Software Architecture Description of Model View Controller for Bikerr, an ERP for Bike Manufacturers

Sacha Elkaim	29779698
James El Tayar	40097755
Jamil Hirsh	21236962
Ashraf Khalil	40066289
Shashank Patel	40094236
Adam Richard	27053929
Derek Ruiz-Cigana	40096268
Michael Takenaka	40095917

Revision History			
Version	Description of Changes	Responsible Party	Date
1.0	Initial version	Adam Richard and Jamil Hirsh	February 3, 2021
1.1	Fixed Viewpoints/views Edited Introduction Fixed diagrams Included 4+1 architecture Only one description of UML Clarified Implementation view Included class diagram	Adam Richard	February 22, 2021

Contents

1. Introduction

This section serves to introduce the software architecture and our reasons for choosing it. The software in question for which we are designing this architecture is an ERP for bicycle manufacturers. This will include manufacturing, transportation, sales, and customer relations in its final iteration. This software architecture description serves to aid in simplifying the development process of the ERP by providing a plan for the developers to follow.

1.1 Identifying Information

The architecture being used for this system is the Model View Controller Architecture. This architecture will be implemented for the online ERP for Bike Manufacturers, which will handle tracking inventory, managing payments, and managing accounts among other functionalities mentioned in the above section.

1.2 Scope

The scope of this project will be limited. This is because ERPs are widely complex systems, and it is impossible to make a full one in the span of three months. Therefore, the main focus will be on inventory. This includes tracking bike parts, materials to manufacture parts, and materials needed. In addition, the ERP will keep track of payments. This not only includes transactions for ordering bikes and bike parts, but also keeping track of customer accounts and the finances associated with them. Finally, we will take measures to protect user accounts. This includes the encryption of passwords to protect against negative users.

1.3 Other Information

This section seeks to provide further insight into the decision-making process for our system architecture.

1.3.1 Architecture Evaluations

Our evaluations of the Model View Controller architecture reveal that there is no specific way to code it. The principle idea is to separate the functionality of input, interface, and interaction. While there is no specifically defined method to code this architecture, it also provides us with some creative leeway into how we choose to separate the Model, the View, and the Controller. We have, through various meetings, decided to implement the Model View Controller architecture by separating each part into the front end, back end, and database of the system. The front end will be the view, which the user sees, the database will be the Model, where all the information of the system is stored, and the back end will be the controller, which communicates between the front-end view and the back end model.

1.3.2 Rationale for Key Decisions

The main reason why the Model View Controller architecture was chosen was because it is easiest to develop quickly. It is also well suited for enterprise or business applications, which is exactly what we are looking for in an ERP. Other architecture models would not be capable of this. Event-driven architecture, for example, is better for systems with asynchronous data flow. Testing for this architecture can be complex, however, and since we need to construct our application with limited time, this could prove to be a serious caveat.

2. Stakeholders and Concerns

This section seeks to identify the main stakeholders for our platform and outline their concerns. This will help to inform our architecture decisions in the following sections.

2.1 Stakeholders

This section identifies the main stakeholder for the application. They are divided into several main groups, which can in turn be divided into subgroups. The first group is Users. Among the users are customers, such as business owners, and administrators, such as the manufacturer's managers or employees. Next, there are negative users. These are primarily those who would seek to take advantage of the system's weaknesses. This group consists mainly of hackers who would attempt to break into the database. Finally there are the software developers who are tasked with designing the system.

Stakeholder	Description	Responsibilities
Manager	A manager at a given Bike Manufacturer. This person is in charge of overseeing operations at their company.	This person is responsible for all of the operations at their manufacturing company. They represent the principal customers of the ERP, and are intended to use our system to keep track of their inventory and manufacturing process.
Employee	An employee of the Bike Manufacturer. This person works for the manager, and carries out operations according to their role.	This person, while not our main customer, and we will not be dealing with them directly, will still be a user of the ERP system. For this reason, we must keep them in mind when designing the system, and make sure that it conforms to their needs.

Business Owner	This is a client of the manufacturer. This person purchases goods from the manufacturer.	This person is in charge of acquiring goods to sell at their own store. They go to the manufacturer to obtain these goods. They will be using our ERP system to view the manufacturer's goods and decide what to purchase based on availability.
Software Developer	This is a developer of the Bikerr ERP system. This person creates the software to be used by the clients.	This person is in charge of developing the ERP software based on clients' needs and desires. It is their job to make sure that the system contains all the functionality needed for the ERP to work effectively. This is a complex task, and so this person concerns themselves with all aspects of the design process.
Negative User	The negative user is someone who wishes to use the system in a way other than intended. One example of this person is a hacker.	This person's goal is to break into the system illegitimately, or otherwise use it in some unintended way. Usually this involves stealing the information stored in the database, or taking over an account.

Figure 1: Stakeholder Table

2.2 Concerns

This section seeks to identify the main concerns that the stakeholders may have. The main concerns for the stakeholders are as follows:

- Inventory
- Transactions
- usage log
- account security
- administrative privileges

2.3 Concern-Stakeholder Traceability

The following table indicates which stakeholder has which concern. This is shown by the table below:

	Manager	Employee	Business Owner	Software Developer	Negative User
Inventory	X	X	X		
Transactions	X	X	X		X
Usage Log	X				
Account Security	X	X	X	X	X
Administrative Privileges	X			X	X

Figure 2: Concern-Stakeholder Traceability Table

3. Viewpoints and Views

This section seeks to show the views that make up the architecture of the ERP system, and provide insight as to the viewpoints that govern them. The model used for the views of this system is the 4+1 view model of software architecture, first posited by Philippe Kruchten [1].

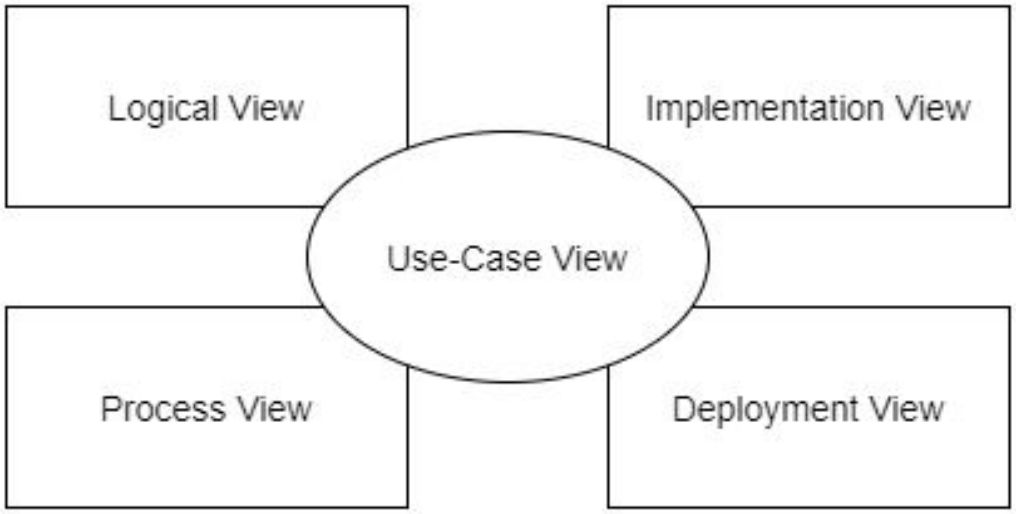

Figure 3: 4+1 View Model of System Architecture

The four views that make up the “4” in the above model will be the Logical View, Process View, Implementation View, and Deployment View. In Kruchten’s original model, the Implementation View and Deployment View were the Development View and Physical View, respectively, but within the scope of this project it is more reasonable to assume the views mentioned here. The Implementation View replaces the Development View, as it is more effective for us to observe the architecture in terms of implementation, and the Deployment View acts as the Physical View, as it will reveal how the system is deployed.

Further to this, the Use-Case View replaces Kruchten’s original Scenario View as the “+1” of the model. These two views are essentially the same, but since we model the scenarios as use cases, we refer to it as the Use-Case View.

3.1 UML Modeling

UML will be used to model all of the views further described in this section. UML is a simple standard that makes it easy to model each of the mentioned views in the section above. Since all views will use UML, it will not be indicated individually for each view.

3.2 UML Conventions

UML has standardized conventions for each of the models that will be given in this document. UML conventions can include the actors acting upon the system, boxes, representing the system itself, or classed within the system, ovals, representing use-cases, and lanes, representing processes. The UML notation also includes arrows that draw relations between different entities in each diagram, based on their relationships.

3.3 Viewpoints

This document will focus on several different viewpoints, each which inform the views that follow them. There are three main viewpoints that we will use in this document to inform the views. These are the development viewpoint, the business viewpoint, and the customer viewpoint.

3.3.1 Development Viewpoint

This viewpoint concerns itself with the development of the system. This includes both designing the architecture and implementing the software. Typical stakeholders for this viewpoint are the Software Developers and Negative Users.

3.3.2 Business Viewpoint

This viewpoint concerns itself with the business that needs to be conducted using the ERP software. This includes manufacturing, transportation, and sales. Typical stakeholders for this viewpoint are the Manager, employee, and business owner.

3.3.3 Customer Viewpoint

This viewpoint concerns itself with how the software is used. This is because it is the customers who will eventually use this ERP software for their manufacturing needs. Typical stakeholders for this viewpoint are the manager, employee, business owner, and negative user.

3.4 The Use-Case View

This view is divided into use-cases. Similar use-cases will be grouped together into single use-case models. These groupings will be defined based on the relevance of the use-cases. The viewpoints associated with this view are the development viewpoint, the business viewpoint, and the customer viewpoint.

3.4.1 Models

The main stakeholders for this view are the Business Owner, Manager, and Employee, which will be grouped together under the “User” stakeholder for

convenience. The main model used for the Use-Case view will therefore be based on the user stakeholder.

3.4.2 Use-Cases Related to the User Stakeholder

For this section, all users (Managers, Employees, and Business Owners) will be grouped together, as their use-cases are similar.

- Registration: A user of the Bikerr application must be able to register an account. A different account will be provided based on the kind of user registering.
- Login: A registered user of the Bikerr system should be able to use the login tool to access their account using username and password.

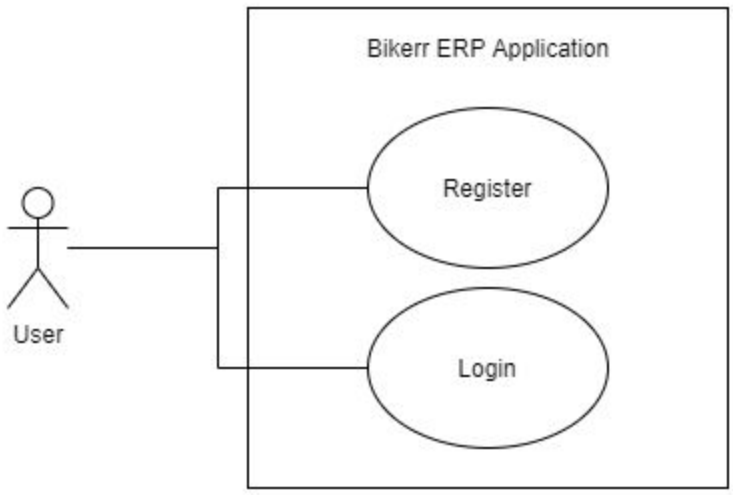

Figure 4: Use Case Diagram for Registration and Login

- View Inventory: A user who has successfully logged in to their account should be able to view inventory available for purchase by clients of the bike manufacturer.
- View materials: A manager or employee of the bike manufacturer who has successfully logged in to the application should be able to view available materials for part creation.
- View needed materials: A manager or employee of the bike manufacturer who has successfully logged in to the application should be able to view the materials that are needed for part creation.

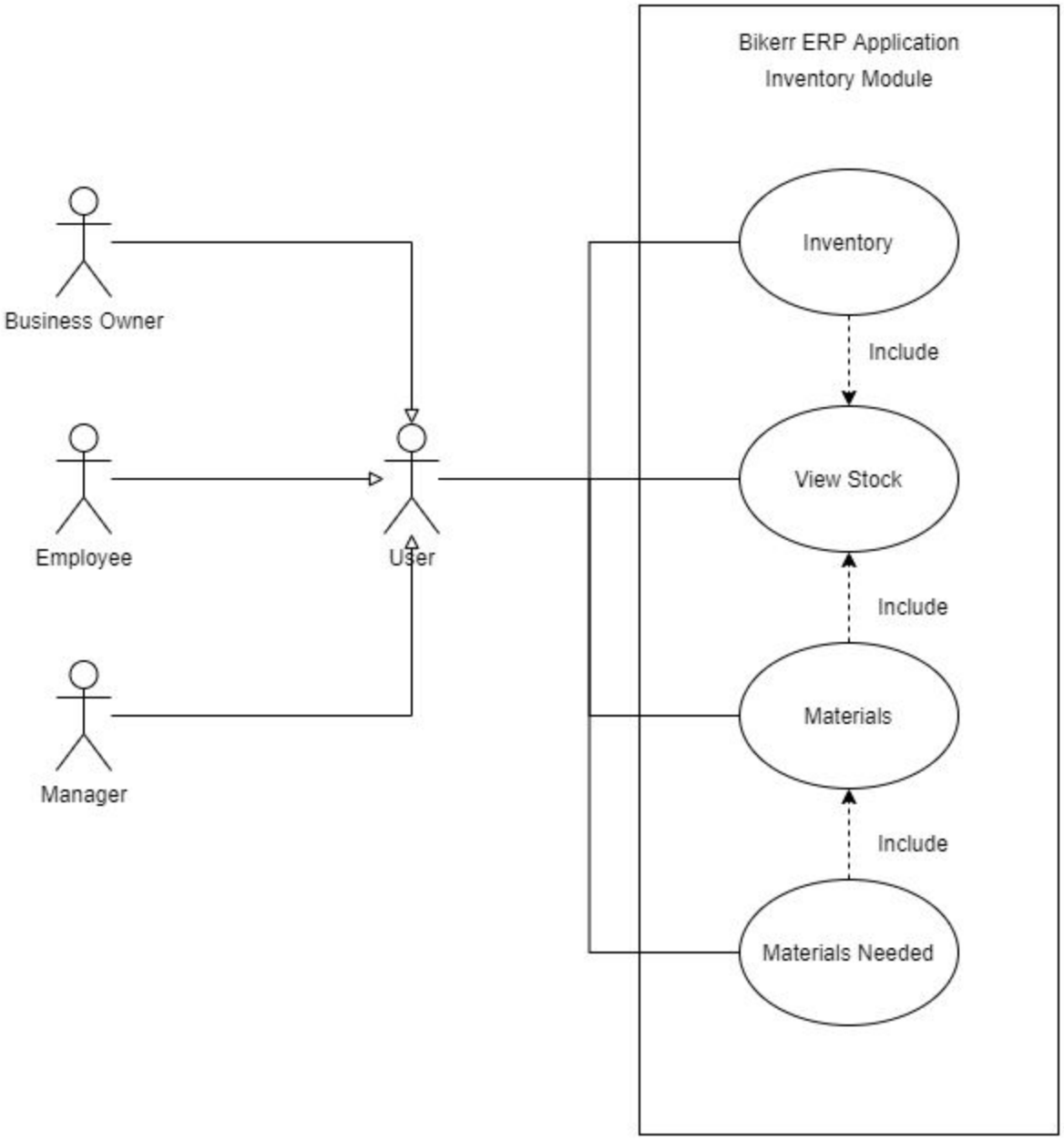

Figure 5: Use Case Diagram for Inventory and Material Management

- Submit Transaction: A user who has successfully logged in to the application should be able to submit a transaction for either inventory or materials, depending on the user's role and their needs.

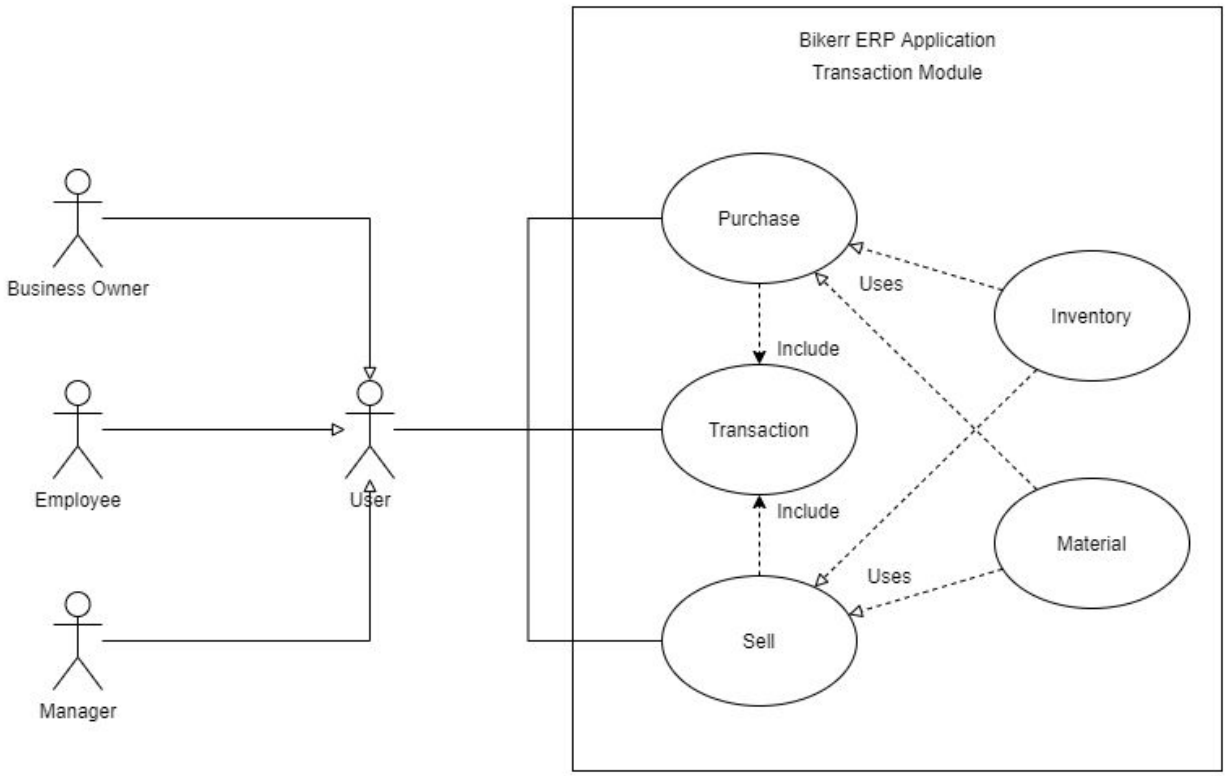

Figure 6: Use Case Diagram for Transaction Submission

3.4.3 Known Issues

The main issue with this view is that it focuses primarily on the user stakeholder. It does not separate the users into their individual stakeholder groups. It also does not include negative use-cases that may be relevant to the negative user stakeholder group.

3.5 The Logical View

The logical view consists of the different classes that makeup the Bikerr ERP system. Below is an updated class diagram of the system in its current state. This view is referenced from the Development viewpoint and in the case of the project outline (see below) the Customer viewpoint as well.

3.5.1 Models

The only two models used to represent the logical view are a simplified class diagram and a project outline diagram. The class diagram is used to represent how the different classes and components relate to each other. The reason we chose to do a simplified class diagram instead of a full one is because the contents of a class can change over time, and a simplified class diagram is easier to update on short notice. This prevents the diagram from going out of date quickly. The project outline exists

because it connects the pieces of the architecture in a way that the class diagram cannot -- it connects them to the user. This diagram allows us to view how the user will interact with the system and allow us to design around that as well.

3.5.2 The Class Diagram

This is a diagram of the different classes and components in the system and how they relate to each other. Each class is represented by its logical component in the implementation of the software. Since we are using ReactJS to code our front-end, it should be noted that the “classes” view in the diagram below are actually called components. Furthermore, this diagram only focuses on the front-end of the system.

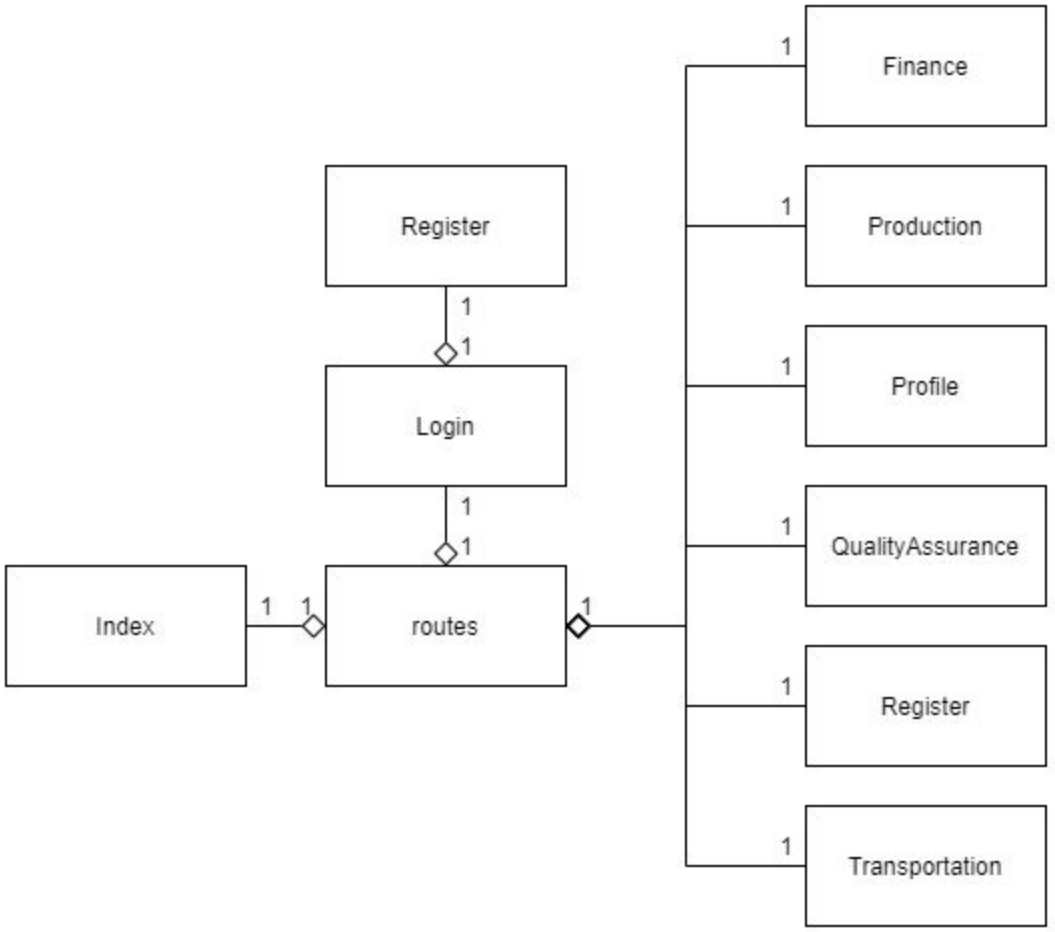

Figure 7: Simplified Class Diagram of Software Architecture

3.5.3 The Project Outline

The project outline is a logical representation of how each of the parts of the Architecture relate to one another, and in turn how the user relates to them. This is a useful model to have while developing the Bikerr ERP software.

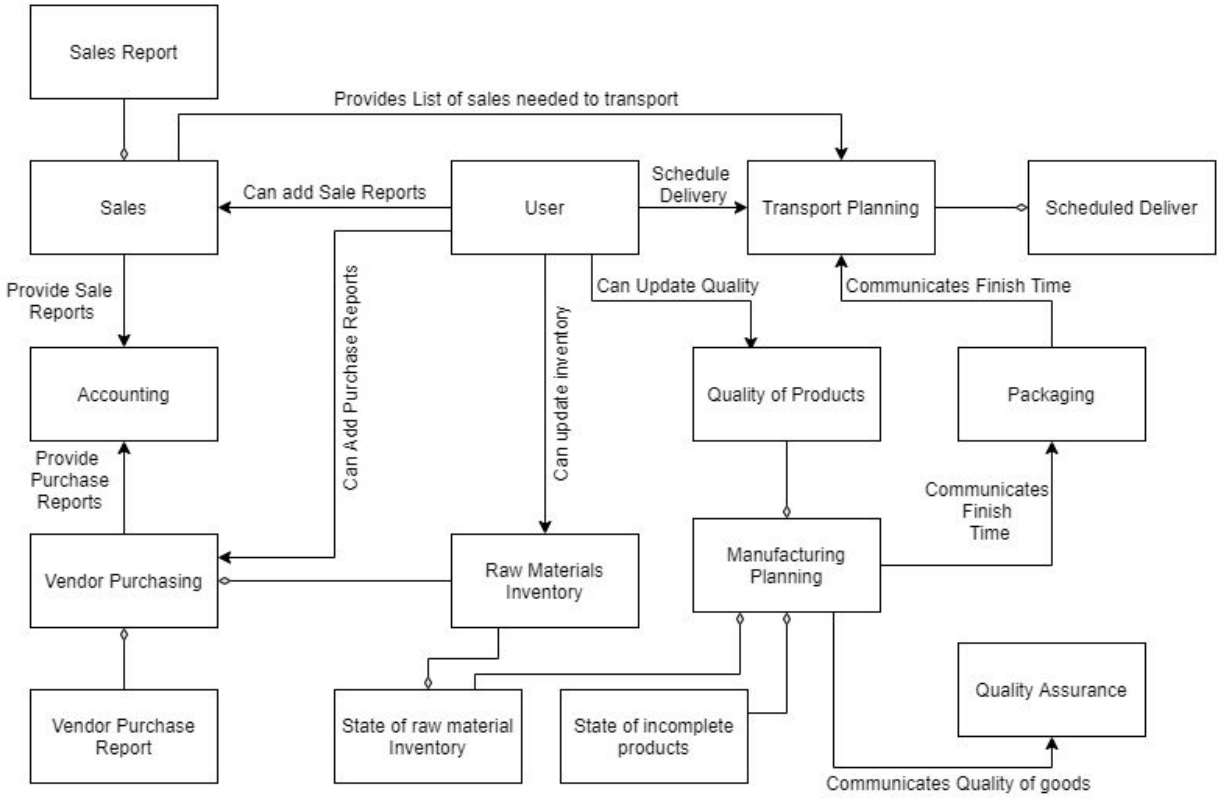

Figure 8: Project Outline of the Bikerr ERP Software

3.5.4 Known Issues

While the simplified class diagram is useful for writing the code and developing the software, it falls short of being able to understand how customers will use the software and interact with it. Furthermore, even though this is a simplified class diagram, it can still go out of date quickly as we update the software. Where the class diagram fails to deliver on user interaction, the project outline provides insight into how a given user may choose to use the system. The down side of this diagram is its complexity.

3.6 The Process View

This view is divided into processes. Each of these processes are main processes that the application should be capable of handling. The viewpoints from which we are looking in the Process View are the business and customer viewpoints.

3.6.1 Models

Since different stakeholders are responsible for multiple concerns in the process view, it is easier to model the processes based on concern. The two main concerns that we will model are Inventory concerns and Transaction concerns.

3.6.2 Inventory Concerns

Any user who has successfully logged in to the system should be able to view inventory, materials, or materials needed based on their system privileges. A check will be performed to see if they are allowed to view the requested material. If they are, they will be shown their request. If not, a message will be returned to them detailing why they were not able to view their request. Any software developer working on the system should be able to manage inventory with administrative privileges at the manufacturer's request

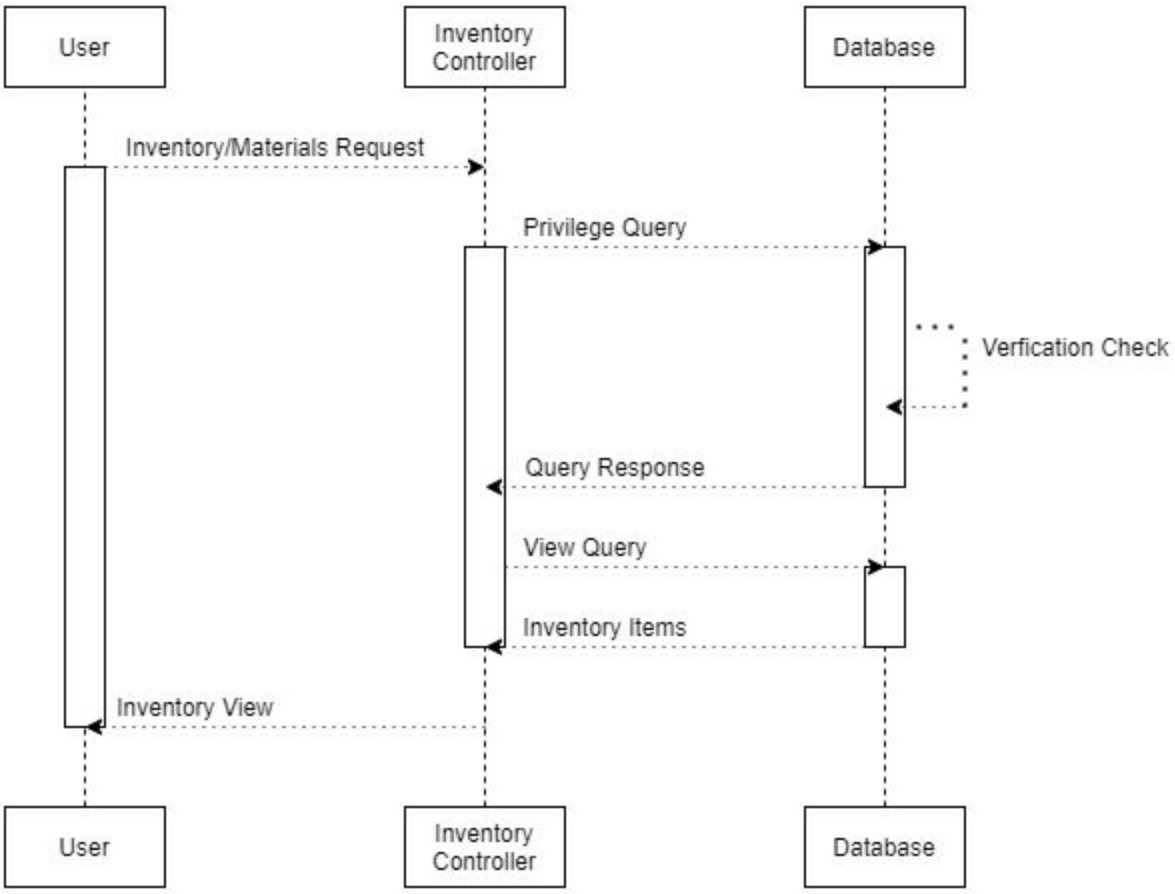

Figure 9: Process Diagram for Inventory and Materials Requests

3.6.3 Transaction Concerns

Any user who has successfully logged in to the system should be able to submit and process a transaction for either inventory, or materials depending on their system privileges. A software developer should be able to manage the transaction process at the manufacturer's request.

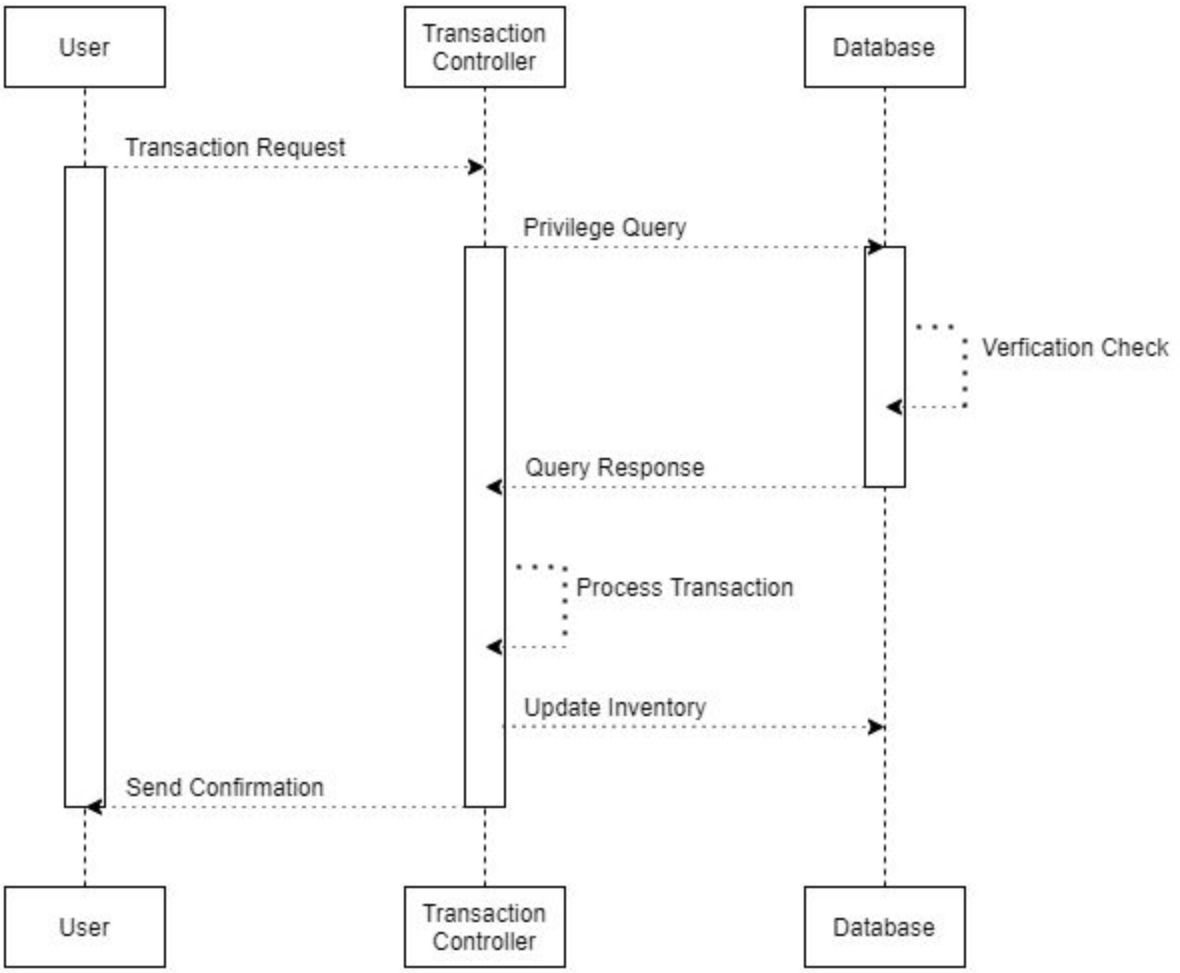

Figure 10: Submitting a Transaction

3.6.4 Known Issues

The main issue with the process view is that it does not include processes for negative users. All the processes described above take into consideration only those processes initiated by any users in the users stakeholder group, or by software developers. It could be reasoned that the check for administrative privileges could also detect negative users, however.

3.7 The Implementation View

This view shows how the software is structured as a whole.

3.7.1 Models

Given that this view focuses on source code implementation, there is only one model, which will emphasize the use of the Model View Controller architecture that is being used for this application. This is finally where we will see how the system will be protected against negative users. The only model that will be presented for this view will be the software model. This primarily concerns itself with the Development Viewpoint.

3.7.2 Software Model

The software model shows how the software is constructed. The diagram below details how the general pieces of the software are fit together to form the Model VIew Controller architecture. The arrows in this model go in both directions, and each module needs to be capable of communicating through the controllers. Both the models and database are considered as the model portion of the model view controller. The client view is considered the view, and the controller that connects the two is the controller.

(Model depicted on following page)

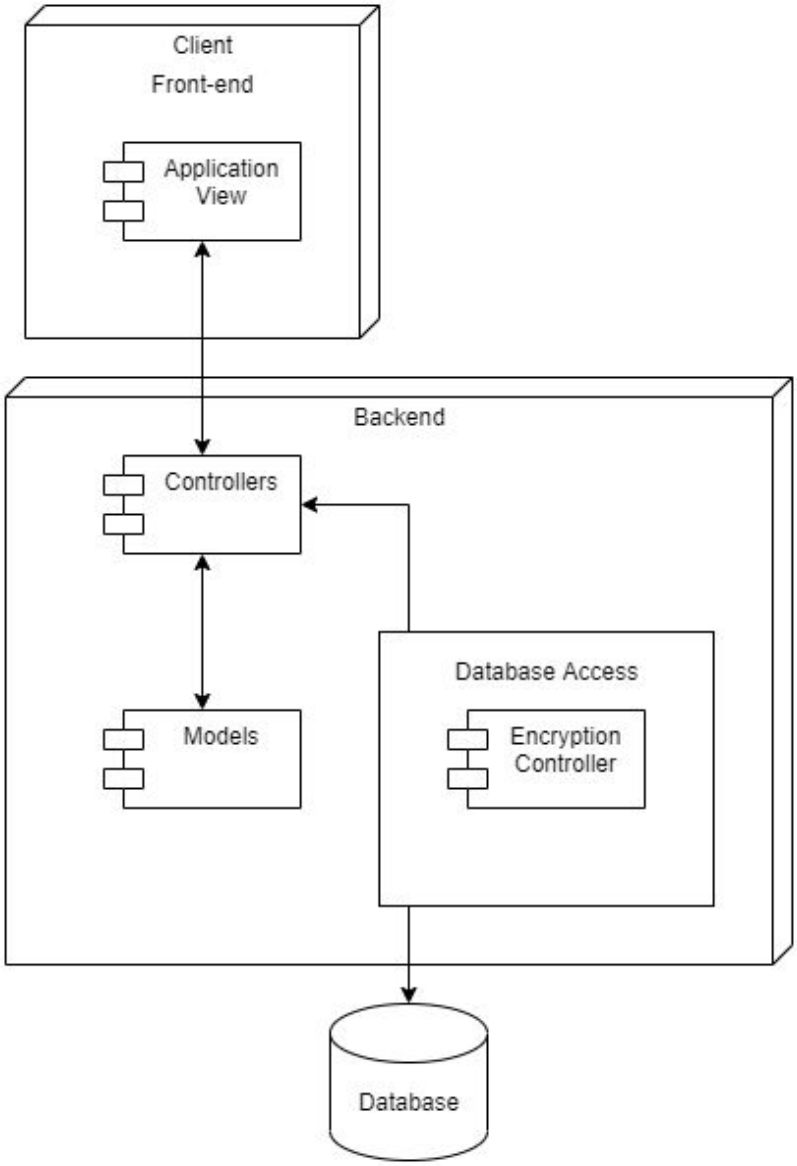

Figure 11: Software Model Implementation Diagram

3.7.3 Known Issues

The Implementation View gives a general view of how the system is constructed and how the front-end, back-end and database fit together. It does not, however, provide a more detailed look into how the classes are structured in the software system.

3.8 The Deployment View

The deployment view seeks to outline how we plan to deploy the Bikerr ERP application. This view is perhaps the simplest but is necessary for understanding how

the product is distributed to the client. This mainly focuses on the Development and Customer viewpoints.

3.8.1 Models

The deployment view takes only a single model, which is the deployment model. Each server or client application is represented by a bos, and are connected to each other with solid lines that represent network connections.

3.8.2 Software Deployment Model

This model focuses on how the application will be deployed. The main servers that act in this model are the front end and back end, along with the database. They connect, through docker, to create a local front end and back end running on the client's desktop. This is where the user interfaces with the application.

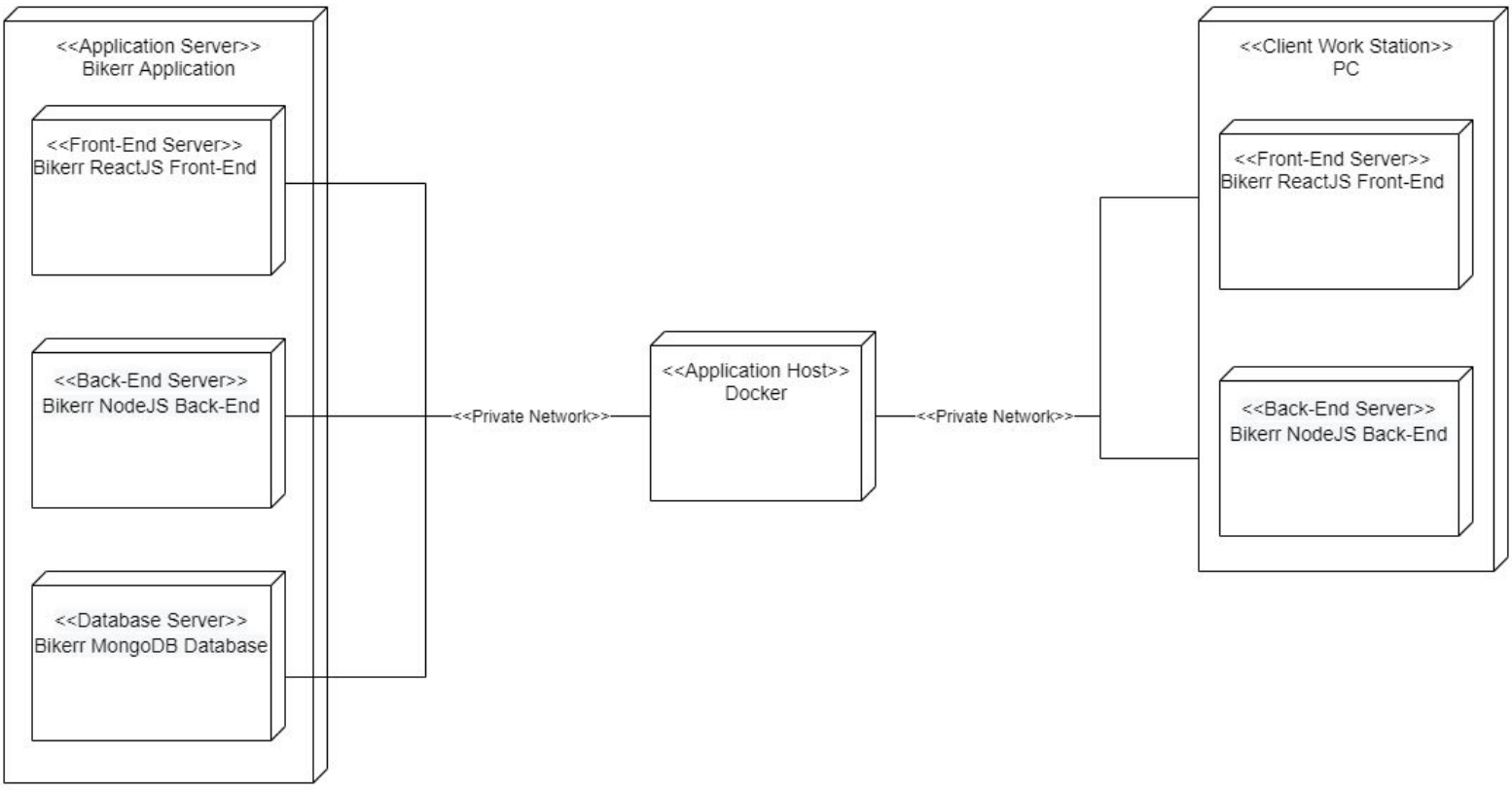

Figure 12: Deployment View Model

3.8.3 Known Issues

This model works well for its purpose. The only issue with it is that it does not show that when downloading and running the application for the Docker host, it requires two commands instead of only a single command. This is because the client needs to run both the front and back end from the client workstation. Otherwise, this model represents the reality of the deployment of the software with a fair amount of accuracy.

4. Consistency and Correspondences

This section seeks to identify the known correspondences and inconsistencies within our software architecture.

4.1 Known Inconsistencies

The largest known inconsistency lies with the grouping of the managers, employees, and business owners together under a single stakeholder group known as users. While this is extremely helpful for time and simplicity, it cannot describe the minute details that differ between each actual stakeholder. For example, the manager and employees should be able to manage materials when dealing with inventory concerns, but the business owners should not. This inconsistency is resolved via checking for privileges when sending a request.

4.2 Correspondences

There is a lot of correspondence between the use case view and the process view. This is because the processes being studied directly relate to the same concerns that the use-cases address. This is simply to be noted, as it will help define how the Bikerr ERP application is built.

The other main correspondence is in the similarity of stakeholder concerns between the managers, employees, and business owners. So much so, that these stakeholders were combined into a single stakeholder group called “Users” for discussion of viewpoints and views. While this does also create our main inconsistency, it also solves a lot of issues when creating the architecture.

4.3 Correspondence Rules

The main rule when considering the first correspondence to consider is that the use-cases and processes should be observed together in order to create a proper system that suits the needs of the users.

The other main rule to consider is the specific privileges assigned to each stakeholder. This will resolve the inconsistency that arises by merging them. By looking at the administrative privileges and employee privileges to determine what each user is allowed to do in the application.

5. References

[1] P. Kruchten, “Architectural Blueprints--The ‘4+1’ View model of Software Architecture,” *IEEE Software,* vol 12, no. 6, November, 1995. [Online serial]. Available: https://www.cs.ubc.ca/~gregor/teaching/papers/4+1view-architecture.pdf. [Accessed Feb. 22, 2021].