NavUP
Architectural Requirements Specifications and Design

Compiled By

Andries Jacobus du Plooy - u15226183
Joseph Phutjane Letsoalo - u15043844
Mathapelo Matabane - u12206522
Munyaradzi Mpofu - u15071830
Ritesh Doolabh - u15075754
Midhun John Cheriyan - u17308632

2017

TEAM FORTRAN

Abstract

This documentation covers all the design requirements for the NavUP system, this includes System's External Requirements Performance Requirements, Technology choices and Design Constraints, to name a few. Furthermore, the documentation brings to mind the appropriate design patterns to implement to construct the system.

Contents

1 Introduction

Information about the activities, points of interest and venues on campus is not readily available to employees, students and visitors to the UP campus. This is especially a problem at the beginning of the academic year when thousands of new students come to campus to attend regular lectures and on open days when large numbers of visitors have to find their way to exhibitions and service points.

1.1 Purpose

NavUP is aimed at providing access to a wealth of information about campus activities, points of interest and venues via the campus WiFi network. The information should be accessible using smart devices. Users should be able to use the information to navigate on campus and be informed about activities and facts of their interest.

1.2 Scope

The core of the system is a navigation tool which is enriched with functionality receiving information of interest.

2 References

Goldman, J.L., Abraham, G. and Song, I. (1998) Generating Software Requirements Specification (IEEE Std. 830 1998) document with Use Cases. Available at: http://www.pages.drexel.edu/ sga72/docs/SRSwithUseCases.pdf (Accessed: 24 February 2017).
Singh, E., Pieterse, V., Omeleze, S., Theunissen, M. (2017) 2017 Class Project NavUP. Available at: http://www.cs.up.ac.za/courses/COS301/NavUPRequirementsSpecification.pdf (Accessed: 27 February 2017).

3 External Interface Requirements

3.1 User Interface

The Notifications subsystem will allow for notifications to be pushed to registered users in the form of E-mails. The E-mails will have a descriptive subject associated with it and will be accessible from any device that allows a form of e-mail service. The subsystem will allow for notifications to be pushed to registered users in the form of SMS's. the SMS's will only have a senders name which will indicate the type of notification and the SMS's content will then go on to expand on the notifications description. The subsystem will interact with users mostly on a cellular device and a colour screen will be required to get the most out of the application in terms of the E-mail service, SMS's on the other hand are a vital part of most cellular devices and in most cases no extra functionality is required.

The system must be able to help the user locate buildings by the use of display, navigate and select functions over the system's built GUI. The system shall provide help options to the user when locating a certain POI.

3.2 Software Interface

In regard to the notifications subsystem, the application will require an e-mail and SMS application on the device of choice in order to receive notifications of this type. The subsystem will need to have access to a database of users such that they can be added to the user lists for the particular type of notification/s of their preference. These lists will then be referenced when the notifications are being sent to the users. The subsystem will also need to interact with all the various other notifying modules that may want to send out information to the users of the application.

A programmatic interface for operations such as adding locations, modifying locations, deleting locations is required.

3.3 Hardware Interface

Using the With the use of most cellular devices, a color screen will be required for the use of e-mails and the device must in all cases be able to display information in a way that is understandable by the user. The device must have the appropriate hardware in place to allow a form of data connection between itself and the Internet. The system needs a database of all the WI-FI hot spots around the campus to map out the location or each Point of Interest. Moreover, the system must update information pertaining to points of interest.

3.4 Communication Interface

The Notifications subsystem will need to communicate with any other subsystem that wishes to create and send out notifications to the users. The subsystem will need to communicate with an automated E-Mail service to send out the mass e-mails to the users. Each notifying module will have access to its own instance of the notification subsystem to prevent any notifications from different notifying modules causing any conflict, and this also allows for no waiting period between creating and sending of notifications. They will be delivered in real time. The subsystem will need to communicate with an automated SMS service to send out the mass SMS's to the users. Each notifying module will have access to its own instance of the notification subsystem to prevent any notifications from different notifying modules causing any conflict, and this also allows for no waiting period between creating and sending of notifications. They will be delivered in real time. The users will need to have the appropriate communications interfaces available to allow them to receive notifications. This would include a working cellphone SIM card with a registered cellphone number and/or a working e-mail account that is accessible on a device with a data connection of some form.

3.5 System Interface

3.6 Memory

4 System Features - Modules

4.1 User

4.1.1 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.
No information is stored for the guest user. When accessing the service, the user assumes the guest role without being logged in. The guest user may use public services and my register or log in. When a user is registered, the fields shown in the domain model are stored for the user using a unique automatically assigned ID. The user provides all other fields. The password should be stored in encrypted format. The user may change the value of any of the fields of his own record except the value of the isAdmin field. The only difference between a User and an Admin is the value of the isAdmin field. Only Admin users may change the value of the isAdmin field of other users. There are three types of users: Admin, User and Guest.

4.1.2 Design details

4.2 Notifications

4.2.1 Scope

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.

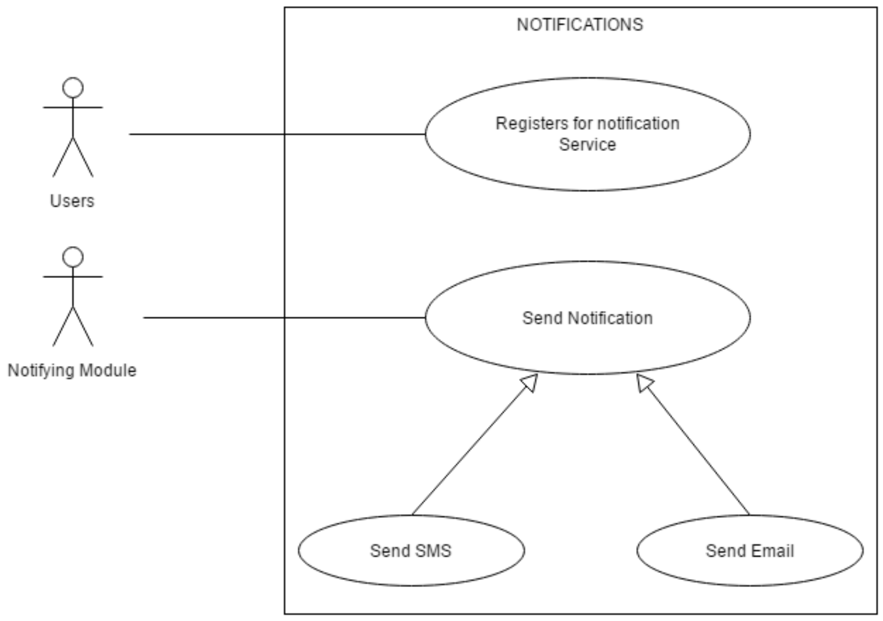

Figure 1: Notification Use Case Diagram

4.2.2 Design details

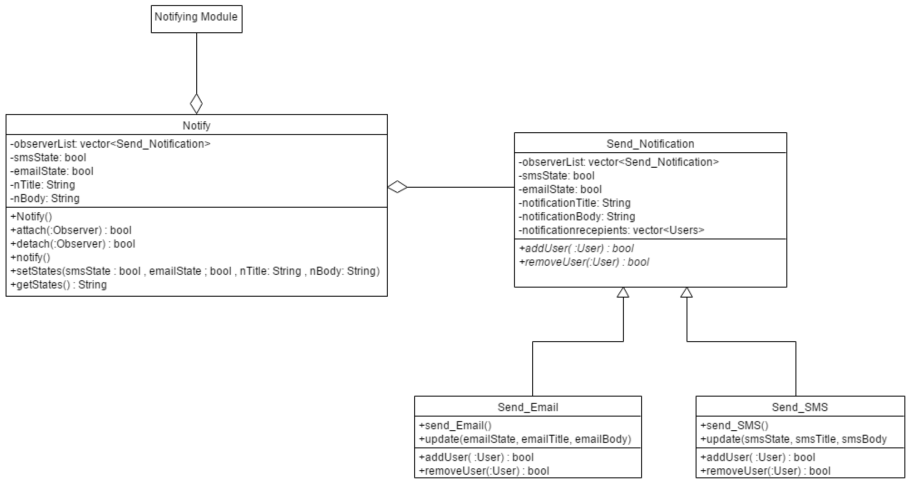
Notifying Module
Notify
-observerList: vector<Send_Notification>
-smsState: bool
-emailState: bool
-nTitle: String
-nBody: String
+Notify()
+attach(:Observer) : bool
+detach(:Observer) : bool
+notify()
+setStates(smsState : bool , emailState ; bool , nTitle: String , nBody: String)
+getStates() : String
Send_Notification
-observerList: vector<Send_Notification>
-smsState: bool
-emailState: bool
-notificationTitle: String
-notificationBody: String
-notificationrecepients: vector<Users>
+addUser(:User) : bool
+removeUser(:User) : bool
Send_Email
+send_Email()
+update(emailState, emailTitle, emailBody)
+addUser(:User) : bool
+removeUser(:User) : bool
Send_SMS
+send_SMS()
+update(smsState, smsTitle, smsBody
+addUser(:User) : bool
+removeUser(:User) : bool

Figure 2: Notification Class Diagram

For the notifications subsystem the observer design pattern came to light. This pattern allows objects to change state and all of its dependents follow suit. This allows a notification state to be used that can update whether or not a notification needs to be sent out. In this case the push method of observer was used. Also evident is the template design pattern that uses the concrete observers to redefine the way in which the notifications are sent. Notifying modules are also given the flexibility to decide which medium(s) are used for the notification. Any notifying module can call the setState() method to initiate the sending of a notification. This function will allow for boolean values to be passed to the sending classes to clarify which notification method/s are to be used. Included in this method are variables to encompass the title and body of the notification. The notify class would then call the notify method which in turn would call the update() functions on the concrete observers and that would trigger the sending of the notifications.

There are methods in the Send Email and Send SMS classes that allow users to be added to the user lists for the notifications to either or both of these mediums. There is also provision for more notification mediums to be added in the future and these can be attached to the Notify class. Should a notification medium be unavailable it can be detached from the Notify class until it is available. The Notify class can be called from any other module that would like to request a notification to be sent out to users which allows for modularity.

Notify: Subject

Send Notification: Observer(Observer) / Abstract Method(Template)
Send Email: Concrete Observer1(Observer) /Concrete Method(Template)
Send SMS: Concrete Observer2(Observer) / Concrete Method(Template)

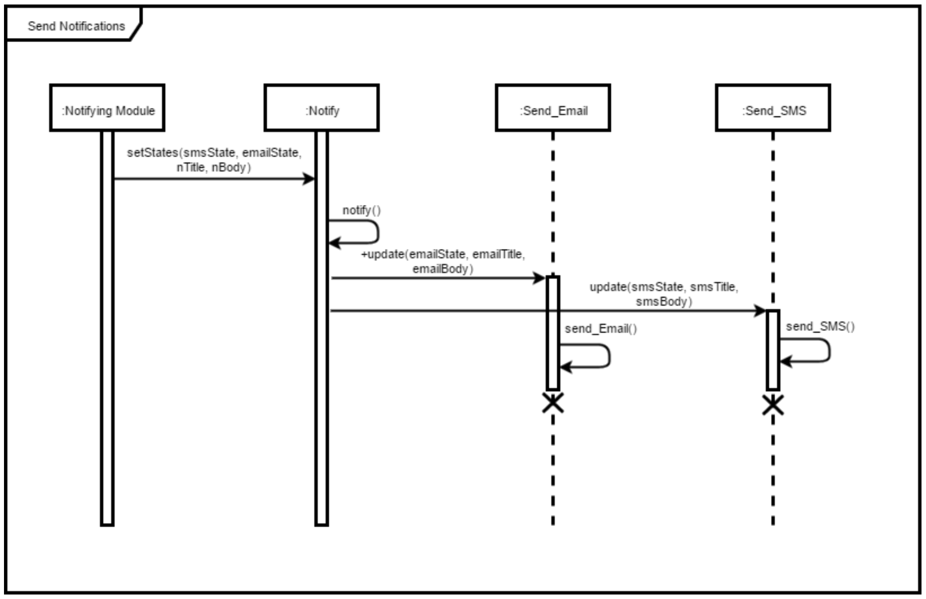
Send Notifications
:Notifying Module
:Notify
:Send_Email
:Send_SMS
setStates(smsState, emailState, nTitle, nBody)
notify()
+update(emailState, emailTitle, emailBody)
update(smsState, smsTitle, smsBody)
send_Email()
send_SMS()

Figure 3: Notification Sequence Diagram

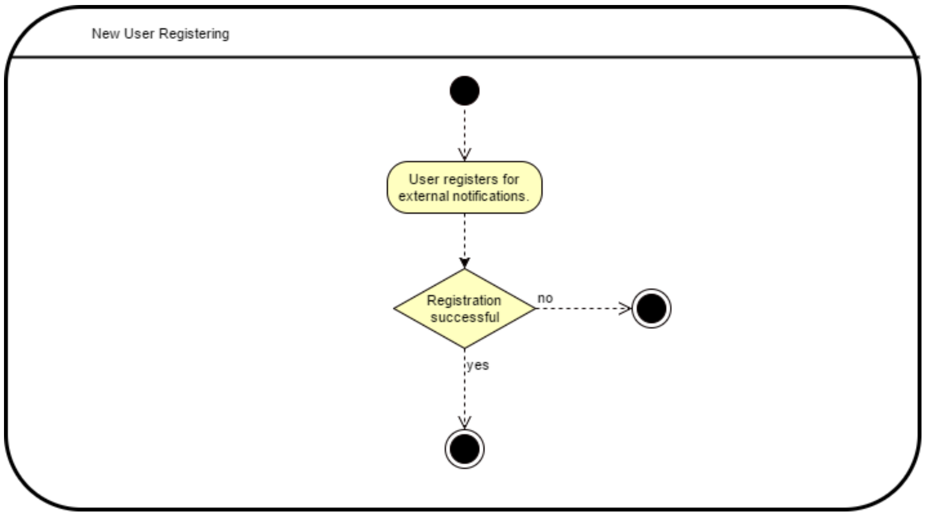

Figure 4: Add User Activity Diagram

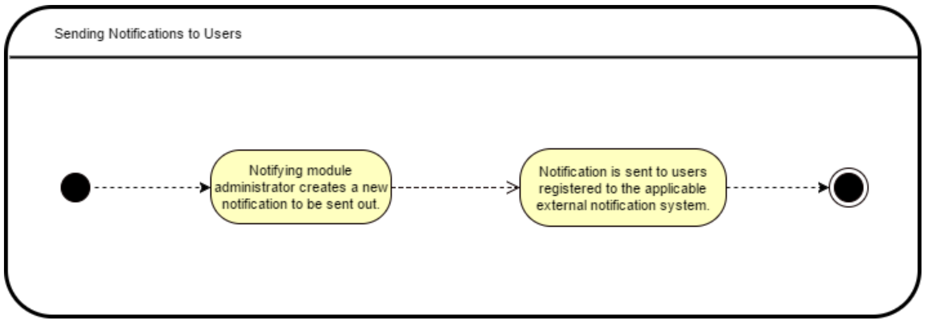
Sending Notifications to Users
Notifying module administrator creates a new notification to be sent out.
Notification is sent to users registered to the applicable external notification system.

Figure 5: Send Notification Activity Diagram

4.3 Points Of Interest

4.3.1 Scope

4.3.2 Design details

4.4 Events

4.4.1 Scope

4.4.2 Design details

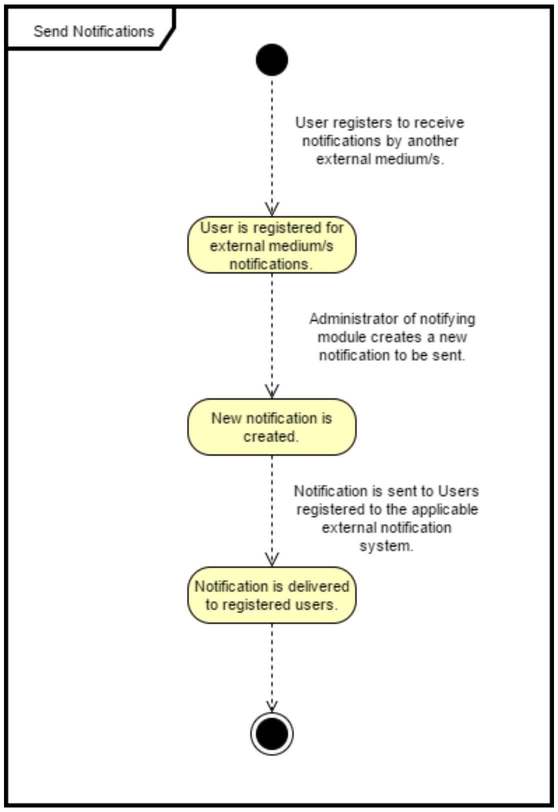
Send Notifications
User registers to receive notifications by another external medium/s.
User is registered for external medium/s notifications.
Administrator of notifying module creates a new notification to be sent.
New notification is created.
Notification is sent to Users registered to the applicable external notification system.
Notification is delivered to registered users.

Figure 6: Notification State Diagram

5 Other Non-functional Requirements

5.1 Performance Requirements

The notifications will have to be pushed out in a timely manner such that the time between creation and time of sending of notifications is minimal.
The system will also have to handle sending bulk e-mails and SMS's from multiple notifying modules in a manner that will not cripple the system or put the system at risk.
The system must accommodate any number of users at any given time period.
All page requests must be processed by the system in no more than the average response time of the Internet connection.
The system must display confirmation of user log ins, as well as display favorite Points of Interest relative to the user profile.

5.2 Design Constraints

Notifications should not be long such that loading times are affected(e-mail) or that partitioning of messages occurs(SMS). Notifications should be pushed out within a given time frame. Notifications should conform to the rules of the medium of transport. No unnecessary data/information should be sent in these notifications.

5.3 Software System Attributes

5.3.1 Reliability

The notifications subsystem is crucial in informing users of possible information. With this it is of utmost importance that all notifications sent are received by the registered users.

5.3.2 Availability

The service must be available to all users and administrators of notifying modules at all times.

5.3.3 Security

Notifications and communications must be secure such that no malicious activity occurs on the system and users are not conveyed wrong information.

5.3.4 Maintainability

The amount of notification mediums is increasing and to accommodate this new forms of communication mediums must be easily appended to the system.

5.4 Technology choices

Below is a summary of the technology choices for the NavUp system and a brief motivation for them.

- TCP/IP - We chose the TCP protocol for information transfer between the server and devices as this method allows for error checking in data and using this we can confirm that the user has received the data/information that was sent.
- PostgreSQL - This would be the database management system used for the system, this platform has features such as providing extensibility in which we can customize it to the needs of the system. Using a separate platform for the database allows it to be separated from the rest of the system.

- PostGIS - Integrating this feature will allow the GIS system to communicate directly with the database system allowing easier integration with location management. PostGIS adds additional types to the database such as geometry, geography, raster and others. This feature will also allow for both processing and analytic functions to be applied for both vector and raster data.

- Traccar - This is a open source GPS tracking platform that will be used to allow tracking on all devices. This service provides client side tracking as well as server side management with location reports done in real time. The client is available for iOS and Android which suits our needs. Traccar also supports multiple communication protocols.

- HTML5 -

- CSS -

6 Appendix A - Glossary of Terms and Definitions