**CS673 Software Engineering** 

**Team Rocket - Project RCM: Rental Car Management**

**Project Proposal and Planning**

| Team Member | Role(s) | Signature | Date |
| --- | --- | --- | --- |
| Alisa Belousova | Configuration Lead | *Alisa Belousova* | 9/9/2023 |
| Nick Cruz | Team Leader/ Requirements lead | *Nickolas Cruz* | 9/9/2023 |
| Devon Duddley | QA Lead | *Devon Dudley* | 9/9/2023 |
| Kris | Security Lead | *Cangqing (Kris) Wang* | 9/9/2023 |
| Chenghao Ye | Design and Implementation | *Chenghao Ye* | 9/10/2023 |

**Revision history**

| **Version** | **Author** | **Date** | **Change** |
| --- | --- | --- | --- |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

[Overview](#_g6igqliy7rm)

[Related Work](#_bf21eadgjj29)

[Proposed High level Requirements](#_rgyo4hi9stmq)

[Management Plan](#_ts358bsdtbcv)

[Objectives and Priorities](#_nxeeppkjxgn4)

[Risk Management (need to be updated constantly)](#_tk7yixobah8p)

[Timeline (need to be updated at the end of each iteration)](#_iksrndohvx29)

[Configuration Management Plan](#_j5uvivmxqcsp)

[Tools](#_dzly5b9kz982)

[Deployment Plan if applicable](#_sd8zu6r3jisd)

[Quality Assurance Plan](#_vra5ptwu59qx)

[Metrics](#_vwjduhc9wuah)

[Code Review Process](#_hx3eaiwb8v3m)

[Testing](#_l9xnpmd6hh0y)

[Defect Management](#_5amsh8h9f0c7)

[References](#_pd9euov6m4du)

[Glossary](#_ty3i2nqffhtc)

# Overview

(Please give an overview of your project. It should include the motivation, the purpose and the potential users of the proposed software system, the basic functionality of the proposed software system and the possible technology stack to be used. )

The purpose of this application is to help car rental companies manage their companies. This would primarily be done by helping them take care of their inventory (type of cars they own, specific details of the cars they own, etc) and their maintenance schedules.

# Related Work

(Please describe any similar software systems that you have found through the online research, and the differences between your software and those software systems.)

There are various business management tools, such as Proofhub, bit.ai, and timely, but our app is focused on helping car rental companies sort and organize their assets. As car rentals have generally need to have a significant amount of resources, it would be helpful to have an app that can stay up to date with those resources, as well as ensuring that things do not get lost due to an extensive catalog.

# Proposed High level Requirements

* 1. Functional Requirements  
     (For each functional requirement, please give a feature title and a brief description using the following format: As (a role), I want to (action), so that (value).)
     1. Essential Features (the core features that you definitely need to finish):

(For each essential features, please give a rough estimation in terms of person hours or an range of person hours)

As a management application, I want to ease the role of the user, so that they might have an easier time managing their resources. Estimated hours: 50 per person.

As an editor, I want to allow users to add vehicles, remove vehicles, and adjust vehicle information, so that the user can be fluid in keeping track of their resources. Estimated hours: 5 per person

As a database, I want to store data that can be found by a search engine, so that my data can be easily accessed. Estimated hours: 10 per person

As a search tool, I want to allow for quick look up of vehicles based on various criteria, so that the user can quickly find what they are looking for. Estimated hours: 5 per person

As a maintenance tracker, I want to help keep track of any maintenance, past or future, so that the user can ensure their vehicles are taken care of, but also check prior cases as needed. Estimated hours: 10 per person

Desirable Features (the nice features that you really want to have too):

As a calendar, I want to be able to hold dates for things scheduled, so that they may be visualized easily. Estimated hours: 10 per person

* + 1. Optional Features (additional cool features that you want to have if there is time):

As a reminder, I want to notify users of maintenance that is overdue or upcoming. Estimated hours: 15 per person

* 1. Nonfunctional Requirements
     1. Security requirements
        1. We would like to implement a login and password combination to ensure that the clients data is secured. We would like to implement a system where passwords would be encrypted in order to maintain integrity. Estimated hours: 15 per person

# Management Plan

## Objectives and Priorities

(Please describe your project objectives with highest priority first. Project Goals can include but not limited to complete all proposed (essential) features, deploy the software successfully, the software has no known bugs, maintain high quality, etc )

* + 1. Develop the necessary pages needed
       1. Home page, login page, signup page, maintenance page, etc.
    2. Ensure that each page can accomplish its main function
       1. Add page adds new vehicle with unique ID
       2. Edit page edits/removes
       3. Maintenance page displays a car’s maintenance history
       4. Maintenance editor allows editing of a car’s maintenance history
    3. Create the database and ensure that information is stored properly in the database and properly accessible by the application
       1. If information about car x is called then information about car y should not be populated.
    4. Ensure that the username and password system work properly
    5. Ensure that the site is easy to operate and has a clean simple design

## Risk Management (need to be updated constantly)

(Please write a summary paragraph about the main risks your group identified and how you plan to manage these risks. (<https://docs.google.com/spreadsheets/d/1zyuz-gFFX54Zf4VquNnzSNqwHtns-p_QF2r0VEanW_8/edit#gid=0> ) for detailed risk management.

**Risk Management Sheet Link:**

The main risks are people leaving the project, having too large of a scope, and having conflicts within the team . In terms of people leaving, we have had 2 people leave the project to date, and in those cases work that was supposed to be completed by them was undone and left the team in a scramble to sort through the work and complete it. In the second case, having too large of a scope could lead to the team struggling to complete the project due to being unable to complete the parts necessary to the project. The third large risk comes with working within a team. As part of any team, people have differing perspectives, views, and ideas. It will be necessary to have conversations and compromise when necessary to not hinder the development of application.

## Timeline

| Iteration | Functional Requirements(Essential/Disable/Option) | Tasks (Cross requirements tasks) | Estimated/real person hours |
| --- | --- | --- | --- |
| 1 | All main pages completed | Creating the pages along with connecting the pages to each other while making it readable and simple to use. | 20 hours or so per team member |
| 2 | Unit testing + Project review and improvement +Security | Create the database, tests, improve design, functionality, and security | 20 hours per per team member |
| 3 | Touch ups | small changes and improvements and feedback | 10 hours per team member |

# Configuration Management Plan

## Tools

In this project, a toolset is established to facilitate efficient development, version control, and continuous integration and continuous delivery (CI/CD).

Git, as a distributed version control system, empowers us to work on code collaboratively while maintaining a clear history of changes. It enables the team to seamlessly track modifications, synchronize work among team members, and resolve any code conflicts efficiently.

GitHub, chosen platform for hosting the project's repository, complements Git perfectly. It offers a centralized hub for the codebase, enhancing collaboration with features like pull requests, issue tracking, and code reviews. These features will streamline communication among team members and provide a structured approach to addressing code changes.

In addition to Git and GitHub, the team leverages popular Integrated Development Environment (IDE) tools such as Visual Studio Code (VSCode) and WebStorm. The choice depends on personal preference but both of these IDEs are renowned for their versatility, providing features like intelligent code completion, debugging capabilities, and extensive plugin ecosystems.

GitHub Actions is an integrated CI/CD solution within the GitHub ecosystem, enabling us to automate various stages of the development process, from building and testing to deployment. By doing so, the team ensures that the codebase remains reliable and deployable throughout the project's lifecycle, ultimately leading to an efficient and dependable software development process.

TailwindCSS to style among with motion js to create visuals

Using MERN - React, nodejs, express and mongo db for this project.

Jira to track project tasks and discord to communicate.

With this well-rounded toolset, the team is well-prepared to tackle the challenges of the project and deliver a high-quality product that meets project goals and exceeds expectations.

* 1. Code Commit Guideline and Git Branching Strategy  
     (Please briefly describe criteria for the code commitment and the branching strategy used, e.g. what are the branches to be used, how the pull request will be used etc. Here is an article to give you some basic knowledge about different git branching strategies: <https://www.flagship.io/git-branching-strategies/>

Typically we will clone the repo, then work on a branch specific to the change the programmer is working on. After completing the article that is being worked on, the programmer would check to see if there had been any changes to main. If there are changes they will pull from main, merge the changes into their branch (to ensure that there are no complications and to fix it if there are complications) then push their changes through before merging to main. This will ensure that everyone can work on their individual assignments without undoing, or creating conflicts. We may also implement an approval system so that other team members have to approve of changes as well, increasing the likelihood of good changes and allowing for other’s perspectives to be seen on any given addition to the project.

## Deployment Plan if applicable

(If you plan to deploy your application (e.g. your web application), briefly describe how you plan to deploy your application).

We will deploy our application using a tool such as Vercel or Heroku that we will determine at a later stage after the team decides what platform they prefer.

# Quality Assurance Plan

## Metrics

(Describe the metrics to be used in the project to measure the quality of your software. Each metric should be measurable and quantifiable. Examples of metrics include product complexity (LOC, # of files, # of classes, # methods, cyclomatic complexity, etc.) , defect rate (# of defect per KLOC), # of test cases, test case pass rate, cost (# of person hours used), # of user stories completed, etc. **The result of these metrics should be reported in the progress report/ iteration summary sheet.**)

| Metric Name | Description |
| --- | --- |
| Cyclomatic Complexity | Calculate the cyclomatic complexity of critical code sections to identify areas prone to defects |
| Unknown | We currently do not know or understand the complexity as we have yet to begin the project. We will update this as we progress. |
|  |  |
|  |  |
|  |  |

* 1. Coding Standard

**Formatting and Indentation:** Coding standards often specify how code should be formatted, including rules for indentation, spacing, and line length. Consistent formatting enhances code readability.

We will use standard practices, such as proper indentation (eg. indenting in nested functions/loops) and spacing(ensuring spaces after punctuation. in order to make code readable.

**Naming Conventions**: Coding standards define rules for naming variables, functions, classes, and other program elements. These rules typically address issues like capitalization (e.g., camelCase, PascalCase, snake\_case), abbreviation, and word choice.

We have decided to use camelCase to be consistent and make for easy readability.

**Comments and Documentation:** Guidelines for adding comments to code are essential. Comments should explain the purpose of code blocks, document functions and classes, and provide information about parameters and return values. Proper documentation is crucial for code maintainability.

We will be commenting on any functions we add to the program. This will ensure that each function will be mapped out, easy to understand by others, and will not cause confusion.

**Code Structure:** Standards may specify the organization of code within files and modules. This includes recommendations on how to structure functions, classes, and modules, as well as guidelines for arranging import statements.

Code should be organized with proper indentation, which will help us understand what functions are nested within other functions.

**Error Handling:** Coding standards often cover error handling practices, including how to handle exceptions, when and how to use try-catch blocks, and how to log errors.

We will try to use try-catch statements, as well as work and communicate with teammates for possible solutions to potential errors.

**Code Readability:** The goal of coding standards is to make code more readable and understandable. This includes avoiding overly complex code, using meaningful variable names, and breaking long functions or methods into smaller, more manageable pieces.

We will attempt to create functions that have simple variable names and to not make functions too complicated. In addition, we will try to avoid code redundancy by creating functions that can be called if needed multiple times.

**Consistency:** Consistency is a key principle of coding standards. Developers should follow the same conventions throughout a codebase to ensure that code looks and behaves predictably.

**Version Control and Collaboration:** Coding standards may include guidelines for using version control systems like Git and collaborating with other developers. This includes rules for commit messages, branching strategies, and code review processes.

**Performance and Efficiency:** In some cases, coding standards may include recommendations for writing efficient code, such as avoiding unnecessary loops or optimizing data structures.

## Code Review Process

(Everyone should review all documents to be submitted. Here you will mainly describe how the code review will be done. Who will review the code, e.g. design or implementation leader will review all code or team members review each other’s code. Do you use pull requests for the code review? Is there a checklist to help review? What feedback should the reviewer provide?)

We will collectively review each other's code so that we might have a broader perspective that one person might miss. We can share screens, screenshots, or pull in code so that we may do this while talking to the person we are reviewing for, or afterwards if the schedules do not align.

## Testing

(Both manual testing and automated testing should be considered. Both unit testing and integration testing should be considered. Briefly describe the testing tools/framework to be used, the personnel involved (e.g. the QA leader will focus on the integration testing and each developer will unit test their own code), when and what types of testing will be performed, the testing objectives, etc)

We can manually test our codes and we may also use outside tools such as codePen and JSfiddle.

## Defect Management

(Describe the tool to be used to manage the defect (e.g github issues). The types of defects to look at. The actions or personnel for defect management. )

We do not currently have a defect management tool, but will use both IDE bug and defect managers (such as those in VScode and github issues) to assist us as well as function as a team to work together to fix defects and bugs.

# References

(For more details, please refer to the encounter example in the book or the software version of the documents posted on blackboard. )

# Glossary

(Any acronym used in the document should be explained here)

RCM: Rental Car Management

Repo: Repository