Table of contents

1. INTRODUCTION

1.1 Client’s Problem

The problem experienced by the client is the significant delays in scheduling repairs and upkeep due to late part ordering and the inability to track which machine has already been updated. When ordering inventory parts, the client often forgets which inventory items to prioritize (different priority due to the task having a higher priority) and schedules the item's arrival date later than the needed date. Due to the large number of work orders, the client takes a while to consider which employees should be sent to fulfill the work order based on their trades, skills, and availability. The significance of this problem is due to a considerable amount of time wasted checking each machine due to being unable to keep track of the machine's status. Another significance would be the large amount of money lost due to having to hire a professional to recheck the machines, being unable to use the machine until it is repaired due to the machine parts reaching their lifespan, or being unable to fulfil work orders due to parts being delivered later than needed.

2. REQUIREMENT ANALYSIS

2.1 Client's Requirement:

2.1.1 Front End (User Interface)

1. Integration with Kronos for:
    - Assigning employees based on trade type, certification, experience, and availability.
2. Integration with our project for:
    - Acquiring KPI feedback.
    - Accessing maintenance schedules.
    - Receiving inventory notifications.
    - Documenting completion time of tasks.
    - Quick access to checklists and manuals.
    - Logging replaced parts, parts needing repair, and additional notes.

2.1.2 Back End (Automation and Database)

1. Integration with Kronos database for employee data access to support scheduling.
2. Our project database for storing:
    - Asset information.
    - Parts inventory and location data.
    - Task lists and machine manuals.
    - Continuously updated KPI feedback.
    - Predictive maintenance timelines based on historical repair data.
    - Automatic notifications for:
        - Maintenance Planner regarding asset efficiency.
        - Inventory Controller when parts stock is low.

2.2 Analysis:

Kronos is an existing software used to help businesses with attendance, scheduling, payroll, and employee management. It also helps the company ensure it is following labour laws. By integrating Kronos into the user interface, we can take the data from the client and input it into Kronos to quickly sort out the information.

Our project is a CMMS, or Computerized Maintenance Management System, which helps track work orders, manage assets and inventory, schedule preventive maintenance, and generate reports.

2.2.1 Front End (User Interface)

Integration of Kronos

The user interface will use Kronos to help find suitable employees, as the Kronos software can help recruit employees to find someone with the right skills and experience for the job.
The user interface will use Kronos to help manage the employees' schedules and automate the scheduling process. This is because Kronos considers employees' availability, vacation days, and time off. In addition, it can also take in time off requests, making it easier for the manager to quickly check if the time off may conflict with another.

This will help supervisors create work orders, as Kronos would make it faster to assign work orders to employees as a result of the system providing the most suitable people for the job based on their experience and skills. This would be very helpful in reducing the time consumed in organizing work orders hence allowing the company to function efficiently under a large demand of service orders.

Integration of our project

The user interface will send supervisors KPI feedback and allow access and modification of the maintenance schedule. The KPI feedback will include employee performance, maintenance cost, downtime, and asset utilization. This will enable them to gain further insight into work performance and help the supervisor understand how to optimize it. Looking at downtime will help supervisors create a maintenance schedule that allows them to repair parts of the machine before they break down, ensuring that the machines do not malfunction during work orders.

The user interface will notify the managers whenever the supplies of specific inventory are low, based on how fast the inventory is being utilized. This will give them enough time to order the parts and have them delivered on time.

Employees assigned with work orders will be able to use the user interface to document the time of completion, which will help determine the next maintenance period of the equipment as well as the time the machinery is ready to be operational. While fulfilling their tasks, workers can quickly access checklists and manuals to complete their work safely and promptly. There will also be a place for them to catalogue the parts replaced, parts that may need repair soon and any other notes they may have about the task or equipment.

2.2.2 Back End (Automation and Database)

Integration of Kronos

In order for the scheduling part of the UI to work, the Kronos database will be needed to access and retrieve important information about the employees. This would work by matching employees to their respective attributes which would assist in scheduling and assigning work orders.

Integration of the project database and automation system

Aside from the Kronos database, there will be a secondary database responsible for holding key system data. The data will include asset information, updated KPI feedback and historical repair data. This will be useful in assessing current machinery conditions and inspecting previous repair data that could be used in scheduling future maintenance accurately. So when repairs are done or the software of the machines is updated, the information is logged into the database, which automatically updates the machine’s information.

Additionally, the database will house inventory and location data which will be helpful in providing updated information about the parts in the inventory and location of the machinery. Employees will also be able to access critical information required to complete their assigned work orders such as task checklists and machine manuals which will promote safe and smooth maintenance.

The CMMS will also have its own in-built, automated system which will generate real-time feedback of KPIs and asset efficiency that will be of use to the maintenance planner in scheduling repairs and maintenance check-ups, aided by predictive maintenance timelines based on historical repair data. Lastly, the inventory controller will be automatically notified if the parts in stock reach a set threshold. This will be useful in planning ahead and ordering parts before the inventory is depleted.

All these features will ensure that operations are running smoothly and efficiently.

3. PROPOSED SOLUTION

To refine our clients maintenance workflow, we will create a CMMS (Computerized Maintenance Management System) that focuses on real-time tracking, automation, and scheduling to complete work orders efficiently and in a smooth manner. This system will provide unified work order management, inventory tracking, and employee scheduling while maintaining fluid communication between different components using a structured software and hardware architecture.

3.1 Technical Specifications

3.1.1 Front End (User Interface)

Design a user interface (UI) that is user-friendly, including an accessible and responsive UI that will allow the app to run smoothly on PCs and tablets. For the front-end we will use:

- **Vue.js** – A JavaScript framework for creating an interactive and dynamic user experience.
- **JavaScript** – Manages client-side functionality.
- **HTML (HyperText Markup Language)** – It gives a structure to the web pages.
- **CSS (Cascading Style Sheets)** – For styling and layout presentation.

The UI will adhere to the standards and specifications described in AODA **(Accessibility for Ontarians with Disabilities Act)** to become user-friendly for all users.

3.1.2 Back End (Logic and Automation)

The backend can manage automation, business logic, and database access. We will use:

- **Java** – Handles server-side logic, automates processing, and connects the UI to the database.
- **API Integration** – APIs will be utilized to facilitate secure data exchange between various components of the system. We'll integrate the Kronos API to pull employee schedules, training records, and availability, and update as necessary.

The backend will also handle:

- Scheduling of tasks automatically according to work orders and priorities.
- Predictive maintenance analysis with historical data.
- Notifications and alerts for low inventory, overdue work orders, and equipment inefficiencies.

3.2 Data Storage and Management

All system data will be stored and managed by a MySQL relational database, including:

- **User Data** (employee credentials, roles, training records).
- **Work orders** (current, completed, pending maintenance tasks).
- **Assets and inventory** (equipment details, manuals, part availability).

Database Structure follows the **MVC (Model- View- Controller)** pattern, its construction being important for scaling and maintainability.

3.3 Security and Data Transmission:

We will cover the following with respect to secure handling of data:

- **API communication over HTTPS** - encrypts data transfers between the UI, backend, and the database.
- **Role-based access control (RBAC**) – Access is controlled based on user roles (maintenance personnel, supervisors, inventory managers).

3.4 Hardware & Network requirements

Hardware Requirements for the system:

- **PCs and tablets** — ideal for planners, supervisors and maintenance personnel to connect to the CMMS.
- **Network/server setup** — This allows all devices to have the same data instantly.

3.5 Advantages of Our Solution

✅Intuitive & user-friendly interface (functional through Vue. js).

✅Effortless automation for maintenance task (Java Backend with API integration)

✅Track work orders, inventory, and asset status in real-time.

✅ Store and manage data securely and efficiently (MySQL database with encryption communication via API).

✅Contains well-structured MVC **(Model- View- Controller)** design (Spring Boot backend) so it is scalable and maintainable architecture.

✅It is compliant with AODA **(Accessibility for Ontarians with Disabilities Act)** standards for accessibility, making it easy to use for various users.

With this solution, we will bring an end to manual inefficiencies, minimize the time taken during maintenance, and automate processes with real-time tracking of the client's workflow.

4. SYSTEM ARCHITECTURE

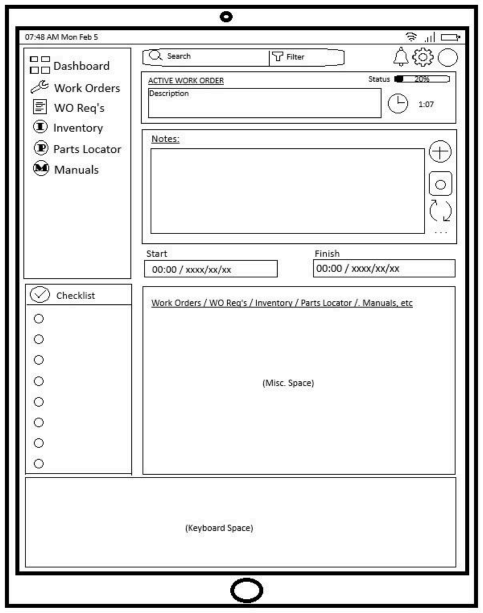
The architecture of our CMMS will require a multitude of component layers to come together in a hierarchically sound and logical fashion. Starting from the external hardware, we will need to set up a dedicated server. Ideally our server will have at least three drives to implement RAID 5 (redundant array of independent disks - striping with parity). This will allow for continuous operation and recovery of data in the case that one of the disks experiences a failure. From here we will integrate our CMMS onto the existing network which is the infrastructure backbone of this type of technology.

4.1 User Interface

The user interface will consist of two similar designs, one will be designed for the planner to have an overview and accessed via PC while the other will be more condensed and designed to work on a portable tablet. Each interface will share some similarity in their design principles such as ensuring both interfaces are AODA “AA” compliant as a minimum. We will aim to follow an intuitive design layout, having our menu list along the left side in both interfaces while dedicating the main screen space for visual feedback or other key functionality where the user will interact.

4.1.1 Tablet User

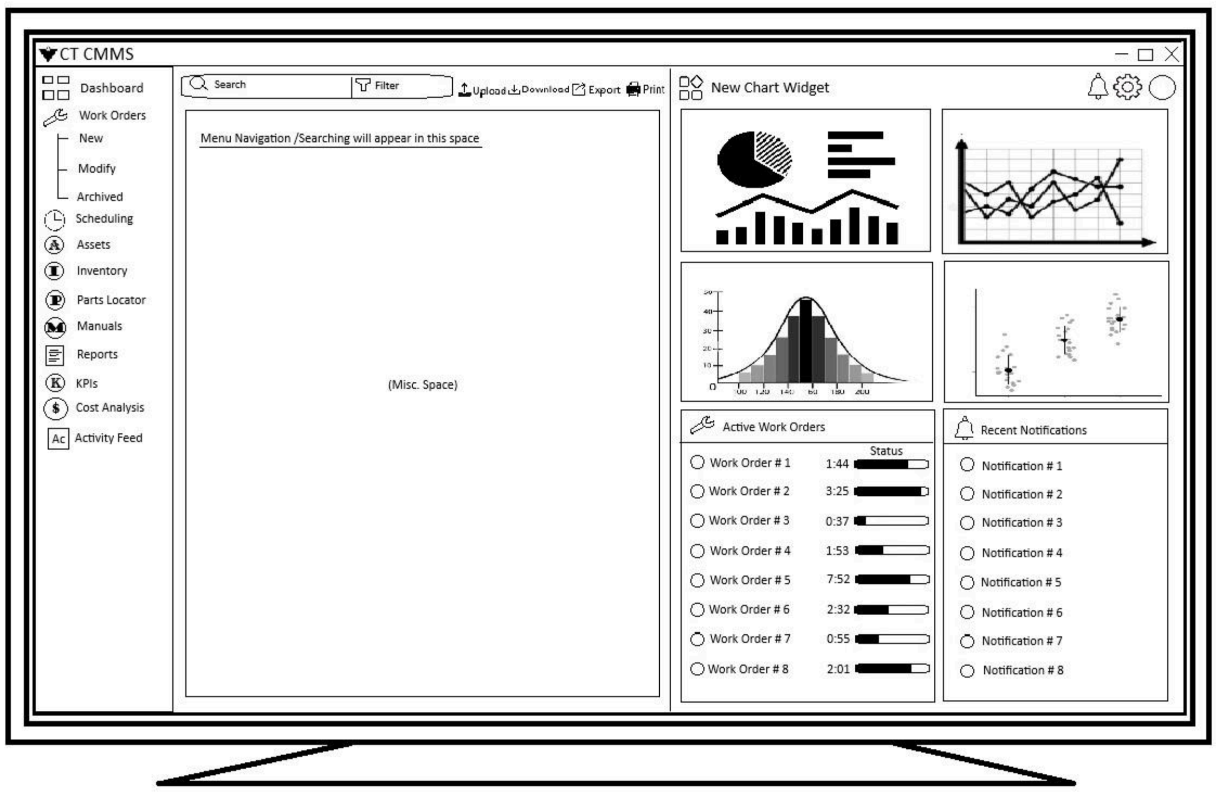
From the workers side, they won’t have quite as complex of an interface but it will be designed to show them what work order is currently in progress, their list of work orders accepted as well as a list of requests. Beyond this they will have a few more menu options which will be for accessing inventory information directly or a search tool for locating parts and otherwise an option that will allow them to access manuals. The lower half of the screen will allow them to view and interact within whichever miscellaneous tab they have open. There will also be a condensed checklist along the lower left side which can be expanded and shrunk as needed. The status on their current work order will be tied to how many checklist items are currently completed which will automatically update the progress bar. The notes section will allow them to attach relevant links such as part/manual references into their notes. They will also be able to use the tablet for the purpose of taking any photos they wish to include in documentation. Aside from this the note section will serve as a general purpose notes block which will send updates that the planner can view whenever they tap the revolving arrows icon on the right. The lowest portion of the screen will be dedicated to keyboard inputs which will pop up from the bottom when they select an input field. The very top of the screen will have a general purpose search bar that they can use to do quick lookups if they wish to quickly search rather than navigate through the dedicated menus.



A sketch depicting the tablet user interface

4.1.2 PC User

On the planner's view, we will have the list on the left side similar to the worker display, however, this time it will include additional features related to monitoring and managing maintenance. The planner interface will also include a large dashboard area which can be customized by the user to display visual information for quick assessment of desired parameters. This could include things such as KPI charts, lists of current work orders with their progress indicated and recent notifications for example. The planner user will also have access to scheduling tools and relevant employee data for pairing work orders to the appropriate person for the task. This leads to another key function which will be creating and managing work order details such as the checklists, description, estimated time and any other defining fields needed to define a work order. Beyond this, the planner interface will include a quick search bar along with data combination filters to group data from different sources for correlations analysis which can aid in strategic planning.



A sketch depicting the PC user interface

4.2 Hardware

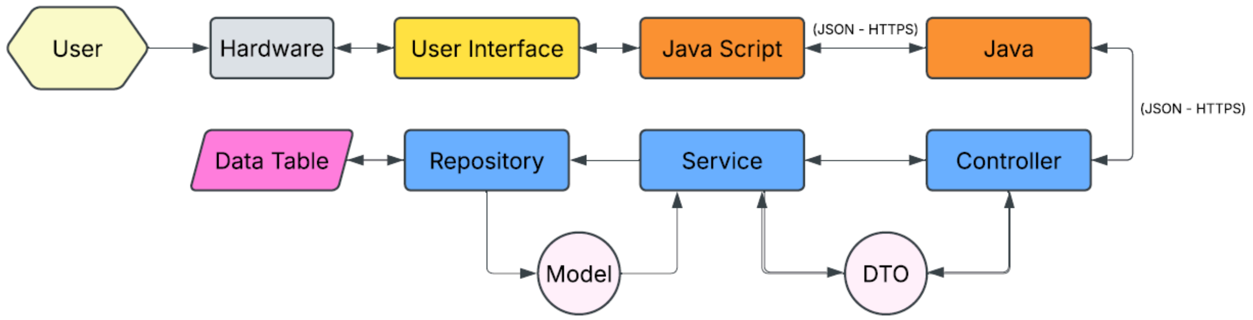
The customer will require a number of rugged work tablets such as the Zebra XSLATE R12 that would be suitable for working in harsh conditions such as industrial or maintenance applications. Although these tablet models are an older design, these appeal for this application as our priority would be suitable durability for the working conditions in place over high performance since the software running on them should not be too CPU intensive. Next, we will need at least one PC workstation at most, presumably these will already be existing hardware that the customer has on hand, so we should not have any issues beyond requiring to install our CMMS software onto their system.

4.3 Framework

Beyond the above addons, the CMMS will be anchored by two different frameworks, the front end will use Vue.js which is a simple and flexible framework suitable for our UI needs, while the backend will use Java's Spring Boot framework. The reason we are choosing to utilize these two frameworks is because they will allow us to develop and integrate our system faster than if we did everything from scratch. The frontend framework will manage our java Script, HTML and CSS programming components which will make up our user interface. On the backend, we will stick with trusty old Java due to it being a versatile OOP language, that is reliable and has a robust community to maintain its libraries, hence making it a solid choice with the long term in mind. The last programming technology we will need to complete our setup is MySQL which will serve as our relational database to hold and link all of our raw data that our CMMS will rely on as part of its tracking abilities and other automated features. The reason we chose this database from the many options available is that it is well tested, reliable and simple to work with while having good scaling up potential for future expansions.

4.4 Programming Technology

Our CMMS software will be backend heavy as most of the operations will occur under the hood outside of the user’s sight. Since we went with Spring Boot as our backend framework, we will organize our backend code in 5 layers, these layers are categorized into controllers, services, repositories, models and data transfer objects (DTOs). The general flow of this type of operation will follow a similar pattern for most processes, starting with the user performing actions and then having Java Script send requests to our Java application to retrieve what it needs to display. Java Script will communicate to java via controllers which will then delegate to their respective service handlers that will manage in accordance with their internal logic. The service component is responsible for taking the request of the controller and performing an action in response. In many cases, this logic will require passing an argument to a repository which will then further access the database tables and generate a model that correlates to the argument passed. This model is then returned to the service which will then in part generate a DTO and pass this back to the controller. The controller then communicates back to Java Script and thus the user interface receives whatever data/output was requested. Based on Spring Boot framework, this will be the general pattern which occurs with each front to back end interaction with flow depth depending on what is requested.



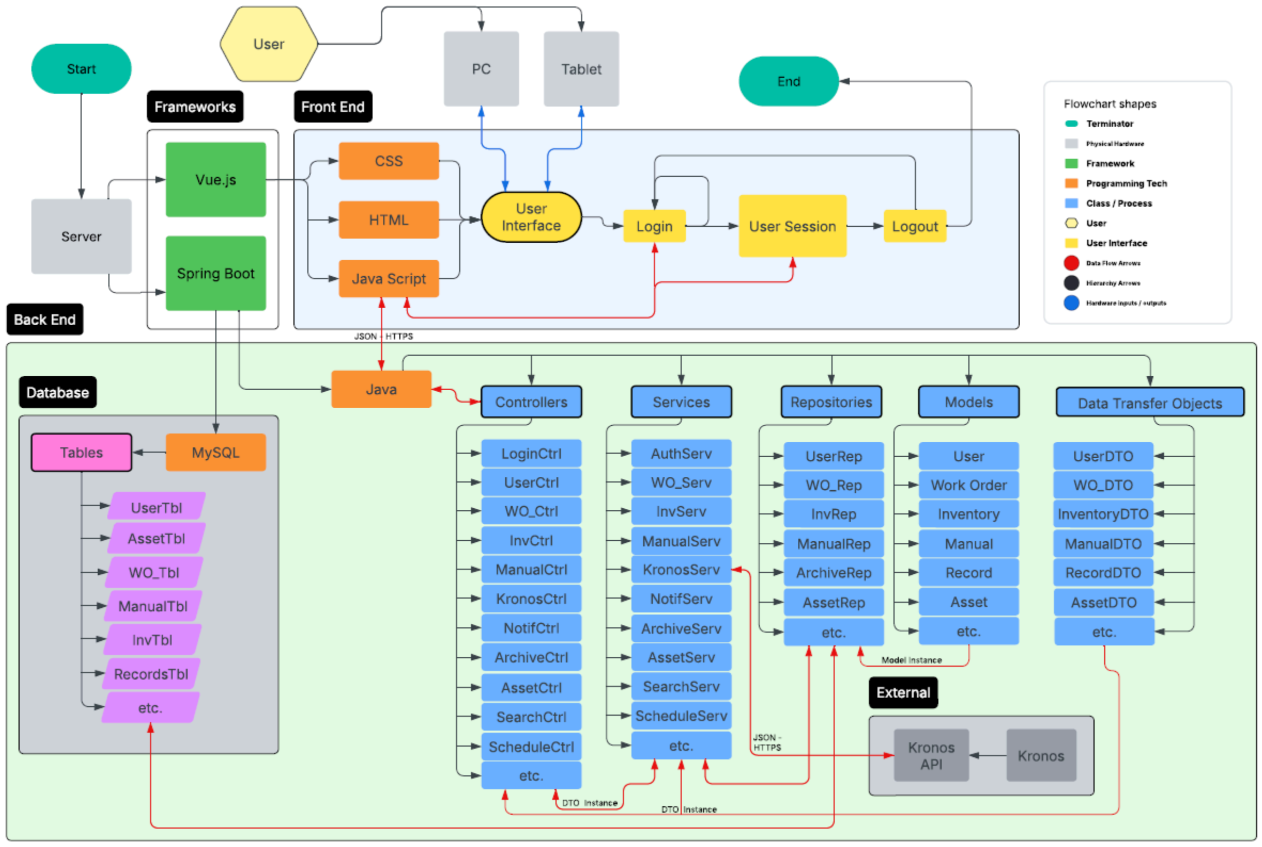
A diagram showing a simplified typical logic flow pattern start to finish

4.5 Communicating with Kronos

A key aspect of our software architecture is that it needs to be able to mesh with Kronos' API to access useful employee data that will be relevant in producing schedules. Kronos has been designed with need in mind and so there are standard APIs in place for this purpose. However, in the case that we have any trouble establishing a connection via the normal route, Kronos is also affiliated with third party support which are specialized in developing custom API solutions to connect Kronos with external software. On our end, all API endpoint communication with Kronos will be handled primarily by a Kronos Service class that will be running in the on our backend Java program. This process will be responsible for maintaining continuous updates as needed by performing data requests when required. Once retrieved, this information will be returned securely back to the user interface for tasks such as setting up work orders and scheduling.

4.6 API and protocols

One last important aspect of the CMMS architecture is going to be handling data transmission correctly. Since we will be passing a lot of simple data over the network – our preferred choice of data format will be the JSON format for organizing our data for transfers. Aside from regular data, we will also be handling more sensitive information such as user data, and other company internal data. As part of our architecture, we will need to implement a login system that will authorize users as there will be a limited number of tablets on hand, and they will need to be distributed based on whomever is scheduled to work that day. To secure this data during transfers, we plan to implement APIs that use encryption protocols such as HTTPS which will help keep the data secure during transit between different parts of the system. With this, we conclude the overview explanation of how the CMMS architecture will be laid out from the external outer layer all the way down to the inner abstract layers as well as how data will be handled in between. Below is a flow chart diagram that helps layout the architecture in a hierarchical manner as well as generally indicates some of the data flow paths which different parts of the system will take during interactions.



A flow chart showing the general overall architecture of our CMMS software.

5. IMPLEMENTATION PLAN

For successful implementation of the client's product, a robust structure must be created to address the pressing financial burdens and time constraints associated with the CMMS system. This section will aim to tackle these problems using four major steps:

5.1 Four Major Milestones

- Identify and define project scope and primary function
- Identify user requirements and ensure compliance with industry standards
- Assess costs and time constraints regarding API integration and IoT (i.e Kronos API and third-party services)
- Assess costs and time constraints regarding full implementation of system hardware and software (Java, SQL,Vue.js, etc)

5.2 Project Scope And Primary Function (Weeks 1-4)

Prior to beginning the client's CMMS project, the attention will be directed towards defining the scope of the product and its desired functionality, user requirements and feasibility to ensure it aligns with the client's operational needs. Discussions should cover system architecture for front and backend development using applicable languages such as Java, HTML and SQL. Furthermore, discussions should include ensuring compliance with industry and legal standards and integration of various systems into existing IoT devices, to ensure compatibility with any new software. All government and institutional approvals must be confirmed before beginning. Consultation with manufacturers and the client is necessary to maintain an organized schedule.

Total Estimated Cost: $50k-$300k

5.3 Frontend UI Integration (Weeks 5-9)

Once approval is met, designing the project will be of priority. Initially, attention should be directed towards designing a user-friendly interface for the CMMS product. Discussions should cover prototypes of wireframes, layout of the UI, navigation of the UI, and overall user experience of the system. During these weeks, a prototype will be developed to demonstrate the system's key functions such as work order tracking and inventory management, allowing feedback from organizers before full-scale development. Furthermore, ensuring compatibility with IoT systems will be necessary in this stage, including compatibility with hardware such as tablets and computers, establishing the foundation for frontend and backend development. This phase will also include collaborating with various companies to integrate hardware components and software into the prototype which will require further expenditure of both time and money towards the project.

Total Estimated Cost: $150k-$300k

5.4 Database, Backend Development And API Integration (Weeks 9-20)

In the latter time frame of the client's project, this phase will focus on the core backend system with an emphasis on processing, automation, data management, and integration with the frontend for operational use. Discussions will be focused on full-scale development of the software and the integration of API systems such as Kronos API for employee scheduling. During this stage, essential features such as task management, maintenance checks, and inventory management will be implemented. Security systems such as role-based access control (RBAC) will be required in this stage as well to manage what CMMS users can access. This will all be possible through a MySQL database which will contain everything, including employee credentials. However, the complexity of database security, backend development and full-integration of the system will necessitate consultation with manufacturers, lawmakers, and require skilled developers to integrate the completed project. This will likely be the costliest phase of the project and may delay the project significantly depending on various factors such as availability of workers, availability of parts, institutional approvals, financial limitations aligning with the client’s budget, and aligning with third-party company schedules.

Total Estimated Cost: $400k-$1M

5.5 Testing And Completion Of The CMMS Project (Weeks 20-34)

During this phase of the project, there will be rigorous system testing of the entire system, including security testing and compatibility with the IoT system. Once testing is satisfactory and the CMMS is approved, it will be deployed on production servers, followed by user training for planners, supervisors, and maintenance teams. Continuous monitoring and maintenance of the system will address any performance issues and require user feedback. Testing and addressing issues that may arise from the product may require more time and finances to be allocated towards the CMMS, delaying final approval and production of the system.

Total Estimated Cost: $300k-$700k