Software Requirements Specification
For
Team 11

February 28, 2020
Version 1

Prepared by:
Mark Ekis, Austin Ross

Table of Contents

Revision History

Version	Date	Name	Description
1	02/28/20	Mark Ekis, Austin Ross	Initial Document

1 Introduction

1.1 Overview

The scheduler app will be a web application designed for a user online to be able to create an account, enter customized scheduling constraints for teachers and then enter a potential schedule to be verified by the application. The application will then assess the schedule based on the given criteria as well as some basic scheduling logic and will return whether or not the schedule is viable.

The purpose of this document is to provide a clear explanation of the requirements for the app. The introduction will lay out goals and scope. Next, section 2 will discuss the constraints and additional information on the environment. Section 3 will go into depth on the nonfunctional requirements. Section 4 will discuss the functional requirements as well as use cases.

1.2 Goals and Objectives

The primary objective of the scheduler app is to give administrators at schools the ability to quickly and easily verify schedules for professors. Things we want out of the app include:

1. The app must be able to access via a browser, as installation can be difficult on school computers.
2. The app must be easy for someone not familiar with web app development to use.
3. The app must have methods for the user to quickly add schedules and scheduling criteria.

1.3 Scope

The scheduler app will provide users the ability to specify scheduling criteria and verify if a schedule meets that criteria all within a web application. Users will be able to save their criteria and uploaded schedules by creating an account on the app which this data will be saved too. It is not criteria that the application actually generates a viable schedule for the user given the criteria. While this may be a feature added in a future iteration of the project it is not in our scope.

1.4 References

Information on Vue: https://vuejs.org/

The Github: https://github.com/umkc-cs-451-2020-spring/semester-project-group-11

2 General Design Constraints

2.1 Application Environment

The scheduler app will be made as a Vue application. It will be making use of the following packages: JS, HTML, SASS, Vue JS, Node JS, Babel, and Webpack. Additionally, the app will make use of a database to keep track of user profiles as well as previously saved schedules.

2.2 User Characteristics

The average user of this app will be an administrator for a school, most likely a university. While it can be expected for the users to have some sort of experience with using a computer. We cannot expect them to have more than an average understanding.

2.3 Mandated Constraints

The application must be a web application. This is because this app will almost always be used on school computers which can be quite the hassle to install applications on.

2.4 Potential System Evolution

The current scope of the project is focused on the basic architecture and the verification algorithm. However, in the future we hope to have the ability to generate a viable schedule given the requirements posted by the user.

3 Nonfunctional Requirements

3.1 Operational Requirements

Usability: The application should be intuitive and simple to use such that 95% of users will not need to read any documentation to be able to use the application.

3.2 Performance Requirements

Efficiency: The application should take no more than 3 to return a list of conflicts with a given schedule(s).

Flexibility: The user should have the option to run the application against more than one schedule if he/she chooses to do so.

Maintainability: Changes made to the website's user interface should not impact the performance of the application.

Reliability: The system shall report back all conflicts with a given schedule(s).

Reusability: All web pages should follow a similar design scheme.

Robustness: The system shall report back conflicts with all university rules and instructor requests.

3.3 Security Requirements

Integrity: The reasoning for instructors' scheduling request(s) should not be visible to non-authorized users.

3.4 Documentation and Training

The Scheduler App will be delivered to stakeholders as a web application. A user guide will be provided.

3.5 External Interface

3.5.1 User Interface

The user interface will be consistent and professional. Navigation to different pages within the app should not alter the users experience with the system and should have a familiar look and feel to all others.

The interface should be intuitive and simple to use, as it is expected that 95% of users are able to successfully use the system without training or referring to the user guide.

3.5.2 Software Interface

A SQL Server database will be used to store information regarding account status and conflict criteria, and serve as an interface between user interaction and the Scheduler App website.

4 Functional Requirements

4.1 Required Features

4.1.1 Use Case: 1

Description: User Login/Sign up
Actors: School Administrator
Value = high
Cost = high

Basic Path

1. User navigates to the website.
2. The web application opens and prompts the user to sign up or log in.
3. The user proceeds to click log in.
4. Next, the system asks the user to enter username and password into provided datafields.
5. Next, the user enters the correct data into the respective fields and clicks "Log in."
6. The system then re-navigates to the homepage and now displays the username in the area where they were previously prompted to log in or sign up.
7. If there were any previously saved schedules, they are populated into an area marked previous schedules. These schedules are identified and sorted by time and date of the last time they were edited.
8. The system then waits for further commands.

Alternate Path

1. Breaking off from point 3, the user instead selects sign up.
2. Now instead, the system asks the user to enter an email address, username, and password. Additionally, the user will be prompted with a test to make sure the user signing up is not a bot.
3. The user then fills out these fields and clicks "Sign up."
4. Provided the fields are filled out with viable information and the test is successfully passed, the system then re-navigates to the homepage and now displays the username in the area where they were previously prompted to log in or sign up.
5. The previous schedules section will not populate with previous schedules as the account does not have any schedules saved yet.
6. The system then waits for further commands.

4.1.2 Use Case: 2

Description: Entering New Scheduling Criteria

Actors: School Administrator
Value = high
Cost = medium

Basic Path

1. The user navigates to the website.
2. If the user desires to use an account, they log in following the steps dictated in use case 1.
3. Now, at the home page, if the user wishes to make a new schedule, they select the button marked, “New Schedule.” Otherwise, if the user wishes to edit an existing schedule, they double click the schedule they wish to edit in the area marked “Previous Schedules.”
4. The app navigates to the schedule editor page where a field marked “Scheduling Criteria” is visible. In this field, there is a button marked with “+.”
5. The user clicks the button.
6. A form is pulled up marked “New Criteria.”
7. The form prompts the user to state the person the criteria is applied to, the nature of the criteria, and the importance of the criteria.
8. The user fills out the fields and hits the button marked “Add.”
9. The criteria is added to the “Scheduling Criteria” field.
10. The system then waits for further commands.

4.1.3 Use Case: 3

Description: Verifying a Schedule
Actors: School Administrator
Value = high
Cost = medium

Basic Path

1. The user navigates to the website.
2. If the user desires to use an account, they log in following the steps dictated in use case 1.
3. Now, at the home page, the user selects the schedule they want to verify from the “Previous Schedules” section by double clicking the desired schedule.
4. The schedule and criteria are pulled up and a button marked “Verify” becomes available.
5. The user clicks the button.
6. The system runs the schedule through its algorithm and returns a score depending on if the schedule passed or failed and if any of the optional criteria were not met.
7. The system then waits for further commands.