Software Requirements Specification: Revision 0

Group 2 - Genzter

Binu, Amit - binua - 400023175
Bengezi, Mohamed - bengezim - 400021279
Samarasinghe, Sachin - samarya - 001430998
October 6, 2017
Professor: Dr. Bokhari
Lab: L01

October 6, 2017

Contents

1 Revision History **3**

2 Project Drivers **4**
- 2.1 Purpose . . . 4
- 2.2 Stakeholders . . . 4
- 2.3 Scope . . . 4
- 2.4 Constraints . . . 4
- 2.5 Naming Conventions . . . 5
- 2.6 Terminology . . . 5

3 Functional Requirements **6**
- 3.1 User Input . . . 6
- 3.2 Generating Schedule . . . 6
- 3.3 Output . . . 6

4 Non-Functional Requirements **7**
- 4.1 Requirements . . . 7
  - 4.1.1 Usability . . . 7
  - 4.1.2 Performance . . . 7
  - 4.1.3 Operational and Environmental Requirements . . . 7
  - 4.1.4 Maintainability and Portability . . . 8

5 Open Issues **9**

6 Off-the-Shelf Solutions **9**

7 New Problems **9**

8 Tasks **9**

9 Risks **10**

10 Costs **10**

11 User Documentation **10**

1 Revision History

Date	Version	Description	Author
05/OCT/17	0.0	Created SRS	Mohamed Bengezi, Amit Binu, Sachin Samarasinghe

Table 1: Revision History

2 Project Drivers

2.1 Purpose

This project addresses a very tedious and time-consuming issue that affects thousands of university students every year. When it comes to course selection, finding the perfect schedule is no easy feat. Various obstacles contribute to this difficulty, whether it be a core, lecture or tutorial conflicting with something else, or a certain course can only be taken in a certain semester, etc. The Timetable Generator solves this issue by taking a list of required courses as input, and outputting a set of viable schedules as output, allowing the user to select the most appealing option. The goal of this project is to successfully accomplish this in an efficient manner, with an easy-to-use interface.

2.2 Stakeholders

There are a couple of stakeholders for this project, including the Genzter team, and obviously the students of McMaster University. In this case, the client and customer are the same: the student body. As for the motivation, seeing as we are students, we have first hand experience with regards to what is expected, and what an acceptable project entails.

2.3 Scope

The scope of this project includes understanding McMaster University's course layout, course selection process, as well as how courses differ throughout the various fields of study at McMaster. Also within the scope is the behaviour of students, and how they prefer timetables, whether they generally prefer early classes, late classes, Friday's off, etc.

2.4 Constraints

- The project must at least re-implement all of the functionalities of the original software.
- Core functionalities must be completed by October 16 for the proof of concept demonstration.
- The backend server must use HTTP to communicate with the client.
- When a valid timetable cannot be generated, the server must communicate that fact to the client.

- The server must be able to serve multiple clients concurrently.
- The server must keep each session isolated from other sessions.

2.5 Naming Conventions

At the moment there are no special naming conventions.

2.6 Terminology

- **Server** is the backend server powered by Node.js. This server manages sessions, calculates timetables and delivers the results to the client. The server uses HTTP to communicate with the client.
- **Client** refers to the web-based client side software that takes the user input and sends it to the server. Through the client, the users can choose the courses that they are interested in adding to their timetables. The client is programmed using HTML, CSS and Javascript.
- **Site** refers to the website that hosts the client. The users can access the application though the site.
- A **Session** refers to a visit of a user to the site. A session begins when the user's web browser connects to the site. And a session ends when the tab or the window that has the site opened is closed. During a session, a user can add, remove and generate a timetable for their selected courses.
- **Selected course list** is a list of courses that are selected by a specific user in the client. When commanded by the user, the client will submit the selected course list to the server and will wait for a response from the server.
- A **valid timetable** is a timetable for all the selected courses without any timing conflicts. This includes times and durations of lectures, tutorials and labs.
- **Scheduler** is the software module that is responsible for generating the timetables.
- **Scheduling algorithm** is the specific timetable generation algorithm used by the scheduler to generate timetables.

3 Functional Requirements

The functional requirements for this project include taking a set of course-codes from the user as an input, checking user's input, and then generating a schedule with no conflicts. The algorithm should generate the schedules within an acceptable period of time. Finally, this schedule will be outputted in a table format.

3.1 User Input

The program will check whether the input is valid or not, and displays an error message if the input is invalid. It also checks whether the user has already added the same course code or not. Also, the user's input must be in capital letters in order for it to be valid.

3.2 Generating Schedule

The user must pass some input(s) in order to generate the schedule. The program takes a set of all the course codes the user entered and generates a schedule with no conflicts. The program should generate the schedule within a decent period of time.

3.3 Output

The output will be displayed in a table format. Each column of the table represents a day of the week and each row of the table represents 30 minutes of a day. There will be a total of 6 columns and 28 rows. Each course will be represented using a unique color.

4 Non-Functional Requirements

The Non-functional requirements (NFR) of this project describe the appearance and feel of the of the Timetable Generator, as well as requirements such as usability, performance, operational, maintainability, and portability.

4.1 Requirements

4.1.1 Usability

The User-Interface (UI) of the project should be very clear and easy to use, and very minimal learning should be required. The user will be able to select the desired courses by inputting the course codes, and pressing an "Add" button to save that selection. Once all desired courses are selected, a "Generate" button is the pressed, and the set of schedules in outputted. The UI couldn't be easier to use, seeing as it simply requires the press of two buttons.

4.1.2 Performance

The performance requirement depends on various factors. Firstly, the speed of the program once the "Generate" button is pressed should take a relatively acceptable time period. That being said, the computation required is not an easy one, so the program is not expected to be lightning quick, but not painfully long. As for safety-critical requirements, there none that apply specifically to this product, seeing as it is hosted online, and no user information is stored. The precision of the product will be measured using schedules, or sets of schedules. The schedules are in the same format as current school schedules, and the precision will be bench-marked using this format. As for reliability, the generator should not available to users at most once a week, for maintenance. Otherwise, it should not produce failures. The capacity of the product should be in the area of 200 users at a single moment.

4.1.3 Operational and Environmental Requirements

For this project, there is no specific physical environment in which the product will operate. The site will be hosted online, so there are mainly technological environments. The expected technological environment must include a reliable internet connection, and any desired internet browser

4.1.4 Maintainability and Portability

The maintainability of the product will be defined by the difficulty to update the data-set used. The reason is because the rest of the project will not need to be maintained, only debugged and improved. The data-set will be easy to update, and so maintainability will not be a big issue. As for portability, the project will be available online, so any user with internet connection will be able to enjoy the product.

5 Open Issues

The data that is used in this project is pulled from http://www.timetablegenerator.io. If the owner of this website decides to modify the format of the data , then this project will have to be re-implemented.

6 Off-the-Shelf Solutions

There are existing similar products, such as http://www.timetablegenerator.io. The Genzter team has been in contact with the developers of this website, and have received assistance and guidance from them. The data-set can be considered a "ready made component", because this project will be using their data-set (with permission). There are also quite a few open-sourced projects on http://www.github.com that can be followed.

7 New Problems

This project shouldn't create any issues or conflicts with the current implementation environment. As for potential user issues, there should not be any adverse reactions that occur.

8 Tasks

The tasks of this project include:

- Defining the Problem Statement
- Creating the Development Plan
- Defining the Requirements
- Developing the Proof of Concept
- Implementation
- Testing

This list can change as the development of the project continues.

9 Risks

Some of the main risks of this project are

- Since the data for this website is gathered from https://www.timetablegenerator.io and if this website gets shutdown, this project won't work. The probability for this being a problem is unlikely. A plan to solve this problem will be to make another program that will parse this data from http://mosaic.mcmaster.ca.
- This project will also not work, if the owner of this website decides to make the data private. The probability for this being a problem is also unlikely. A plan to solve this problem will be to make another program that will parse this data from http://mosaic.mcmaster.ca.

10 Costs

There should a very small cost associated with this project, seeing as all software used is free, or used under a student license. The cost will come from purchasing the domain and hosting the website. There shouldn't be any other costs associated with the project.

11 User Documentation

The only documentation that will be necessary for users are the simple instructions that are on the homepage of the Generator.

References