UNIVERSITEIT VAN PRETORIA
UNIVERSITY OF PRETORIA
YUNIBESITHI YA PRETORIA
Faculty of Engineering, Built Environment and Information Technology

PYTHON ROUND2

Architectural Design

Dewald de Jager	15188800
Johan du Plooy	12070794
Juan du Preez	15189016
Marthinus Hermann	15081479
Maria Qumayo	29461775
Orisha Orrie	13025199
Azhar Patel	15052592

STAKEHOLDERS

Computer Science Department
of University of Pretoria

Vreda Pieterse

Contents

1 Introduction

1.1 Purpose

1.2 Scope

2 External Interface Requirements

2.1 User Interface

2.2 Hardware Interface

2.3 Software Interface

2.4 Communication

3 Performance Requirements

The system should be able to run in the following way:

- The system will be run through campus servers as well as web servers.
- The performance of the application will be dependent on the users hardware device.
- The loading time of each page will be dependent on the strength of the users internet connection as well as the number of users currently using campus Wi-Fi.
- The time it takes to log in will be dependent on the internet connection.
- All new users should be added to the system in less than a second however, it may take the user longer to see this, depending on the internet connection.
- Users should be sent notifications as soon as they are available on the system.
- Admin should be able to access the GIS module as well as user module with ease in less than a second as these details will be located on campus servers.
- Points of interest should appear as soon as a user searches for a location or searches for a specific point of interest however, this will once again be dependent on the internet connection.
- Users should be able to search, delete, save as well as modify their route. This should be able to take place quickly. Saving and modifying will be dependent on the user and searching and deleting will be dependent on the internet connection.
- Rendering display service will differ from device to device while requesting service relies on the internet connection.
- Push services for events should happen as soon as the event has been added to the application

- Fitness service and heat maps should be updated in real time.

4 Design Constraints

5 Software System Attributes

5.1 Reliability

When transferring data from point A to point B, it is crucial that the integrity of every piece of information be maintained and accounted for. One piece of misinformation could very well mean the difference between the locations of two different class rooms or even completely different buildings. One of the applications features is to ensure the fastest route, which relies heavily on making the right decisions based on the data it receives, therefore if the information is not accurate or even correct, the user could end up late to their desired destination.

5.2 Robustness

All functionality of NavUP must work correctly even under exceptional conditions; The most important being correctness when there is no Wi-Fi connection, extremely slow Internet connectivity or low availability of resources on either the host system for each access channel or the server.

The data module should provide means of communication even when the server is under strain in order to be robust. This may be done in terms of a pipeline system where messages are queued and receipt of them are acknowledged, along with the error detection and correction discussed under Reliability.

5.3 Efficiency

Sending data between remote locations relies heavily on the network traffic and its capabilities/limitations. Thus the idea would be to put the least amount of stress and congestion on that network and only make strategic requests from client to server. The idea would be to make relatively frequent requests to small yet crucial information, and much less frequent requests to information that is less likely to change within a small period of time.

5.4 Interoperability

5.5 Maintainability

5.6 Testability

5.7 Portability

5.7.1 Hardware Independence

Seeing as how the system will be deployed with a web base as a back-end API, the hardware choices are not limited to a specific set of components, so long as they meet the minimum or preferably recommended requirements to be able to handle the estimated network traffic to keep the system performing at optimal levels.

5.7.2 Software Dependence

The software on the other hand will be limited to being web specific, i.e. being able to use or integrate with applications like AJAX, Node etc.

5.8 Reusability

5.9 Modularity

5.10 Cohesion

5.11 Coupling

The subsystems should have minimal impact on and dependency on the other subsystems of the system. The system as a whole applies the low coupling principle. Subsystems should be abstracted and implementation changes should only affect the subsystem that was modified. This abstraction layer will provide an interface between all the subsystems that prevents interference and augments communication between them.

The data module will be represented as a module with requests and responses that are used both by access channels and the server. This allows for other subsystems to easily use it without having to worry about implementation specific details and the implementation can be changed to use a different technology or update the application programming interface (API).

6 Technologies

6.1 Data

Message passing is done in a transactional manner in this subsystem. The access channels, or clients, need a way to send requests to the server and receive responses. Likewise, the server needs a way to service requests and issue responses. The University of Pretoria Wi-Fi infrastructure can be advantageous in this scenario as we can assume the user is likely connected to the Wi-Fi and fall back on cellular data otherwise. This is done by having the server accessible both locally and publicly via the Internet. If a connection cannot be established to the local server a connection via cellular data or a different Wi-Fi connection is made to the server.

The connection is established via either the HyperText Transfer Protocol (HTTP), or it's secured counterpart HTTPS that makes use of the Secure Socket Layer (SSL) depending on the nature of the message contents, and is maintained only for the duration of the message send and receive process. The requests will, more specifically, be made using Asynchronous JSON and XML (AJAX) and will always be paired with a response. The protocol that HTTP is built on, the Transmission Control Protocol (TCP), ensures the delivery of messages in the correct order and that the messages remain intact.

6.2 Users

6.3 Events

6.4 Points of Interest

7 UML Diagrams

7.1 Data

7.2 Users

7.3 Events

7.4 Points of Interest