**Rush Hour – Effort Management Application**

**SWE 6633 – Software Project Management & Planning**

**Feb. 24th, 2023**

**Group 3 Members:**

Justin Hall

Jhaedyn Gibbs

Mielat Habte

Jonathan Gomez Torres

Toluwase Gbenle

Table of Contents

[Introduction 2](#_Toc128340461)

[Project Overview 2](#_Toc128340462)

[Development Team 2](#_Toc128340463)

[Functionality & Features 3](#_Toc128340464)

[Goals 4](#_Toc128340465)

[Deliverables 5](#_Toc128340466)

[Development Tools 5](#_Toc128340467)

[Project Schedule 6](#_Toc128340468)

[Development Model 6](#_Toc128340469)

[Deliverable Schedule: 6](#_Toc128340470)

[Estimated Timeline: 7](#_Toc128340471)

[Cost Assessment 7](#_Toc128340472)

[Risk Assessment 8](#_Toc128340473)

[Assumptions: 8](#_Toc128340474)

[Risks: 8](#_Toc128340475)

[Summary 9](#_Toc128340476)

[References 10](#_Toc128340477)

# Introduction

Project management is often one of the most difficult and ephemeral aspects of developing software. Though the coding process has more than its fair share of issues and nuances, management remains a larger issue because of the need to supervise all of those issues and nuances. Managers and development teams often require the use of software tools to help them track and estimate the progress of a software project. Furthermore, there often exists a need to condense all of the relevant technical information into a format that stakeholders and financial backers can easily understand, leading to a larger need for effective communication skills and tools to help support that communication.

Considering all of these responsibilities and needs, project managers and their teams need automated tools that can support and extend their capabilities. We plan to develop a tool that can help shoulder the burden of project planning and management, from handling project requirements and tracking time investments all the way down to organizing documentation and generating reports. The Rush Hour project is dedicated to supporting the most vital aspects of project development, while also giving those overworked managers and coders some much-needed relief.

# Project Overview

## Development Team

**Group Leader:** Justin Hall

* Contributions: (Did his share)
  + Led weekly sprint meetings
  + Investigated backend frameworks
  + Completed Risks & Assumptions Quick Plan section
  + Edited Quick Plan document

**Group Members:**

* Jhaedyn Gibbs
  + Contributions: (Did his share)
    - Attended weekly sprint meetings
    - Completed Cost Assessment section
    - Edited Quick Plan document
* Mielat Habte
  + Contributions: (Did her share)
    - Managed Jira board
    - Attended weekly sprint meetings
    - Completed Project Schedule section
    - Edited Quick Plan document
* Jonathan Gomez Torres
  + Contributions: (Did his share)
    - Attended weekly sprint meetings
    - Completed Functionality & Features, Goals, and Deliverables sections
    - Edited Quick Plan Document
* Toluwase Gbenle
  + Contributions: (Did his share)
    - Attended weekly sprint meetings
    - Completed Functionality & Features, Goals, and Deliverables sections
    - Edited Quick Plan Document

## Functionality & Features

The most central features for the project were specified in a project overview document provided to the development team. It was agreed that all of the provided features would be implemented as a baseline, with some additional features introduced at the recommendation of the development team. All originally specified features will be indicated in bolded text, with any additional recommended features remaining normal. As the project develops further and the team settles into their chosen development methodology, it will quickly become apparent how many of the recommended features can be added alongside the baseline.

***Generic Functionalities:***

* **A General section that allows the input of the following information:**
  + **A high-level description of the software project**
  + **The name of the owner or project manager**
  + **A list of project team members (which may vary as the project progresses)**
  + **A list of risks and risk status (which may vary as the project progresses)**
* **Allow the user to enter a list of functional and non-functional requirements for the software project.**
* New users should be introduced to the Rush Hour application with a skippable tutorial.
* The application should allow any users to create and manage multiple projects.
* The application should allow people to sign up for accounts with an email address, name, and password.
* The application should allow users to log in with a username or email address and password.
* The application should allow users to reset their password if necessary.
* The application should allow users to invite others to contribute to their project. This should be by inputting their email address, which should send an email notification and invite link if they are not already a user or a notification within the app if they already have an account.
* The application should allow users to contribute to projects that they did not create if they accept an invite from another user.
* Allow users to store documents in association with a project.
* The application should allow users to download documents from projects that they have access to.
* The application should allow users to indicate whether a requirement has been met, not met, or in progress.
* The application should have an aesthetic and easy-to-understand user interface.
* The application should store its information in a relational database.
* The application should be able to communicate with the relational database through REST API calls.

***Project Effort Monitoring and Tracking:***

* **Allow the user to enter (at a daily or weekly basis) the effort in number of person hours expended on Requirements Analysis, Designing, Coding, Testing, Project Management (by each of the earlier entered requirements).**
* **Allow the user to view the total expended hours by requirements analysis, designing, coding, testing, project management (by each of the requirement entered).**
* The application should provide a search functionality that can filter data according to all relevant metadata such as user, sprint name, project name, project phase, etc.
* The application should allow users to separate time investments according to a provided sprint schedule.
* Allow the user to export a formatted report of the project’s requirements and efforts in pdf format.

## Goals

The ultimate goal of the Rush Hour project is to produce a user-friendly project management tool that allows software development teams to coordinate project planning activities such as time expenditure tracking and project requirements in a simple and intuitive way. The Rush Hour application is seen as a sort of supplemental tool that targets aspects of the software development left out by other major workflow managers such as Jira, Trello, and VisCon.

To our current knowledge, the other major competitors do not offer a fully functional service that tracks the full assortment of time expenditures and project requirements while also offering free report generating and document coordinating functionalities. Simplicity and a focus on providing the most useful features are the edge that should provide our users with a superior experience. We hope that software developers choose to use our simplified but useful application in addition to their preferred work management tools.

As this is the first stage of development, some of the more advanced features might prove to be too ambitious for the time frame that the project must be developed in. Further thought needs to be done on positioning and possible cross-compatibility with the other major tools, but the development efforts behind those features will most likely have to occur in later stages of development.

## Deliverables

The following list details some of the deliverables for the initial stage of project development:

* Prototype web application and database source code
  + Any code or other software artifacts generated during the development effort.
* Quick Project Plan
  + Initial planning document made before development work begins.
* Comprehensive Project Plan
  + Planning document made once development work begins. Contains more details and nuanced projections of the product.
* Software Design Document
  + Document detailing the architecture and rationale of the Rush Hour software, including the frontend, backend, and database schemas.
* Software Presentations
  + Any presentations or materials made to better convey the achievements and progress of the development efforts.
* Weekly Meeting Reports
  + Any notes or documents generated during the weekly sprint review and planning meetings.
* GitHub Repo
  + The GitHub repository containing the source code and git history for the prototype.

## Development Tools

In order to properly manage and develop high quality software, the following tools will be used:

* GitHub Repo
  + Store source code and other software artifacts
  + Handle source control for the project
  + Use GitHub actions for any automated testing suites
* Jira Board [1]
  + Track Sprints and Tasks
  + Assign tasks to development team members
* ReactJS Framework (Frontend) [2]
  + JavaScript framework to develop the user interface
  + Handles routing and initiates API calls
* Appwrite (Backend) [3]
  + Open-source Backend-as-a-service (BaaS) framework
  + Handles API calls and communicates with the database
  + Also handles authorization and other functions
  + Utilizes Docker containers [7]
* MySQL (Database) [4]
  + Software to create and manage relational databases
  + Could possibly be another framework depending on compatibility
* Visual Studio Code (IDE)
  + Most popular Integrated Development Environment currently available
* AWS (Hosting Service) [6]
  + Allows for remote hosting of the web application and database
  + Would prefer to use free resources and accounts if possible
  + Can be replaced with other hosting services if they offer better functionality

## Project Schedule

The following sections will go into detail on the chosen SDLC (Software Development Life Cycle) development model and the tentative schedule for deliverables and development. The project is currently constrained to a timeframe of a few months, with a few key deliverables and checkpoints.

### Development Model

Our team will be using the Agile Methodology, as it will help our team deliver functionality in steady, continuous increments. Requirements, plans, and development will be evaluated continuously so our team will have a natural mechanism to respond to bugs quickly. We will have 5 stages in our software development lifecycle: Ideation, Development, Testing, Deployment, and Operations. We decided to follow this methodology, because most of the team is already familiar with the agile process. We also believed it would be the easiest way to make sure everyone is doing their part in this project.

### Deliverable Schedule

Below is the schedule when the team is planning to submit each deliverable. The deliverable due date means that the finalized version of the team’s deliverable submission must be completed by that date, and the team leader will submit the deliverable on the team’s behalf.

|  |  |  |  |
| --- | --- | --- | --- |
| **Deliverable #** | **Deliverable Name** | **Description** | **Due Date** |
| 001 | Quick Project Plan | Quick pre-development plan | 02/26/2023 |
| 002 | Comprehensive Project Plan | Detailed plan with full development details | 03/26/2023 |
| 003 | Developed Prototype | All deliverables submitted | 05/01/2023 |

### Estimated Timeline

The development team decided to have week-long sprints due to the shorter timeframe of the project and the numerous parallel projects the developers are working on. It is hoped that the shorter turnaround time will increase responsiveness and keep the development team on schedule. The weekly sprint review and planning meetings will occur on the Thursday of each week. The developers will utilize Jira to keep track of the sprints and their associated tasking and backlog. The table below displays the proposed sprint schedule and high-level focus of each sprint.

|  |  |  |  |
| --- | --- | --- | --- |
| **Sprint** | **Sprint Start Date** | **Sprint End Date** | **Focus** |
| 1 | 02/02/23 | 02/08/23 | Deliverable 1 |
| 2 | 02/09/23 | 02/15/23 | Deliverable 1 |
| 3 | 02/16/23 | 02/22/23 | Deliverable 1 |
| 4 | 02/23/23 | 03/01/23 | Deliverable 1 |
| 5 | 03/02/23 | 03/08/23 | Deliverable 2 & 3 |
| 6 | 03/09/23 | 03/15/23 | Deliverable 2 & 3 |
| 7 | 03/16/23 | 03/22/23 | Deliverable 2 & 3 |
| 8 | 03/23/23 | 03/29/23 | Deliverable 3 |
| 9 | 03/30/23 | 04/5/23 | Deliverable 3 |
| 10 | 04/6/23 | 04/12/23 | Deliverable 3 |
| 11 | 04/13/23 | 04/19/23 | Deliverable 3 |
| 12 | 04/20/23 | 04/26/23 | Deliverable 3 |

## Cost Assessment

The estimates of the chart below were sourced from a service offering estimates on the resources necessary for developing an application with the primary requirements listed in the section above [5]. None of the developers for the project will be charging for their hours, but it seemed to be a prudent first step to get an outside estimate from an official service. The Comprehensive plan will have a much better estimate from firsthand development experience and a reassessment of requirements, but it is believed that the current estimate offers a unique perspective into how much effort should go into each of the phases and development cycles for a full product.

It is important to note that we will be rapidly developing a functional prototype with several specified features and may have to shorten certain phases depending on unforeseen issues and the learning curve experienced by the development team. Similarly, while a full development would require software licenses and subscriptions, the prototype will be developed with as many free and open-source components as possible to speed up development.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Services** | **Price** | **Hours (Min/Max)** | | **Price** |
| Front End Developer | 60 $ per Hour | 170/310 | | $10,200/$18,600 |
| Back End Developer | 65 $ per Hour | 320/480 | | $20,800/$31,200 |
| Project Manager | 75 $ per Hour | 80/100 | | $6000/$7500 |
| Designer | 60 $ per Hour | 90/160 | | $5400/$9600 |
| Scrum Master | 80 $ per Hour | 80/100 | | $6400/$8000 |
| Data Engineer | 70 $ per Hour | 320/480 | | $22,400/$33,600 |
| Hosting services | 45 $ per Month | N/A | | Dependent on Host Time |
| **Min Est. Man Hours** | | | **Max Est. Man Hours** | |
| 1060 | | | 1630 | |

## Risk Assessment

The following lists detail some of the biggest risks and assumptions about the project, covering an initial estimate of the weaknesses of the team and how able they are to achieve the stated goals. Many of the assumptions are made with the content of the limited available project guidance and will have to be challenged or supplemented with further Product Owner feedback in the Comprehensive Plan.

### Assumptions:

* Client has all the information about the project
* Multiple people will want to be able to adjust/view the stats of a project
* All people will have access to wifi and a device/software capable of viewing the website
* The application will not need to handle extreme numbers of users at any one time
* A locally hosted web application will suit the user’s needs
* The customer is alright with a prototype product

### Risks:

1. **Risk:** Developers choose a framework or language that not everyone is familiar with
   1. **Consequences:** slowing down development and degrading software quality with additional mistakes
   2. **Solutions:** 
      1. Choose the framework and language that the highest number of developers are familiar with, or the one that the most experienced developer is familiar with.
      2. Find resources to teach the developers how to use the framework and include them in documentation.
      3. Conduct much of the initial development together so that group members are confident
      4. Pair experienced developers with inexperienced developers.
      5. Can also designate less experienced developers to other roles such as task coordinator, diagramming and documentation.
2. **Risk:** There is no clearly defined Product Owner/Visionary for the product
   1. **Consequences:** End result lacks purpose or does not fully adhere to the requirements provided. The quality of the software is lowered? Really, it just wouldn’t be as good as a more purposefully designed narrower app
   2. **Solutions:**
      1. Assign one dev team member to act as the Product Owner/Visionary
      2. Create application vision together with input from all dev team members
      3. Create diagrams and mockups showcasing the application’s structure before choosing dev frameworks or doing other development work
3. **Risk:** Project has too much or too little planning before development
   1. **Consequences:**
      1. Too much planning: There won’t be enough time to complete development on the application and/or the development proceeds much too slowly
      2. Too little planning: Loss of tempo as problems arise, slowed development and lack of focus
   2. **Solution:** Declare the items needed to move from one phase to another, define the process needed to move forward from one iteration to the next.
4. **Risk:** Development team lacks enough time to work on the project due to other commitments
   1. **Consequences:**
      1. Project won’t progress quickly enough to finish in time
      2. Grade will suffer
   2. **Solutions:**
      1. Get dev teammates more coordinated with class schedules
      2. Factor extenuating circumstances into the project schedule as soon as possible
      3. Create the project one step at a time, creating base functionality and improving it as possible
5. **Risk:** Documentation and Testing are completely ignored
   1. **Consequences:** Less significant, but a lack of testing can result in issues with the code and degrade software quality. Lack of documentation is less serious, but can still cause issues if the development team forgets crucial information
   2. **Solutions:** 
      1. Researching how to test the framework we choose
      2. Requiring unit/integration tests for pull requests
      3. Writing the Jira tasks with enough information to act as documentation

# Summary

Rush Hour is a project management application dedicated to easing the management burden of software developers as they undergo the Planning, Organization, Monitoring, and Adjustment phases of software projects. The development team will be focused on adding the required functionalities to the application while also focusing on addressing an overlooked niche in the management software space. This document serves as the first step in the continual journey of progressing software management and planning, where each improvement is minute, but vital to our current lifestyle.

# References

1. Atlassian, “Get started with agile project management | Atlassian,” Atlassian, 2019. https://www.atlassian.com/agile/project-management
2. ‌React, “Getting Started – React,” Reactjs.org, 2019. https://reactjs.org/docs/getting-started.html
3. “Docs,” Appwrite. https://appwrite.io/docs (accessed Feb. 27, 2023).
4. ‌“MySQL :: MySQL Documentation,” Mysql.com, 2019. https://dev.mysql.com/doc/
5. “Crowdbotics | Build Apps Fast, No Coding Required,” www.crowdbotics.com. <https://www.crowdbotics.com/>
6. Amazon, “Amazon Web Services (AWS) - Cloud Computing Services,” Amazon Web Services, Inc., 2022. https://aws.amazon.com/
7. ‌Docker, “Enterprise Application Container Platform | Docker,” Docker, 2018. https://www.docker.com/

‌