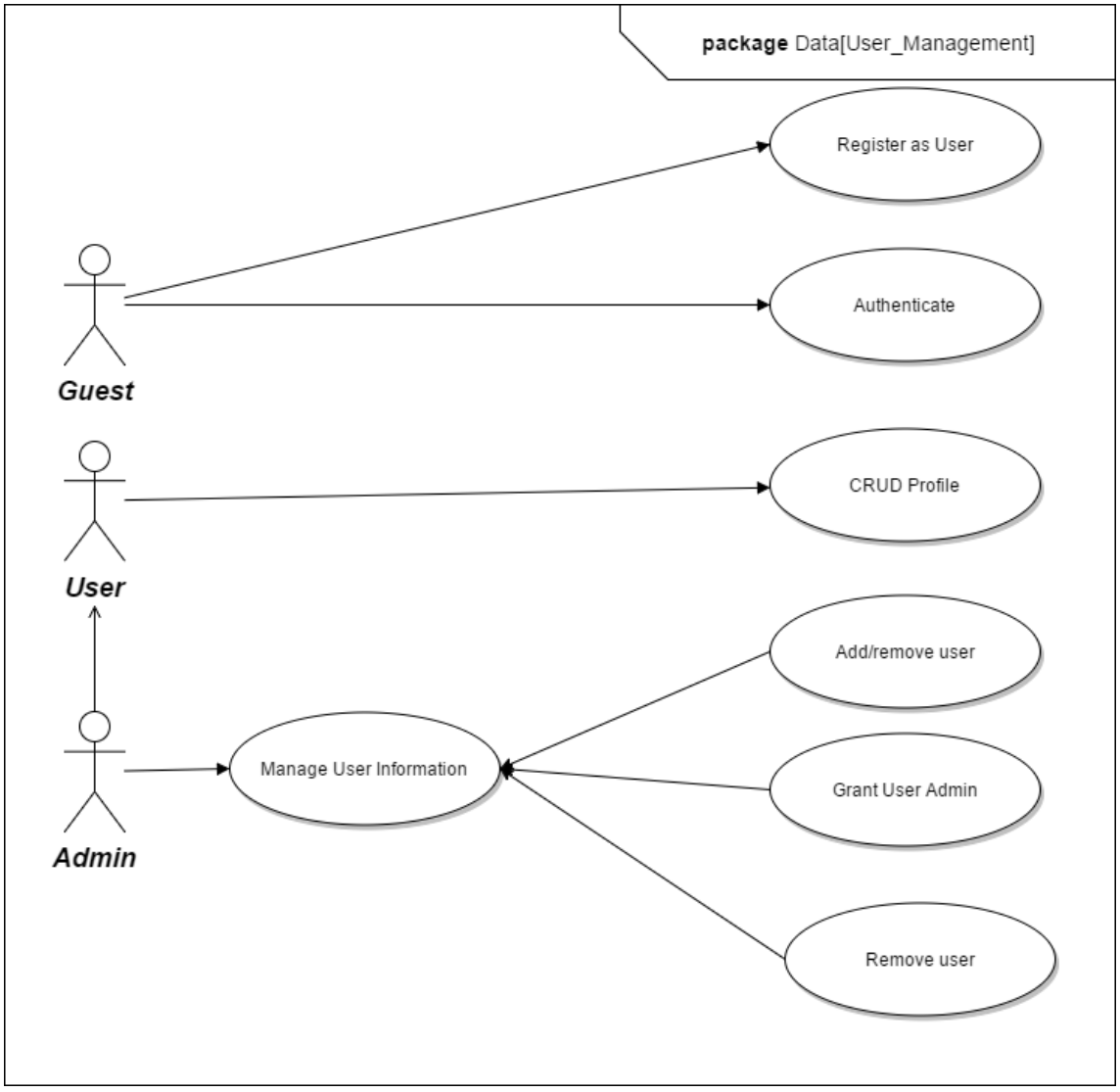

Figure 1: Use Case Diagram

There are three types of users: Guest, User and Admin.

2 User Management Functions

2.1 getUser

This module takes a username string and returns the user object with the users data

2.2 authenticate

Module used to login a user. The module takes in a username and a password which is hashed first before it is used to check if it is the same as the hashed password stored in the database that corresponds to the username used. If a match is found, a session is created in the webservice and the user is logged in. The module returns true if successful and false if the user was not found in the database or if the password did not match the password stored in the database for the specified username.

2.3 isAuthenticated

This module is used to check if the user is already logged in. If there is an active session that corresponds to the current user, the module returns true as a string using the gson library to change the Boolean variable into a string, else it returns false.

2.4 registerAsUser

This module registers the user and persists their information in the database.It accessess the database via the jdbc Driver and updates the table of users and stores the users' information on the user table

2.5 grantAdminRight

The grantAdminRight module is used to modify the administration privileges of a specific user.The user is allowed the same privileges as an admin, The user can then assign other users as admin, they can add, remove and edit tables. The module returns a boolean value as a string indicating whether the operation was a success or not, the module returns true if the user was granted admin rights or false if its unsuccessful. The module starts by accessing the database and searching for the user to see if a user with the supplied username exists, if they do, the module then checks to see if the user has already been granted admin rights, if so the module returns true and does not make any changes. If the User has not yet been assigned any admin rights, the user is then assigned user admin rights by setting the isAdmin value to true and then the module returns true to indicate that it was successful. If the user was not found in the database, the module returns false to indicate that it was unsuccessful.

2.6 deleteUser

Removes the user from the database and returns true or false as a string using the gson library to change the Boolean variable into a string. The module first searches for the user in the database to see if the user exists in the database, if the user is found, the users information is removed from the database and the module returns true. If the user is not found, an exception is thrown and the user returns false.

3 Helper Functions

3.1 persistUser

Used to persist user information in the database.

3.2 check

checks if the user exists in the database using their email address and their username

3.3 getUserFromDb

searches the database for a user using their username, if the user is found in the database, the module returns all the users information as a User object.

3.4 updateUser

Used to update the users information in the database.

4 Technologies

User subsystem is best illustrated by the Model-View-Control(MVC) architectural pattern. The following technologies were used to implement this subsystem.

4.1 API Restful

This service calls users function to registering a user, adding a users and updating user details.

4.2 Java Persistence API

Java Persistence API(JPA) is a Java specification for accessing, persisting and managing data between the UserManager, User, and Us

4.3 SQL

Standard Query Language(SQL) is used commonly in programming and designing for managing data held or streamed in a relational database management system.