Project Report

Andromeda

COMS3002

Group 1

2 October, 2017

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

Contents

1 Introduction

This document contains information on SchedEx, an examination-venue scheduling software to be implemented into university/college institutions. This includes the requirements specification which is composed of the project scope, product description, formal requirements definition and use cases. The system design comprises the domain model, informal description, system use cases and system framework. The system implementation contains the dependencies, testing and pair work as well as the presentation, business and data layers.

1.1 Project Overview

A class-examination to venue scheduling system software is required to be implemented to aid university/college institutions.

1.2 Problem Statement

We want our examination Time-table scheduling software to allocate examinations to venues on the recommended date-time slot (if possible), maximising student-venue capacity and minimising examination clashes.

Recommended date-time slots may be the same for different examinations. If this is ignored when generating the time-table then examination clashes can be made resulting in an unorganised timetable and overbooked venue.

Separate venues will have to be allocated to clashing recommended date-time slots and should be ignored when it is not possible to assign that examination to the recommended date-time slot. We will use a graph colouring algorithm to minimise examination-venue clashes.

1.3 Project Objectives

The system will need to produce a clash free examination time-table, taking the following into consideration for each course:

- Recommended examination date-time slot
- Number of students registered
- Current venue where lectures take place
- Main buildings where the faculties of the courses are located

1.4 Project Stakeholders

The following is a list of stakeholders:

- Students
- Lecturers
- Course Coordinators
- Administrators
- University Management

2 Requirements Specification

2.1 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.

2.2 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.

2.2.1 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.

2.3 Formal Requirements Definition

2.3.1 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 down-time)		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	

2.3.2 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.

2.3.3 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.
- Python networkx - this package is used to graph the required data to generate the examination-venue timetable.
- Python sqlite3 - this library is used to interact with the institution's database.

2.3.4 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

2.3.5 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.

2.3.6 Portability

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

2.4 Use Cases

2.4.1 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

2.4.2 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.

3 System Design

The design for the system is provided below. The first sub-section includes UML-based logical models whilst the second explains the design, providing specific details about each of the major components.

3.1 Domain Model

3.1.1 Informal Description

A system for the scheduling of university or college examinations is to be designed and created. The system is meant to create an exam timetable (optimally) by assigning course exams to a certain venue at a specific time, without creating clashes in the process. The system should function as follows:

- Administrators are able to login, specify start and end dates of the examination period, invoke the generation of a timetable and view the generated timetable.
- Course co-ordinators are able to login, enter preferences relating to the scheduling of their courses' exam(s) and view the final timetable.
- Students and lecturers are able to login to the system and view the full timetable. Additionally, students are able to view a personalised timetable.

3.2 System Use Cases

The system use cases represent the various scenarios (user stories) which occur during the use of the system.

3.2.1 Final Use Case List

The list below represents the final project Use Case Set identified from the Requirements Specification (Section 2.4, above). 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.

Core business-related use cases:

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

Maintenance-related use cases:

7. Update Timetable
8. Update Course Preference

Reporting-related use cases:

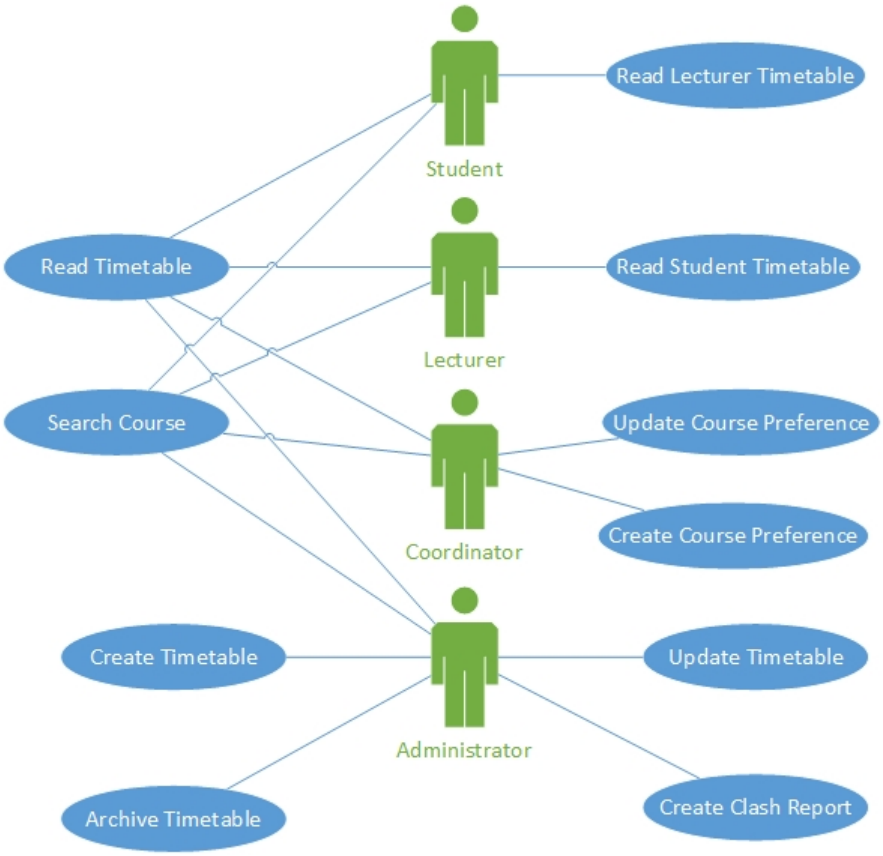
9. Create Clash Report

Archiving-related use cases:

10. Archive Timetable

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

3.2.2 Use Case Diagram



The primary actors in this system are the Administrator, Course co-ordinator and Student whilst the Lecturer is a secondary actor.

The administrator may invoke the Create Timetable use case in order to generate a new timetable or the Update Timetable Use Case in order to update the timetable. Additionally, the administrator may invoke the Create Clash Report use case to generate a list of clashes (if any) and email the affected parties. Finally, the Archive Timetable may be invoked by the administrator in order to archive a previously-generated timetable.

The course co-ordinator is able to invoke the Create Course Preference use case which allows him/her to enter the chosen preferences (date, time and venue) relating to one of their courses.

Students and lecturers invoke the Read Student Timetable and Read Lecturer Timetable use cases respectively to view their personalised timetables based on the courses they take/lecture.

All users (actors) are able to invoke the Read Timetable use case which allows them to view the full timetable (for the whole exam period) which has been generated. Additionally, all users may invoke the Search Course use case which enables them to see the exam details of the course which they have entered.

3.3 System Framework

The design is split up into three distinct sections, namely the Presentation, Business and Data Layers:

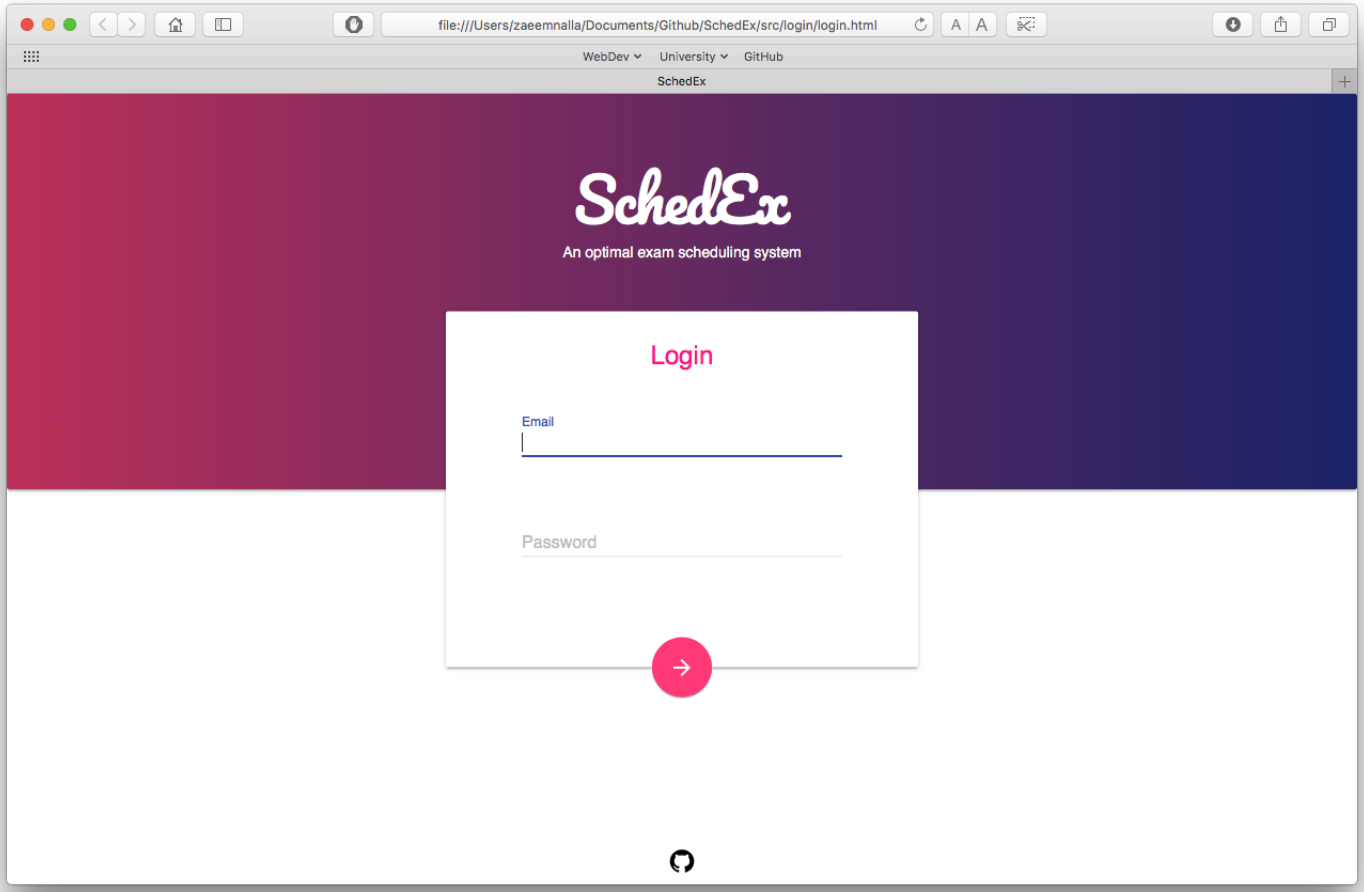
3.3.1 Presentation Layer

The presentation layer involves the user interface of the application which will run on the client side of the system. It is the front end with which the user interacts and all transfer of data between the application and the user occurs in this layer. [¡x¿]

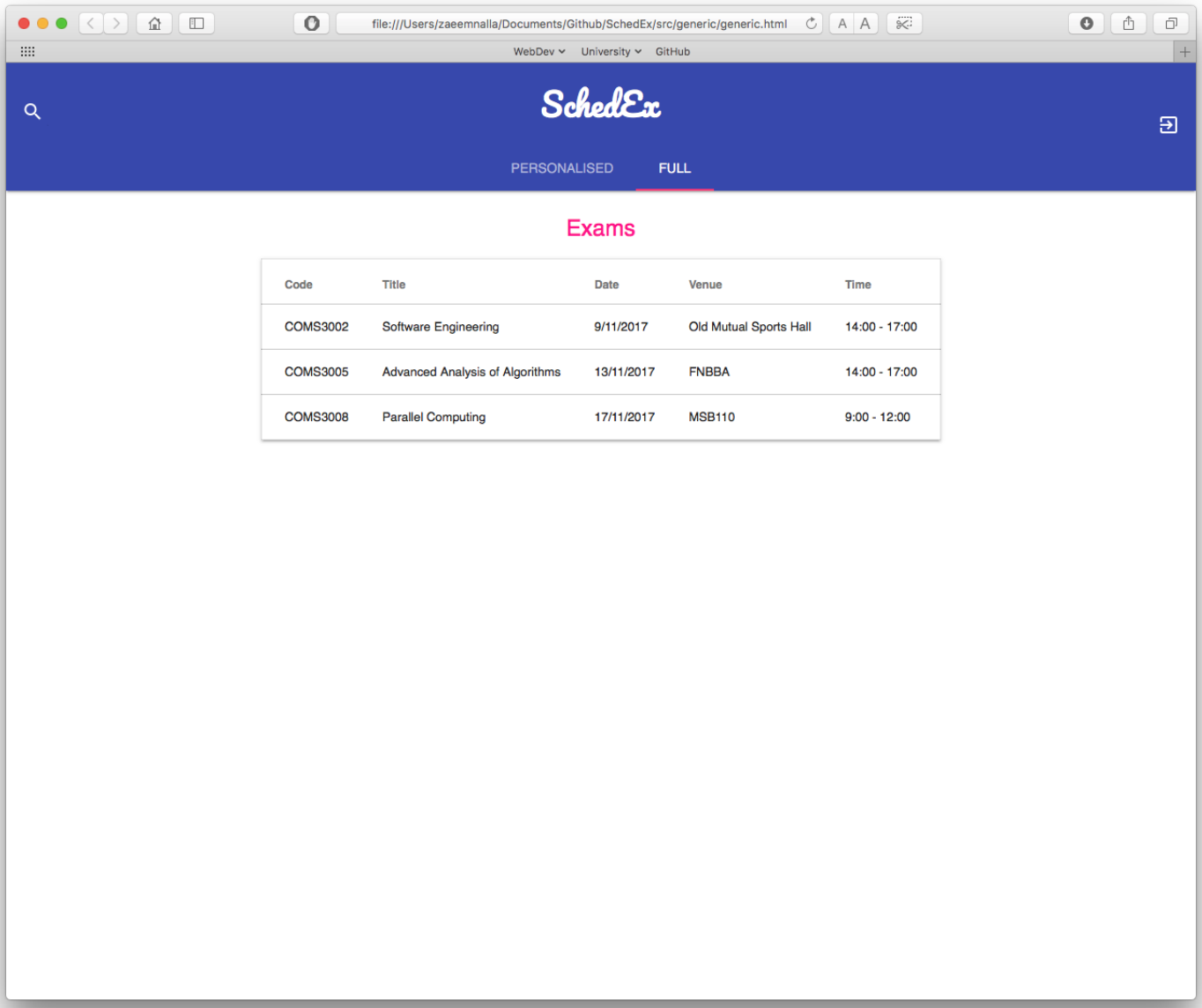
The User Interface is designed to be clean, informative and aesthetically pleasing. There is a logical flow between screens which ensures that the user always knows where they are currently in the execution of the program and

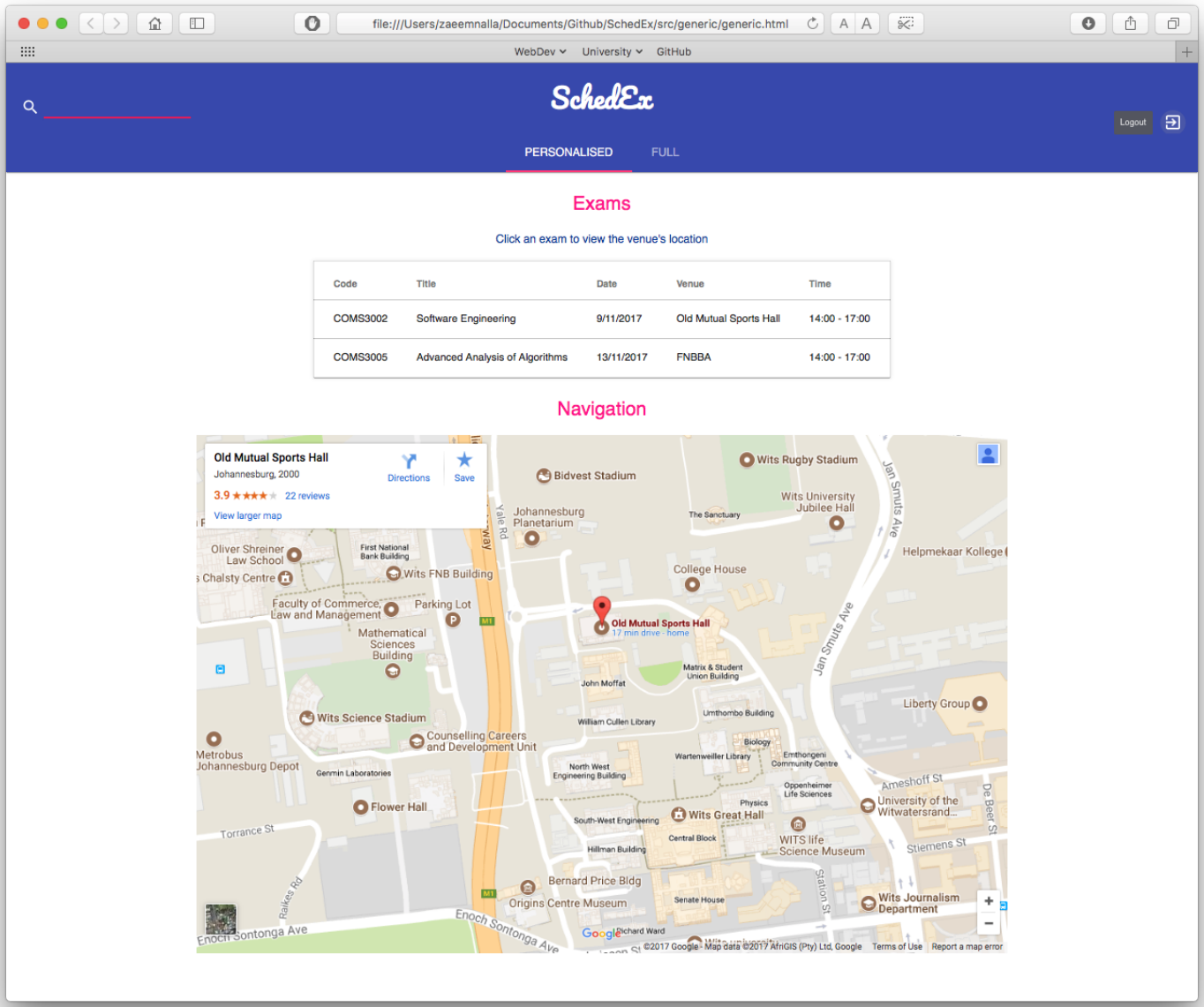
where they can get to from that state.

With the above in mind, the interface has four screens: Login, Generic (for students and lecturers), Admin and Coordinator that fulfil the aforementioned design requirements.

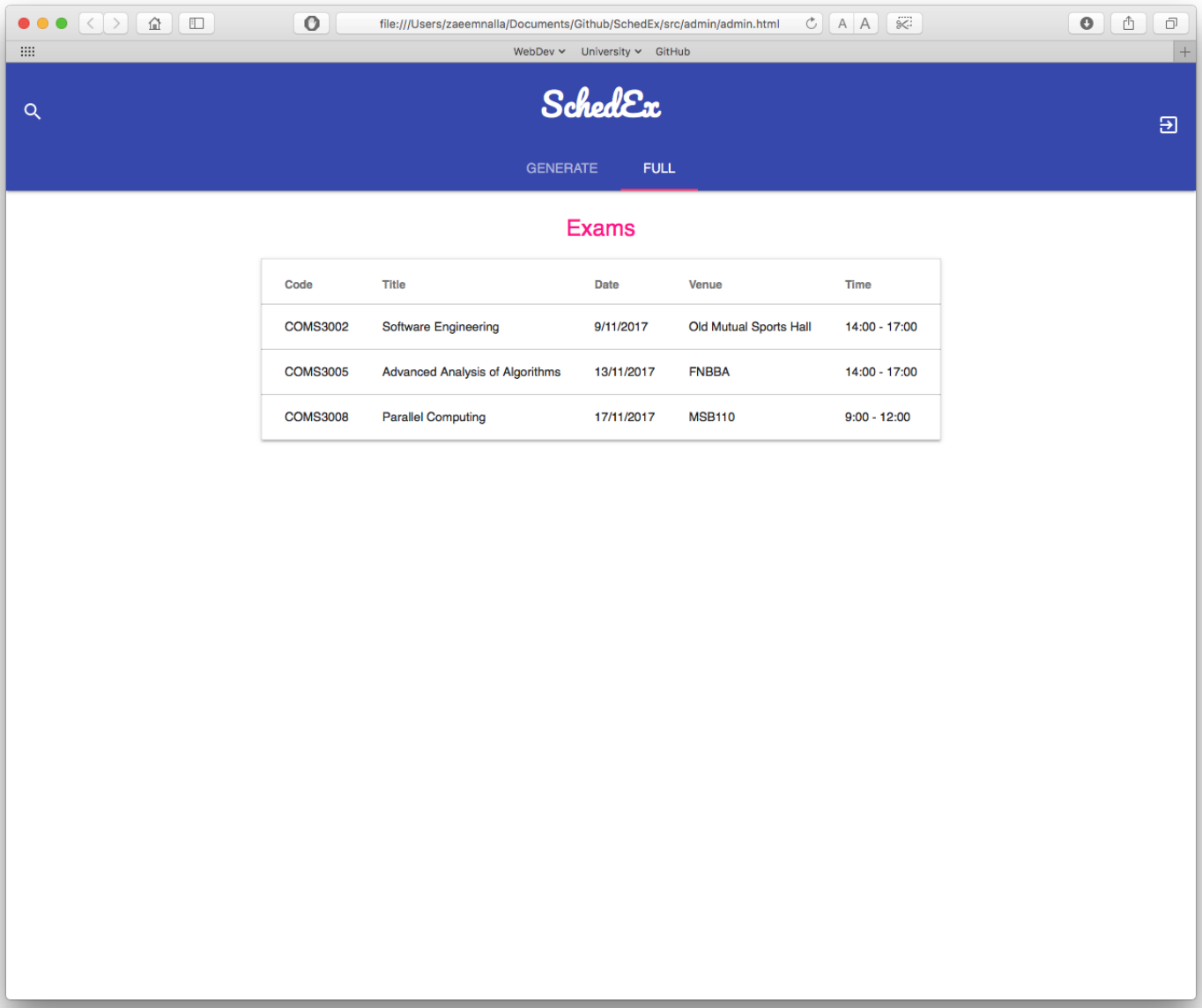


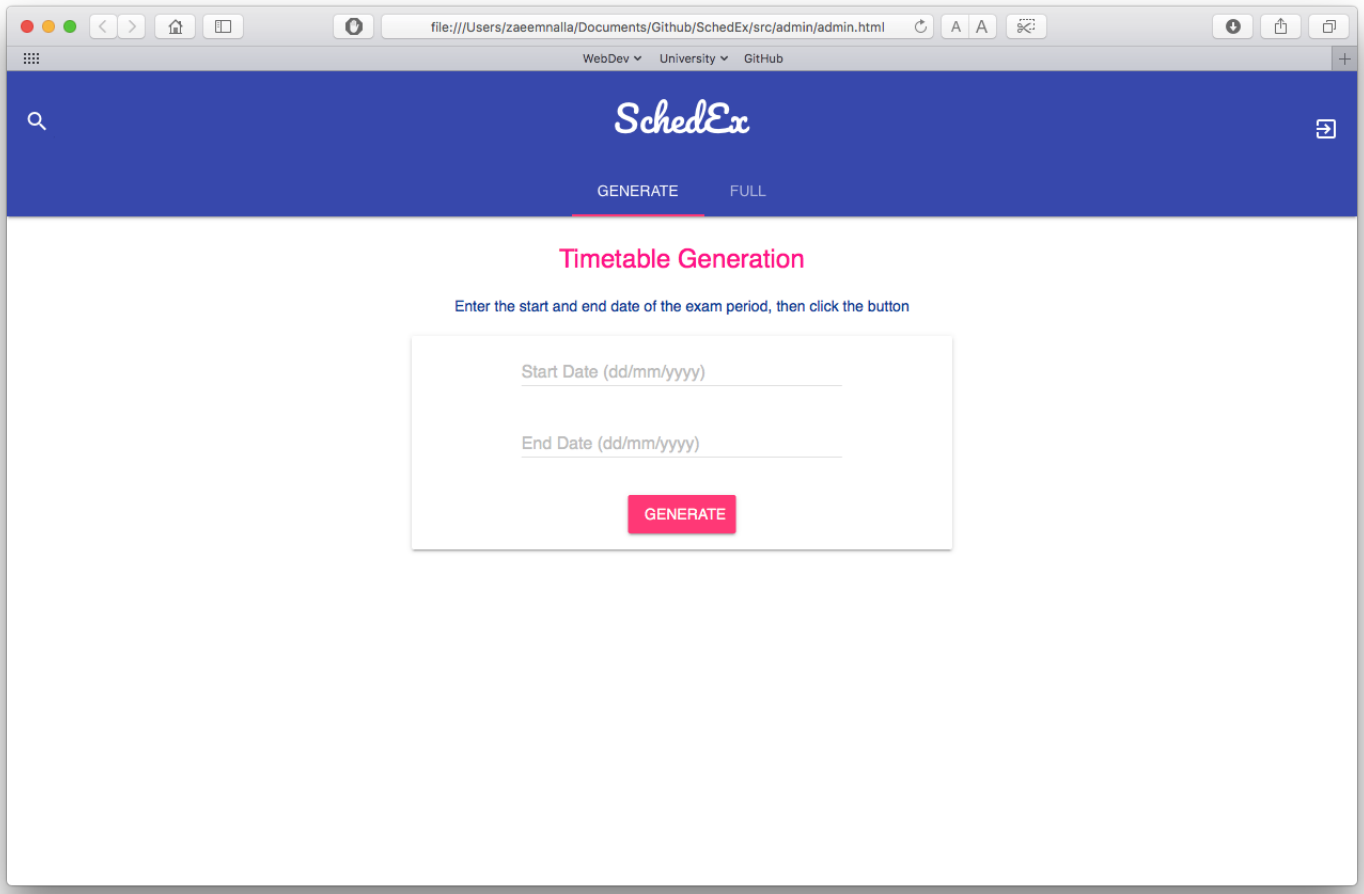
The Login screen enables the user to securely login to the system and, after valid credentials have been entered and the login button is clicked, displays the appropriate screen (Generic, Admin or Coordinator) based on the privilege level of the user - determined by role level in the data layer (Section 3.3.3) associated with the credentials entered.


file:///Users/zaeemnalla/Documents/Github/SchedEx/src/generic/generic.html
WebDev
University
GitHub
SchedEx
PERSONALISED
FULL
Exams
Code
Title
Date
Venue
Time
COMS3002
Software Engineering
9/11/2017
Old Mutual Sports Hall
14:00 - 17:00
COMS3005
Advanced Analysis of Algorithms
13/11/2017
FNBBA
14:00 - 17:00
COMS3008
Parallel Computing
17/11/2017
MSB110
9:00 - 12:00

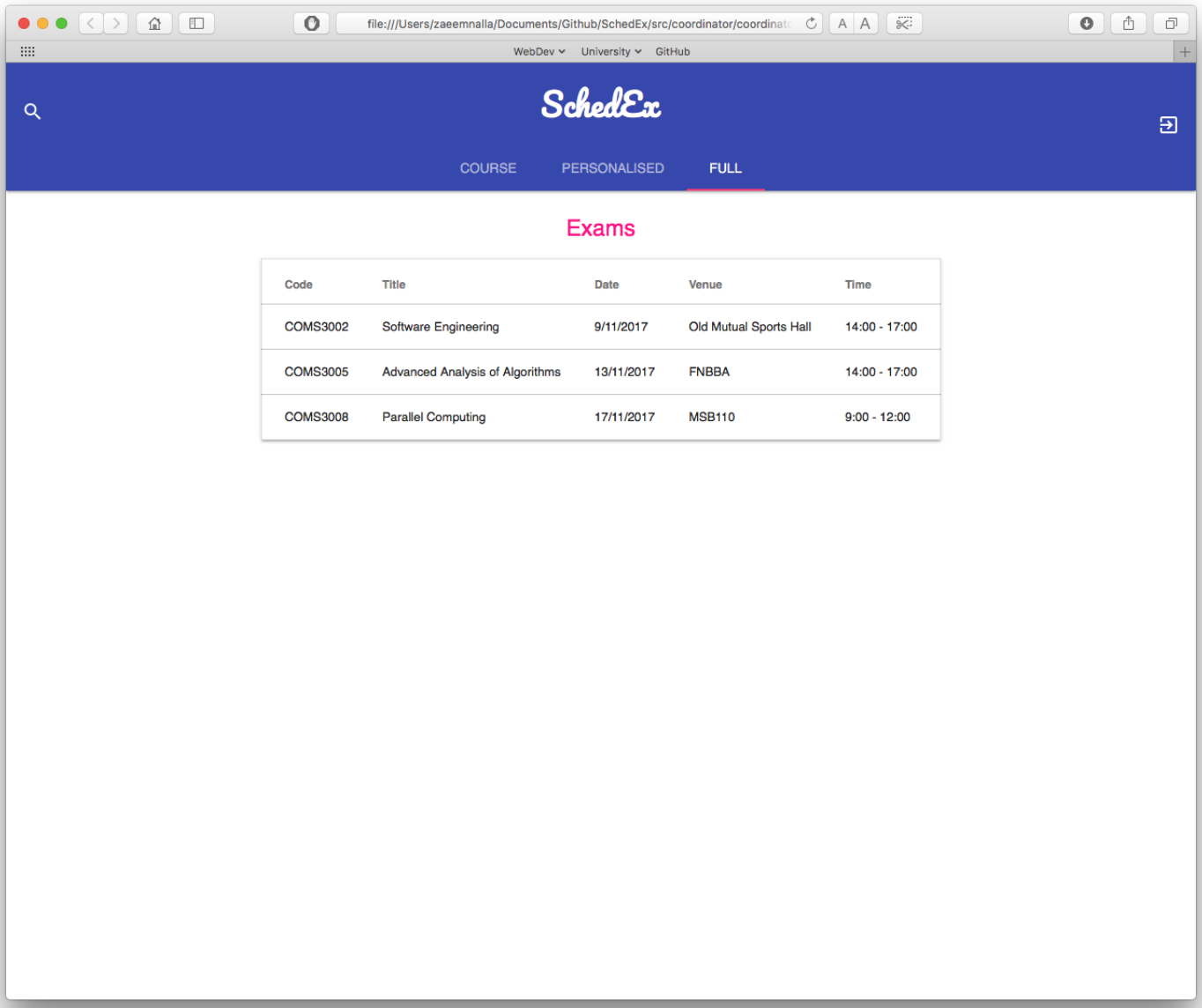


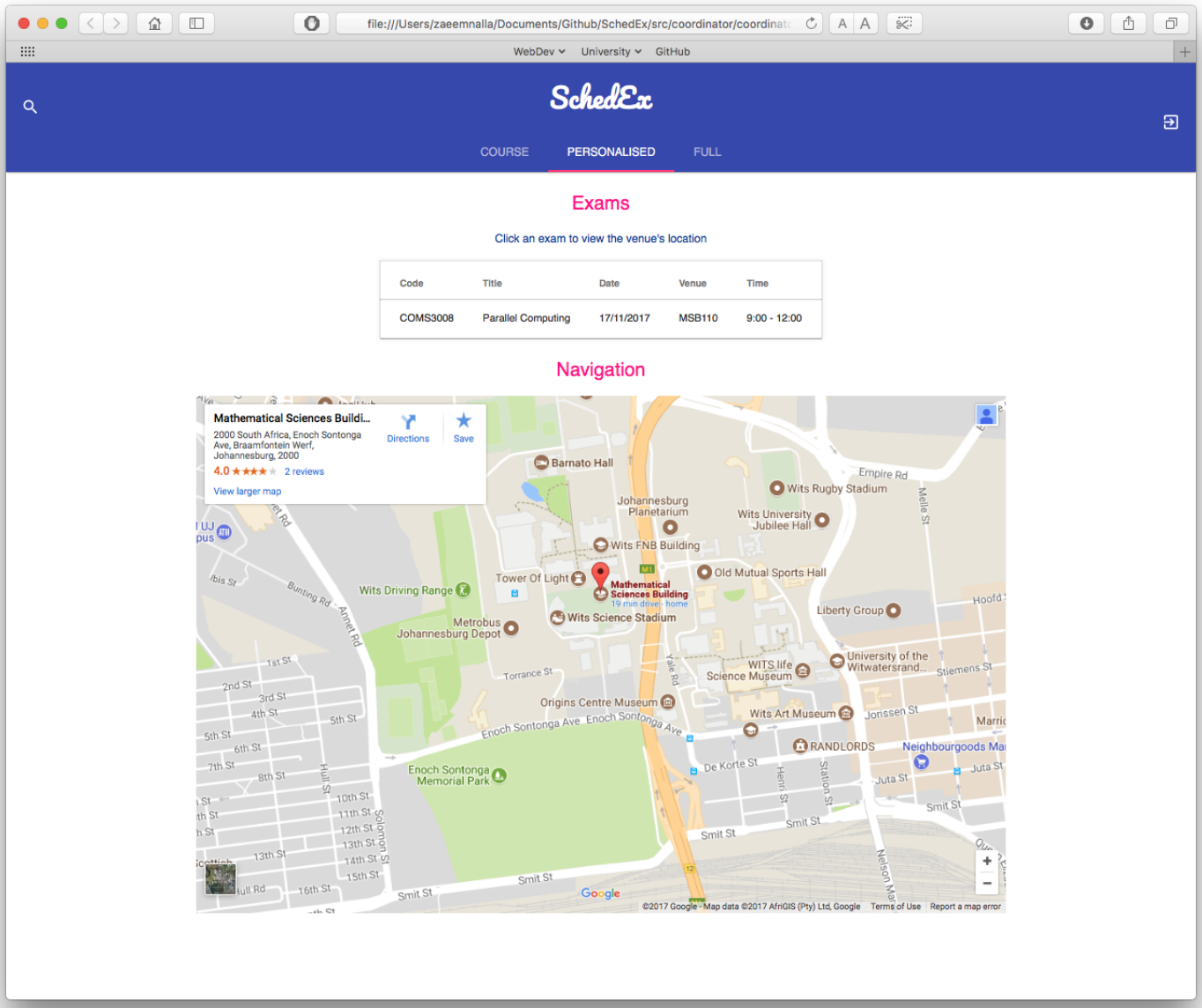
The Generic screen is designed for students and lecturers. The screen facilitates the viewing of the various timetables (both the full timetable, as well as a personalised version). The screen also allows the user to search for a specific course and displays the exam details associated with it.

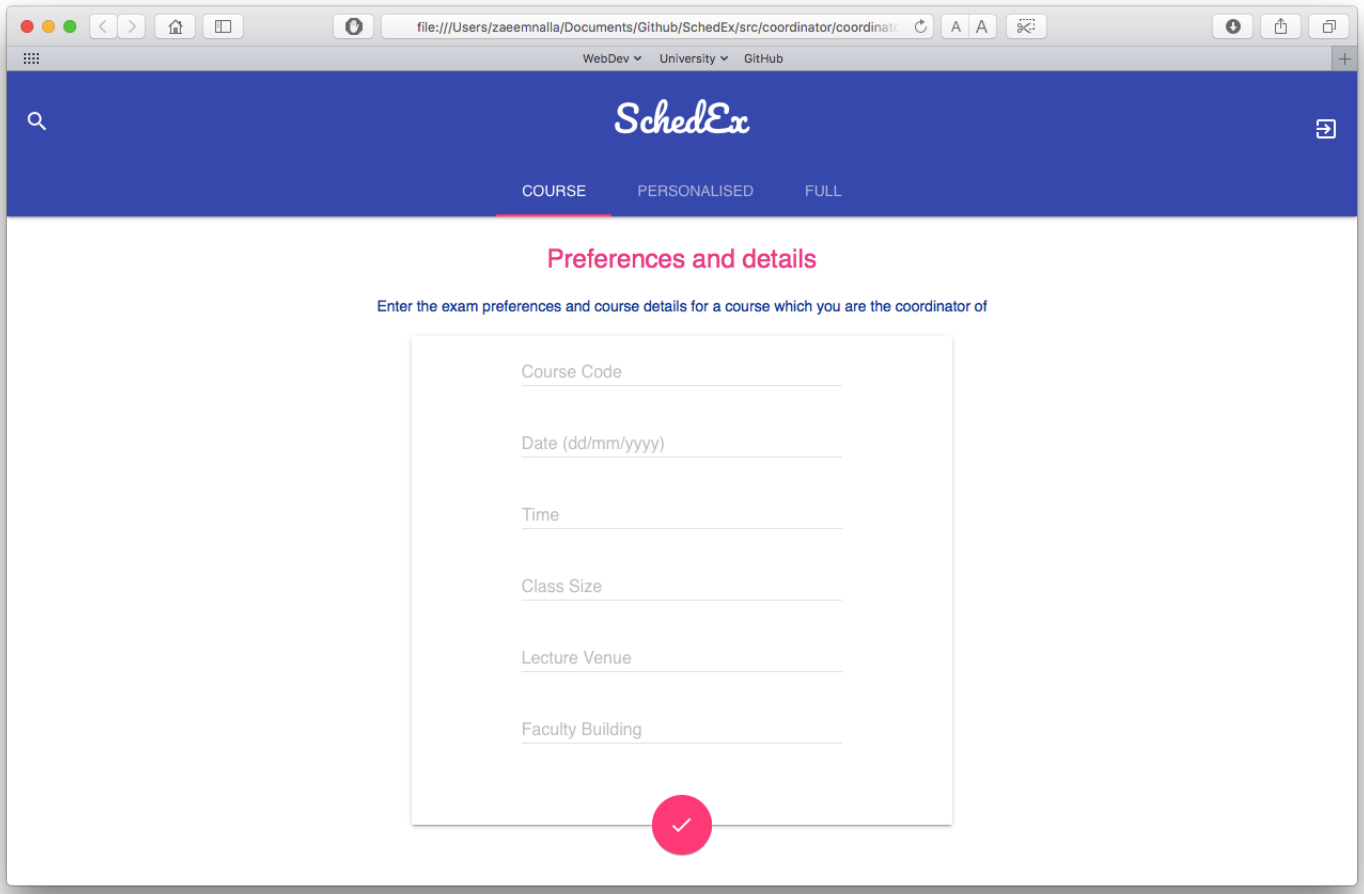

file:///Users/zaeemnalla/Documents/Github/SchedEx/src/admin/admin.html
WebDev
University
GitHub
SchedEx
GENERATE
FULL
Exams
Code
Title
Date
Venue
Time
COMS3002
Software Engineering
9/11/2017
Old Mutual Sports Hall
14:00 - 17:00
COMS3005
Advanced Analysis of Algorithms
13/11/2017
FNBBA
14:00 - 17:00
COMS3008
Parallel Computing
17/11/2017
MSB110
9:00 - 12:00



The Admin screen is designed for university/college administrators. It enables the admin to generate a new timetable through provision of start date and end date fields for the exam period. Additionally, the screen provides the administrator with the option of viewing the generated timetable and editing it, should it prove necessary.


file:///Users/zaeemnalla/Documents/Github/SchedEx/src/coordinator/coordinat
WebDev
University
GitHub
SchedEx
COURSE
PERSONALISED
FULL
Exams
Code
Title
Date
Venue
Time
COMS3002
Software Engineering
9/11/2017
Old Mutual Sports Hall
14:00 - 17:00
COMS3005
Advanced Analysis of Algorithms
13/11/2017
FNBBA
14:00 - 17:00
COMS3008
Parallel Computing
17/11/2017
MSB110
9:00 - 12:00


file:///Users/zaeemnalla/Documents/Github/SchedEx/src/coordinator/coordinat
WebDev
University
GitHub
SchedEx
COURSE
PERSONALISED
FULL
Exams
Click an exam to view the venue's location
Code
Title
Date
Venue
Time
COMS3008
Parallel Computing
17/11/2017
MSB110
9:00 - 12:00
Navigation
Mathematical Sciences Buildi...
2000 South Africa, Enoch Sontonga Ave, Braamfontein Werf, Johannesburg, 2000
4.0
2 reviews
View larger map
Directions
Save
Barnato Hall
Johannesburg Planetarium
Wits Rugby Stadium
Wits University Jubilee Hall
Wits FNB Building
Tower Of Light
Old Mutual Sports Hall
Mathematical Sciences Building
19 min drive - home
Wits Driving Range
Wits Science Stadium
Metrobus Johannesburg Depot
Liberty Group
WITS life Science Museum
University of the Witwatersrand...
Origins Centre Museum
Wits Art Museum
RANDLORDS
Enoch Sontonga Memorial Park
Enoch Sontonga Ave
Torrance St
De Korte St
Smit St
Juta St
Empire Rd
Bunting Rd
Annet Rd
Yale Rd
Jorissen St
Siemens St
Melle St
Hull St
Solomon St
Nelson Ma
Google
©2017 Google - Map data ©2017 AfriGIS (Pty) Ltd, Google
Terms of Use
Report a map error



The Coordinator screen is designed for course co-ordinators. The ability to add course preferences (such as the time, date and requested venue where they wish their exam to be scheduled) is provided by this part of the interface, along with the option to view the full timetable and a personalised timetable (based on the courses for which he/she is a co-ordinator).

As can be seen from the above screen designs and their associated descriptions, the presentation layer of SchedEx follows a simple, clean, systemised methodology which facilitates ease-of-use whilst still providing all of the requested functional and non-functional capabilities of the system.

3.3.2 Business Layer

The business layer is where the functionality of the application is implemented and runs on the server side of the system. It acts as the middle-ground between the presentation and data layers (data is received, processed and sent to or from both the presentation and data layers).

The system is designed with the important principles of user experience in mind including: ease-of-use, practicality and efficiency.

The business layer is set to work as follows: data is received from the presentation layer indicating the specific function which the user is requesting. This data, along with the appropriate class and method, are used to retrieve (from the data layer) or generate the requested data and process it as necessary before returning it to the presentation layer to be displayed to the user.

Object orientation is to be used in the implementation of the business layer in order to create an accurate representation of the 'real-life' scenario. The exact details of this object oriented design may be found in the class diagram below (Section 3.3.4). Each of the relevant entities in the scenario are modelled as a class with variables and methods which enable them to perform their specific task.

The design of the system ensures the various use cases may be executed intuitively. Each request made from the presentation layer is carried out in a well-defined manner using the aforementioned object oriented approach and returns the correct information to successfully complete the request.

The business layer connects to the data layer through one of the classes. This class contains all of the methods necessary to retrieve any data which made be needed for the processing of requests. In this way, an instance of this class may be made and used wherever communication with the data layer is needed.

3.3.3 Data Layer

The data layer acts as the interface with external data storage systems and is stored on the server side of the system. It allows data to be written and retrieved to and from files and other external systems such as databases.

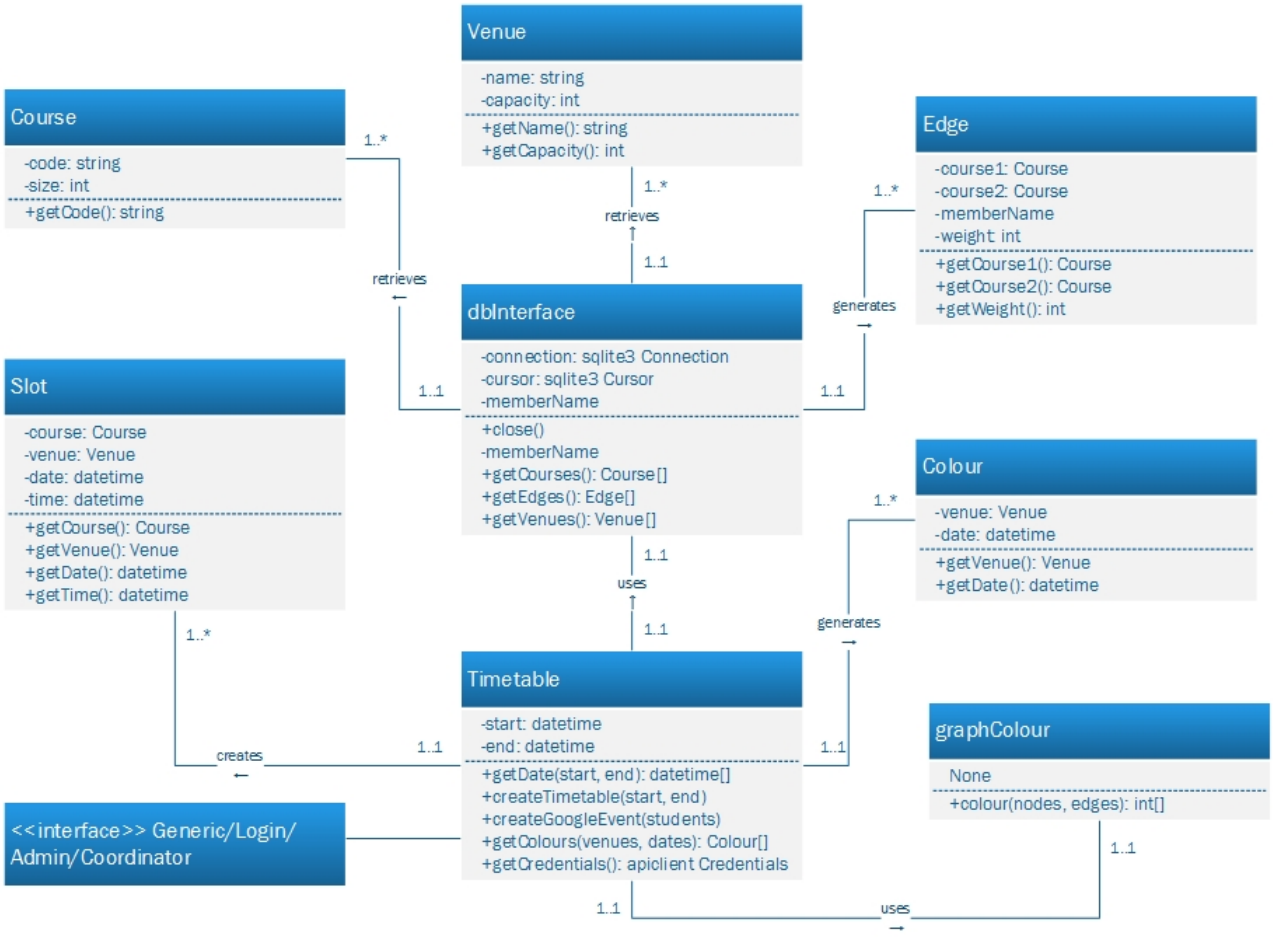
When designing the data layer, it is important to consider, amongst other things, which attributes should be stored and the security this data requires. As per the assumptions made in the Requirements Specifications (Section 2.3.2), the system has access to the institution's data stores in order to obtain various student and course-related data. In addition to this, the system requires its own data stores to act as persistent storage for various aspects of the exam scheduling system.

With the above in mind, the data layer is to be implemented in the form of a database which will maintain all of the relevant attributes. The chosen type of database is relational due to the flexibility it provides, along with security and ease-of-access using Structured Query Language (SQL) querying.

The data layer interacts with the business layer using the intermediary class (as mentioned in the design of the business layer, above). The class makes use of custom SQL queries to retrieve the necessary data from the database tables. The design of the data layer allows this data to be pulled and used in an efficient manner.

3.3.4 Class Diagram

In object-oriented programming the code is implemented as a set of classes. The final step in object- oriented design is to produce a class diagram, which is the detailed description of the code to be produced. The class diagram shows all of the classes in the application, including their associated attributes and methods. The class diagram also shows the relationships (association) between classes.



The figure above represents the design of the system (specifically the business layer) and shows the various classes in the object oriented design.

As can be seen, the dbInterface and Timetable classes make up the backbone of the system. The dbInterface class is used to connect to the database - the intermediary class discussed in the design of the business and data layers (Sections 3.3.2 and 3.3.3) - and creates Course (university courses), Venue (university venues) and Edge (combinations of dates and venues used in graph colouring) objects by pulling the relevant data from the database through an established connection. The Timetable class is the key component in the functioning of the back end. It uses an instance of the dbInterface class to retrieve the necessary data. Thereafter, this data is used to instantiate the graphColour class which takes in the courses and date-venue combinations and uses a custom graph colouring algorithm to create an optimal timetable (if possible). For each match between a course and date-venue, an instance of a Slot is created. The dbInterface class is then used again to store the slots in the database as a Timetable table which allows the presentation layer to display it and perform operations on it for all relevant users.

The class diagram as presented above is the final output of the design process discussed above. The implementation of the system is drawn from the class diagram. The system depicted by the class diagram is discussed in Section 4.

3.4 Pair I - Front End

The front end pair are responsible for the design of the presentation layer. This entails creating mock-ups (wireframes) of the relevant screens. Thereafter, the screens are turned into an actual interface design which includes all of the necessary components. Once the interface deign is in place, the logic to make it work and receive user requests is designed and checked to ensure it will successfully satisfy both the functional and non-functional requirements aspects relating to it. Once the above criteria have been satisfied, the front end is designed and ready for the implementation phase of the project, discussed below.

3.5 Pair II - Back End

The back end pair are responsible for both the business and data layers of the SchedEx system. The business layer involves the creation of a class diagram - through the translation of requirements and use cases to a design that may be implemented in code - which provides a detailed description of the attributes and methods in each class of the object oriented solution. The data layer entails the consideration of various aspects (such as security, data types and querying) which leads to the selection of an appropriate solution and the creation of an entity relationship diagram which provides insight into the relational database, the various tables, attributes, primary and foreign keys and the relationships between each of the entities. Having fulfilled the above requirements, the back end is fully designed and ready for the implementation phase of the project, discussed below.

4 System Implementation

The implementation of the above design is split up into three distinct sections, namely the Presentation, Business and Data Layers.

4.1 Presentation Layer

The general theme of the interface is modern and based on industry standards such as Google Material Design. This is meant to let the user feel comfortable and experience no learning curve since it mimics the appearance of common, standout websites with a card focus heavily in action, as reiterated by Google. The login screen has two input fields for the users credentials (email and password) once entered and the login button is clicked, the system then verifies the credentials by issuing these encrypted details from the client side to the server side. The server side then checks the credentials against records in the database for each possible entity (student, lecturer, coordinator and administrator) and returns a status (true or false) which indicates the validity of the details. If valid the interface will proceed to the relevant view based on the type of user logging in, otherwise theyll be notified of the invalid details.

The generic screen displays two tabs (personalised and full) allowing the respective view of the exam timetable. When personalised is clicked, the client requests a list of the exams related to the student or lecturer. This is done by sending the users institution identity number to the server which then retrieves the list of exams from the relational database and returns it in json format. A table is then populated with these exams and their details for the users convenience and viewing pleasure. When the user clicks on an exam listing, the exam venue of a scheduled exam is extracted from the object and the location is displayed on a map for the users navigation purposes by setting the src attribute of the iframe. The full timetable tab deploys a process of simply requesting the full exam timetable to be sent that is currently stored in the database.

The coordinator screen is the same as above with the extension of a course tab which allows the coordinator to enter course details and exam preferences for the course. When the tab is clicked, a card with the required inputs components are displayed. Once the coordinator enters these details and clicks the button, input validation occurs and if valid, the details are sent to the server to be stored in the database. These are considered by the algorithm when constructing an exam timetable. If invalid, the coordinator is notified of this.

The admin screen has a generate tab along with the full timetable tab. This provides the admin with a button to schedule the system to generate an exam timetable based on all the courses and preferences and start and end dates of the examination period. This occurs on the back-end where the algorithm runs and then stores the timetable in the database.

4.2 Business Layer

The timetable is generated using a greedy graph colouring algorithm that utilizes the networkx python library and retrieves the data using sqlite queries. The data retrieved from the front-end is just the start and the end dates of the exam period.

4.3 Data Layer

The database was created using sqlite and the business layer connects to it using the sqlite 3 python library.

4.4 Dependencies

Reinventing the wheel is not good practice and as a result, the use of external libraries plays a vital part in enabling efficiency during our design and implementation of the system. Frameworks and APIs also allow us to implement certain functionality that connects to external services. On the front-end, libraries such as jQuery, material design lite, material icons and material selectfield all help with the UX and overall appearance of the UI. Material design lite (MDL) and material selectfield are used for the html components and particularly the css styling (through classes) and animation of these components applied through the javascript made available by these libraries. Material icons is used for the inclusion of Google designed icons in our UI such as login, logout and search. jQuery is used for ease of manipulation of DOM elements in javascript. All of these libraries are widely used in industry for modern front-end development. Google maps is also a front-end dependency as it displays an exam venues location. On the back-end, a combination of libraries, packages and APIs are used for implementation. Python sqlite3 is used to read data (concerning entities within the academic institution) and write data (concerning course preferences and the generated exam timetable) to the database. The lightweight nature of this DBMS made it a perfect fit for the system. Python networkx performs the graph-colouring algorithm on our graph of courses in order to generate the exam timetable. Google Calendar API allows us to add exam events on the Google account for each related entity.

4.5 Pair I - Front End

The tools used to perform the tasks in section 3.5 are Axure for the wireframe and html, css, js for the interface design.

4.6 Pair II - Back End

The tools used to perform the tasks in section 3.5 are Microsoft Visio for the class diagram and entity relationship diagram.

4.7 Testing

Testing is a critical part of the development of any system as it ensures the functional and non-functional requirements of the project are met and that the system functions as intended without any serious issues or bugs. Both black box and white box testing are employed in this system as detailed below.

4.7.1 Server

Verification was done to ensure the various back end components such as the database connectivity, data retrieval, Google calendar events and timetable generation work correctly.

Unit tests were conducted to ensure that, given certain parameter values, the algorithm produces the correct results. These algorithm was first traced by hand and then the unit tests were used to assert whether the hand calculations and the output of the functions do match.

The output of the Google calendar event creation was also monitored by observing whether an event was actually created given a valid Google account.

Both of the above tests successfully completed and verified that the server-side works correctly.

4.7.2 Interface

Unit tests are used to test each of the interface views i.e. generic (used for students and lecturers since they share mutual features), coordinator and administrator. This is done by entering pre-defined login credentials for each type of user on the login screen and observing the resulting interface transition. This ensures the relevant interface view and related components are created based on the type of user so that the user is given the system features that they require.

The interface view is then tested by a user. The user attempts to use the interface to navigate between the various screens. The user clicks on various places on the screens to ensure that no unintended behaviour occurs and then clicks on an available button to test that this is the only place where a change due to a mouse click occurs.

The above tests are completed successfully and determine that the interface works correctly according to the design.

5 SCRUM Sprints

5.1 Project Sprint Planning

5.1.1 Team

1. Project Owner - the project owner was determined to be the lecturer and his instructions would come from emails, consultations and project brief.
2. Scrum Master - the Team Lead (Harvey Muyangayanga) also assumed the role of SCRUM master, insuring that the due dates are kept and that the sprints are completed.
3. Development Team - the team of four students was split into two pairs: Front-end and Back-end. The front-end team includes Harvey Muyangayanga (941446) and Zaeem Nalla (1178454). The back-end team includes Jason Parry (1046955) and Uvir Bhagirathi (1141886).

5.1.2 Sprint

It was decided that the project will run over four sprints and each of them will have a duration of roughly two weeks. This however varies depending on due dates.
Sprint 1 was scheduled to run from the 10th of August to the 24th of August.
Sprint 2 was scheduled to run from the 25th of August to the 1st of September
Sprint 3 was scheduled to run from the 4th of September to the 18th of September
Sprint 4 was scheduled to run from the 18th of September to the 2nd of October

5.1.3 Stand Ups

Stand up meetings were held on Mondays for a period of fifteen minutes and were replaced on the other days of the week by video conferences, Slack messages and emails to keep everyone updated according to their availability.

5.1.4 Detailed Sprint Execution

Sprint 1

Initial Documentation (SCHED-2)

[SCHED-9] Type out Initial document Created: 10/Aug/17 Updated: 24/Aug/17 Resolved: 24/Aug/17

Status:	Done
Project:	Schedex
Component/s:	None
Affects Version/s:	None
Fix Version/s:	None

Type:	Sub-task	**Priority:**	Medium
Reporter:	Harvey Muyangayanga	**Assignee:**	1178454
Resolution:	Done	**Votes:**	0
Labels:	None		
Remaining Estimate:	Not Specified		
Time Spent:	Not Specified		
Original Estimate:	Not Specified		

Sprint:	SCHED Sprint 1

Description

As per team agreement, this includes work allocation for the front end and back end, project specifics(naming, repository,...), inputs and outputs and optional features.

[SCHED-2] Initial Documentation Created: 10/Aug/17 Updated: 24/Aug/17 Resolved: 24/Aug/17	
Status:	Done
Project:	Schedex
Component/s:	None
Affects Version/s:	None
Fix Version/s:	None

Type:	Story	**Priority:**	Medium
Reporter:	Harvey Muyangayanga	**Assignee:**	Unassigned
Resolution:	Done	**Votes:**	0
Labels:	None		
Σ Remaining Estimate:	Not Specified	**Remaining Estimate:**	Not Specified
Σ Time Spent:	Not Specified	**Time Spent:**	Not Specified
Σ Original Estimate:	Not Specified	**Original Estimate:**	Not Specified

Sub-tasks:					
	Key	**Summary**	**Type**	**Status**	**Assignee**
	SCHED-6	Logo Design	Sub-task	Done	Uvir Bhagirathi
	SCHED-9	Type out Initial document	Sub-task	Done	1178454
Sprint:	SCHED Sprint 1				

Description

Determine the team members roles and project description

Initial Documentation (SCHED-2)

[SCHED-6] Logo Design Created: 10/Aug/17 Updated: 11/Aug/17 Resolved: 11/Aug/17

Status:	Done
Project:	Schedex
Component/s:	None
Affects Version/s:	None
Fix Version/s:	None

Type:	Sub-task	**Priority:**	Medium
Reporter:	Harvey Muyangayanga	**Assignee:**	Uvir Bhagirathi
Resolution:	Done	**Votes:**	0
Labels:	None		
Remaining Estimate:	Not Specified		
Time Spent:	Not Specified		
Original Estimate:	Not Specified		

Sprint:	SCHED Sprint 1

Description

Design Andromeda logo

Github Repository Creation (SCHED-1)

[SCHED-8] Create Branches Created: 10/Aug/17 Updated: 10/Aug/17 Resolved: 10/Aug/17

Status:	Done
Project:	Schedex
Component/s:	None
Affects Version/s:	None
Fix Version/s:	None

Type:	Sub-task	**Priority:**	Medium
Reporter:	Harvey Muyangayanga	**Assignee:**	Jason Parry
Resolution:	Done	**Votes:**	0
Labels:	None		
Remaining Estimate:	Not Specified		
Time Spent:	Not Specified		
Original Estimate:	Not Specified		

Sprint:	SCHED Sprint 1

Comments

Comment by Harvey Muyangayanga [10/Aug/17]

Creation of members' branches and a documentation branch.

Github Repository Creation (SCHED-1)

[SCHED-7] Create main repo Created: 10/Aug/17 Updated: 10/Aug/17 Resolved: 10/Aug/17

Status:	Done
Project:	Schedex
Component/s:	None
Affects Version/s:	None
Fix Version/s:	None

Type:	Sub-task	**Priority:**	Medium
Reporter:	Harvey Muyangayanga	**Assignee:**	Harvey Muyangayanga
Resolution:	Done	**Votes:**	0
Labels:	None		
Remaining Estimate:	Not Specified		
Time Spent:	Not Specified		
Original Estimate:	Not Specified		

Sprint:	SCHED Sprint 1

Description

Creation of the main repository

[SCHED-1] Github Repository Creation Created: 10/Aug/17 Updated: 10/Aug/17 Resolved: 10/Aug/17	
Status:	Done
Project:	Schedex
Component/s:	None
Affects Version/s:	None
Fix Version/s:	None

Type:	Story	**Priority:**	Medium
Reporter:	Harvey Muyangayanga	**Assignee:**	Harvey Muyangayanga
Resolution:	Done	**Votes:**	0
Labels:	None		
Σ Remaining Estimate:	Not Specified	**Remaining Estimate:**	Not Specified
Σ Time Spent:	Not Specified	**Time Spent:**	Not Specified
Σ Original Estimate:	Not Specified	**Original Estimate:**	Not Specified

Sub-tasks:	**Key**	**Summary**	**Type**	**Status**	**Assignee**
	SCHED-7	Create main repo	Sub-task	Done	
	SCHED-8	Create Branches	Sub-task	Done	Jason Parry
Sprint:	SCHED Sprint 1				

Sprint 2

Software Requirements Specifications (SCHED-10)

[SCHED-15] Assumptions Created: 25/Aug/17 Updated: 01/Sep/17 Resolved: 01/Sep/17

Status:	Done
Project:	Schedex
Component/s:	None
Affects Version/s:	None
Fix Version/s:	None

Type:	Sub-task	**Priority:**	Medium
Reporter:	Harvey Muyangayanga	**Assignee:**	Jason Parry
Resolution:	Done	**Votes:**	0
Labels:	None		
Remaining Estimate:	Not Specified		
Time Spent:	Not Specified		
Original Estimate:	Not Specified		

Sprint:	SCHED Sprint 2

Software Requirements Specifications (SCHED-10)

[SCHED-18] Data Management Created: 25/Aug/17 Updated: 01/Sep/17 Resolved: 01/Sep/17

Status:	Done
Project:	Schedex
Component/s:	None
Affects Version/s:	None
Fix Version/s:	None

Type:	Sub-task	**Priority:**	Medium
Reporter:	Harvey Muyangayanga	**Assignee:**	Jason Parry
Resolution:	Done	**Votes:**	0
Labels:	None		
Remaining Estimate:	Not Specified		
Time Spent:	Not Specified		
Original Estimate:	Not Specified		

Sprint:	SCHED Sprint 2

Software Requirements Specifications (SCHED-10)

[SCHED-12] Project Scope Created: 25/Aug/17 Updated: 01/Sep/17 Resolved: 01/Sep/17

Status:	Done
Project:	Schedex
Component/s:	None
Affects Version/s:	None
Fix Version/s:	None

Type:	Sub-task	**Priority:**	Medium
Reporter:	Harvey Muyangayanga	**Assignee:**	1178454
Resolution:	Done	**Votes:**	0
Labels:	None		
Remaining Estimate:	Not Specified		
Time Spent:	Not Specified		
Original Estimate:	Not Specified		

Sprint:	SCHED Sprint 2

Software Requirements Specifications (SCHED-10)

[SCHED-17] Operating Environment

Created: 25/Aug/17 Updated: 01/Sep/17 Resolved: 01/Sep/17

Status:	Done
Project:	Schedex
Component/s:	None
Affects Version/s:	None
Fix Version/s:	None

Type:	Sub-task	**Priority:**	Medium
Reporter:	Harvey Muyangayanga	**Assignee:**	Jason Parry
Resolution:	Done	**Votes:**	0
Labels:	None		
Remaining Estimate:	Not Specified		
Time Spent:	Not Specified		
Original Estimate:	Not Specified		

Sprint:	SCHED Sprint 2

Software Requirements Specifications (SCHED-10)

[SCHED-21] Use Case Description Created: 25/Aug/17 Updated: 01/Sep/17 Resolved: 01/Sep/17

Status:	Done
Project:	Schedex
Component/s:	None
Affects Version/s:	None
Fix Version/s:	None

Type:	Sub-task	**Priority:**	Medium
Reporter:	Harvey Muyangayanga	**Assignee:**	Uvir Bhagirathi
Resolution:	Done	**Votes:**	0
Labels:	None		
Remaining Estimate:	Not Specified		
Time Spent:	Not Specified		
Original Estimate:	Not Specified		

Sprint:	SCHED Sprint 2

Software Requirements Specifications (SCHED-10)

[SCHED-16] Dependencies Created: 25/Aug/17 Updated: 01/Sep/17 Resolved: 01/Sep/17

Status:	Done
Project:	Schedex
Component/s:	None
Affects Version/s:	None
Fix Version/s:	None

Type:	Sub-task	**Priority:**	Medium
Reporter:	Harvey Muyangayanga	**Assignee:**	Jason Parry
Resolution:	Done	**Votes:**	0
Labels:	None		
Remaining Estimate:	Not Specified		
Time Spent:	Not Specified		
Original Estimate:	Not Specified		

Sprint:	SCHED Sprint 2

Software Requirements Specifications (SCHED-10)

[SCHED-19] Portability Created: 25/Aug/17 Updated: 01/Sep/17 Resolved: 01/Sep/17

Status:	Done
Project:	Schedex
Component/s:	None
Affects Version/s:	None
Fix Version/s:	None

Type:	Sub-task	**Priority:**	Medium
Reporter:	Harvey Muyangayanga	**Assignee:**	Jason Parry
Resolution:	Done	**Votes:**	0
Labels:	None		
Remaining Estimate:	Not Specified		
Time Spent:	Not Specified		
Original Estimate:	Not Specified		

Sprint:	SCHED Sprint 2

Software Requirements Specifications (SCHED-10)

[SCHED-13] Product Description Created: 25/Aug/17 Updated: 01/Sep/17 Resolved: 01/Sep/17

Status:	Done
Project:	Schedex
Component/s:	None
Affects Version/s:	None
Fix Version/s:	None

Type:	Sub-task	**Priority:**	Medium
Reporter:	Harvey Muyangayanga	**Assignee:**	Harvey Muyangayanga
Resolution:	Done	**Votes:**	0
Labels:	None		
Remaining Estimate:	Not Specified		
Time Spent:	Not Specified		
Original Estimate:	Not Specified		

Sprint:	SCHED Sprint 2

Software Requirements Specifications (SCHED-10) [SCHED-14] Requirements Created: 25/Aug/17 Updated: 01/Sep/17 Resolved: 01/Sep/17	
Status:	Done
Project:	Schedex
Component/s:	None
Affects Version/s:	None
Fix Version/s:	None

Type:	Sub-task	**Priority:**	Medium
Reporter:	Harvey Muyangayanga	**Assignee:**	Harvey Muyangayanga
Resolution:	Done	**Votes:**	0
Labels:	None		
Remaining Estimate:	Not Specified		
Time Spent:	Not Specified		
Original Estimate:	Not Specified		

Sprint:	SCHED Sprint 2

Software Requirements Specifications (SCHED-10)

[SCHED-11] Executive Summary Created: 25/Aug/17 Updated: 01/Sep/17 Resolved: 01/Sep/17

Status:	Done
Project:	Schedex
Component/s:	None
Affects Version/s:	None
Fix Version/s:	None

Type:	Sub-task	**Priority:**	Medium
Reporter:	Harvey Muyangayanga	**Assignee:**	1178454
Resolution:	Done	**Votes:**	0
Labels:	None		
Remaining Estimate:	Not Specified		
Time Spent:	Not Specified		
Original Estimate:	Not Specified		

Sprint:	SCHED Sprint 2

[SCHED-10] Software Requirements Specifications Created: 25/Aug/17 Updated: 01/Sep/17 Resolved: 01/Sep/17	
Status:	Done
Project:	Schedex
Component/s:	None
Affects Version/s:	None
Fix Version/s:	None

Type:	Story	**Priority:**	Medium
Reporter:	Harvey Muyangayanga	**Assignee:**	Unassigned
Resolution:	Done	**Votes:**	0
Labels:	None		
Σ Remaining Estimate:	Not Specified	**Remaining Estimate:**	Not Specified
Σ Time Spent:	Not Specified	**Time Spent:**	Not Specified
Σ Original Estimate:	Not Specified	**Original Estimate:**	Not Specified

Sub-tasks:

Key	Summary	Type	Status	Assignee
SCHED-11	Executive Summary	Sub-task	Done	1178454
SCHED-12	Project Scope	Sub-task	Done	1178454
SCHED-13	Product Description	Sub-task	Done	Harvey Muyangayanga
SCHED-14	Requirements	Sub-task	Done	Harvey Muyangayanga
SCHED-15	Assumptions	Sub-task	Done	Jason Parry
SCHED-16	Dependencies	Sub-task	Done	Jason Parry
SCHED-17	Operating Environment	Sub-task	Done	Jason Parry
SCHED-18	Data Management	Sub-task	Done	Jason Parry
SCHED-19	Portability	Sub-task	Done	Jason Parry
SCHED-20	Use Case Set	Sub-task	Done	Uvir Bhagirathi
SCHED-21	Use Case Description	Sub-task	Done	Uvir Bhagirathi

Sprint:	SCHED Sprint 2

Software Requirements Specifications (SCHED-10)

[SCHED-20] Use Case Set Created: 25/Aug/17 Updated: 01/Sep/17 Resolved: 01/Sep/17

Status:	Done
Project:	Schedex
Component/s:	None
Affects Version/s:	None
Fix Version/s:	None

Type:	Sub-task	**Priority:**	Medium
Reporter:	Harvey Muyangayanga	**Assignee:**	Uvir Bhagirathi
Resolution:	Done	**Votes:**	0
Labels:	None		
Remaining Estimate:	Not Specified		
Time Spent:	Not Specified		
Original Estimate:	Not Specified		

Sprint:	SCHED Sprint 2

Sprint 3

Back End (SCHED-4)

[SCHED-22] Python Code Created: 04/Sep/17 Updated: 18/Sep/17 Resolved: 18/Sep/17

Status:	Done
Project:	Schedex
Component/s:	None
Affects Version/s:	None
Fix Version/s:	None

Type:	Sub-task	**Priority:**	Medium
Reporter:	Harvey Muyangayanga	**Assignee:**	Uvir Bhagirathi
Resolution:	Done	**Votes:**	0
Labels:	None		
Remaining Estimate:	Not Specified		
Time Spent:	Not Specified		
Original Estimate:	Not Specified		

Sprint:	SCHED Sprint 3

Description

Algorithm Implementation

[SCHED-3] Front End Created: 04/Sep/17 Updated: 18/Sep/17 Resolved: 18/Sep/17	
Status:	Done
Project:	Schedex
Component/s:	None
Affects Version/s:	None
Fix Version/s:	None

Type:	Story	**Priority:**	Medium
Reporter:	Harvey Muyangayanga	**Assignee:**	Unassigned
Resolution:	Done	**Votes:**	0
Labels:	None		
Σ Remaining Estimate:	Not Specified	**Remaining Estimate:**	Not Specified
Σ Time Spent:	Not Specified	**Time Spent:**	Not Specified
Σ Original Estimate:	Not Specified	**Original Estimate:**	Not Specified

Sub-tasks:					
	Key	**Summary**	**Type**	**Status**	**Assignee**
	SCHED-24	HTML	Sub-task	Done	1178454
	SCHED-25	CSS	Sub-task	Done	Harvey Muyangayanga
Sprint:	SCHED Sprint 3				

Front End (SCHED-3)

[SCHED-25] CSS Created: 04/Sep/17 Updated: 18/Sep/17 Resolved: 18/Sep/17

Status:	Done
Project:	Schedex
Component/s:	None
Affects Version/s:	None
Fix Version/s:	None

Type:	Sub-task	**Priority:**	Medium
Reporter:	Harvey Muyangayanga	**Assignee:**	Harvey Muyangayanga
Resolution:	Done	**Votes:**	0
Labels:	None		
Remaining Estimate:	Not Specified		
Time Spent:	Not Specified		
Original Estimate:	Not Specified		

Sprint:	SCHED Sprint 3

Front End (SCHED-3)

[SCHED-24] HTML

Created: 04/Sep/17 Updated: 18/Sep/17 Resolved: 18/Sep/17

Status:	Done
Project:	Schedex
Component/s:	None
Affects Version/s:	None
Fix Version/s:	None

Type:	Sub-task	**Priority:**	Medium
Reporter:	Harvey Muyangayanga	**Assignee:**	1178454
Resolution:	Done	**Votes:**	0
Labels:	None		
Remaining Estimate:	Not Specified		
Time Spent:	Not Specified		
Original Estimate:	Not Specified		

Sprint:	SCHED Sprint 3

[SCHED-4] Back End Created: 04/Sep/17 Updated: 18/Sep/17 Resolved: 18/Sep/17	
Status:	Done
Project:	Schedex
Component/s:	None
Affects Version/s:	None
Fix Version/s:	None

Type:	Story	**Priority:**	Medium
Reporter:	Harvey Muyangayanga	**Assignee:**	Unassigned
Resolution:	Done	**Votes:**	0
Labels:	None		
Σ Remaining Estimate:	Not Specified	**Remaining Estimate:**	Not Specified
Σ Time Spent:	Not Specified	**Time Spent:**	Not Specified
Σ Original Estimate:	Not Specified	**Original Estimate:**	Not Specified

Sub-tasks:	**Key**	**Summary**	**Type**	**Status**	**Assignee**
	SCHED-22	Python Code	Sub-task	Done	Uvir Bhagirathi
	SCHED-23	Google Calendar Integration	Sub-task	Done	Jason Parry
Sprint:	SCHED Sprint 3				

Back End (SCHED-4)

[SCHED-23] Google Calendar Integration Created: 04/Sep/17 Updated: 18/Sep/17 Resolved: 18/Sep/17

Status:	Done
Project:	Schedex
Component/s:	None
Affects Version/s:	None
Fix Version/s:	None

Type:	Sub-task	**Priority:**	Medium
Reporter:	Harvey Muyangayanga	**Assignee:**	Jason Parry
Resolution:	Done	**Votes:**	0
Labels:	None		
Remaining Estimate:	Not Specified		
Time Spent:	Not Specified		
Original Estimate:	Not Specified		

Sprint:	SCHED Sprint 3

Sprint 4

Final Documentation (SCHED-26)

[SCHED-31] Scrum Sprints Created: 18/Sep/17 Updated: 01/Oct/17 Resolved: 01/Oct/17

Status:	Done
Project:	Schedex
Component/s:	None
Affects Version/s:	None
Fix Version/s:	None

Type:	Sub-task	**Priority:**	Medium
Reporter:	Harvey Muyangayanga	**Assignee:**	Harvey Muyangayanga
Resolution:	Done	**Votes:**	0
Labels:	None		
Remaining Estimate:	Not Specified		
Time Spent:	Not Specified		
Original Estimate:	Not Specified		

Sprint:	SCHED Sprint 4

Description

Scrum Planing and Retrospective

Final Documentation (SCHED-26)

[SCHED-30] System Implementation

Created: 18/Sep/17 Updated: 01/Oct/17 Resolved: 01/Oct/17

Status:	Done
Project:	Schedex
Component/s:	None
Affects Version/s:	None
Fix Version/s:	None

Type:	Sub-task	**Priority:**	Medium
Reporter:	Harvey Muyangayanga	**Assignee:**	1178454
Resolution:	Done	**Votes:**	0
Labels:	None		
Remaining Estimate:	Not Specified		
Time Spent:	Not Specified		
Original Estimate:	Not Specified		

Sprint:	SCHED Sprint 4

Final Documentation (SCHED-26)

[SCHED-29] System design Created: 18/Sep/17 Updated: 01/Oct/17 Resolved: 01/Oct/17

Status:	Done
Project:	Schedex
Component/s:	None
Affects Version/s:	None
Fix Version/s:	None

Type:	Sub-task	**Priority:**	Medium
Reporter:	Harvey Muyangayanga	**Assignee:**	Jason Parry
Resolution:	Done	**Votes:**	0
Labels:	None		
Remaining Estimate:	Not Specified		
Time Spent:	Not Specified		
Original Estimate:	Not Specified		

Sprint:	SCHED Sprint 4

Final Documentation (SCHED-26)

[SCHED-28] Requirement specifications Created: 18/Sep/17 Updated: 01/Oct/17 Resolved: 01/Oct/17

Status:	Done
Project:	Schedex
Component/s:	None
Affects Version/s:	None
Fix Version/s:	None

Type:	Sub-task	**Priority:**	Medium
Reporter:	Harvey Muyangayanga	**Assignee:**	Uvir Bhagirathi
Resolution:	Done	**Votes:**	0
Labels:	None		
Remaining Estimate:	Not Specified		
Time Spent:	Not Specified		
Original Estimate:	Not Specified		

Sprint:	SCHED Sprint 4

Final Documentation (SCHED-26)

[SCHED-27] Introduction Created: 18/Sep/17 Updated: 01/Oct/17 Resolved: 01/Oct/17

Status:	Done
Project:	Schedex
Component/s:	None
Affects Version/s:	None
Fix Version/s:	None

Type:	Sub-task	**Priority:**	Medium
Reporter:	Harvey Muyangayanga	**Assignee:**	Uvir Bhagirathi
Resolution:	Done	**Votes:**	0
Labels:	None		
Remaining Estimate:	Not Specified		
Time Spent:	Not Specified		
Original Estimate:	Not Specified		

Sprint:	SCHED Sprint 4

[SCHED-5] Integration Created: 18/Sep/17 Updated: 01/Oct/17 Resolved: 01/Oct/17	
Status:	Done
Project:	Schedex
Component/s:	None
Affects Version/s:	None
Fix Version/s:	None

Type:	Story	**Priority:**	Medium
Reporter:	Harvey Muyangayanga	**Assignee:**	Unassigned
Resolution:	Done	**Votes:**	0
Labels:	None		
Remaining Estimate:	Not Specified		
Time Spent:	Not Specified		
Original Estimate:	Not Specified		

Sprint:	SCHED Sprint 4

[SCHED-26] Final Documentation Created: 18/Sep/17 Updated: 01/Oct/17 Resolved: 01/Oct/17	
Status:	Done
Project:	Schedex
Component/s:	None
Affects Version/s:	None
Fix Version/s:	None

Type:	Story	**Priority:**	Highest
Reporter:	Harvey Muyangayanga	**Assignee:**	Unassigned
Resolution:	Done	**Votes:**	0
Labels:	None		
Σ Remaining Estimate:	Not Specified	**Remaining Estimate:**	Not Specified
Σ Time Spent:	Not Specified	**Time Spent:**	Not Specified
Σ Original Estimate:	Not Specified	**Original Estimate:**	Not Specified

Sub-tasks:

Key	Summary	Type	Status	Assignee
SCHED-27	Introduction	Sub-task	Done	Uvir Bhagirathi
SCHED-28	Requirement specifications	Sub-task	Done	Uvir Bhagirathi
SCHED-29	System design	Sub-task	Done	Jason Parry
SCHED-30	System Implementation	Sub-task	Done	1178454
SCHED-31	Scrum Sprints	Sub-task	Done	Harvey Muyangayanga

Sprint:	SCHED Sprint 4

5.2 Project Sprint Retrospective

Sprint Number	Start date	End date	Successes	Issues	Suggestions for improvement
1	10th of August	24th of August	Work was split evenly and the team cohesion is perfect	Stand Ups cannot be implemented every day	Combine stand ups with short videos calls and other communications methods to ensure that the work is being executed
2	25th of August	1st of September	Communication issues solved	Lack of Precision in the project brief on some points	Contact the project owner (Lecturer) to make sure that what was done is what was expected
3	4th of September	18th of September	Clarification obtained. Project has successfully entered the coding phase	Team stresses over the imminent deadline	Members of the team that are done or awaiting a part their own depends on will assist other members
4	18th of September	2nd of October	Prototype working. Project almost complete	Final documentation incomplete	Write up documentation as the software is being developed

6 Conclusion

The above document details the design and implementation of a web application which schedules college/university examinations. Requirements gathering has been undertaken to determine the functional and non-functional requirements of the system. A design has been drawn up using UML models as well as logical splitting of the design into various layers. This design is implemented as a web application which, when tested, adheres to the specifications that were determined during the requirements gathering process. Finally, a detailed description of the agile SCRUM process followed by the development team is provided through sprint planning and a sprint retrospective.

A Pair Responsibilities

A.1 Pair I - Front End

The front-end pair were responsible for all coding and documentation related to the front-end (presentation layer of the project).

A.2 Pair II - Back End

The back-end pair were responsible for all coding and documentation related to the back-end (business and data layers of the project).