Software Requirements Specification

For

Class Scheduler

3/02/2021

Version 1.0

Prepared by:

Lake, Zunair, Thom, Adam, Elizabeth, Ryan

**Table of Contents**

[**REVISION HISTORY 3**](#_heading=h.32hioqz)

[**1**](#_heading=h.1hmsyys) **INTRODUCTION 4**

[**1.1**](#_heading=h.41mghml) **Overview 4**

[**1.2**](#_heading=h.2grqrue) **Goals and Objectives 4**

[**1.3**](#_heading=h.vx1227) **Scope 4**

[**1.4**](#_heading=h.3fwokq0) **Definitions 4**

[**1.5**](#_heading=h.1v1yuxt) **Document Conventions 5**

[**1.6**](#_heading=h.4f1mdlm) **Assumptions 5**

[**2**](#_heading=h.2u6wntf) **GENERAL DESIGN CONSTRAINTS 5**

[**2.1**](#_heading=h.19c6y18) **Product Environment 5**

[**2.2**](#_heading=h.3tbugp1) **User Characteristics 5**

[**2.3**](#_heading=h.28h4qwu) **Mandated Constraints 6**

[**2.4**](#_heading=h.nmf14n) **Potential System Evolution 6**

[**3**](#_heading=h.37m2jsg) **NONFUNCTIONAL REQUIREMENTS 6**

[**3.1**](#_heading=h.1mrcu09) **Usability Requirements 6**

[**3.2**](#_heading=h.44sinio) **Operational Requirements 7**

[**3.3**](#_heading=h.46r0co2) **Performance Requirements 7**

[**3.4**](#_heading=h.2lwamvv) **Security Requirements 7**

[**3.5**](#_heading=h.111kx3o) **Safety Requirements 7**

[**3.6**](#_heading=h.3l18frh) **Legal Requirements 7**

[**3.7**](#_heading=h.206ipza) **Other Quality Attributes 7**

[**3.8**](#_heading=h.4k668n3) **Documentation and Training 7**

[**3.9**](#_heading=h.2zbgiuw) **External Interface 7**

[**3.9.1 User Interface**](#_heading=h.1egqt2p) **7**

[**3.9.2 Software Interface**](#_heading=h.3ygebqi) **8**

[**4**](#_heading=h.2dlolyb) **SYSTEM FEATURES** 8

[**4.1**](#_heading=h.sqyw64) **Feature**: all features 9-15

Revision History

|  |  |  |  |
| --- | --- | --- | --- |
| **Version** | **Date** | **Name** | **Description** |
| 1.0 | 3/2/21 | Lake | Initial Document |
|  |  |  |  |

# **Introduction**

## ***Overview***

This document defines the requirements for the Class Scheduling system that is being created for UMKC faculty. The Client and Stakeholders of this system should be able to read this document to get an understanding of the system requirements and their correctness. As well as serve as a guide for the development team to design the product

This document does not cover subjects as cost, schedule, development methods, development phases, deliverables and testing processes. They will be covered in a separate document and test plan.

The Class Scheduling system is a web-based tool for setting schedules for semesters at UMKC. It provides a way for the staff to directly take notes and see past semesters to get a better understanding of the needs for the semester.

## ***Goals and Objectives***

The main goals of the class scheduling system is:

1. Provide an easy way to store and keep track of past semester schedules. This will make it easier to see what worked in the past and what did not.
2. Provide a space where notes about professors and specific classes can be stored. This is so if someone new needs to make the schedule it will make it easier on them.
3. Make creating a schedule for the next semester more time efficient.

## ***Scope***

The class scheduling system would enable users to utilize the software to keep track of the past semester schedules and also see the list of courses being provided for the current semester alongside their teachers. If given substantial time the scope could include different views for different users by utilizing their respective logins and could provide the option of creating example schedules.

## ***Definitions***

**Class Scheduling System Application** – the product that is being described here; the software system specified in this document.

**Project** – activities that will lead to the production of the Class Scheduling System application.

**Client** – the person or organization for which this Class Scheduling System application is being built.

**User** – the person or persons who will actually interact with the Class Scheduling System application.

**Use case** – describes a goal-oriented interaction between the system and an actor. A use case may define several variants called scenarios that result in different paths through the use case and usually different outcomes.

**Scenario** – one path through a user case

**Actor** – user or other software system that receives value from a user case.

**Developer** – the person or organization developing the system, also sometimes called the supplier.

**Stakeholder** – anyone with an interest in the project and its outcomes. This includes clients, customers, users, developers, testers, managers and executives.

## ***Document Conventions***

This document will contain unfinished parts and will be denoted by TBD.

## ***Assumptions***

It is assumed that the client has access to the internet and a computer with a web browser installed.

The team will be supplied with mock professor information for populating the DB. Gina wants to be able to use the product from anywhere she has the internet. This leads to implementing a 3-tiered product for accessing the DB from a web browser. Bingham has approved using the DB product produced in the 470 class in this project: attribution notes must be kept and a robust product is expected. If there are changes to this approval, any required decisions will be made by the entire group.

# **General Design Constraints**

## ***Product Environment***

The Scheduling system will be a component of a larger system. This will be using MySQL server and a server hosting service to store the data. It will also use MySQL for security and verify users. This server will hold a relational database for all the data storage.

## ***User Characteristics***

**Gina:** SCE’s scheduling specialist will be the main user of the application. To start with, she will be the only user. She should have complete control of the system and be able to modify any data and -most importantly- run the scheduling algorithm and approve schedules, as well as, see historical data.

**Professors:** (Stretch Goal) Any secondary users will be professors. Their use of the system will be limited to adding, modifying, and deleting their scheduling preferences and will be unable to see other professors information or any of the scheduling process, until approved and ‘posted’ by Gina.

## ***Mandated Constraints***

The application will be accessed via any common web browser. This was chosen based on experience with Python, SQL, and team consensus.

## ***Potential System Evolution***

Potential evolution of this system would likely see the ability to implement more restrictions and allowances. One example could include a function to factor in teacher holidays. Another likelihood would be, should the program operate to the client’s standards, that the client would want to consolidate any other programs being used for scheduling purposes of any kind. An example of this would be the ability to include accumulated vacation/sick time, time off requests, and scheduling for substitute teachers.

This program could be modified in the future based on changes in rules, attributes, and/or user base. For the rules, we -hope to- implement via a web service in order to allow future modifications without needing to update the DB or front-end. For attributes, we are unsure how this would be managed. For the user base, if we get to the point of having user logins, making a tiered privilege system seems like a good way to go.

# **Nonfunctional Requirements**

Final product should

* Only ever be offline for 15 minutes at a time and no more than 30 minutes per week
* Be accessible to multiple users at once
* Run on standard web browsers (chrome, firefox, edge)
* Keep user information, such as names and account details, secure

## ***Usability Requirements***

The flow of adding, modifying or removing professors, classes, classrooms, and professor scheduling preferences should be direct, easy, and undo-able (if a mistake was made). Creating potential schedules and choosing an approved schedule should be visually represented for ease of identifying any potential issues with the proposed schedules. Most of the usability validation will be per Gina, as she will be the main user. We hope to keep close contact with her while designing and implementing the UI for feedback.

## ***Operational Requirements***

There are no environmental requirements.

## ***Performance Requirements***

System connect time should be less than 3 seconds. With 30 concurrent users no operation should take more than 5 seconds and 95% of the operations should take less than 2 seconds. Generating a schedule might take longer if using AI guidance.

## ***Security Requirements***

Only authorized users would be allowed to access the data and features of the application. There could be the option of keeping track of the authorized users that have utilized the application on a given day or time.

## ***Safety Requirements***

There are no safety requirements for the use of this system.

## ***Legal Requirements***

All Personal information about professors will only be visible to authorized personnel and no sensitive information will be stored.

## ***Other Quality Attributes***

There are a few non-functional quality attributes that will need to be looked into for the application such as the availability of a classroom or if there is a time constraint for a professor to conduct a class due to their personal schedule.

## ***Documentation and Training***

The class scheduling system application will be delivered to users as a usable application without documentation or training. A user guide and system documentation may be provided to project stakeholders.

## ***External Interface***

### **User Interface - FrontEnd Person - TBD**

The user interface should look very professional and clear what classes are when. 75% of the users should know how to use 80% of the features within 1 min without training. 90% of users should understand at first look what time class is scheduled and what room it is in.

### **Software Interface -** FullStack Person - TBD

Using python and django all features will be available via HTTP output to the web browser. The exact protocol is TBD

# **System Features**/**Functional Requirements - Adam** - ForReview

This section describes the core requirements of the system. The use cases are used to describe these requirements. This will include cost, value, priority of the features. As well as Alterjnate paths to the i

## ***Required Features***

**4.1.1 Use Case: 1**

**Description: User Login / Check Schedule**

Actors: Head of Scheduling

Value = High

Cost = High

Basic Path

1. User clicks icon for scheduling application
2. System prompts user to enter user email and password.
3. User enters correct user email and password and clicks Login
4. System displays schedule, with options to click
   1. Scheduling Requests
   2. Instructors
   3. Generate Schedule
   4. Edit Schedule
   5. Logout
5. User clicks Logout
6. System exits

Alternate Path

1. User clicks icon for scheduling application
2. System prompts user to enter user email and password.
3. User enters incorrect user email or password and clicks Login
4. System displays error message: “Invalid Email or Password ..... Or you may have exceeded the number of consecutive attempts allowed. Please try again later.”
5. User may choose to login again, returning to step 1, or exit.
6. System exits.

**4.1.2. Use Case: 2**

**Description: Entering One or More Schedule Requests**

Actors: Head of Scheduling

Value = High

Cost = Medium

Basic Path

1. User successfully logs into application
2. User clicks the “Scheduling Requests” option
3. System prompts user to select “Enter”, “Edit”, or “Approve”
4. User clicks the link for “Enter”
5. System displays entry form with fields for the Teacher, Class, Time(s), and Date(s), as well as a field for notes.
6. Once all fields are filled, system prompts user to select “Cancel” or “Submit.”
7. User selects “Submit.”
8. Scheduling request is submitted and saved into the system for later access.
9. System prompts user to enter another request or return to home page.
10. User selects return to home page.
11. User is directed back to the home page.

Alternate Path

1. User successfully logs into application
2. User clicks the “Scheduling Requests” option
3. System prompts user to select “Enter”, “Edit”, or “Approve”
4. User clicks the link for “Enter”
5. System displays entry form with fields for the Teacher, Class, Time(s), and Date(s), as well as a field for notes.
6. Once all fields are filled, system prompts the user to select “Cancel” or “Submit.”
7. User selects “Cancel.”
8. Scheduling request is not submitted and user is directed back to the home page.

Alternate Path

1. User successfully logs into application
2. User clicks the “Scheduling Requests” option
3. System prompts user to select “Enter”, “Edit”, or “Approve”
4. User clicks the link for “Enter”
5. System displays entry form with fields for the Teacher, Class, Time(s), Date(s), Request Weight as well as a field for notes.
6. Once all fields are filled, system prompts user to select “Cancel” or “Submit.”
7. User selects “Submit.”
8. Scheduling request is submitted and saved into the system for later access.
9. System prompts user to enter another request or return to home page.
10. User selects “Enter Another Request.”
11. Repeat steps 5 - 9 until user selects “Cancel” or “Return to Home Page.”
12. User is directed back to the home page.

**4.1.3. Use Case: 3**

**Description: Editing Schedule Requests**

Actors: Head of Scheduling

Value = High

Cost = High

Basic Path

1. User successfully logs into application
2. User clicks the “Scheduling Requests” option
3. System prompts user to select “Enter”, “Edit”, or “Approve”
4. User clicks “Edit” option
5. System directs user to database of saved and approved scheduling requests, with the option to “return” to the home page
6. User selects desired request from database
7. User is able to edit room, time, date, and notes field as needed
8. User is prompted to select “Cancel” or “Submit”
9. User clicks “Submit”
10. System saves new request information
11. User is automatically redirected back to database of saved and approved scheduling requests
12. User clicks “return”
13. User is returned to the home page

Alternate Path

1. User successfully logs into application
2. User clicks the “Scheduling Requests” option
3. System prompts user to select “Enter”, “Edit”, or “Approve”
4. User clicks “Edit” option
5. System directs user to database of saved and approved scheduling requests, with the option to “return” to the home page
6. User selects desired request from database
7. User is able to edit room, time, date, and notes field as needed
8. User is prompted to select “Cancel” or “Submit”
9. User clicks “Cancel”
10. System does not save new request information
11. User is automatically redirected back to database of saved and approved scheduling requests
12. User clicks “return”
13. User is returned to the home page

**4.1.4. Use Case: 4**

**Description: Approving Schedule Requests**

Actors: Head of Scheduling

Value = High

Cost = Medium

Basic Path

1. User successfully logs into application
2. User clicks the “Scheduling Requests” option
3. System prompts user to select “Enter”, “Edit”, or “Approve”
4. User clicks “Approve”
5. System directs user to a database of scheduling requests that have not yet gone through approval process and is also given an option to “Return” to the home screen
6. User selects an unapproved request from the database
7. System displays the request details for review as well as the option to “Approve”, “Deny” or “Cancel”
8. User clicks “Approve”
9. Details of the request are stored as variables to be factored into the schedule generation
10. System directs user back to list of pending schedule requests.
11. User clicks “Return”
12. System directs user back to home page

Alternate Path

1. User successfully logs into application
2. User clicks the “Scheduling Requests” option
3. System prompts user to select “Enter”, “Edit”, or “Approve”
4. User clicks “Approve”
5. System directs user to a database of scheduling requests that have not yet gone through approval process and is also given an option to “Return” to the home screen
6. User selects an unapproved request from the database
7. System displays the request details for review as well as the option to “Approve”, “Deny” or “Cancel”
8. User selects “Deny.”
9. System removes request from all databases
10. System directs user back to list of pending schedule requests
11. User clicks “Return”
12. System directs user back to home page

Alternate Path

1. User successfully logs into application
2. User clicks the “Scheduling Requests” option
3. System prompts user to select “Enter”, “Edit”, or “Approve”
4. User clicks “Approve”
5. System directs user to a database of scheduling requests that have not yet gone through approval process and is also given an option to “Return” to the home screen
6. User selects an unapproved request from the database
7. System displays the request details for review as well as the option to “Approve”, “Deny” or “Cancel”
8. User selects “Cancel”
9. System directs user back to list of pending schedule requests
10. User clicks “Return”
11. System directs the user back to the home page.

**4.1.5. Use Case: 5**

**Description: Viewing Instructor Information**

Actors: Head of Scheduling

Value = High

Cost = Low

Basic Path

1. User successfully logs into application
2. User clicks “Instructors” option on home screen
3. System directs user to database of all instructors, as well as options to “Add” an instructor and “Return to Home.”
4. User selects any given instructor
5. System displays instructor name, class(es) taught, and any related schedule requests or “none” if there are no schedule requests associated with that instructor. System also displays “Return”.
6. User clicks “Return”
7. System automatically redirects user back to database of all instructors.
8. User clicks “Return to Home.”
9. System redirects user back to the home page of the application.

Alternate Case

1. User successfully logs into application
2. User clicks “Instructors” option on home screen
3. System directs user to database of all instructors, as well as options to “Add” an instructor and “Return to Home.”
4. User clicks “Return to Home.”
5. System redirects user back to the home page of the application.

**4.1.6. Use Case: 6**

**Description: Adding Instructor to the Database**

Actors: Head of Scheduling

Value = High

Cost = Medium

Basic Path

1. User successfully logs into application
2. User clicks “Instructors” option on home screen
3. System directs the user to the database of all instructors, as well as options to “Add” an instructor and “Return to Home.”
4. User clicks “Add”
5. System directs user to “add instructor” form that includes entry fields for name and classes taught by the instructor. System also displays options to “cancel” and “submit”
6. User enters relevant information
7. User clicks “Submit”
8. Information is stored to the database of all instructors
9. System redirects user to the previous screen that includes the updated database and “Add” and “Return to Home” options
10. User clicks “Return to Home”
11. System redirects user to the home page

Alternate Path

1. User successfully logs into application
2. User clicks “Instructors” option on home screen
3. System directs the user to the database of all instructors, as well as options to “Add” an instructor and “Return to Home.”
4. User clicks “Add”
5. System directs user to “add instructor” form that includes entry fields for name and classes taught by the instructor. System also displays options to “cancel” and “submit”
6. User clicks “Cancel”
7. Information is not stored to the database of all instructors
8. System redirects user to the previous screen that includes the updated database and “Add” and “Return to Home” options
9. User clicks “Return to Home”
10. System redirects user to the home page

**4.1.7. Use Case: 7**

**Description: Generate Schedule**

Actors: Head of Scheduling

Value = High

Cost = High

Basic Path:

1. User successfully logs into application
2. User clicks “Generate Schedule” option on home screen
3. System processes all approved schedule request parameters and applies information to create ideal, non-conflicting schedule of classes for the semester
4. System redirects user to new screen with generated schedule in an easy-to-read format that can be viewed in its entirety, along with options to “Approve” or “Cancel”
5. User selects “Approve”
6. System redirects user to home page
7. Home page now displays most recently approved schedule
8. User clicks “logout”
9. System logs user out of application

Alternate Path

1. User successfully logs into application
2. User clicks “Generate Schedule” option on home screen
3. System processes all approved schedule request parameters and applies information to create ideal, non-conflicting schedule of classes for the semester
4. System redirects user to new screen with generated schedule in an easy-to-read format that can be viewed in its entirety, along with options to “Approve” or “Cancel”
5. User selects “Cancel”
6. System redirects user to home page
7. Home page displays most recently approved schedule unchanged
8. User clicks “logout”
9. System logs user out of application

### **Additional Requirements**

Include in this section additional functional and non-functional requirements not specified in the use case(s) above.

## Optional ***Features - TBD***

### 