Virtual Project Management Web App
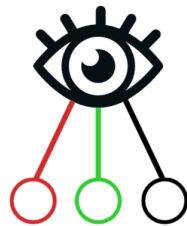
Prototype Project Report

Washington State University

Engineering and Technology Management Department

Dr. James R. Holt

The Visualizers

Jing Ren Tay, Pragun Kalra, Ava Stromme

September 28, 20222

TABLE OF CONTENTS

I. Introduction

This document provides details regarding the Virtual Project Management Web Application developed by The Vizualizers. It provides an overview of the project, including features and goals. It also describes the rationale behind Visual Project Management. The document also provides an overview of the requirements and use cases for the web application.

I.1.   Project Introduction

The Visualizers have been asked to create a project management application for our client, Dr. Holt. This application can be used as a tool by companies, managers, and individuals for tracking their progress on a particular project. The tool will help solve the problem of inefficiency in project management. The creation of such an application requires the implementation of a web application with a front-end web interface and a database in the backend which will be used to determine the progress of the user in the project. The core of the project is the re-creation of the graphical interface which provides a visual representation of the task and project status. The end goal of this project is to provide a new product with improvements to the current solution.

I.2.   Background and Related Work

Virtual Project Management is an application that Dr. Holt currently owns [2]. The application serves the purpose of facilitating efficient time management during a project. It has three main features which aid it in fulfilling its purpose. It displays the status of a task based on the input by the user, tells which tasks need to be prioritized based on their life cycle, and aids in determining the right hierarchy of time allocation to follow based on the status of the task. The essence of this application is the graphical user interface which is the visual representation of the project life cycle.

Previously, Dr. Holt built a basic version of the application on a spreadsheet program for demonstrating his idea and once had a mobile application. Over the years, the software transitioned into a web application. The current version of the application is built using obsolete technology, and key security changes are being implemented under the supervision of Scott Mattes, an associate of Dr. Holt. The changes include transitioning from HTTP to the more secure HTTPS, improvements and changes to the graphical interface, and bug fixes. The client suggests that the team create the product in a way that it can be further used for the creation of a mobile application.

Throughout the project, the team is expected to acquire new skill sets such as gaining an understanding of a query language for the management of databases, learning tools similar to GraphJs for developing a graphical user interface, and improving knowledge of UX for a better user interface and user experience.

I.3.   Project Overview

Much of the work that people do today is centered around projects [1]. Companies hire project managers to plan and manage projects, and they hire workers to carry out the actual project work. However, sometimes predictions for project timelines are incorrect, resulting in late projects or projects completed too early [1]. According to Dr. Holt, management may have difficulty determining when to start new projects and when to wait for current projects to progress further [1]. The purpose of VPM is to make the process of planning and tracking the

progress of projects easier. One aim of VPM is to provide project managers with a visual representation of whether a project is on track, falling behind, or whether it can be paused to allow other projects to get back on track. VPM strives to help organizations, or anyone who works on projects, become more efficient.

In VPM, projects are represented as a plot on a graph, with the x-axis as the percentage of the project that is completed and the y-axis as the percentage of the buffer consumed. This is known as the Fever Chart. Each project is allocated a 50% buffer to account for errors in the predicted time for the project. A diagonal line is drawn from the lower left corner to the upper right corner; the space below this line is colored green while the space above this line is colored red. Projects in the green zone are on track. Projects in the red zone have used up too much of their buffer for the amount of the project that is completed, and they may need some help in getting back on track. Projects "in the black," or plotted directly above the red zone on the graph, are projects that will be late.

When the project is in the green zone, the project is on track. If it lands on the diagonal line, it is progressing as expected. Depending on where the project falls in the green zone, resources for the green project may be spared to help projects in the red zone get back on track. When a project is in the red zone, it has consumed too much of its buffer too soon and it is at risk of being late. Depending on where the project falls in the red zone, experts or resources from other projects may be called in to help the project get back on track. Projects in the black zone are projects that will be late. They have used all of their allocated buffers too soon in the project timeline. For these projects, the customer may need to be notified of the problem or the project may need to be canceled.

The desired outcome of the project is a functioning web application that serves as a tool for managing projects according to VPM guidelines. The web app will allow users to create projects and specify the duration, tasks, and other information associated with the project. Projects in the planning stage can be duplicated and edited. When a user starts a project, they will be able to check off tasks that they complete and see a visualization of the project's progress on a colored graph.

Some of the other features of this web application include a graph known as the Multi-Project Fever Chart that shows a plot of all projects currently in progress. The app will feature a prioritized list of the tasks that are not yet completed from all projects. These tasks will be prioritized based on which zone their project is currently in. For example, tasks from a project in the black zone will be prioritized over those from a project in the red zone. Projects can exist in the "In Planning" state, the "In Progress" state, or the "Archive" state.

While these are the most important features to implement in this web application, there are many other features to mention. The web app will need to support the creation of user accounts protected by passwords as well as the ability for a user to become part of an organization. Users should be warned when different projects enter, for example, the red zone, and require attention. While this describes the basic functionality of the app, many other features will be described in detail later on. A web application created under these guidelines and with these features should help make project management easier and more efficient for its users.

I.4.   Client and Stakeholder Identification and Preferences

Our client is Washington State University's Engineering and Technology Management (ETM) program, and Dr. James Holt is our primary contact for the project. Scott Mattes is also involved in the project and is a programmer brought in by Dr. Holt to help with the development of the

current version of the VPM website. Our instructor, Ananth Jillepalli, will be our mentor throughout this project. The VPM website is open and free for anyone to use.

For this project, Dr. Holt requests the development of a web application for VPM. This web app will help users visualize whether a project is on track to finish on time before the due date. As described above, we will be implementing the main features of VPM, including the presentation of a project in progress through a graph showing the Project Percent Complete versus Percent Project Buffer Consumed. This graph will be displayed with information on whether a project is on time (green) or behind the initially assigned due date (red). Furthermore, there is a Multi-Project Fever Chart to keep track of many projects' timelines in one graph.

Another important use of VPM is to prioritize the list of uncompleted tasks. This ensures the uncompleted tasks in the black zone (where the buffer consumed 100%) will always appear above tasks in the red and green zones. For the programming language, the client does not have a preference and requests that the most suitable tools be used for the project. The client also suggests the use of tools that make for an easier transition to a possible mobile app version in the future.

VPM aims to aid people in visualizing project progress and ensure that they are on track to finish them on time. The current product is not ready to be fully used on the web or mobile devices as there are bugs and some features are yet to be implemented. As a result, the product will need to be fixed and packed with more features to give users a complete and satisfying experience.

II. Team Member Bios

Ava Stromme is a computer science student interested in software engineering, full-stack development, and web development. She has had internship experience where she has worked with Go applications, Node.js web apps, and data monitoring boards. She is also skilled in C/C++, C#, Python, Git, and SQL. Ava is the team lead for this project and has worked on the UI of the application as well as the API, backend, and the Redux functionality for managing the central state of the application.

Pragun Kalra is a computer science student interested in project management and full-stack development. He has experience in software testing and developing front end for web applications, web crawlers. He is skilled in Python, JavaScript, C/C++, Git. Pragun is responsible for the development of the front end and has worked on the form page, login page and sub components within the UI. He is responsible for the software testing and UI development of the application.

Jing Ren Tay is an undergraduate student that is majoring in Computer Science on software track. He had worked with the CUB as an setup technician to set up events for VIPs, various organizations, and for the public. He also is a president of the badminton club and the co-programming chair of the International Student Council. He is skilled in C/C++, Python, C#, Haskell and SQL. He is great when it comes to HTML, CSS and using Flask for web development. Tay is one of the development members on the frontend, forms pages such as view project page, and edits projects involving uploading the information to the backend side. He also worked on some parts in the API and redux in order to create some features of the application.

III. System Requirements Specification

III.1. Use Cases

Virtual Project Management Web App
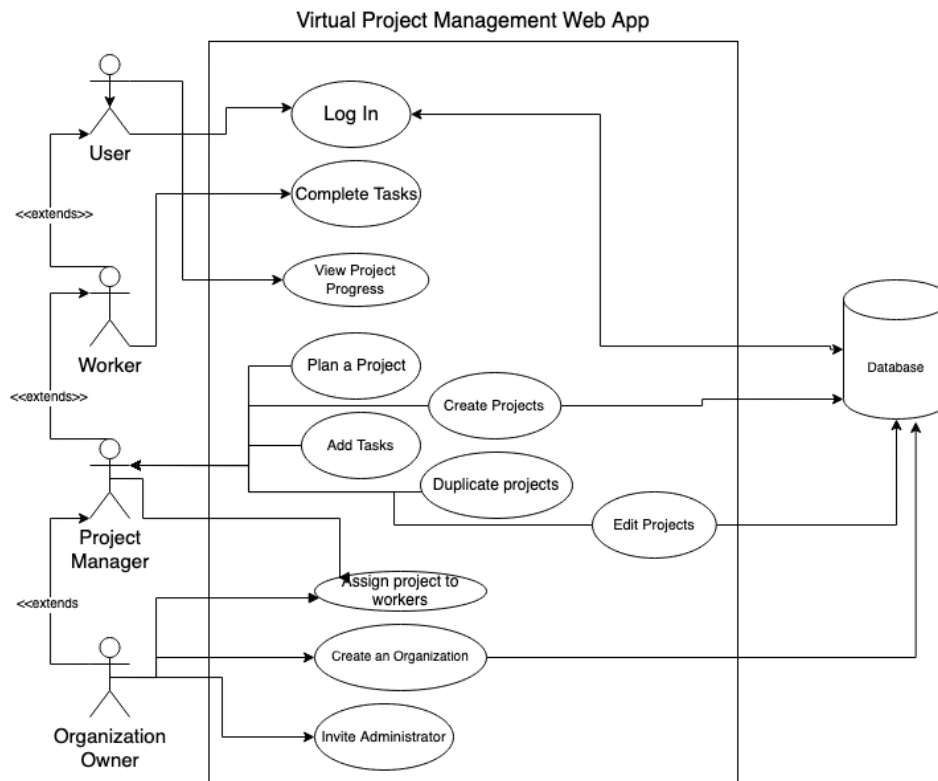
Story: As a project manager, I want to manage project(s) so that I can keep track of the progress and supervise the workers.
Criteria:
- Users should be able to create and login into their accounts.
- Able to create project(s).
- Can duplicate project(s) once a project is created.
- Assign to workers in projects.
- Keep track of the progress of the tasks created and assigned to workers.
- Able to access all features of workers.
- Can extend or shorten the due date of the project.

Story: As a worker, I want to see my tasks that are assigned to me so that I can have an understanding of my part in project(s).
Criteria:
- Users should be able to create and login into their accounts.
- Should be able to see tasks they are assigned to in project(s).
- They can mark their tasks completed or not completed.
- Can access multiple projects if they are under various projects.

Story: As the owner of the organization, I would like to see the overall situation on all projects and the information about them so that I can know what are the progresses in the company.
Criteria:

- Users should be able to create and login into their accounts.
- They can see all of the progress of the subprojects for the overall project.
- Able to access all of the information on a project.
- They can cancel or delay the subprojects of a project.
- They can assign project managers to subprojects.
- They can access all the features of both project managers and workers.
- They should be able to invite administrators with privileges as the owner.

III.2. Functional Requirements

III.2.1. Project Tracking

Fever Chart: The system needs to plot the Percent Complete versus the Percent Buffer Consumed on a line graph for a project that is in progress. The system also plots the "last known task completion" for each current project on a Multi-Project Fever Chart with labels.
Source: Dr. Holt originated this requirement and the requirement is necessary for users to view the progress of their projects.
Priority: Priority Level 0: Essential and required functionality

Prioritization of Tasks: The system must prioritize the list of tasks associated with each project. The tasks are prioritized first by the project status (black, red, or green), then by the expected task start date. Projects in the black zone should be prioritized over projects in the red and green zones.
Source: Dr. Holt originated this requirement and the requirement is necessary for users to view which tasks are currently most important in each of their projects.
Priority: Priority Level 0: Essential and required functionality

View Project Status: An authorized user must be able to view new projects ("in planning" stage), started projects ("in progress" stage), and archived projects ("archived" stage). The user must be also able to access projects in each stage and move projects between these stages.
Source: Dr. Holt originated this requirement and the requirement is necessary for users to view the status of their projects.
Priority: Priority Level 0: Essential and required functionality

III.2.2. Project Planning

Creating and Editing Projects: The system needs to allow Project Managers (or authorized users) to create, edit, and duplicate projects. A unique identifier must be created for each project. Projects should be created with at least one task.
Source: Dr. Holt originated this requirement and the requirement is necessary for the app to function and for the user to track their projects.
Priority: Priority Level 0: Essential and required functionality

Adding Tasks: The system needs to allow an authorized user to add tasks to a project. A unique identifier must be created for each task, and an estimated, aggressive time for completion should be assigned to the task. Tasks should also have a name and a description. Tasks may be assigned to specific workers.
Source: Dr. Holt originated this requirement and the requirement is necessary for calculating the progress of the project and plotting it on the Fever Chart.
Priority: Priority Level 0: Essential and required functionality

III.2.3. Authentication

User Accounts: The system must allow users to create an account with a login email and a password. The user will have the option to create and be the owner of an organization.
Source: Dr. Holt originated this requirement and the requirement is necessary for a user to view projects in their organization or projects that they own.
Priority: Priority Level 0: Essential and required functionality

Organizations: The system must be able to keep track of organizations created by users. An organization will own projects and share projects to members of the organization. Organizations will also have data items associated with it, including a name, address, phone number, etc.
Source: Dr. Holt originated this requirement and the requirement is necessary for projects to be shared across multiple users' accounts.
Priority: Priority Level 1: Desirable functionality

User Roles: The system must allow users to have different roles in an organization. Every user in an organization is a Worker. The Owner is the user that created the organization. Owners can assign Administrators, Project Managers, Resource Managers, or other Workers, all with varying levels of access to all the features of the app.
Source: Dr. Holt originated this requirement and the requirement is necessary for controlling user access to certain features and resources.
Priority: Priority Level 1: Desirable functionality

III.2.4. Project Data Storage

Current Projects: The system must be able to show the user the projects that are currently in planning or in progress and their related tasks. This data must be stored persistently.
Source: Dr. Holt originated this requirement and the requirement is necessary for project data to be saved and accessed across different user accounts or sessions.
Priority: Priority Level 0: Essential and required functionality

Archived Projects: The user must be able to archive projects and view projects that they have archived. These projects must be stored persistently and retain all necessary information concerning the project, including the tasks within the project, the time it took for the project tasks to be completed, and the overall Fever Chart for the individual project.
Source: Dr. Holt originated this requirement and the requirement is necessary for users to retain and view their past project data across different sessions.
Priority: Priority Level 1: Desirable functionality

III.2.5. Project Optimization

Predecessor Tasks: Allow tasks within a project to be completed in any order. Allow some tasks to be specified as a "Firm Predecessor" to other tasks.
Source: Dr. Holt originated this requirement and the requirement is necessary for tasks to be completed in the correct order or prioritized correctly.
Priority: Priority Level 2: Extra Features or Stretch Goals

Sub-Projects: Allow a more complicated project to have sub-projects that show parallel paths of tasks that can be done at the same time as the project.

Source: Dr. Holt originated this requirement and the requirement is necessary for tasks to be prioritized correctly.
Priority: <u>Priority Level 2:</u> Extra Features or Stretch Goals

Warnings: The system will alert a user when a project is entering the black or red zones.
Source: Dr. Holt originated this requirement and the requirement is necessary for teams to track their progress and ensure they are managing their time well.
Priority: <u>Priority Level 2:</u> Extra Features or Stretch Goals

III.3. Non-Functional Requirements

Performance and scalability: The software should be able to function properly even in the case of a large input and multiple sub-tasks.

Reusability: It needs to be designed in such a way so that in the future it can be used to create a mobile application and be able to keep up with advancements in technology.

Security: The security of the servers needs to be increased which will be performed by upgrading the existing module from HTTP to more secure HTTPS.

Portability: The software needs to be designed in such a way that it is able to work across various operating systems in the future.

Maintainability and Manageability: The software needs to be designed using object-oriented principles and using appropriate naming conventions such that engineers can understand it and it is easy to maintain the software in the future.

Easy to use: The software needs to be designed in a way that it can be used by anyone who does not have a specialization in the field.

Reliability: The software needs to be highly reliable as the nature of its function can be very sensitive to organizations and users for appropriate planning and managing deadlines.

III.4. System Evolution

Critical chain project management is the core on which our software is based. The first task our team is focused on is the development of the VPM graph which gives a visual representation of the project status. The graph needs to be developed in a way that multiple tasks and higher workloads are computed with adequate speed, performance, and scalability. The current VPM application is based on HTTP protocol and requires a transition to more secure HTTPS, but in future even more secure protocols might be available and the project will therefore be created in a way that most of the components of the code are reusable. In addition the product needs to be created in a way that it can have utility ranging from a web-app to a multiple OS application. Furthermore, with advancements in technology and methods the VPM model we are creating might need future updates, therefore it needs to be created with the flexibility of adding new attributes so that it is prone to future updates.

IV. Solution Approach

IV.1. System Overview

The Virtual Project Management application is composed of web-based interfaces which respond to requests from the user and display the project details as part of its functionality. The interfaces are a result of integration of the front end, composed of the user interface framework, and the backend, composed of the database framework. The system requires multi-level integration of these components for its functionality.

IV.2. Architecture Design

IV.2.1. Overview

The Visualizers have composed the design for the development of the Virtual Project Management application by taking into account our client Dr. Holt's requirements. The web application involves dynamic communication between the application and the client, therefore its composition is based on the client-server architectural pattern. This type of architecture ensures correct input-process-output as the client can request service and the server can respond to client requests by providing the appropriate service. The diagram given below gives the blueprint of the components involved in our architectural design. The frontend communicates with the API, which then communicates with the database to extract useful data. The extracted data then undergoes processing done by business logic to display the required instances. The API and routing here functions as a navigator for the various components. Branching off below the API and routing is the authenticator which in turn connects externally. The authenticator, as its name implies, is responsible for all user permissions and authentication tasks. Another essential component is temporary data storage, all data pulled from our external database will be stored here temporarily for use in the application. This component is developed using instances of MERN stack and utilizes Node.js and Express.js to express modular and maintainable functions. The external component of the architecture is the database framework which is developed using MongoDB Cloud Atlas and is accessed from the server side of the application. The subsystem decomposition provides extensive detail on each of these components in the diagram below.
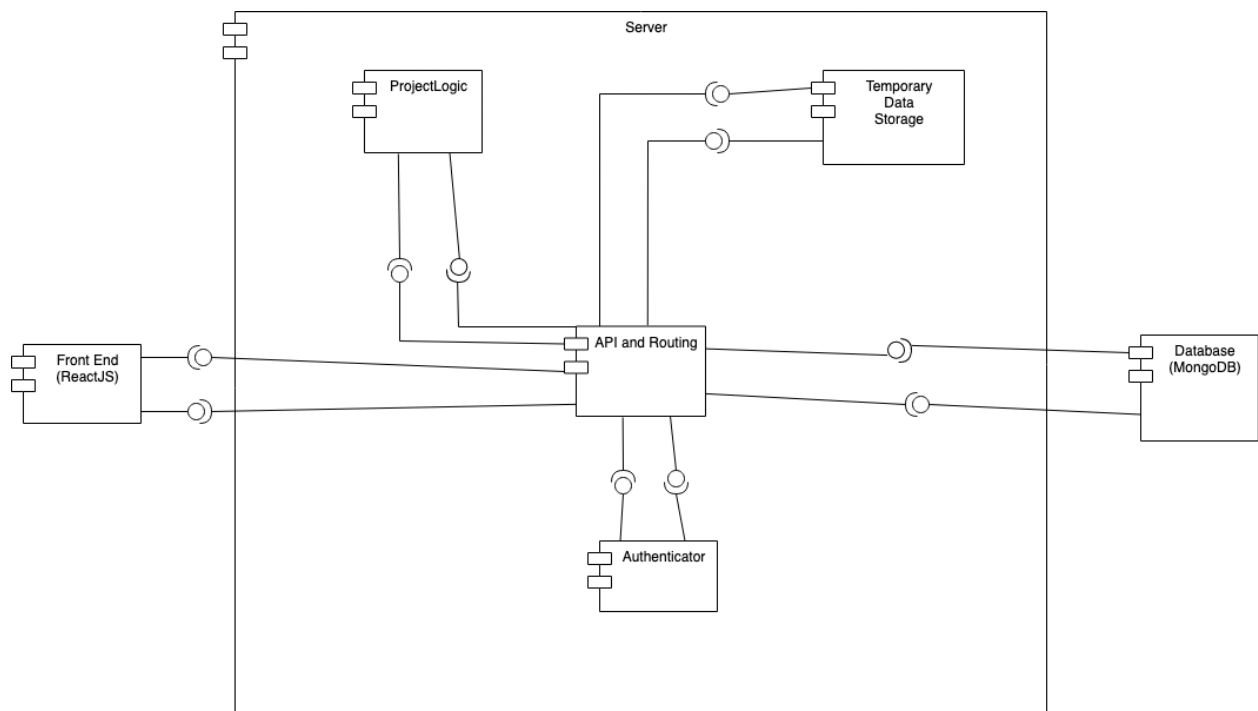
IV.2.2. Subsystem Decomposition

IV.2.2.1.　　API and Routing

a) Description

The API and Routing subsystem acts as the connection between the server and all the external components in the architecture, including the frontend and the database. It handles API and routing calls in the framework. It is an essential component which acts as a navigator between the various components.

b) Concepts and Algorithms Generated

Routing is how Web API matches a URI to an action. Attributes would be used to define routes, these would give us more control over the URIs in the interfaces. There would be multiple attributes defined for both the client and server frameworks as well as attributes to link the two frameworks.

c) Interface Description

Services Provided

1. Service Name: ReadProjectDetails
    Service Provided To: Temporary Data Storage
    Description: This service will take the project details entered by the user and then pass it to the temporary storage. The output will either be successful and a new project will be created which would then be stored in the database or it will fail in case the temporary storage will find an already existing project in place, then it will send an error message to the API controller which will be passed further back to the frontend.

2. Service Name: PostProgressDetails
    Service Provided From: Database (MongoDB)
    Description: This service will get the project details requested by the user and then pass it to the temporary storage. The output will either be successful and a project and its graph will be displayed or it will fail in case the database will not find an already existing project with the same name in place, then it will send an error message to the API controller which will be passed further back to the frontend.

3. Service Name: UserData Retriever
    Service Provided From: Database (MongoDB)
    Description: This is a trigger function where if there are any login page that is being used and a user is trying to login into their account, it will be triggered to retrieve data from database about all user's username and their password in order to check and match with the username and password that the user has just entered from the login page. The output of this service is to obtain data so that the user can be verified to login.

IV.2.2.2.　　ProjectLogic

a) Description

The Project Logic subsystem is responsible for any calculations that will be used during creating or editing or even deleting projects. It will be triggered when a user does specific

actions that will need to be calculated or any actions that might affect other data in the database. For example, if a new task is assigned into an existing project and that task will change the due date of project, then the Project Logic will be triggered for not only calculating the time estimated for that individual tasks, but also for the overall project and probably a bigger project that includes this project since the due dates will need to be change or updated as well.

b) Concept and Algorithms Generated

The Project Logic subsystem will include all of the calculations that are required in the VPM application. As one of the key features in VPM, the aggressive estimated time formula will be included in the Project Logic subsystem. Once the new value is evaluated and approved, then it will be sent back to the APIController for updates to the database or temporary data storage.

c) Interface Description

Services Provided

1. Service Name: TimeCalculator
   Service Provided To: Temporary Data Storage
   Description: This service includes all of the calculations that are required in the VPM application. The output of the service will provide a result to various calculations needed to display information about the projects.

2. Service Name: CreateProject
   Service Provided To: Temporary Data Storage and API
   Description: This service will enable users to create a brand new project that includes descriptions and information about the project. Once the project is created, it will send the data into the temporary data storage and enable the user to see the progress of the new project through API.

3. Service Name: DuplicateProject
   Service Provided To: Temporary Data Storage and API
   Description: The service duplicates projects that exist and copies all of the information about the duplicated project. The duplicated project will be shown through API and be able to be seen by the user.

4. Service Name: EditProject
   Service Provided To: Temporary Data Storage and API
   Description: The service takes an existing project and allows the project to be edited. This includes adding tasks, changing due dates, and other features. Once the editing is done, it will be committed to the temporary data storage and to the API.

Services Required

1. Service Name: StoreTemporaryData
   Service Provided From: Database (MongoDB)

IV.2.2.3.   Authenticator

a) Description

Authenticator is a subsystem that verifies every time a user login to an account or creates an account. It will be triggered when a user is logging in into their account by authenticating and verifying username and password from the database. It is connected to the APIController so that when when the API want is doing GET request to the server when logging in, it will send request to the database first to retrieve user information and then send a request to the authenticator to confirm that the login credentials match with the retrieved information from the database. Once it is successful, then the authenticator will reply back to the request and approve the login. It is important to have an authenticator for both user security and database encryption to ensure that the system is safe to be accessed.

b) Concepts and Algorithms Generated

This subsystem contains one class which is the authenticate class. It will have two main functions; one of them is getting the data from the database while another one will take the user input for login and try to match with the user info in the database. If the process passes, then the user will be taken to a new page through routes. If it is not, then they will be asked to try again or perhaps suggest the creation of a new account.

c) Interface Description

Services Provided

1. Service Name: UserLogin
   Service Provided To: API and Routing
   Description: This service is used for verification of a user when they are logging in into the VPM app. It compares the input username and password with the database in order to get the user logged in.

2. Service Name: UserAuthenticator
   Service Provided To: API and Routing
   Description: This is a service that will check whether a user is authorized to perform some tasks. It will send a warning or error if the user is not authorized to carry out the action.

Services Required

1. Service Name: ReceiveData
   Service Provided From: Database (MongoDB)

IV.2.2.4.   Temporary Data Storage

a) Description

Temporary Data Storage acts as a temporary place to store the data that will be used or retrieved by the Project Logic. This is because we do not want to store a large amount of data locally or it will overwhelm the system. Instead, we store our data in the external database and only query for the data we need for a specific action. This data will be retrieved through an API call that connects our application with the database. We will save all temporary data within our application for rendering the UI and making calculations. Then the database will be updated from this temporary data.

b) Concepts and Algorithms Generated

The temporary data storage will be doing queries from the database based on the user actions and retrieving any related data to that action. For example, if a user wants to edit a project, all related information to that project will be queried from the database and stored into the temporary data storage. Then, if the user adds or removes any tasks, it will be saved into the temporary data storage first and saved once all changes are made.

c) Interface Description:

Services Provided

1. <u>Service Name:</u> StoreTemporaryData
   <u>Service Provided to:</u> ProjectLogic
   <u>Description:</u> Retrieve data that was requested from GET request from a user or retrieved from the UI, and store it temporarily in the storage in order for ProjectLogic to compute graphs, perform calculations, and save project data.

IV.3. Data Design

A database stored in a MongoDB Cloud Atlas cluster will be our means for external data storage. It is a NoSQL database that stores data in a JSON-like format. The server side of the application is connected to the database, which allows the application to retrieve the necessary data when needed. For example, the application will need to store a user's account data and organizations. The database will store this data such that a user's email, password, and other account data is stored together. A list of organizations and the members and roles of that organization also need to be stored. This will allow for user accounts and roles to be correctly tracked and authenticated.

For the projects, project data such as tasks and time requirements will need to be stored. All data items relating to the plans for projects "in planning" and the details for projects currently "in progress" will be stored for retrieval by the application. The database will also store data related to projects that are archived, including the time it took to complete each task, the expected timeline for the project, and which zone (green, red, black, etc.) in which the project was completed.

As for temporary data storage, our application will store data temporarily in the form of variables in our code that will be used by the application in the short-term. For example, when the application needs to display the details of a certain project that is currently in progress, it will need to send a GET HTTP request to retrieve the necessary information from our database. This data will be stored as a temporary variable in our code so that it can be used to render the information in the UI. Similarly, information from the user input will be stored temporarily before it is stored in the long-term database with a POST HTTP request. This data will not be saved persistently and will likely be replaced by data from the next user interaction, therefore, the application requires a means of persistent data storage like a database in a cluster in MongoDB Cloud Atlas.

IV.4. User Interface Design

A partial UI for the application is detailed in Appendix A of this document. The design was based largely on the previous UI design in Dr. Holt's current version of the VPM web app [2]. This is because the functionality is the same and many of the web pages that are required are the same. There is the addition of the sidebar for various navigation tabs. Some components have

also been rearranged for the purpose of adding a sidebar and for ensuring the UI maintains a logical flow the user can navigate through easily.

The application will require a login page as shown in Page 1 of Appendix A. The title of the application, "Virtual Project Management" will be featured prominently on the screen. The user will be prompted to enter their email and password and have the option to log in. Buttons below that will allow the user to sign up for an account or recover their password if they have forgotten it. This page satisfies the use case where any user, whether they are an owner, a project manager, or a worker, will be able to log into their account. Though not pictured, the button for signing up should lead to another page where a user can enter their details and make an account, as well as have the option to create an organization. An extra feature would be to allow users to recover their password if it is lost.

The UI will have a tab that contains all pages related to projects. Clicking that tab on the sidebar will display a page with two tabs: "In Progress" and "In Planning." The "In Planning" section is shown in Page 3 of Appendix A. This page will have a list of projects that are in the planning stage and have not been started yet. There is an option to delete the project or edit the project by clicking on it. There is also a button to create a new project. The use cases that this page corresponds to are the use cases that allow the user to create new projects, edit and specify the details of the project, add tasks, duplicate projects, and start the projects. An authorized user, such as a project manager, will have access to these features.

The "In Progress" section is shown in Page 2 of Appendix A. This section will show the Multi-Fever chart that plots the last known completion point for each current project listed below. Hovering over a point on the graph will show the name of the project and the percentage of the buffer it has consumed. Clicking on one of the projects in the list will display the Fever Chart for that specific project, as shown in Page 4 of Appendix A. It will show the plots for completed tasks. Below that, a list of incomplete and complete tasks. The details of these tasks can be edited by clicking on them. The details of this project can be viewed and edited within the "Details" button. The use cases that these pages correspond to are the use cases that pertain to tracking the progress or status of a project, editing a project, viewing the prioritized tasks for a project, and viewing the details of a project.

In the sidebar of Pages 2-4 in Appendix A, there is a place to display the current user's name. A tab in the sidebar or the three dots in the top right corner will allow the user to log out of their account. There is the "Projects" tab in the sidebar that contains all content related to the projects, as described above. Below that, there is an "Archive" tab that will display the projects that the user has archived. This corresponds to the use case that allows users to archive their projects. The last tab is the "Settings" tab, which will allow users to configure their setting and possibly edit their personal information.

Other features that may not be shown in the images and are still in planning include the "Home" button. This page could possibly be a dashboard or could show a list of prioritized uncompleted tasks for the current user. The application will also need to include spaces for displaying all information related to organizations, such as which projects belong to organizations, which role a user holds for that project, which organizations a user is part of, etc. This would require additional pages to ensure that users only are able to access those features that are available to their role in the project. Sub-projects and warnings, as well as project freezing, are also extra features that may need to be included in the UI at a later date.

V. Testing and Acceptance

V.1. Overview

V.1.1. Testing Summary

In this project, we will use two platforms to help us test both the frontend and backend side of the application. First and foremost, we will be using Jest, a JavaScript testing framework to mainly test the ReactJS that is written within the project. Jest framework will cover all the client testing that is required for the user interfaces. Meanwhile, for Mocha, it will be used to test the backend of the application, which is the server that is connected with MongoDB with the user databases and other data that will be stored within it. This will ensure that any data that is either retrieved from the database or sent into the database matches correctly with the input or output.

V.1.2. Test Objectives and Schedule

In this testing process, there will be three objectives that we are looking to achieve: Creating a separate repository that will solely be used for testing our VPM application, a full documentation that will record any testing during the development of the app, and also all the resources that are used in testing our VPM app, such as Mocha and Jest. Our approach to the testing plan will be following the Agile development style together with Kanban to produce an efficient and effective strategy for our testing. For instance, when we finish a small feature that is a part of a milestone in a project, that particular feature will be tested straightaway. This is to prevent clusters of bugs or errors that might occur during a huge feature/big milestone testing. All testing should be commented to make sure that all possible errors are actually tested and run successfully. For milestones, we will be focusing on creating the features or writing code that is required for the app. Then once it is finished, we will begin the testing from unit tests, integration tests, and system tests at the end.

V.1.3. Scope

The main purpose of this document is to show our approach in testing the VPM application. It tells the frameworks that we will be using to test both frontend and backend of the application, as well as showing our strategy on testing different levels of VPM. It also gives a brief overview of the whole testing process from start to finish and also explains each stage of tests for the VPM.

V.2. Testing Strategy

For this web application, the main functionality that needs to be tested is the backend functionality. This includes testing whether data can be posted to and retrieved from the database and ensuring this data is stored correctly. Additionally, the internal logic of the program must be tested to catch errors, exceptions, or incorrect functionality. For example, we need to test the logic that makes time and date calculations so that correct work hours are entered into the database. Finally, some parts of the frontend need to be tested to ensure input validation is working correctly and pages are rendering properly. Overall, tests should be written with a focus ensuring the functionality works as expected and catching any bugs or edge case errors.

The parts described above, the backend (internal logic and data storage) and the frontend can be treated as separate components that may need different types of testing. We will follow a testing process to evaluate these different components and determine how specific requirements and features should be tested. The flowchart for the testing process is as follows:

1. Determine the requirements to test based on the requirements report and the list of features in the VPM feature document provided by the client. All important features and details outlined in the feature document and the requirements section of this report should be listed.
2. Determine which component this requirement or feature is part of as well as the best testing framework to use for requirements or features within this component.
3. Set up the testing framework and write the tests. Ensure every important main feature has tests associated with it and the necessary tests to catch edge cases and cover all details of the feature. This involves evaluating what the expected outcomes are for each requirement and determining all possible cases that could result.
4. Organize the tests and set up the environment so that the tests can run smoothly on the application and catch errors. First, focus on unit testing and then move on to other forms of testing.
5. Run the tests on the application. Compare expected outcomes to actual results within the testing framework to determine errors and causes. Document any failed bugs in issues and return to the code to focus on bug fixes.
6. When all tests pass, push the code to the repository for testing the code as a part of the CI/CD approach. Merge new code through a reviewed pull request.
7. Report on and document these outcomes in a revised section of this report.
8. Repeat for the other necessary forms of testing: system and integration testing.

We will be using CI/CD approach to testing, meaning that we will continuously integrate and deliver any changes or bug fixes made to our code. This involves building the code and running tests whenever we push new changes to the application. This allows us to continuously work on features and push fixes while ensuring they do not break the existing code and that the functionality works as expected.

V.3.   Test Plans

V.3.1. Unit Testing

Unit testing will take place on each individual component using Mocha: a feature-rich JavaScript test framework running on Node.js and in the browser [3]. Mocha ensures asynchronous testing of each component as well as flexible and accurate mapping of uncaught expressions to the correct test cases. Tests in this framework would run serially using an assertion library called chai. This would ensure verification of the test results. The testing activity will take place in the app.js component of the client as well as server side of the application. In order to conduct the testing - beforeEach() and afterEach() test hooks will pass each test case in the script, these two hooks would encapsulate the <div> element using which the components would get mounted. To define the test case the describe() function would be defined and then the it() function will be used for function calls to identify each individual test case. To run the test a babel-register which is a developer dependency would be installed as a plugin. The babel register would store .babelrc file in the project root and add the contents in the form of a json document in it.

V.3.2. Integration Testing

Integration testing will be conducted between components of the code that interact with each other, such as interaction between multiple services like the API and database and the

interaction between react components that work together in a redux environment. For conducting these tests, the framework required includes a combination of Enzyme and Jest [4]. Jest runs in Node and uses jsdom which simulates a DOM environment. The objective of our application is to use MaterialUI and react-bootstrap for styling, but some components of our application might involve the use of CSS. In that case, services like saucelabs would allow us to test that the layouts are looking good across multiple browsers. The tests in this framework would run serially using Jest as an assertion library, which would ensure verification of the test results. The following Jest functions will be utilized:

makeMountRender — a util function that accepts a component and some default props. It returns a function so that we can override the default props if desired. This function returns a react wrapper using enzymes mount.

makeStore — a util function that is used to create a new redux store. We import the rootReducer and the createStoreWithMiddleWare function from our actual application which makes it easy to create a new store for testing purposes.

reduxify — a util function takes a component, props and an initial redux state and returns a new react component which renders the passed in component in a redux environment.

snapshotify — a util function that accepts an enzyme react wrapper returning an HTML string.

We pass our component into **reduxify** and then pass that into **makeMountRender**. The result is an enzyme react wrapper that can be passed to **snapshotify**.

V.3.3. System Testing

The objective of our system testing is to check the integration of the components, check that the code runs in a real browser, and to handle cross browser JS concerns. In order to replicate a real browser, a tool called Karma will be used. The reason for choosing Karma is that its testing framework spreads across multiple frameworks, including the ones which we have utilized for our integration and unit testing–Mocha, Enzyme and Jest. On top of that, React web applications need to be versatile across various browsers. Therefore, in order to check cross-browser versatility of modern frameworks, a transpiler called Babel will be used. This will ensure that old JS engines do not face issues in utilizing modern features of the application.

V.3.3.1.   Functional Testing

The objective of our functional testing is to test the core features of our application, each feature in this case would be tested for all its components using shallow methodologies of enzyme and mounting methodologies using Jest and the inbuilt react testing libraries [5]. On top of that, project tracking, project planning, and project data storage would be separately tested using the inbuilt react test libraries to ensure that they are working as per the requirements of the system. The various layers of testing for each component would be recorded and all the issues would be added to the kanban board so that the developer who performed the test could add to the team repository and assign the team member whose work is responsible for the given error. That team member must then rectify the bug and write appropriate tests to catch the bug in the future.

V.3.3.2.   Performance Testing

The objective of our performance testing is to check the code for its qualitative aspects. The team's goals for achieving the non-functional requirements consists of using object-oriented

software principles to ensure reusability, performance, scalability, manageability, and maintainability. In order to confirm that the code is in correspondence with these requirements, a static code analyzing tool will be used. This will ensure that the team uses one coding style and avoids known anti-patterns in development. Arguably, one of the best lint tools for our project would be ESLint due to its support for a variety of plugins to extend the functionality. It also has rich easy-to-use documentation [6]. The create-react-app command (used for creating a react application) comes with ESLint already installed and is inside the package.json under the eslistConfig object. Another non-functional requirement is portability. In order to ensure that we can port the application to a mobile based application in future, the use of Karma will take place. Karma supports cross browser as well as mobile-app compatibility testing frameworks for our code.

V.3.3.3.   User Acceptance Testing

User acceptance testing will be carried out during the demo for our projects. This would be an opportunity for our team to test our prototypes over various phases of the software development cycle. The essential requirements of the user acceptance testing would include, project creation, project management (creation, deletion, and editing) and fever chart status. This would not be the only limit to user testing, as during the testing phase our objective would be to get feedback from users on possible unknown issues. The objective of this testing is to make the application as user-friendly as possible. The process for evaluation would be getting feedback from users, if a bug is found or there is user feedback to add a feature it would be created as an issue on github and a team member would be assigned to patch the bug or create a new feature.

V.4.   Environment Requirements

For this project, there will be few tools that are required to let us do the testing on VPM. On our frontend side, which is also the client, we will be using Jest for the testing. On the other hand, Mocha will be used to test the Node js within the application. This is for the backend side of the application which includes the MongoDB that we are using as the database for this project. There are no requirements needed for hardware in this testing process.

VI. Alpha Prototype Description

For our application's current alpha prototype, we have implemented the database completely by connecting the backend with our frontend through our API. The database is receiving the correct information that was submitted by the users and it was accurately shown in MongoDB. Next up, our current API and routing section is implemented well, however there are still some API that we need to figure out in order for us to create new functionalities or refine our created features to achieve its fully functional features. We will be constantly updating and refining our API and routing throughout the current and next semester to improve the application. Meanwhile, on the frontend side, we managed to create multiple pages such as "In Planning" and "In Progress" tab as our main pages with correct styling and fields to provide a good UI and UX for the users. We will be adding more pages in the future to make progress towards the full completion of our app in providing the best experience to our users. For ProjectLogic, we created a new file called "ProjectSlice.js", and its main purpose is to obtain the correct particular project(s) in order to access information and update the information within and upload them back to the database. It is still not fully implemented with all functions required for the project, but they will be developed in the future. There are still two subsystems we have not been implemented completely, these

are the temporary data storage and authenticator. Those two subsystems will be implemented in the future, with more details at the "Future Work" section.
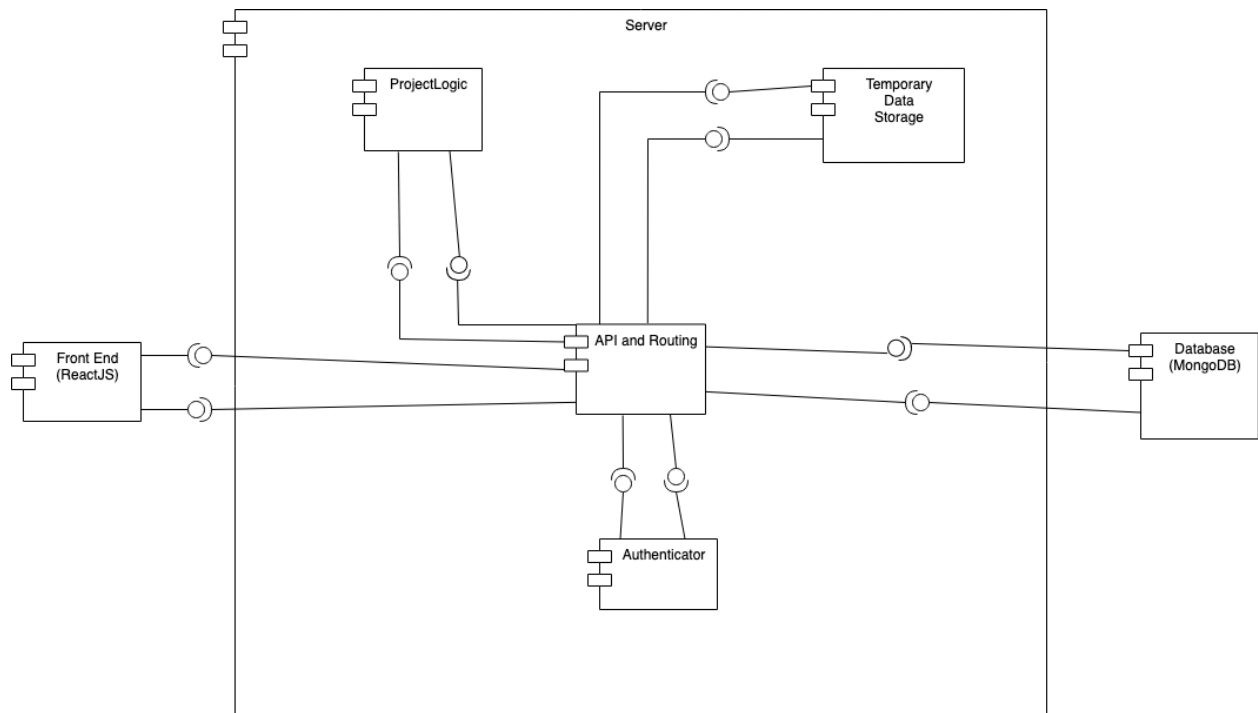
VI.1. API and Routing

VI.1.1. Functions and Interfaces Implemented
Our API is fully implemented for our current functionality, including obtaining project(s) using Axios with the project's ID. We also have a function "createProject" to help us create a brand new database entry for a newly created project. A post request helps create the project in the database. Next, we have an "updateProject" function that uses axios.patch, which takes a particular project ID and the url with the project prompted from the database and updates any information that was changed in the database. New code will be added to the API for features such as delete and duplicate, among others.

VI.1.2. Preliminary Tests
We will implement unit testing for this component in the first semester. Testing this component will involve unit tests that check, for example, if an API call to the server returns the correct project when a project ID is passed.

VI.2. ProjectLogic

VI.2.1. Functions and Interfaces Implemented
In our application, we created a file called "ProjectSlice.js" on our client side, and we have multiple functions for creating and updating projects through using redux toolkit. We are able to load and store the project's information to our frontend through these functions and they are

fundamental for connecting to the database for obtaining information and updating them. There will be more functions added for calculations of aggressive time duration and updating them through here if there are any changes to them.

VI.3. Front End (ReactJS)

VI.3.1. Functions and Interfaces Implemented
The front end side of the VPM application is mostly implemented with our current features. We separated the main home page of the VPM app by classifying it into "In Progress" and "In Planning" tabs. This helps us to differentiate the projects that have been published and those that are yet to be published. The published ones will go to the In Planning tab while the others will go to the In Progress. We also have a ProjectForm page which is the user interface page for users to create a new project. Once they are published, the project gets added to the details page that allows users to view the descriptions within the project and also change information about the projects using the edit tab. We are planning to add more pages for our future features such as project archives, app settings etc.

VI.4. MongoDB Non-Relational Database

VI.4.1. Functions and Interfaces Implemented
The implementation of our web application is based around the MERN stack and therefore we have used MongoDB. The use of MongoDB is ideal for us because our objective was to utilize a non-relational database. MongoDB stores data extracted from the form on its server and also data is extracted from MongoDB to be displayed on the web application. The data stored on MongoDB includes the project ID, project description, and the time required to complete the task. The data can further be manipulated using the edit feature on the application and the values are subsequently updated on the database. To further add to the functionality of the database, our UI can even display the data from MongoDB under the view projects section. This process was implemented using the Mongoose protocol. Following the alpha prototype, we will need to add in collections for the currently non-existent data types, as well as assess if any others are needed.

VI.5. Authenticator

VI.5.1. Functions and Interfaces Implemented
Authenticator is still in the development process, but sub-components of the application have authentication to ensure smooth functionality of the application. Currently, in order to validate all the form submission requests, mongoose is used. It validates the data inserted by the user before uploading it to the server. There is also form validation, which authenticates the user input in the frontend before sending it to the database. The use of get and post in these protocols further catches any errors and keeps the subsystem up to date. The current subsystem has the required implementation in order to serve our proof-of-concept demonstration. In the final iteration of this project, this subsystem will make use of authentication for all of its services.

VI.6. Temporary Data Storage

VI.6.1. Functions and Interfaces Implemented

The "Add New Project" pops up a form which inputs data from the user. In the JavaScript framework we require temporary data storage in the form of a central state which can be used to store temporary data for reuse within the application. Due to the integration with the backend, the temporary data storage incorporates the Redux environment which has been developed in our web application to ensure smooth processing of data between the frontend and the backend as well as easier use of temporary data to render our UI.

VII.   Alpha Prototype Demonstration

In the coming weeks, we will demo our current prototype VPM app to our client, Dr. Holt. We will present the work we have done on the prototype thus far and collect feedback for the second semester. The features that have been implemented in the prototype are on the list of critical features that have been provided to us. The following list details the features we would like to present during our prototype demonstration:

1. Firstly, the prototype consists of a placeholder login page that leads into the "projects" dashboard of the app, since the authentication feature has not yet been implemented.
2. Throughout the entire app, a title navigation bar runs along the top of the screen while a sidebar is on the left side of the screen for navigating to the "projects" dashboard, the archive page, and the settings page, which is not yet implemented.
3. The archive page shows a list of all projects that have been completed. A project can be clicked to display the full details of the project, including tasks and a fever chart.
4. Within the "projects" dashboard, there are two tabs for "in planning" and "in progress" projects.
5. In the "in planning" tab, a user can create a project and view a list of projects that have not been started. A project can be clicked to view a page displaying the details of the project and a page to edit the project. An icon next to the project name allows a user to delete the project.
6. In the "in progress" tab, a multi-fever chart is shown that displays points for the last-known completed task for each project in the list below it. The project can be clicked to view the fever chart for that specific project, the tasks that are part of the project, and the project's details.
7. The user can plan their project, start it, complete tasks that are in the project, view and edit the project's details, and finish a project. They are able to manage the projects that are "in planning," "in progress," and in the archive and determine the completion of a project based on its color zone status and its position on the chart.

VIII. Future Work

In the second semester, we will focus on expanding our VPM app prototype and completing delayed tasks from the first semester. The first major tasks to be completed in the second semester are authentication and user roles. There must be a fully functional login page where users can create accounts to house their projects. A user's account needs to be authenticated with a username and password. The user must only be able to access authorized content for the duration of their session.

There must be a way for a user to create an organization that owns projects. Users can have special roles within an organization. They can be owners, project managers, administrators, resource managers, or workers. Depending on what role a user has, they will have varying

authority over projects. For example, workers can only complete tasks while project managers can manage the entire project. A major task will be ensuring that each user can only perform the duties that are allowed to do according to their role. Only project managers should be able to create, duplicate, and edit a project. An authorized user should be able to manage the people added to the project and what level of control they have over the project.

Another task for the second semester will be to refine the functionality to create, delete, and edit projects completed in the first semester. The tests for this functionality should be expanded to ensure that the user input is processed correctly on the UI. The work hours and aggressive duration for each task must be calculated correctly to ensure that the times are correct. Tasks should also be prioritized by their project color status. The project timeline from creation to in progress to archive must be smooth and functional. Other lower-priority tasks that can be completed include warning users when a project enters the red zone or black zone. Overall, the functionality and testing completed in the first semester will be cleaned up and refined, while the next important features will be developed. If time allows, the lower-priority advanced features will be implemented.

IX. Glossary

VPM: Visual Project Management

ETM: Engineering and Technology Management

UX: User Experience

UI: User Interface

HTTP: Hypertext Transfer Protocol

HTTPS: Hypertext Transfer Protocol Secure

URI: Universal Resource Identifier

CI: Continuous Integration

CD: Continuous Delivery

X. References

[1] Holt, Dr. James, WSU Academic Outreach and Innovation - AOI. Visual Project Management by Dr. James Holt. (Apr. 13, 2012). Accessed: Sep. 20, 2022. [Online Video]. Available: https://youtu.be/DPFTJayYrnk.

[2] J. Holt. "In Progress." Visual Project Management. https://www.visualprojmgmt.com (accessed Sep. 20, 2022).

[3] OpenJS foundation. (n.d.). *The fun, simple, flexible JavaScript test framework*. Mocha. Accessed: October 28, 2022, from https://mochajs.org/.

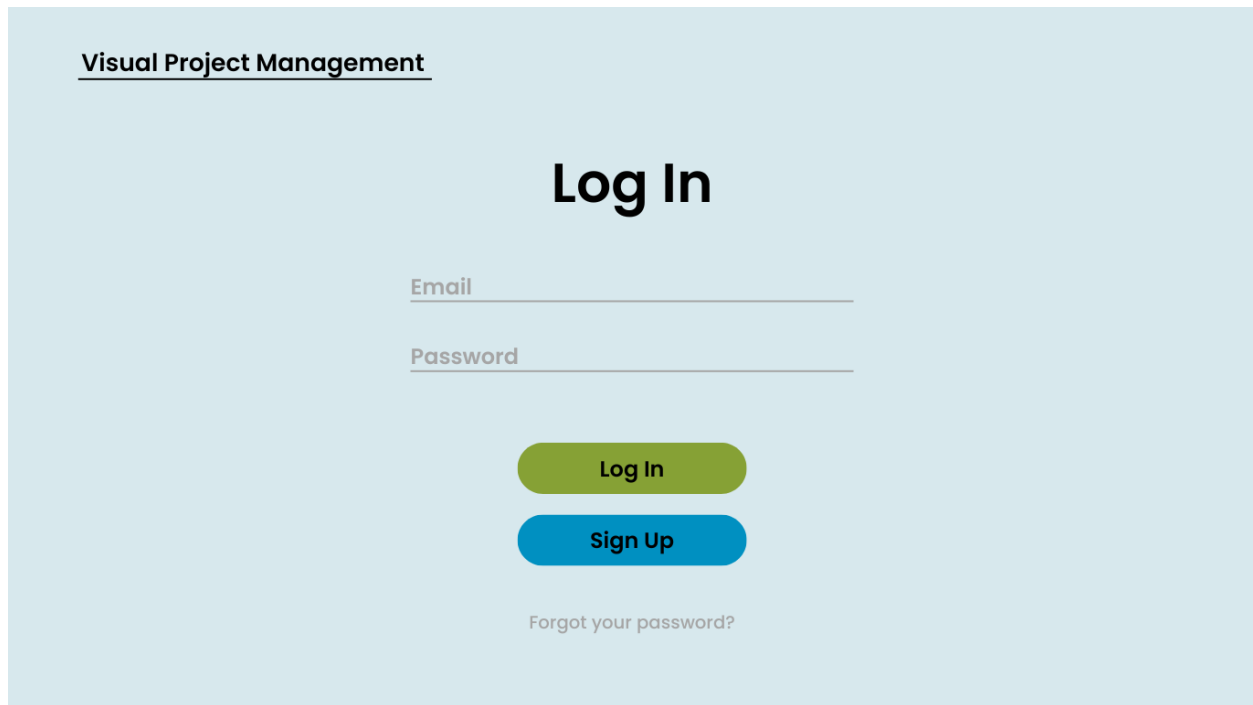[4] Facebook Open Source. (n.d.).,"Testing React Apps." Jest. Accessed: October 28,2022. from https://reactjs.org/docs/testing.html.

[5] Facebook Open Source. (n.d.)."Testing Overview." React. Accessed: October 28,2022. from https://reactjs.org/docs/testing.html

[6] OpenJS foundation. (n.d.). "Configuring ESLint."ESLint. Accessed: October 28, 2022, from https://mochajs.org/

XI. Appendices

XI.1. Appendix A
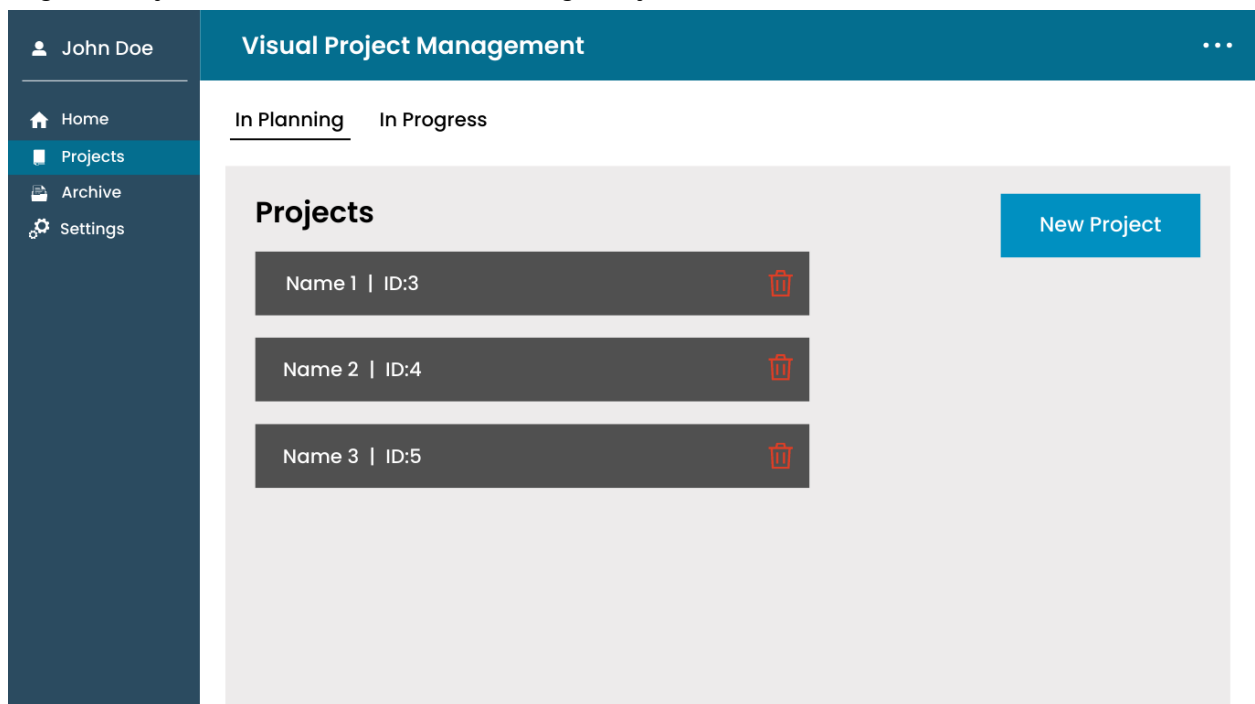
Page 1: Login Screen

Page 2: Projects Dashboard for "In Progress" Projects
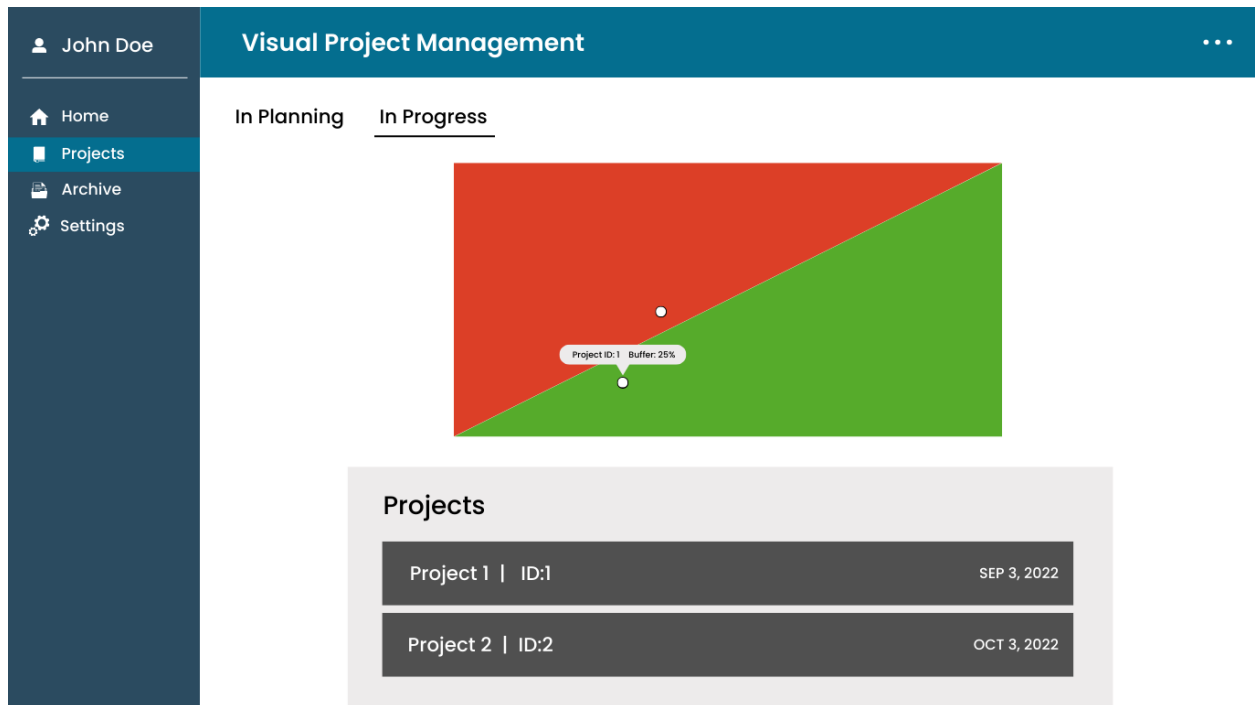
Page 3: Projects Dashboard for "In Planning" Projects

Page 4: Individual Project Dashboard for a Project "In Progress"

Prototype Project Report   27