**Software Project Management Plan**

**GradRooAte**

March 6, 2020

**Team Members**

David Leatherwood

Joshua Wendl

Alec Sears

Kendal Sutton

Document Control

**Change History**

|  |  |  |
| --- | --- | --- |
| **Revision** | **Change Date** | **Description of changes** |
| V1.0 | 3/6/2020 | Initial release |
|  |  |  |

**Document Storage**

**Google Drive:**

<https://drive.google.com/drive/folders/11kIuwggv3lp0DlZSqAeHGnxIArGHGj5i>

**Github:**

https://github.com/umkc-cs-451-2020-spring/semester-project-group-10/projects/2

**Document Owner:**

David Leatherwood is responsible for developing and maintaining this document.

**Table of Contents**

1. [**Overview**](#_heading=h.ea3ni9yecp4y) **4**
   1. [Purpose and Scope](#_heading=h.1fob9te) 4
   2. [Goals and Objectives](#_heading=h.3znysh7) 4
   3. [Project Deliverables](#_heading=h.2et92p0) 4
   4. [Schedule and Budget Summary](#_heading=h.3dy6vkm) 5
   5. [Success Criteria](#_heading=h.1t3h5sf) 5
   6. [Definitions](#_heading=h.4d34og8) 6
   7. [Evolution of the Project Plan](#_heading=h.8j20l0ppxvil) 6
2. [**Startup Plan**](#_heading=h.2s8eyo1) **7**
   1. [Team Organization](#_heading=h.17dp8vu) 7
   2. [Project Communications](#_heading=h.3rdcrjn) 7
   3. [Technical Process](#_heading=h.26in1rg) 7
   4. [Tools](#_heading=h.lnxbz9) 7
3. [**Work Plan**](#_heading=h.35nkun2) **8**
   1. [Activities and Tasks](#_heading=h.1ksv4uv) 8
   2. [Release Plan](#_heading=h.44sinio) 9
   3. [Budget](#_heading=h.z337ya) 10
4. [**Control Plan**](#_heading=h.3j2qqm3) **12**
   1. [Monitoring and Control](#_heading=h.1y810tw) 12
5. [**Supporting Process Plans**](#_heading=h.yjjxvs60s4xm) **12**
   1. [Risk Management Plan](#_heading=h.c00u103i9i9m) 12
   2. [Configuration Management Plan](#_heading=h.r3hwqs49tikw) 12
   3. [Verification and Validation Plan](#_heading=h.y49xdx7ljrkk) 12

# 

# Overview

## Purpose and Scope

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. This desktop application is being initiated for the needs of the UMKC scheduling department. Previous scheduling was done by hand and paper.

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 environment for professors and students. Without a fair number of rules put into the application, GradRooAte will not be able to help create the desired schedule.

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.

## Goals and Objectives

GradRooAte was created with the employees of the University of Missouri - Kansas City in mind. 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.

## Project Deliverables

This section lists the deadline dates for project deliverables:

1. 03/16/2020 – Technical Prototype
2. 04/03/2020 – Architecture Document
3. 04/27/2020 – User and System Guide
4. 05/01/2020 – Project Results

Assumptions and Constraints

With the creation of GradeRooAte, there are a few key components that are assumed:

* University of Missouri - Kansas City labs and conference rooms will be available for meetings and group work.
* all members of the group will contribute their portion and give suitable feedback when requested.
* communication will be readily available with the employees the application is intended for
* The application will be a desktop application run 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.

Constraints of GradeRooAte:

- The GradRooAte application will be designed using .NET and WPF(Windows Presentation Foundation).

- This desktop application will be a windows only application.

- 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 run on a SQL database.

## Schedule and Budget Summary

**Estimated Time:**

1 project manager at 3 hours per week for 12 weeks 36 hours

1 requirement engineer at 3 hours per week for 12 weeks 36 hours

2 software engineers at 3 hours per week each for 12 weeks 72 hours

**144 hours total**

**Cost:**

Project Manager = $60/hr \* 36 hr = $2160

Requirement Engineer = $50/hr \* 36 hr = $1800

Software Engineer = $35/hr \* 72 hr = $2520

**Total Budget = $6480**

Milestone Name Start Date Closeout Date

Iteration 1 02/17/2020 03/02/2020

Iteration 2 03/02/2020 03/16/2020

Iteration 3 03/16/2020 04/06/2020

Iteration 4 04/06/2020 04/20/2020

Iteration 5 04/20/2020 05/04/2020

Project Closeout 02/17/2020 05/04/2020

## Success Criteria

GradRooAte will be a success if users can generate a schedule that violates the least amount of constraints and be used to effectively be implemented. Given that all constraints might not be obtainable, success is determined on the least amount of violations. High level schedule for the required material should be completed and delivered by May 5, 2020.

## 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.

## Evolution of the Project Plan

Because the team is using an adaptive project process, it is known that plans will change and migrate iterations during the project. However, at the beginning of each iteration the teams saw what went right and what went wrong and plans accordingly. If things did not get completed, the team moves the task to a future iteration or puts it in backlog if it is not a requirement. During an iteration closeout, the team looks over what tasks they have remaining or what tasks have been completed for future iterations. Those tasks that need to be moved are then discussed throughout the team and moved accordingly.

# Startup Plan

## Team Organization

Project Manager:

The project manager is in charge of creating, maintaining, and tracking documentation. Also, the project manager is responsible for scheduling team meetings, tracking team member’s progress and making due dates.

Project Manager Assistant:

The project manager assistant will be a combination of a software developer and a project manager. They will be responsible for little coding, minor documentation updates and maintaining progress throughout iterations.

Software Developers (2):

Software Developers are mainly responsible for coding and unit testing. They are also responsible for updating the project manager on progress, planning and milestone requirements.

## Project Communications

Communication between team members will operate by using the team Slack channel. This is mandated by the professor overseeing the project. Communication with the client will take place via emails and documentation turn ins. The team will receive feedback after each milestone has been turned in and reviewed. Questions are expected to be asked and answered during the entire project by form of emails.

## Technical Process

As a team, the agile working process is being used. We are using project boards, iteration plans, global communication and other forms of keep progress up to pase. Also, because the team knows plans will actively change, we have agreed on using an adaptive way of working. Things will constantly change during iterations due to hour requirements and resources available. Having the adaptive project process will help with any obstacles that come along during the project.

## Tools

Programming Languages - SQL and .NET/WPF.

Version Control - All documentation and source code will be kept in team Github and Google Drive listed in this report.

Progress Control - Github will house the project’s boards that help track work progress.

Testing - Testing will be mostly manual, but automated testing will be used on different rule testing.

Communication - Team communication will be tracked in the team’s Slack channel.

# Work Plan

## Activities and Tasks

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
|  |  |  | **Role** | **Owner** | **Estimated** | **Effort** | **Actual** |
|  |  |  |  | **(primary owner) (**secondary owners**)** | **By Task** | **Subtotals** | **By Task** |
| **Preliminary &** |  |  |  |  |  |  |  |
| **Parallel Tasks** | **Overview** |  |  |  |  | **4** |  |
|  |  | Design Project Plan | Project Manager | **Alec, David, Kendal, Joshua** | 4 |  | 4 |
|  | **Requirements** |  |  |  |  | **9** |  |
|  |  | Usage of Required Communication Outlets | All | **Alec, David, Kendal, Joshua** | 1 |  | 1 |
|  |  | Analyze Requirements | All | **Alec, David, Kendal, Joshua** | 4 |  | 4 |
|  |  | Determine strengths/assign roles | All | **Alec, David, Kendal, Joshua** | 2 |  | 2 |
|  |  | Schedule Meet Up Times/Days (Agile) | All | **Alec, David, Kendal, Joshua** | 2 |  | 2 |
|  | **Documentation** |  |  |  |  | **19** |  |
|  |  | Project Charter | Project Manager | **Kendal** | 2 |  | 3 |
|  |  | Release Plan | Project Manager | **Joshua** | 2 |  |  |
|  |  | Requirements Document | Project Manager, Requirements Engineer | **Joshua** | 4 |  | 4 |
|  |  | Project Plan | Project Manager, Developers | **David** | 2 |  |  |
|  |  | Architecture Document | Architect, Developers | Kendal | 3 |  |  |
|  |  | Test Report | Developers, Tester | **David,** Alec | 2 |  |  |
|  |  | User Guide/Admin Doc | Developers, Requirements Engineer | **Kendal, Joshua** | 4 |  |  |
| **Planning** | **Development** |  |  |  |  | **23** |  |
| **Iteration 1:** |  | Design Overall Scope and Objectives |  | **Alec, Kendal, Joshua** | 3 |  | 3 |
|  |  | Determine Desired Technologies | Developer, Project Manager | **Alec, David, Kendal, Joshua** | 4 |  | 5.5 |
|  |  | Read/Understand Documentation on New Technologies | Developer | **Alec, David, Kendal, Joshua** | 10 |  | 12 |
|  |  | Design Database Requirements | Developer | **Joshua, David** | 6 |  | 6 |
|  | **Analysis** |  |  |  |  | **7** |  |
|  |  | Prepare for next iteration | Project Manager, Developers | **Alec, David, Kendal, Joshua** | 4 |  | 6 |
|  |  | Finalize Present Iteration | Project Manager, Developers | **Kendal, Joshua, David** | 3 |  | 3 |
| **Coding** | **Design** |  |  |  |  | **18** |  |
| **Iteration 2:** |  | Design High Level Front End Design | Developer | **Alec,** Joshua | 6 |  |  |
|  |  | Design Models and Relationships | Developer | **Joshua,** Alec | 8 |  |  |
|  |  | Design Checking Process | Architect, Developers | **Alec,** Joshua, | 4 |  |  |
|  | **Development** |  |  |  |  | **16** |  |
|  |  | Implement Database Structure | Developer | **Joshua,** David | 4 |  |  |
|  |  | Create Basic UI for Further Development | Developer | **Alec**, Kendal | 4 |  |  |
|  |  | Add Unit Testing Framework | Developer, Tester | **Kendal**, Alec | 2 |  |  |
|  |  | Implement Schedule Checking Process | Developer | **Joshua,** Alec | 6 |  |  |
|  |  | Update UI with schedule checking | Developer | Kendal, Alec |  |  |  |
|  | **Analysis** |  |  |  |  | **9** |  |
|  |  | Prepare for next iteration | Project Manager, Developers | **Alec, David, Kendal, Joshua** | 4 |  |  |
|  |  | Finalize Present Iteration | Project Manager, Developers | **Kendal, Joshua, David** | 3 |  |  |
|  |  | Prepare Iteration 2 Presentation | Project Manager |  | 2 |  |  |
| **Testing** | **Design** |  |  |  |  | **0** |  |
| **Iteration 3:** |  |  |  |  |  |  |  |
|  | **Development** |  |  |  |  | **0** |  |
|  | **Analysis** |  |  |  |  | **0** |  |
|  |  | System Testing | Tester, Developer | Kendal | 3 |  |  |
|  |  | Bug fix from testing results | Tester, Developer | Alec | 4 |  |  |
| **Total** |  |  |  |  | **0** | **0** | **0** |

## Release Plan

Expected completion date will be May 5, 2020. The project will have 5 milestones which are iterations. For approximately two weeks, there will be an iteration starting and the past iteration closing. Iteration 5 will be the final iteration and will also be the product release.

## Budget

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
|  |  |  | **Role** | **Estimated** | **Effort** | **Actual** | **Cost** |
|  |  |  |  | **By Task** | **Subtotals** | **By Task** |  |
| **Preliminary &** |  |  |  |  |  |  |  |
| **Parallel Tasks** | **Overview** |  |  |  | **4** |  | $**240** |
|  |  | Design Project Plan | Project Manager | 4 |  | 4 |  |
|  | **Requirements** |  |  |  | **9** |  | $**450** |
|  |  | Usage of Required Communication Outlets | All | 1 |  | 1 |  |
|  |  | Analyze Requirements | All | 4 |  | 4 |  |
|  |  | Determine strengths/assign roles | All | 2 |  | 2 |  |
|  |  | Schedule Meet Up Times/Days (Agile) | All | 2 |  | 2 |  |
|  | **Documentation** |  |  |  | **19** |  | $**1040** |
|  |  | Project Charter | Project Manager | 2 |  | 3 | $180 |
|  |  | Release Plan | Project Manager | 2 |  |  | $120 |
|  |  | Requirements Document | Project Manager, Requirements Engineer | 4 |  | 4 | $190 |
|  |  | Project Plan | Project Manager, Developers | 2 |  |  | $95 |
|  |  | Architecture Document | Architect, Developers | 3 |  |  | $225 |
|  |  | Test Report | Developers, Tester | 2 |  |  | $70 |
|  |  | User Guide/Admin Doc | Developers, Requirements Engineer | 4 |  |  | $160 |
| **Planning** | **Development** |  |  |  | **23** |  | **$970** |
| **Iteration 1:** |  | Design Overall Scope and Objectives |  | 3 |  | 3 | $150 |
|  |  | Determine Desired Technologies | Developer, Project Manager | 4 |  | 5.5 | $190 |
|  |  | Read/Understand Documentation on New Technologies | Developer | 10 |  | 12 | $420 |
|  |  | Design Database Requirements | Developer | 6 |  | 6 | $210 |
|  | **Analysis** |  |  |  | **7** |  | **$320** |
|  |  | Prepare for next iteration | Project Manager, Developers | 4 |  | 6 | $190 |
|  |  | Finalize Present Iteration | Project Manager, Developers | 3 |  | 3 | $130 |
|  |  |  |  |  |  |  |  |
| **Coding** | **Design** |  |  |  | **18** |  | **$630** |
| **Iteration 2:** |  | Design High Level Front End Design | Developer | 6 |  |  | $210 |
|  |  | Design Models and Relationships | Developer | 8 |  |  | $280 |
|  |  | Design Checking Process | Architect, Developers | 4 |  |  | $140 |
|  | **Development** |  |  |  | **16** |  | **$560** |
|  |  | Implement Database Structure | Developer | 4 |  |  | $140 |
|  |  | Create Basic UI for Further Development | Developer | 4 |  |  | $140 |
|  |  | Add Unit Testing Framework | Developer, Tester | 2 |  |  | $70 |
|  |  | Implement Schedule Checking Process | Developer | 6 |  |  | $210 |
|  |  | Update UI with schedule checking | Developer |  |  |  |  |
|  | **Analysis** |  |  |  | **9** |  | **485** |
|  |  | Prepare for next iteration | Project Manager, Developers | 4 |  |  | $190 |
|  |  | Finalize Present Iteration | Project Manager, Developers | 3 |  |  | $175 |
|  |  | Prepare Iteration 2 Presentation | Project Manager | 2 |  |  | $120 |
| **Testing** | **Design** |  |  |  | **0** |  |  |
| **Iteration 3:** |  |  |  |  |  |  |  |
|  | **Development** |  |  |  | **0** |  |  |
|  | **Analysis** |  |  |  | **0** |  | **$245** |
|  |  | System Testing | Tester, Developer | 3 |  |  | $105 |
|  |  | Bug fix from testing results | Tester, Developer | 4 |  |  | $140 |
| **TOTAL COST:** |  |  |  |  |  |  | **$4940** |

# Control Plan

## Monitoring and Control

Weekly – Team meeting. Project participants report status, progress and potential problems.

3/14/2020 - Technical Prototype Review

4/1/2020 - Architecture Document Review

4/25/2020 - User and System Guide Review

4/30/2020 - Review Final Project

# Supporting Process Plans

## Risk Management Plan

Conflicting Schedules - The process to handle this risk is to meet once a week. Even though we all might not be able to meet, others who attended will be able to inform those who did not attend.

Lack of Communication with Client - schedule a set contact time with client.

Technology Failure - To prevent this risk, all members will be able to efficiently use GradRooAte on personal devices.

## Configuration Management Plan

1. Documentation, codes, testing, and other project requirements will be stored/submitted in GitHub.
2. Any changes made to the project after the project’s due date must be communicated with the team members. Must provide the information of the change and reason for the change.
3. Any changes made to the project after the project’s due date must be entered in the change history should include the revision, the date, and the description of change.

## Verification and Validation Plan

For the verification plan, due dates for milestone’s interteration, peer review, and designing prototype. Validation includes testing of the languages, presentations, and review.