Software Requirements Specification

For

GradRooAte

March 1, 2020

Version 1

Prepared by:

Joshua Wendl

**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 5**

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

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

[**2**](#_heading=h.2u6wntf) **GENERAL DESIGN CONSTRAINTS 6**

[**2.1 Product Environment 6**](#_heading=h.19c6y18)

[**2.2 User Characteristics**](#_heading=h.3tbugp1) **6**

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

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

[**3**](#_heading=h.37m2jsg) **NONFUNCTIONAL REQUIREMENTS 7**

[**3.1 Usability Requirements**](#_heading=h.1mrcu09) **7**

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

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

[**3.4 Security Requirements**](#_heading=h.2lwamvv) **8**

[**3.5 Safety Requirements**](#_heading=h.111kx3o) **8**

[**3.6 Legal Requirements**](#_heading=h.3l18frh) **8**

[**3.7 Other Quality Attributes**](#_heading=h.206ipza) **8**

[**3.8 Documentatio**](#_heading=h.4k668n3)**n and Training 8**

[**3.9**](#_heading=h.2zbgiuw) **External Interface 9**

[3.9.1 User Interface](#_heading=h.1egqt2p) 9

[3.9.2 Software Interface](#_heading=h.3ygebqi) 9

[**4**](#_heading=h.2dlolyb) **SYSTEM FEATURES 9**

[**4.1**](#_heading=h.sqyw64) **Feature: Rule Insertion 10**

[4.1.1 Description and Priority](#_heading=h.3cqmetx) 10

[4.1.2 Use Case:](#_heading=h.1rvwp1q) New Rule 10

[4.1.3 Additional Requirements](#_heading=h.4bvk7pj) 10

[**4.2**](#_heading=h.2r0uhxc) **Feature: Rule Deletion 10**

[4.2.1 Description and Priority](#_heading=h.1664s55) 10

[4.2.2 Use Case:](#_heading=h.3q5sasy) Delete Rule 10

[4.2.3 Additional Requirements](#_heading=h.25b2l0r) 11

[**4.3**](#_heading=h.2r0uhxc) **Feature: Rule Edit 11**

[4.3.1 Description and Priority](#_heading=h.1664s55) 11

[4.3.2 Use Case: Change Rule](#_heading=h.3q5sasy) 11

[4.3.3 Additional Requirements](#_heading=h.25b2l0r) 11

[**4.4**](#_heading=h.2r0uhxc) **Feature: Schedule Checking 12**

[4.4.1 Description and Priority](#_heading=h.1664s55) 12

[4.4.2 Use Case: Check Schedule](#_heading=h.3q5sasy) 12

[4.4.3 Additional Requirements](#_heading=h.25b2l0r) 12

Revision History

|  |  |  |  |
| --- | --- | --- | --- |
| **Version** | **Date** | **Name** | **Description** |
| 1 | 03/01/20 | Joshua Wendl | Initial Document |
|  |  |  |  |

# **Introduction**

## ***Overview***

The GradRooAte application is a desktop application designed for specific University of Missouri - Kansas City employees that have the job to construct a proper class schedule. When an employee creates a schedule for UMKC, there are a set of criteria that must be fulfilled in order to please as many people as possible. The basis for GradRooAte is to assist in checking for situations in a schedule that does not meet the rules that have been input into the system.

The UMKC administrator will input a set of rules for every schedule that the GradRooAte will store and then overtime will check every schedule to make sure it follows the given rules. It will be the job of the administrator to make sure the rules are set correctly so that the system can check correctly.

This document will provide different requirements of the application. Alongside the requirements, this document will list out different goals, scope, define specifics and explain different variables that go into the GradRooAte. However, certain things like schedule, cost, technical details and more will not be addressed. That information will be provided in other documentation.

## ***Goals and Objectives***

GradRooAte was created with the employees of the University of Missouri - Kansas City in mind. With that being said, the application is expected to:

1. Have a user-friendly graphical user interface that allows employees to input rules, classes, professors, rooms and other items that are required to check a schedule.
2. Show any conflicting items that are contained in a certain schedule.
3. Stretch Goal: Give the ability to generate a functional schedule after enough variables are set in the system.

## ***Scope***

The GradRooAte application was created with the purpose of easing the jobs of certain University of Missouri - Kansas City employees. This will allow them to create a more perfect schedule, spend more time and effort in different areas of their job and create a better university eviernment for professors and students. Without a fair amount of rules put into the application, GradRooAte will not be able to help create the desired schedule.

## ***Definitions***

This section defines potentially unfamiliar or ambiguous words, acronyms and abbreviations. If terms such as “shall”, “should” and “may” are used to indicate importance the meaning of these terms should be defined here.

**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 use case

**Employee** – the intended user of the application. Usually refers to the University of Missouri - Kansas City.

**Administrator -** the key employee at the University of Missouri - Kansas City that will oversee the usage of this application.

**Product** – what is being described here; the software system specified in this document.

**Project** – activities that will lead to the production of the application described in the document.

**Should** – adverb used to show expectations. In most cases, this shows importances but not a guarantee.

**May** – adverb used to indicate a certain option. When used, there could be many different options and one or many will be chosen.

**Controls** – different options or tools that are options within the application; many of which will be visible in the graphical interface.

**System -** the underlying mechanics of the application. The system will run to save data, change data and run checks.

## ***Document Conventions***

This document is intended to be updated as progress of the application is made. Certain things may seem incomplete or in a beta state. Those will be addressed in more clarity through time.

## ***Assumptions***

With the creation of GradRooAte, there are a few key components that are assumed. Firstly, it is assumed that the University of Missouri - Kansas City labs and conference rooms will be available for meetings and group work. It will also be assumed that all members of the group will contribute their portion and give suitable feedback when requested. Also, communication will be readily available with the employees the application is intended for. Alongside project assumptions, there are a few assumptions made after the creation of the project. The application will be a desktop application ran off a single machine. The database and its functions will be stored off a single machine. This will ensure all data will be retained and stored properly.

# **General Design Constraints**

## ***Product Environment***

The GradRooAte application will run in parallel with a database that houses the rules for schedules, and time permitting, will house classes, rooms, professors and to other schedule components. These rules will be able to be stored for sure checks and future semesters when creating a new semester schedule will have some of the similar rules and conflicts that previous semesters did. The front end of the application will take the database and present it in a natural way that a user can edit, delete and create items in the database.

## ***User Characteristics***

Although the GradRooAte application will affect the lives of many different people, such as students and professors, the only users will be certain employees of the University of Missouri - Kansas City.

**Administrator -** The administrator of the system will have full permissions to do any creating, deleting and editing of rules and variables within the application. They will be in charge of the overall system.

**General Employees -** The general employees of the University of Missouri - Kansas City that have access to the application will have the ability to check schedules, and time permitting, have the ability to input rules to general a fresh schedule for further evaluations.

## ***Mandated Constraints***

The GradRooAte application will be designed using .NET and WPF(Windows Presentation Foundation). This desktop application will be a windows only application due to this. In further iterations, if there is a desire for other operating system functionality, porting the application to be a web application is a possibility. The backend database will be run off Oracle’s MySQL.

## ***Potential System Evolution***

As stated in prior sections, there is a desire to further the application to a state where it can generate a fresh schedule after given an appropriate amount of rules and variables. This will further ease the job of the University of Missouri - Kansas City employees. It is understood that many different things go into creating a schedule, some of which cannot be understood by a computer, but the application would have the ability to help the users get a better grasp on different variables and ideas.

Also, if desired, the application could be ported to be a web application for most streamline use. This will depend on the number of users the application has using it, as well as the desire for different operating systems to house the application. As the development of the application furthers, there will be more ideas to further develop it.

# **Nonfunctional Requirements**

## ***Usability Requirements***

Because the GradRooAte will only be used by the University of Missouri - Kansas City employees, the effort to make it usable is a specific one. The application will have a learning curve due to the method of which rules are input to the system. The majority of users should not have to read the user manual, but they will benefit from it greatly. Having a base knowledge on how rules and variables are input into the system will make for a much better user experience.

Inputing a schedule will be far much easier. There will be places to enter each attribute of the schedule and the ability to add more fields/classes to the schedule. This aspect of the application should require no reading of the user manual for most users.

## ***Operational Requirements***

The users’ environment is their office, because of this, the user interface should be designed well enough to keep their interest. Also, because working environments are quiet, the sound the application makes will be kept to a minimum.

## ***Performance Requirements***

The GradRooAte application should take no more than 10 seconds to start up. Because it is a simple desktop application that does not require internet access, the application will not bottleneck a system while running in the background. Given that there are no more than 100 rules, it should take no longer than 30 seconds to check any given schedule. When schedule generation is implemented, projected time is around one minute; this is due to the amount of checking, and optimization that is taking place. When inputting rules, it should take no longer than three seconds to submit at rule after pressing the submit button.

All of these values are subject to change and vary depending on the quality of the machine it is running on.

## ***Security Requirements***

Because the GradRooAte application can hold semi-permanent information that if overridden can impact the jobs and lives of many people, there will be a login page that will help separate users, as well as secure that only permitted users are granted access within the system. The database will house the information for the users to sign in to the application.

## ***Safety Requirements***

Overall, the GradRooAte application does not have the ability to endanger the safety of people. It is a simple desktop application that simply runs when opened. The greatest risk it presents is the possibility to suggest a bad schedule. In that case, it is the University of Missouri - Kansas City employee’s job to catch that something is wrong and not submit that schedule for semester submission.

## ***Legal Requirements***

The GradRooAte application does not interfere with any legal organizations. The only sensitive information the application holds is the user credentials to access the system. This information will be protected by the backend of the system and only be accessible by the given user.

## ***Other Quality Attributes***

The GradRooAte application will start out on only a single machine. Over time, the team plans to make the application portable and have the ability to be housed in many different locations. This implementation will require a database housed on a remote server where it can be accessed from many different locations with the correct security permissions. Also, because it is a simple application during the first iteration, it will be available at any moment when the user decides to use it. When it becomes multi-platform, the server will require maintenance for regular availability.

## ***Documentation and Training***

Training for GradRooAte will be kept minimal. For the user, it is a basic application for setting rules and checking information over time. The original user, administrator, there will be a short information training session that will teach them how to input rules and perform a quick check. For every employee to learn the application thereafter, the only training that will be needed is for the original administrator to show them how to input rules. Because the application is going to be kept relatively simple, inputting and editing rules will be the only area of confusion. After a short instructional session with the administrator, any user should be able to use the application with no problem.

## ***External Interface***

### **User Interface**

Because the GradRooAte application was designed with the professional in mind, there is not much artistic design implemented within it. It has 4-5 pages that are used to input data. Data driven applications are designed for that purpose, efficiency for inputting data. The user interface will help streamline the input of data so the user can do their job in a reasonable amount of time.

With the idea of data input in mind, it is important to help the user not be bored when using the application. The graphical user interface will incorporate the colors and design of the University of Missouri - Kansas City. Because the colors are bright and entertaining, that will make the user have interest from the beginning. As well, the user interface will have different buttons and options to interact with, ie. places to find out more information, ability to read data and more. It is important to show the user the correct data, but also keep them interested with the task at hand.

As the application grows, there will be more room for it to incorporate different functions. With the growth of different functions, the team plans to implement different graphical features to enhance the visual design of the application. With that said, the main goal of the application is ease of input and reading data. This will never change as the main goal of the application. For example, over 80% of users should be able to easily find and read the data given to them.

### **Software Interface**

The client of the GradRooAte application has requested that minimal to no programming knowledge will be needed to use the application. Because of this the team has made sure that the application can be used without any user interaction with software besides the user interface to the application. The only interaction the user will have to have with software is during installation. However, this should be relatively easy due to the packaging of the application.

# **System Features**

From the application origin there are two main features it will incorporate: insertion of rules and schedule checking. The two features are directly correlated by the fact that schedules will be checked against rules that have been input into the system.

## ***Feature:*** Rule Insertion

### **Description and Priority**

The function of inserting a rule into the application is a requirement due to the main purpose of checking a schedule against inserted rules. These rules will have many different options such as professor limits, student limits, classroom limits, and more yet to be seen. They will be inserted and then checked against any given schedule.

Cost: Medium

Risk: Low

Value: High

### **Use Case:** New Rule

The user will have a known rule or restriction that restricts certain situations in a schedule. The user will first log into the system. Afterwards, they will select that they want to input a new rule. Then, they will be prompted with what kind of rule they wish to insert. They will then be prompted to fill in values that would explain the given rule; the more detail the user fills in, the more solid the rule will be. After everything they wish is filled in, they will press submit and the rule will be inserted into the database.

### **Additional Requirements**

Because it is hard for a system to know all different options a rule could have, the user may need to add new fields or rules over time. This will assist the application in learning new ways of checking.

## ***Feature:*** Rule Deletion

### **Description and Priority**

Over time, rules will become outdated and will be required to be deleted. If a professor does not require their rule or a class is canceled it will require new rules and old ones to get deleted. The user will then decide which rules to delete and remove it from further checks.

Cost: Medium

Risk: High

Value: Medium

### **Use Case:** Delete Rule

After an event takes place the user will know to delete rules pertaining to that event. The user will first log into the system, then choose they want to delete a rule. They can search for that rule, delete all rules, or not delete any from there. After searching they will be given a list of rules that pertain to their search. When the rule they want to delete is found, they will check that rule to be deleted. Once submitted, that rule will be deleted from the database.

### **Additional Requirements**

It may be that the user desires to only temporarily remove a rule from the system, in this case, the team may implement the ability to suspend rules. This will mark the rule in a suspended state and make it so the rule is no longer checked against schedules unless unsuspended.

## Feature: Rule Edit

## Description and Priority

Over time, rules may change and it will require users to edit the already inserted rules instead of deleting one and adding a new one entirely. This will help save time and add new dimensions to already inserted rules. The user will have the choice to add, change and remove aspects to every rule that has been inserted.

Cost: Low

Risk: Medium

Value: Medium

### Use Case: Change Rule

Multiple things will require a user to have to edit a rule. An error may have occurred when inserting, something may have changed temporarily or changed permanently. The application will allow a user to change any rule in any way whenever they need to. They will start by logging into the system and then select to edit a rule. When wanting to edit a rule, the user will be required to select which rule they want to edit. They will do this by searching from all of the rules within the system. When they have found the rule they wish to edit, they will be brought to a page where they can change the values of the rule, add values, remove values or change the type of rule. Once the proper edits have been inserted, they will have to submit the edit. After submitting, the database will update with the new values the user has submitted.

### Additional Requirements

It may be possible the user will want to create different situations per rule. If the client wishes to may this a feature, it will be added later that a rule can contain its own rules. For example, if professor A teaches class X then rule Y. However, if the same professor teaches class W then rule Z. This feature would take additional information on the user end of the application.

## Feature: Schedule Checking

## Description and Priority

All of the rules that have been inserted into the system are only there for the purpose of checking them against a schedule. This function is the main reason for the creation of the application. It will take a schedule created by a user and check all rules against it to make sure there are no conflicts with it. In further iterations, the feature for the application to recommend changes to make the schedule work, if possible.

Cost: Medium

Risk: Low

Value: High

### Use Case: Check Schedule

The user will create a semester schedule and in the case there are conflicts within it, they will want to check it against the application. First, the user will log into the system and then select to check a schedule. They will then insert each class, room, building, professor and all variables a instance will have. After all items have been inserted, they will submit the schedule for checking. The application will make checks against all of the rules that have been inserted. After a few moments, a report with all the conflicts will be presented. From there, the user can make changes to the schedule to run another check.

### Additional Requirements

Assuming the user will want to check the schedule again at a later time, the ability to save schedule within the system will be a function. Also, any checking reports will be saved with that schedule. This will give the option for users to save records and make modifications over time.