Software Requirements Specification: Revision 1

Group 2 - Genzter

Binu, Amit - binua - 400023175
Bengezi, Mohamed - bengezim - 400021279
Samarasinghe, Sachin - samarya - 001430998
December 5, 2017
Professor: Dr. Bokhari
Lab: L01

December 5, 2017

Contents

List of Tables

List of Figures

1 Revision History

Date	Version	Description	Author
05/OCT/17	0.0	Created SRS	Mohamed Bengezi, Amit Binu, Sachin Samarasinghe
26/NOV/17	1.0	Modified Requirements	Mohamed Bengezi, Amit Binu, Sachin Samarasinghe
01/DEC/17	1.0	Updated terminology, requirements and added figure(s)	Mohamed Bengezi, Amit Binu, Sachin Samarasinghe

Table 1: Revision History

***Please Note, all updated material is written in blue

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 back-end 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

It is assumed that the client or individual using this document understands basic web development terminology, such as Node.js or HTTP

- **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.

2.7 Functional Requirements

2.7.1 Startup and User Input

1. When the server starts, the back-end must retrieve and parse and store a file from a remote web-server that contains the schedules of all courses offered by McMaster. The URL of the file will be determined during the implementation of the back-end. The file will be in JSON format.

2. When a user visits front-end, it must at least offer the user a text-box to search the courses (search-box), a button with text "Add" (add button) and a button with text "Generate" (generate button). The front-end could contain additional elements.

3. When a user searches for a course in the search-box, the front-end must show the user the courses that match text in the search-box. The matches could be partial. For an example, when the user types in "econ", the front-end will display a list of courses that begin with "econ".

4. The search must not be case sensitive.

5. When the add button is clicked, the front-end must retrieve the text in the search-box and send it to the server. This text is treated as the course code. Then the server will check if there exists a course that has the provided course code. If such a course does not exists, then the server will alert the front-end of it and the front-end will alert the user visually. If such a course exists, then the server will add that course to its own internal list that is specific to this session and will alert the front-end of the existence. Then the front-end will display the course in a list along with a button called "Remove".

6. The front-end and the server must remove a course from their lists when the "Remove" button for that specific course is clicked.

2.7.2 Generating and Displaying a Schedule

1. When the generate button is clicked, the front-end must alert the server of this event. Then the server will check if its internal list of courses for this session is empty. If the list is empty, then the server will alert the front-end of this and then the it will alert the user visually. If the list is not empty, then the server will proceed and will begin to generate the timetable.

2. When the server is generating a timetable, it must attempt to generate at least one valid timetable.

3. If the server found at least one valid timetable, then the it must send the timetable to the front-end which in-turn will display it to the user with "Re-generate" button. If the server was unable to generate any valid timetables, then it must alert the front-end of this. Then the front-end will display the user the "timing conflict" error message.

4. If the server generated more than one valid timetable during generation, then it must send the first one in its valid timetable list to the front-end.

5. During a session, the server must store all the generated valid timetables until the session end.

6. 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.

7. When the front end displays the generated timetable, it must display it in two separate tables. First table displays timetable of the first semester, and the second table displays timetable of the second semester.

Time	Monday	Tuesday	Wednesday	Thursday	Friday
8:00					
8:30					
9:00					
9:30	SFWRENG-3BB4 C01 HSC 1A6 Emil Sekerinski	SFWRENG-3XA3 C01 T13 125 Asghar Bokhari		SFWRENG-3BB4 C01 HSC 1A6 Emil Sekerinski	
10:00					
10:30	SFWRENG-3RA3 C01 TSH B128 Ryszard Janicki	SFWRENG-3BB4 C01 HSC 1A6 Emil Sekerinski	SFWRENG-3RA3 C01 TSH B128 Ryszard Janicki	SFWRENG-3RA3 C01 TSH B128 Ryszard Janicki	
11:00					
11:30			SFWRENG-3MX3 T03 ITB 139 Martin George Von Mohrenschildt		
12:00					
12:30					
13:00					
13:30	SFWRENG-3MX3 C01 TSH B128 Martin George Von Mohrenschildt	SFWRENG-3RA3 T01 BSB 137 Ryszard Janicki	SFWRENG-3MX3 C01 TSH B128 Martin George Von Mohrenschildt	SFWRENG-3MX3 C01 TSH B128 Martin George Von Mohrenschildt	
14:00					
14:30	SFWRENG-3BB4 T03 T13 107 Emil Sekerinski			SFWRENG-3XA3 L01 ITB 236 Asghar Bokhari	
15:00					
15:30					
16:00					
16:30	SFWRENG-3XA3 L01 ITB 236 Asghar Bokhari				
17:00					
17:30					
18:00					
18:30					
19:00					
19:30					
20:00					
20:30					
21:00					
21:30					
22:00					

Figure 1: Example Output

3 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. Underneath each requirement, there is the fit criterion that is used to measure the specific requirement.

3.1 Look and Feel Requirements

1. The User-Interface (UI) must be graphical.
    - The interface must be entirely made of text-boxes, buttons, graphical lists and tables.
    - The users must be able to use all the features of the product without the use of a command line terminal.
2. The UI must be minimal.
    - At any given instance, the interface must contain no more than one text-box, two buttons, one graphical list and two tables.
3. The UI looks professional.
    - At least 80% of the users must agree that the interface is professional.
4. The UI must look and feel modern.
    - At least 80% of the users must agree that the interface is modern.
5. The UI must feel responsive.
    - For all actions, the results must be displayed no later 500 milliseconds.

3.2 Usability Requirements

3.2.1 Ease of use

1. The User-Interface (UI) must be minimal.

- At any given instance, the interface must contain no more than one text-box, two buttons, one graphical list and two tables.

2. The UI must be intuitive.
   - Text on all buttons must be self explanatory.
   - Text on buttons must describe the functions that they perform when they are clicked.
3. The UI must be clear.
   - There must be at least 10 pixels of space between different section of the interface.
   - The text size must be at least 16pt.
   - There must be a sharp contrast between texts and their backgrounds.
4. The UI must be responsive.
   - For all actions, the results must be displayed no later 500 milliseconds.
5. The front-end must minimize as much as possible the number of mouse-clicks that are necessary to generate a timetable after a user enters the website.
   - After the users enter their desired courses, only a single click of the generate button must be needed to begin the generation process.

3.2.2 Ease of learning

1. There must be visual aids (helpful text, images, animation, etc...) to guide the user.
   - There must a written walk-through displayed on the website that can guide the user without any external help.
   - Text on all buttons must be self explanatory.
   - Text on buttons must describe the functions that they perform when they are clicked.
   - There must be text that explain parts of the user interface that are not buttons.
2. The error messages must be short and easy to understand.

- Messages that explain the errors must be no longer than two sentences each.
- The timing conflict error must display the conflicting courses.

3.3 Performance

3.3.1 Speed Requirements

1. After the user clicks the "Generate" button the generated timetable must be presented to the user in an acceptable time period.
   - The generated timetable must be displayed no later 3 seconds.

3.4 Operational and Environmental Requirements

3.4.1 Physical Environment

1. For this project, there is no specific physical environment in which the product will operate.

3.4.2 Operational Requirements

1. The back-end must be compatible with the environment of FireBase since it will hosted there.
   - The back-end must be tested in Firebase to make sure that it functions as intended.
2. The front-end will be operating in users' web-browsers. Therefore, it must be compatible with browser environment.
   - The front-end must be tested on Mozilla Firefox, Google Chrome, Opera, Safari, Microsoft Edge and Internet Explorer.
3. The environment must include a reliable high-bandwidth internet connection in order to be able to server many users concurrently.

- The internet connection must at least be able to download and upload 50 Megabits/second.

3.5 Maintainability and Support Requirements

3.5.1 Maintainability Requirements

1. The program must be decomposed into a set of small modules that each perform a well-defined set of tasks.
   - The developers must use the module pattern.
2. The source code must be well documented.
   - Module interface specification documents must be written and made available for all modules.
   - Logic behind the code must be explained using comments in the source code explaining the.
3. The code must follow a standard coding convention.
   - All of the source code must conform to the Mozilla coding style.

3.5.2 Support Requirements

1. Users must be able report bugs and concerns with the product.
   - Users can contact one of the three developers of the product, Sachin, Amit or Mohamed using email.

3.6 Security Requirement

- There are no security requirements that apply specifically to this product, seeing as it is hosted online, and no user information is stored.

3.7 Cultural Requirements

- There are no cultural requirements.

3.8 Legal Requirements

- There are no legal requirements.

3.9 Health and Safety Requirements

- There are no health and safety requirements.

4 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.

5 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.

6 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.

7 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.

8 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.

9 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.

10 User Documentation

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

References