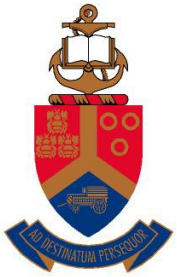

UNIVERSITEIT VAN PRETORIA
UNIVERSITY OF PRETORIA
YUNIBESITHI YA PRETORIA

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

1 Notification Module

1.1 Overview

The notifications module provides notifications to system users regarding particular system updates that a user would like to be notified about through some medium external to the application.

1.2 External Interface Requirements

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

1.2.1 System Interfaces

- The notification module will interface with any subsystem or module that wishes to send out notifications to users. The subsystem requesting for users to be notified is modelled as the Notifying Module. The Notifying module will interface with the Notification Module through the setState() function to send a notification request.
- The notification module will interface with the User module in order to sufficiently dispatch notifications. Each instance of the Notifier class will maintain a list of attached users and will send notifications through the notify() function.

1.2.2 User Interfaces

- The system will dispatch notifications in the form of an **SMS** to the appropriate users. The SMS will contain the content of the notification and the interface will depend on the user's device.
- The system will dispatch notifications in the form of an **email** to the appropriate users. The email will have an appropriate subject and will contain the content of the notification. The interface will depend on the mail application used by the user.
- The system will dispatch notifications in the form of **Push Notifications**. push notification will contain a title, content as well as a timestamp. The interface will depend on the user's operating system.

1.2.3 Hardware Interfaces

The Notification Module does not have any explicit hardware interfaces

1.2.4 Software Interfaces

- The notification module will communicate with the database in order to get the details of users to be notified.

1.2.5 Communication Interfaces

- The notification module will need to communicate with an Email API when sending out emails to users
- The notification module needs to interface with the SMS Manager API to send SMS's to users.
- The notification module will interact with the Operating system push notification service (OSPNS) to allow users to receive push notifications.

1.3 Performance Requirements

Notifications should be able to perform and guarantee message delivery under high volumes, allow real time notifications to be received and have good network access for this purpose.

1.4 Design Constraints

The notifications must be:

- Visually appealing and easy to read, have a gamification point of view
- Support viewing on different sizes of device screens
- Support colour for all devices
- Being able to integrate easily with notification providers

1.5 Software System Attributes

1.5.1 Reliability

The system should make use of a queue to send notifications, so that no messages can get lost on the way and must guarantee that the notification must be delivered. Notifications should be authentic and be able to verify that delivery of the notification.

1.5.2 Security

The sending of the notifications must be secure using proper SMTP setup with SSL encryption for emails, and a authentic SMS provider validated by the WASPA, and a secure means of push notifications using IONIC App that allows secure remote push notifications to all user devices that contain the app.

1.5.3 Auditability

All notifications that have been sent by any means, by email, SMS, or even push notifications should be logged on the back-end server.

1.6 Design

The Notification Subsystem is designed using 2 design patterns:

1. **Strategy -** This pattern encapsulates the send() function within the Notification class, making it interchangeable. This allows the send() function to vary based on whether the notification is an email notification, SMS notification or a push notification.
2. **Observer -** This pattern defines a one-to-many relationship between the Notifier and User classes so that when the Notifier's state changes, all users are notified automatically.

1.6.1 Class Diagram

This diagram models the structural elements, their composition and classification.

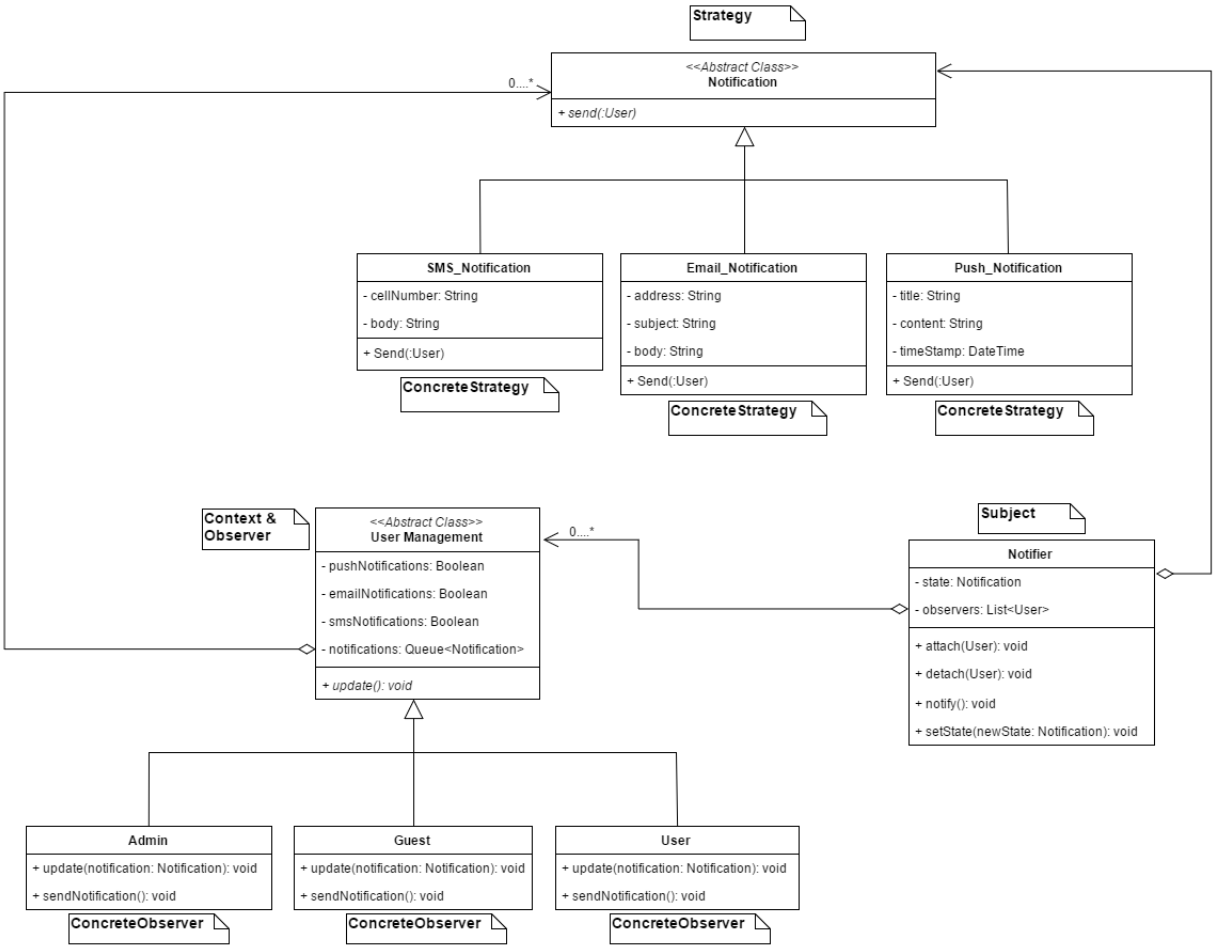

Figure 1: Notification Class Diagram

1.6.2 Activity Diagram

This diagram models workflow activities and exhibits sequencing, exclusion, synchronizaion and concurrency.

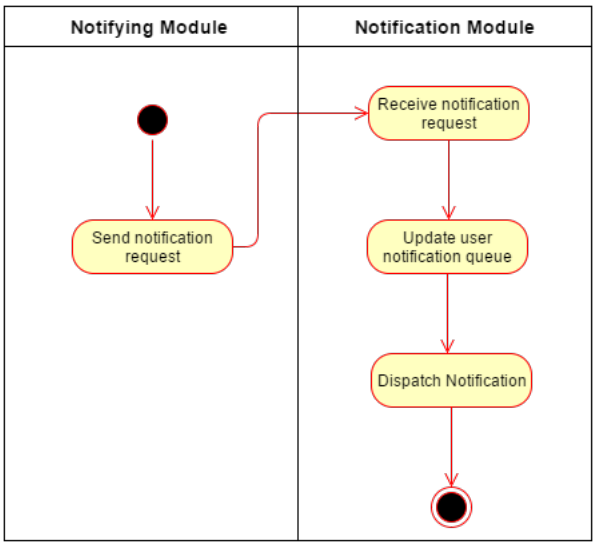

Figure 2: Notification Activity Diagram

1.6.3 Sequence Diagram

This diagram models time-ordered interaction behaviour between the objects within this subsystem.

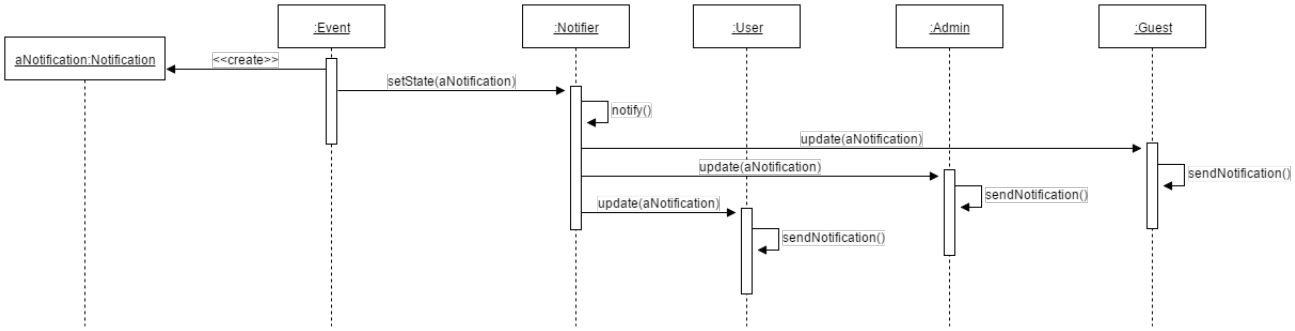

Figure 3: Notification Sequence

1.6.4 State Diagram

This diagram models state dependant behaviour of an object.

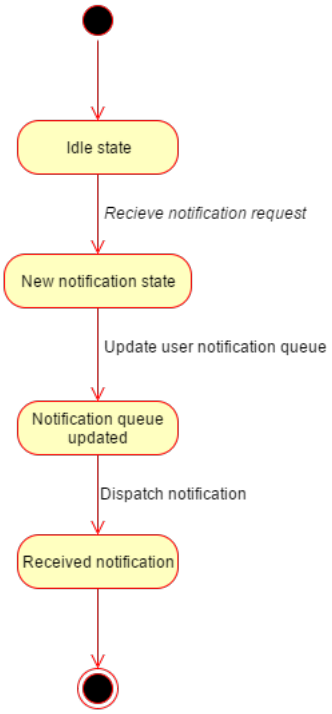

Figure 4: Notification State Diagram

1.6.5 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.

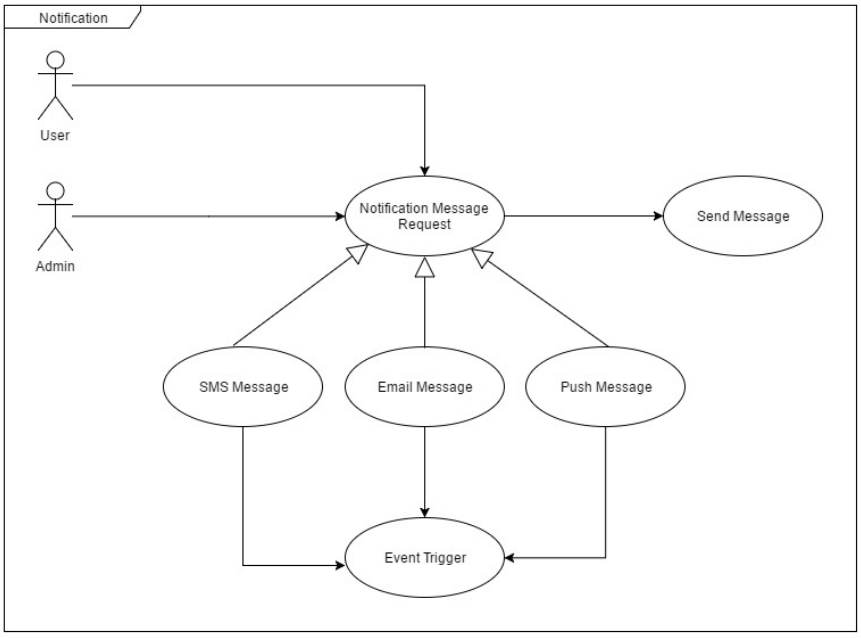

Figure 5: Notification core functionality