Software Requirements Specification

for

Majorizer

Version 1.0

Prepared by Jacob Desiletts, Jonathan Nordby, Ryan Stewart, and Kaitlyn Witt

Condor

10/17/21

Table of Contents

Revision History

Name	Date	Reason For Changes	Version
All of us	10/17/2021 (Due date)	Initial Writeup	1.0

1. Introduction

Purpose

The purpose of this Software Requirements specification is to provide additional information and details about the functionalities of the Majorizer application. This document will cover each feature of the application in great detail and provide a glance at what the user interface will look like. It will also cover any constraints, hardware, and software requirements needed to use the application.

Intended Audience and Reading Suggestions

This document is intended for anyone using Majorizer. This includes individuals aiding in the maintenance and development process along the way. Readers who want a brief overview of this application should read the rest of Part 1 (Introduction) and Part 2 (Overall Description) to get a more in depth view of key components. If applicable, readers can look at Part 6 (Key Milestones) to get a timeline of the project.

Readers who want to learn more about the key features of this application should take a look at Part 3 (System Features), which addresses all of the key features in more detail. In addition, Part 5 (External Interface Requirements) will include more technical details regarding the user interface and hardware and software interfaces that will be needed to run the application.

Lastly, readers who want to learn more about non-technical requirements should take a look at Part 6 (Other Nonfunctional Requirements), which goes into detail about performance, safety, security, and software quality that are important to the application functionality and for user knowledge.

Project Scope

Majorizer is a two part system consisting of a client-side interface and a server-side backend. The client-side interface is accessed through a web browser on Windows, Mac and Linux devices. The backend web server will handle the storage and serving of content. This system is designed to allow students and advisors to build course schedules. Course schedules can be created to plan out a road map based on courses that have already been taken and any majors or minors that can be added. Students should be allowed to share their schedule with any of their advisors for review. The student and any of their advisors will then have the ability to suggest changes to the schedule using a commenting feature and suggestion feature. This will allow advisors to add or remove courses which the student then should be able to accept or reject. This same feature will exist in reverse if an advisor proposes a schedule for a student of theirs where the student can comment on and suggest changes to the schedule.

2. Overall Description

Product Perspective

Majorizer is a web application designed for creating schedules for different major and minor combinations at Clarkson University. This application consists of a server-side hosting the web server and the database to store all of the required information for system functionality. While the client-side will be responsible for the functionalities of the different users. Users will interact with this application in a web browser on a Windows, Mac and Linux based system. All aspects of the server-side and client-side interactions will more covered in more detail throughout this document

Product Features

The main feature of this application allows users to build class schedules, which includes searching classes, adding classes, dropping classes, and being able to block off specific times based on their own personal commitments. Users will be able to search for classes through a variety of categories like course title, department, professor name, time of days, days of week, credit count, and major requirements. When searching for classes, there will be a tree-like view of all of the prerequisites for the chosen class. This will then tell the user if they are eligible to take the desired class based on the previous classes they have taken. Users can also create “What-If Scenarios”. These give students an idea on whether or not they are able to complete their desired majors and minors in their 4 year undergraduate career, or if that will change their graduation term. Students are able to share their schedules and What-if Scenarios with their advisors to get their feedback where each of them can reply to comments and communicate with each to get the best plan.

User Classes and Characteristics

There are several types of users that will use our product. Those are students, guests, and advisors. All users will have access to the scheduling features, but only students with accounts to the system students will have the ability to connect to advisors and save data to the database. Guests only have the ability to export the data in a JSON format. Furthermore, advisors will have the ability to create and save schedules to the database. Advisors will also have the ability to view schedules that are not their own, although this is limited to students who have shared schedules with the advisors. Students will not be able to share their schedules with other students. This software is designed primarily for students and advisors, making them the most important stakeholders to support.

	Create Schedules	Export Schedules to File	Save Schedules to the database	Share schedules with an advisor	View Other User's Schedules
Guest	Y	Y	N	N	N
Student	Y	Y	Y	Y	N
Advisor	Y	Y	Y	N	Y

Operating Environment

The client-side operating environment will be a desktop web app compatible with Chromium and Gecko based browsers. We will not provide support for mobile views such as smartphones or tablets. The client side application will communicate with the server side application frequently, so a stable network connection is recommended. The server side operating environment will include an operating system capable of running Docker.

Design and Implementation Constraints

We will be using Django for web server development, which restricts the development language to Python. In addition, we will be using PostgreSQL to develop the database. All of this will occur within Docker, so the host system must have Docker installed in order to run the application. This means the user who wants to host the server-side part of the application must have a device that is able to run Docker containers. Lastly, a network connection is required for the application, so any firewall in place of the application and the potential clients must make an exception for the host and port this application is hosted on.

User Documentation

The primary function of Majorizer is to allow users to build their course schedules. This application is designed in a way so that it's easy to use for all users. However, some features may need additional explanation to teach the user exactly how it operates. So, there will be a tutorial feature and a help menu that will help people navigate the menu options.

The help menu will contain topics covering how each of the application's functions work. This will include how to use corresponding menus and all relevant site functionality. The user should be able to access the help menu from any page on the site.

The Majorizer tutorial will demonstrate each of the topics in the help section by turning them into a step-by-step demonstration on how to use the application. This tutorial will be designed for end users rather than systems administrators and maintainers.

Assumptions and Dependencies

In order to run the server-side of Majorizer, we will assume that the user has Docker installed. This will allow all required dependencies to be installed. As a substitute for Docker, the user must have Python 3 installed and a bare PostgreSQL database for this application. In addition, the user must install all Python libraries required for this application, which will be in a requirements.txt file.

1. Authentication

1.1. Description and Priority

1.1.1. Allows users to log into the service in a secure way to ensure data integrity and user privacy

1.1.2. Authentication itself is a high priority, but implementing complex authentication is not a high priority.

1.1.3. The authentication consists of a username and a password sent over an encrypted connection (SSL). Bcrypt hashes of passwords will be stored in the database alongside usernames and user session tokens.

2. Schedule Functionality

2.1. Allows users to view a calendar view of the classes they have selected

2.2. This is the most important feature in the entire application

2.3. The schedule functionality ties into many other systems. Users will search the list of classes to sign up for, then the system will automatically determine if the class fits in your schedule and if you meet the requirements and alerts you if any of this is the case. It also automatically saves the classes you've already selected, so you can use them for enrollment immediately when it comes time.

3. Export as Calendar or Image

3.1. The export feature allows you to take the classes that were selected by feature 2 and export it as a calendar file or an image. The calendar format is intended to be used by Google Calendar.

4. Suggestions based off previous classes

4.1. Suggests classes based off classes already taken

4.2. This feature is somewhat important to the application

4.3. The suggestion feature acquires its functionality through classes that students have completed their prerequisites for, classes that are required for your selected program of studies, and classes that are being offered the next semester.

5. What-If

5.1. What-if scenarios give a view of future semesters based on a student's selected majors and minors.
5.2. This feature is considered to be very important
5.3. This feature allows students to view how adding an additional major or minor will affect their course progression. This includes graduation term and allows for planning co-ops into their schedule.

6. Link Student to Advisor

6.1. Allow a student to link to an advisor
6.2. This feature is somewhat important to the application
6.3. This feature allows students to easily connect to an advisor when they have a registered account. An advisor will generate a one time code to link with a student. They will send the code to a student through some external medium and the student will copy it to the code entry dialog box and the linking will occur.

7. Share Schedule with Advisor

7.1. After a schedule has been created, it can be shared with any advisors that have been linked to the student's account.
7.2. This feature is quite important to the application
7.3. This feature allows students to create sample schedules to send to advisors, so that an advisor can have easy access to what a student thinks is the best course of action to streamline the advising process.

8. Comment Feature

8.1. This allows advisors to comment on schedules to allow for an improved asynchronous communication experience.
8.2. This feature is fairly important to the application
8.3. This feature will allow professors to click on a part of the schedule and write a comment about what they think should happen. Then, students should be able to reply in a small box under the comment what they think about the change. At any point, the student can resolve or reject the comment and the thread will disappear. Both users can continue to use the reply feature as long as no other actions are taken.

3. External Interface Requirements

1. User Interfaces

1.1. General layout

1.1.1. There will be a navigation bar at the top of every page containing two buttons

1.1.2. There will be a home button on the Majorizer logo on the left side of the navigation bar.

1.1.3. There will be a profile button linking to the user's profile on the right side of the navigation bar.

1.1.4. There will be a help button linking to the help page.

1.2. Class search and tree view

1.2.1. There will be a search bar at the top of the panel where users can enter classes and filter their search results

1.2.2. There will be a tree view in the middle for a selected class and its immediate prerequisites as well as classes it is a prerequisite for.

1.2.3. There will be an information panel on the bottom of the panel which will display a summary of the selected class, its credits, and the semesters it is offered.

1.2.4. Users can drag and drop a course from the tree view to the calendar panel to add it to their schedule

1.3. Calendar

1.3.1. There will be a weekly calendar displaying Monday through Friday on the horizontal axis and times from 8:00 am to 9:00 pm in half-hour increments on the vertical axis.

1.3.2. Classes will be displayed differently based on if the user has met the prerequisites to take them

1.3.3. There will be an "export" button linking to the export page

1.4. Home page

1.4.1. The contents of the home page will vary based on whether the user is a guest or a registered student.

1.4.2. The home page for unregistered users will consist of two sections in addition to the navigation bar.

1.4.2.1. On the left side of the page there will be a class search and tree view panel as described in User Interface 1.2.

1.4.2.2. On the right side of the page there will be a Calendar panel as described in User Interface 1.3.

1.4.3. The home page for students will be identical to the home page for unregistered users except for the following:

1.4.3.1. The navigation bar will include an "advisor" button linking to the student's advisor page.

1.4.3.2. The calendar panel will include a "save" button that saves the current schedule.

1.4.4. The home page for advisors will be identical to the home page for students except for the following:

1.4.4.1. The “advisor” button in the navigation bar will be replaced with a “students” button linking to the advisor’s students page
1.4.4.2. There will be a student selection panel under the navigation bar where the advisor can select which of their linked student’s schedules they would like to see.
1.4.4.3. The calendar panel will display the saved schedule of the selected student.
1.4.4.4. There will be a comment panel to the right of the calendar panel where the advisor can send and respond to messages with the selected student.

Hardware Interfaces

Majorizer is a web application supported on Chromium and Gecko browsers. Users interact with Majorizer through the web page using the HTTP protocol.

Software Interfaces

Majorizer will be developed using Python 3 and the Django web framework, as well as PostgreSQL for the database.

1. Incoming and outgoing items
    a. Outgoing data consists of content to be displayed on the website such as search results, API response data, saved schedule information, and exported schedules as images or calendar files
    b. Incoming data consists of search and filter terms, major and class information, and communications sent by the user.
2. Services and communications
    a. Using Majorizer requires a constant internet connection with the server.
    b. Majorizer requires an authentication service so that the user can store information and communicate with their students/advisor.
    c. Majorizer requires a class prerequisite service that is used by the search functionality, the recommendation functionality, and the schedule functionality.

Communications Interfaces

Majorizer requires the user to have a Chromium or Gecko based browser. The user’s browser will use the HTTP protocol to communicate with the Majorizer server. The server will be created using the Django web framework and will use a PostgreSQL database for storing class and user information.

4. Other Nonfunctional Requirements

Performance Requirements

Provided the user has a fast and stable connection to the Majorizer server, performance should not be an issue for most elements of the application. The most computationally expensive functionalities are the class filter search and the tree view of prerequisites, but these should not take longer than 10 seconds on average.

Safety Requirements

Because Majorizer is a web application, it poses virtually no risk to the user's computer. There is a risk that Majorizer could provide the user with incorrect information, however this risk can be mitigated by the presence of advisors who can double check a student's plans.

Security Requirements

Users can access Majorizer with or without making an account. For unregistered users, information is not stored beyond their session, except schedules that the user chooses to export. User accounts will require a username and a password, and whether they are a student or an advisor. Registered users will have to log in at the start of every session in order to access their information. Registered students can choose to share information with a registered advisor, and vice versa.

Software Quality Attributes

Majorizer should be maintainable and correct above all else. It should be easy to update the service with new class information as it is released, and the user should be able to rely on that information to be accurate. Majorizer should also prioritize usability, at least more so than existing alternatives such as those provided by schools.

Business Rules

Users can choose what information is shared with others or exported. Advisor/student relationships require action from both parties to establish. Once established, students can choose to share their schedules with their advisor or ask their advisor questions, and the advisor can respond. Advisors can remove students from their list of students.

5. Key Milestones

#	Milestone	Target Completion Date	Comments
1.	Completion of requirements document	10/15/2021	Revisions will be made over the following weeks.
2.	UI design finalized	10/25/2021	Should be completed before design document for revision
3.	Backend design finalized	10/25/2021	Should be completed before design document for revision
4.	Completion of design document	11/2/2021	Tasks will be assigned to individuals in order to complete by date
5.	Project completion	12/4/2021	Tasks will be assigned to individuals in order to complete by date

6. Key Resource Requirements

Major Project Activities	Skill/Expertise Required	Internal Resource	External Resource	Issues/Constraints
Authentication	Databases, security	Ryan has experience with authentication; Django has an authentication system	Internet, Django/Python documentation	Constraint set by client for very simple username/password
Database Creation	Databases	All members have database experience	SQL Documentation	Retrieving the data needed for the database
User Interface	Design experience, frontend development experience	Jacob has design experience, all members are familiar with Django	Internet, Django documentation	Limited time to implement and knowledge on limitations with html
Scheduling ability	Databases, Python/Django experience	All members are familiar with databases and Python	Django documentation, python libraries	Limited time to implement
Import and Export	Databases, Python libraries	All members are familiar with databases and Python	Internet, Python Documentation	We will have to find Python libraries for exporting to iCal and images
User Connections/Shareability	Databases, security	All members are familiar with databases	Internet	Limited time to implement