Software Requirements Specification

Andromeda

COMS3002

Group 1

28 August, 2017

Team Members	
Harvey Muyangayanga	941446
Jason Parry	1046955
Uvir Bhagirathi	1141886
Zaeem Nalla	1178454

Contents

1 Executive Summary

The Software Requirements Specification document defines the requirements of the proposed system to be designed by Andromeda for the SchedEx project - scheduling examinations for various college/university courses in various venues and time slots - and recognises each requirement as either Functional (sub-categorised into Mandatory or Optional) or Non-functional.

The software solution will serve as a portal for all the identified users: students, lectures, course coordinators and administrators. It will create an optimal exam timetable and then provide them with relevant information and features based on this such as personalised timetables, Google calendar event creation, exam details and navigation to the venue.

The Formal Requirements Definition tabulates the requirements under key categories: Interfaces, Functional Capabilities, Performance Levels, Reliability, Security/Privacy and Quality. This classifies the functionalities that the system should consist of accordingly and in a manner that assists the development team during the design phase of the system. Thereafter constraints of the project are listed as a consideration of pseudo-requirements that should be respected and adhered to.

A useful conclusion which can be compiled from the list of functional requirements is a Use Case Set consisting of use cases grouped by the business features to which they are related i.e. Core (e.g. generating a new timetable), Maintenance (e.g. updating course preferences), Reporting (e.g. a report of clashes - if any exist) and Archiving (e.g. storing old timetables). This shows the actions that needs to be carried out and that the system should support in order for the functional requirements to be captured.

2 Project Scope

The project scope is based around implementing a software system to schedule class-examination to examination venues. The system will take into consideration details such as class size, preferred examination date and time when scheduling the examination and venue. Once a examination timetable is generated students and lecturers will be provided with a personalised version of the timetable consisting of just the courses he/she is associated with.

The purpose of this project is to provide a robust examination scheduling software to be used by tertiary education facilities with minimum examination clashes. If an examination clash occurs then the relevant personnel will be notified and will be corrected manually.

This project contributes to the need for an examination scheduling solution that's capable of generating an optimal timetable with minimal clashes and as accommodating as possible for the various course combinations for students and lecturers alike.

The purpose of the project (i.e. exam scheduling) has the expectation of ensuring that each set of exams at a tertiary (or other similar) institution has a minimal amount of clashes and fair spread of a degree's exams across the examination period (i.e. a student won't have a situation where they'll have to write all 4 of their exams in a week given the examination period is four weeks)

The goals of the project include the creation of a web application that optimises the exam scheduling process and reports to the system actors with meaningful information, creating the system in such a manner that it is aligned with the client's requirements (functionality), making a robust and reliable system, meeting phase deadlines, planning thoroughly according to the SDLC and operating cohesively as an organisation to provide the requested product.

The deliverables of the project include the fully functional web application in the allocated time frame, documents detailing the various phases of the project (including iterations) which will provide the foundation for the final product, presentations which will provide user-centric information to the clients and proof of concept and feasibility of the project, detailing and implementing the many requirements of the project (including but not limited to the mechanical data processing requirements i.e. identifying and getting the business processes and work-flows up and running, and reporting requirements) which will lead to a successful system.

The project is time sensitive and, as such, should be completed before the required deadline. This will be achieved by using an Agile approach to the Software Development Life Cycle, specifically, the SCRUM methodology. This will ensure that each phase of the project gets the required attention leading to a project of high standards which is complete and considered.

Andromeda hopes to provide a robust platform that will ease the process of exam scheduling and be a useful solution for tertiary institutions.

3 Product Description

The SchedEx examination scheduling system is designed to schedule examinations for various college/university courses in various venues and time slots without clashes (if possible) whilst taking certain optimisations into account.

3.1 User characteristics

The following text mentions the purpose of the system for each type of user that's identified to interact with the system:

Student
The system creates a clash-free timetable (if possible) and allows the student to view the full timetable as well as a personalised calendar version. If the system is unable to create a clash-free timetable for the student, he/she is notified and is able to see the relevant clash(es).

Lecturer
The system creates a clash-free timetable (if possible) and allows the lecturer to view the full timetable as well as a personalised calendar version. If the system is unable to create a clash-free timetable for the lecturer (regarding the courses he/she lectures), he/she is notified and is able to see the relevant clash(es).

Course coordinator
The system allows the course coordinator to enter a preference for when and where they wish their exam to be scheduled. The system creates a clash-free timetable (if possible), taking these preferences into account and allows the course coordinator to view the finalised timetable as well as a report on clashes if any are present.

Administrator
The system allows the administrator to generate the provisional and final timetable on demand, let's them view the timetable and gives them the ability to edit or create course preference (for the course's exam) on behalf of a lecturer.

4 Formal Requirements Definition

4.1 Requirements

The various tables below will list and describe the requirements as understood by Andromeda. Each table will list various requirements (grouped by category), each followed by columns that classify the requirement as either Functional (**F** - core functionality that is integral to the system) or Non-functional (**NF** - the methods in which the core functionality will be delivered) as well as either Mandatory (**M** - requirements key to project success) or Optional (**O** - requirements that are not necessarily key to a successful project but may improve on or add to the final product) in the case of functional requirements.

	Interface	**F**	**NF**	**M**	**O**
1.	Provide a graphics intensive interface (GUI)		X	X	
2.	The interface should be intuitive which will facilitate ease-of-use		X	X	

	Functional Capabilities	**F**	**NF**	**M**	**O**
1.	Enable the creation of a clash-free (if possible) examination timetable	X		X	
2.	Allow a student to view their examination timetable	X		X	
3.	Enable a course co-ordinator to enter preferences for their course examination	X		X	
4.	Allow a lecturer to view the examination timetable for the courses associated with them	X		X	
5.	Allow any user to view the whole examination timetable	X		X	
6.	Allow users to view a clash report if any are present after timetable creation	X		X	
7.	Create Google Calendar events to enable users to view the personalised timetables (mentioned above)	X		X	

	Performance Levels	**F**	**NF**	**M**	**O**
1.	The system should be able to handle the traffic that is expected without severe drops in performance		X		X
2.	The system should be built and optimised according to the performance of the computing device architecture provided by the university in order to ensure that all students are able to easily access their timetables		X	X	
3.	The algorithm to create examination-venue-time slot combinations should run efficiently		X	X	

	Reliability	**F**	**NF**	**M**	**O**
1.	The system should provide reliable access to students by being constantly available (aside from necessary maintenance downtime)		X	X	
2.	The system should maintain data consistency so as to ensure that functional requirements are carried out with the most recent and correct data		X	X	

	Security/Privacy	**F**	**NF**	**M**	**O**
1.	A student should have to login (with a valid username and password) before private information is accessible		X	X	
2.	User levels can be used by the system to distinguish between students and administrators, offering more options to the administrator and being more restricted towards the student		X	X	
3.	User levels can be used by the system to distinguish between students and administrators, offering more options to the administrator and being more restricted towards the student		X	X	

	Quality	**F**	**NF**	**M**	**O**
1.	The system should maintain professional build quality which may encourage implementation and use of the platform		X	X	

4.2 Assumptions

The list below contains some constraints (or limitations) that affect the system:

- The system has access to the institutions personnel and student databases - students, course coordinators/lecturers and administrators do not have to register in order to use the system.

4.3 Dependencies

The list below contains some functionalities that the system depends on:

- A valid course listing/database with related courses (i.e. courses that can be done simultaneously) clearly indicated - the number of clashes which the system should minimise and generation of a timetable is dependent on this.
- Internet connection - the system is based on a client-server architecture with the server being centralised and remote.
- Google Calendar API - if the API drastically changes, the system won't be able to add exam events for students, lecturers and course coordinators.

4.4 Operating Environment

- **Architecture**: web-based application (client-server) - three-tier
- **Operating System**: Windows, Linux, MacOS, Android, iOS - any OS wherein a web browser can be used since a web application is being developed
- **Database**: SQLite database
- **Database Type**: centralised
- **Platform**: Python/Javascript/HTML/CSS - used to develop a web app to be accessed through a web browser

4.5 Data Management

Data management details how the input and output of the system will be stored. The majority of the data relating to the operation of the SchedEx system will be stored in a relational database. The details of the staff and students are stored in the database of the institution at which the system is being used. The data which is received and generated by the SchedEx system (e.g. course preferences and timetables) is also stored in a database and is received using the business layer of the three-tiered web application.

4.6 Portability

The system should be able to run on any platform with a web browser.

5 Use Cases

5.1 Use Case Set

The list below represents the project Use Case Set identified for the system. The Use Cases are grouped by the business features that they are related to (core, maintenance, reporting and archiving) because this will help with administration purposes. For example, if a new use case has to be added due to system improvements or changes then it will be easier to see where it fits in.

Core business-related use cases:

1. Create Timetable
2. Create Course Preference
3. Read Lecturer Timetable
4. Read Student Timetable
5. Read Timetable

Maintenance-related use cases:

6. Update Timetable
7. Update Course Preference

Reporting-related use cases:

8. Create Clash Report

Archiving-related use cases:

9. Archive Timetable

Actors identified that will link to the Use Case Set include: Student, Lecturer, Course coordinator and Administrator

5.2 Use Case Description

1. **Create Timetable**
   Basic Flow: The administrator clicks the Generate Timetable button. The system utilises our scheduling algorithm to match courses and venues with non-conflicting examination slots. The system stores the generated timetable. Thereafter, the system proceeds to create a personalised timetable for each student by creating entries in the respective student's Google Calendar (linked to their student email account).
   Alternate Flow: If the algorithm is unable to create a clash-free timetable, the best possible timetable is created, a clash report is logged in the database and the administrators and affected students receive an email containing a clash report.

2. **Create Course Preference**
   Basic Flow: The course coordinator clicks the Create Preference button. The system receives and stores input from the course coordinator on date, time, class size, lecture venue, faculty building.

3. **Read Lecturer Timetable**
   Basic Flow: The lecturers personalised timetable is retrieved from his/her exam related events that were added to his/her Google calendar (associated with his/her university account) and displayed. Navigation to the exam venue for a selected course is displayed beneath.
   Alternate Flow: If the exam timetable (provisional or final) hasnt been created yet then a personalised timetable will be irrelevant and text notifying the lecturer of this will be displayed instead.

4. **Read Student Timetable**
   Basic Flow: The students personalised timetable is retrieved from his/her exam related events that were added to his/her Google calendar (associated with his/her university account) and displayed. Navigation to the exam venue for a selected course is displayed beneath.
   Alternate Flow: If the exam timetable (provisional or final) hasnt been created yet then a personalised timetable will be irrelevant and text notifying the student of this will be displayed instead.

5. **Read Timetable**
   Basic Flow: The administrator logs in or the student, lecturer and course coordinator requests to view the full timetable. The stored timetable in the database is then displayed in a friendly format (list or a calendar). Navigation to the exam venue for a selected course is displayed beneath.
   Alternate Flow: If the exam timetable (provisional or final) hasnt been created yet then a personalised timetable will be irrelevant and text notifying the admin, student, lecturer or coordinator of this will be displayed instead.

6. **Update Timetable**
   Basic Flow: When a new Course Preference is created the Update Timetable use case is invoked for the algorithm to take into account the new course preference.

7. **Update Course Preference**
   Basic Flow: The course coordinator or admin clicks the Update Preference button. The system receives, searches and then updates the relevant course preference.

Alternate Flow: If the course coordinator or admin tries to update the course preference of a course that doesnt have any preferences set, the Create Course Preference use case is invoked.

8. **Create Clash Report**
   Basic Flow: The administrator clicks the Generate Timetable button which creates the timetable. A report of clashes is then generated and displayed to the administrator as well as emailed to the students and lecturers associated with the courses.
   Alternate Flow: If there are no clashes one the timetable has been created, the administrator is informed of this through the system.

9. **Archive Timetable**
   Basic Flow: The admin clicks the Generate Timetable button. The Create Timetable use case is then invoked. The timetable is then archived by storing the dates, times and venues for each course in the database.