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

Contents

1. Introduction

This section serves to introduce the software architecture and our reasons for choosing it.

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.

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.

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.

2.2 Concerns

This section seeks to identify the main concerns that the stakeholders may have. The main concerns for the stakeholders are inventory, transactions, usage log, account security, and 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 1: Concern-Stakeholder Traceability Table

3. Viewpoints

This section details the various viewpoints that our architecture is concerned with. There are three main viewpoints to consider. They are the Use-Case Viewpoint, the Process Viewpoint, and the Implementation Viewpoint.

3.1 The Use-Case Viewpoint

This section seeks to describe the details of the Use-Case Viewpoint.

3.1.1 Overview

The Use-Case Viewpoint concerns itself with the various use-cases that a stakeholder may encounter. For this viewpoint, we will not include the negative use-cases that the negative users are concerned with.

3.1.2 Concerns

The main concerns that the use-case viewpoint addresses are the inventory, and transactions. This is because the majority of user interaction concerns itself with these two concerns.

3.1.3 Typical Stakeholders

The typical stakeholders of these concerns are the Manager, Employee, and Business Owner. While the negative user also has the transaction concern, negative use-cases will not be included in this viewpoint.

3.1.4 UML

This viewpoint will be represented by UML diagrams of Use-Case models.

3.1.5 UML Conventions

A Use-Case UML diagram will typically include the actors who act upon the system, a box representing the system itself, and the individual use-cases within that system and how they relate to each other.

3.2 The Process Viewpoint

This section seeks to describe the details of the Process Viewpoint.

3.2.1 Overview

The Process Viewpoint concerns itself with the different software processes that will occur based on actor interactions. Again, for this viewpoint, we will not include the action of the negative users.

3.2.2 Concerns

This viewpoint concerns itself with inventory management, transactions, administrative privileges, and the usage log. Each of these concerns is related to different processes that may occur in the ERP.

3.2.3 Typical Stakeholders

All of the previously identified stakeholders have at least one of the mentioned concerns. Again, we will not be focusing on negative users for this viewpoint however, as we will focus on what the system is intended to do.

3.2.4 UML

The Process Viewpoint will be represented using UML notation.

3.2.5 UML Conventions

The Process Viewpoint will be denoted by different actors in boxes above, each having their own “lane” for a given process. The UML model will show how each of these lanes interact with each other as the process plays out.

3.3 The Implementation Viewpoint

This viewpoint focuses on how the application will be constructed from a software point of view.

3.3.1 Overview

The Implementation Viewpoint concerns itself with how the software is put together. For this viewpoint, we will include the concerns of the negative user, as the software should be created with keeping user data safe from negative users.

3.3.2 Concerns

The main concern of the implementation viewpoint is account security. It also concerns itself with Inventory and Transactions through communication with the database.

3.3.3 Typical Stakeholders

All stakeholders are concerned with account security. Inventory and transactions are mainly concerns of the managers, employees, business owners, and negative users.

3.3.4 UML

Again, the Implementation viewpoint will use UML modelling to express its views.

3.3.5 UML Conventions

The model for the implementation view will concern itself with the separation of front-end and back-end, and will use 3D boxes to represent these. It will then show how major pieces of the software interact, and note where the concerns are handled in each.

4. Views

This section seeks to show the views governed by the viewpoints described in section 3. These views will be more comprehensive and include diagrams where appropriate.

4.1 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.

4.1.1 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.

4.1.2 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: Any 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: Any registered user of the Bikerr system should be able to use the login tool to access their account using username and password.

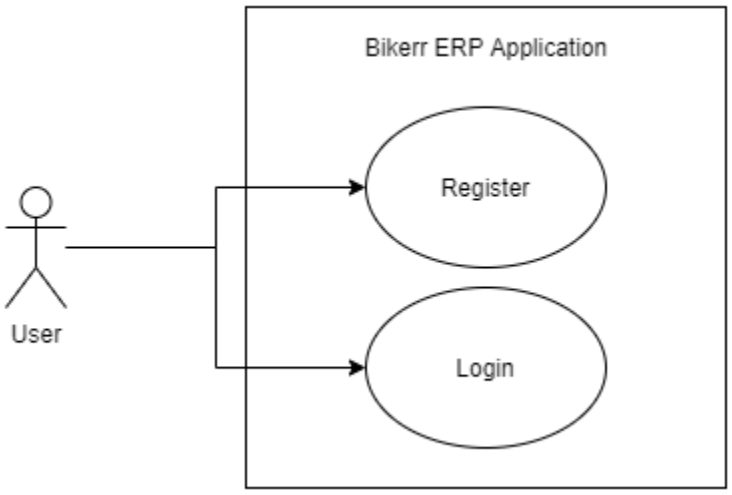

Figure 2: Use Case Diagram for Registration and Login

- View Inventory: Any 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: Any 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: Any 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.

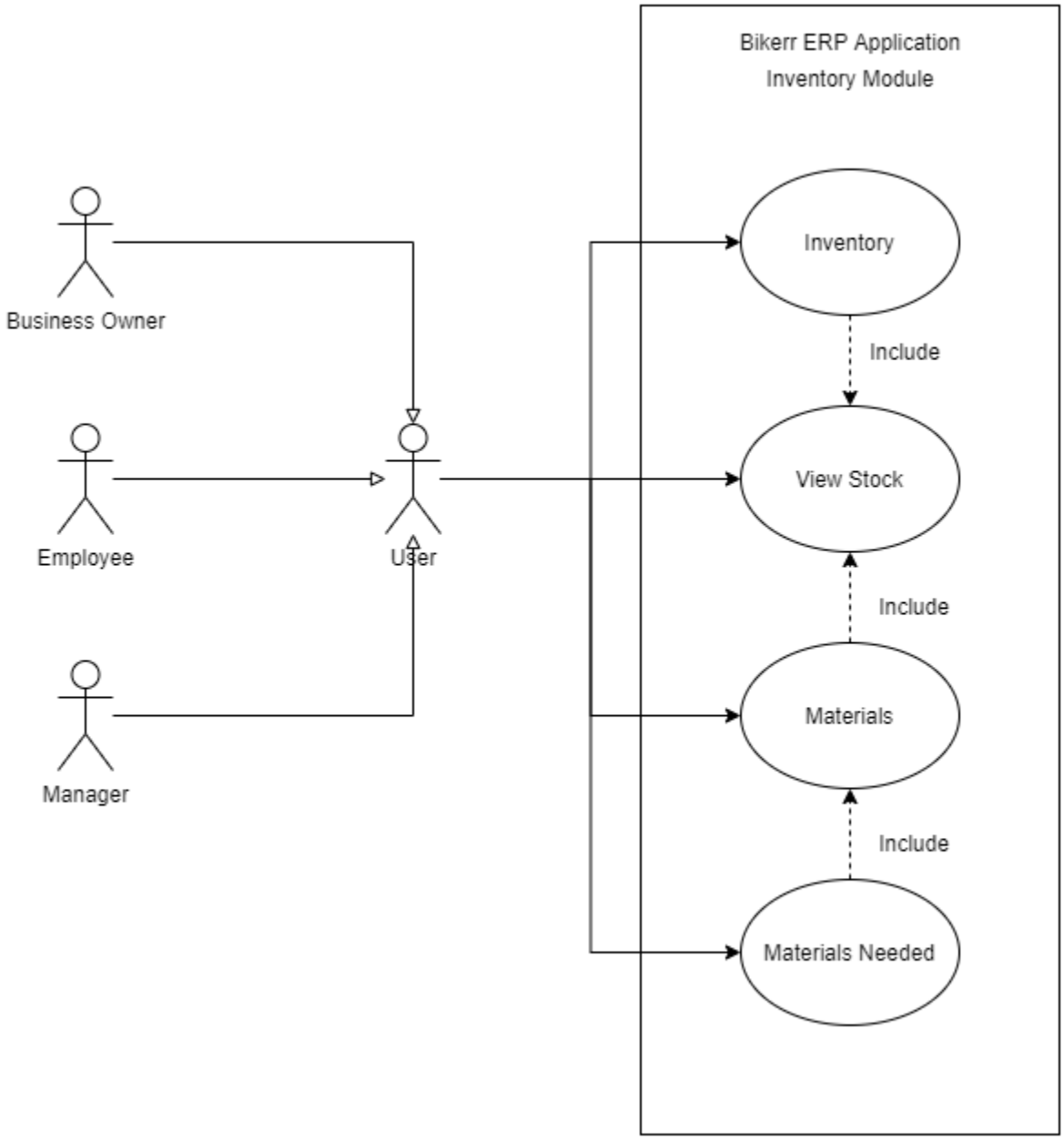

Figure 3: Use Case Diagram for Inventory and Material Management

- Submit Transaction: Any 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.

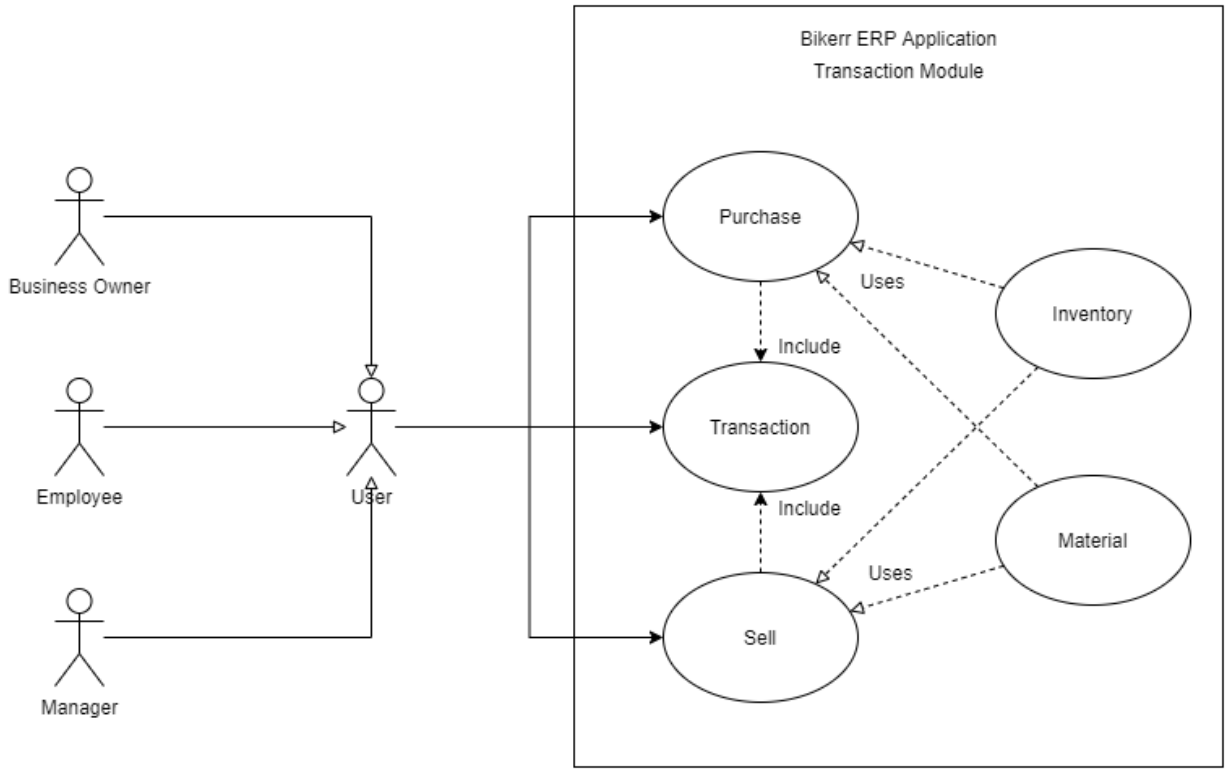

Figure 4: Use Case Diagram for Transaction Submission

4.1.3 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.

4.2 The Process View

This view is divided into processes. Each of these processes are main processes that the application should be capable of handling.

4.2.1 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.

4.2.2 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

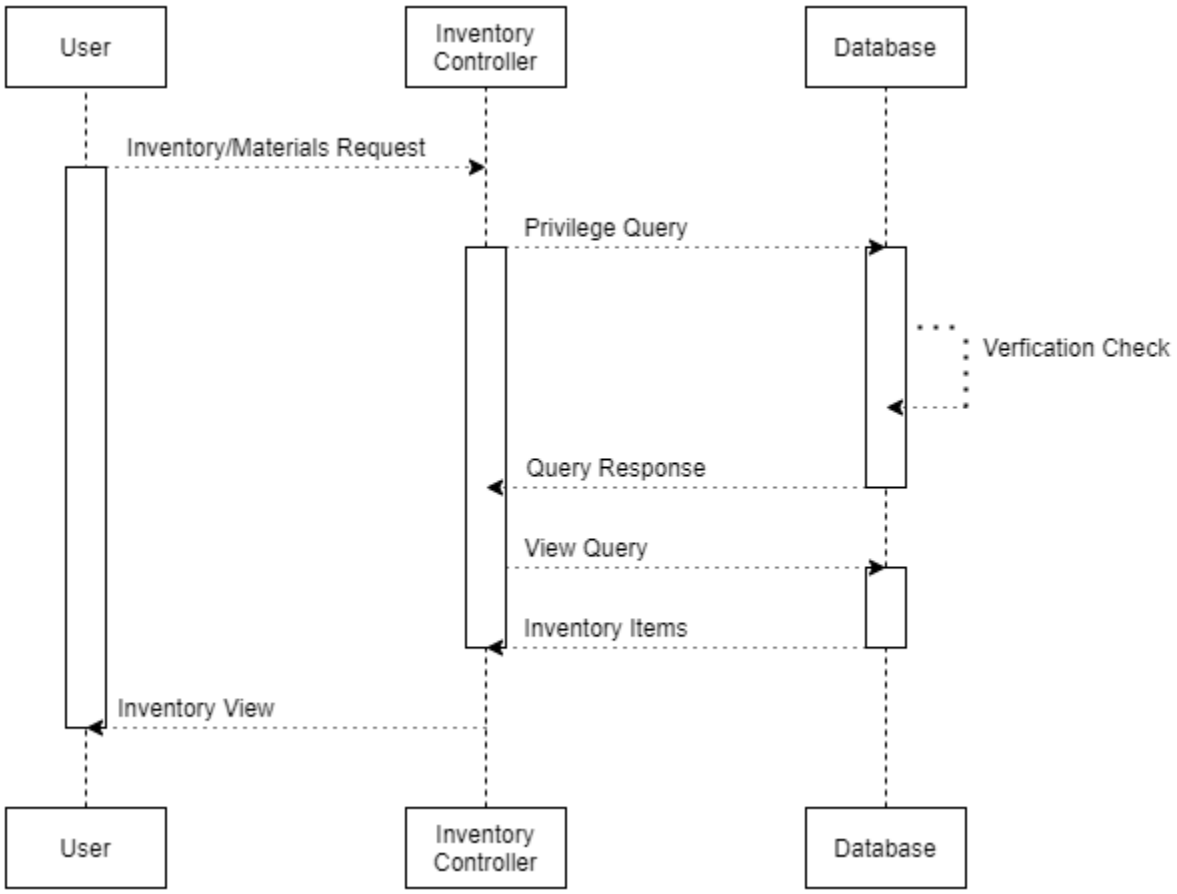

Figure 5: Process Diagram for Inventory and Materials Requests

4.2.3 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.

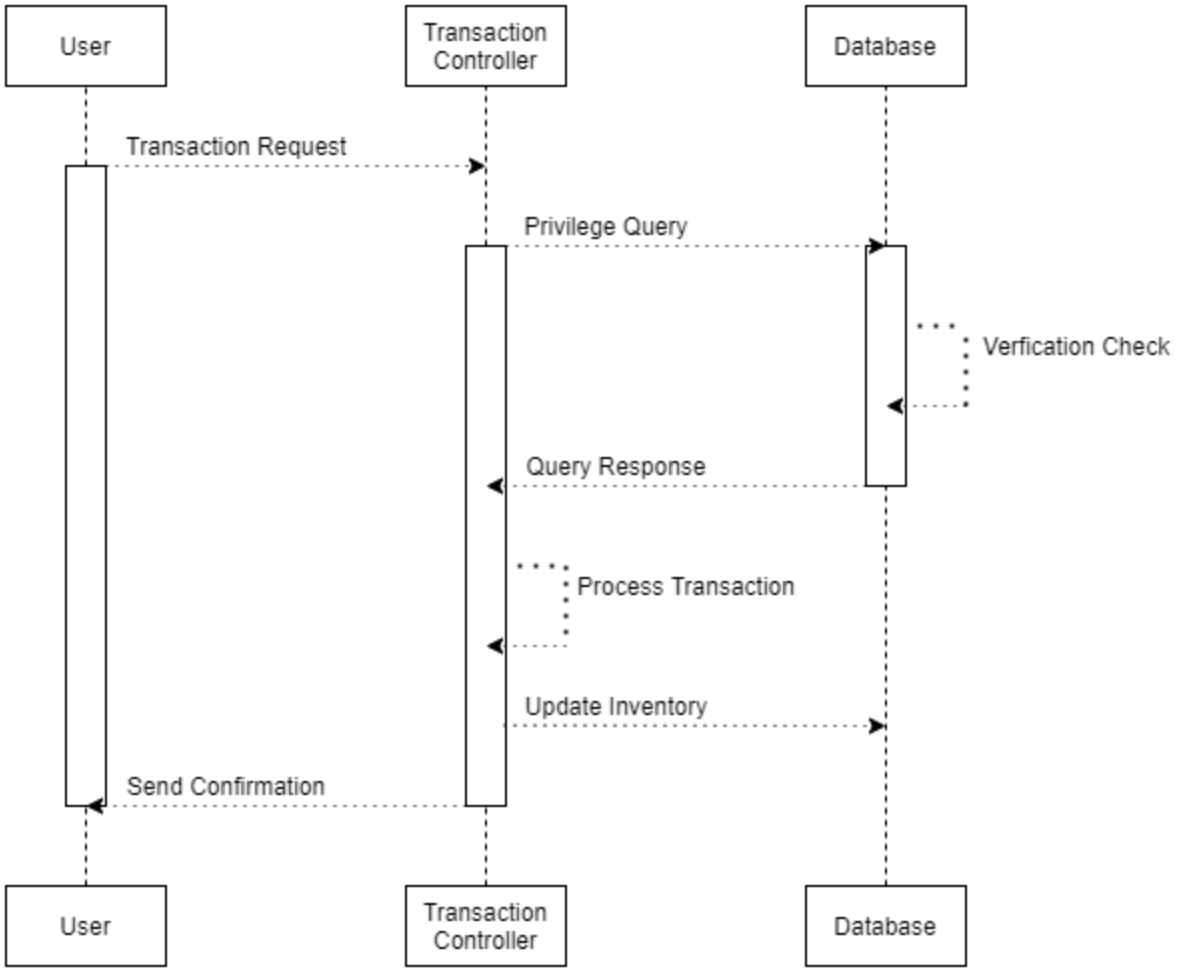

Figure 6: Submitting a Transaction

4.2.4 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.

4.3 The Implementation View

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

4.3.1 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.

4.3.2 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.

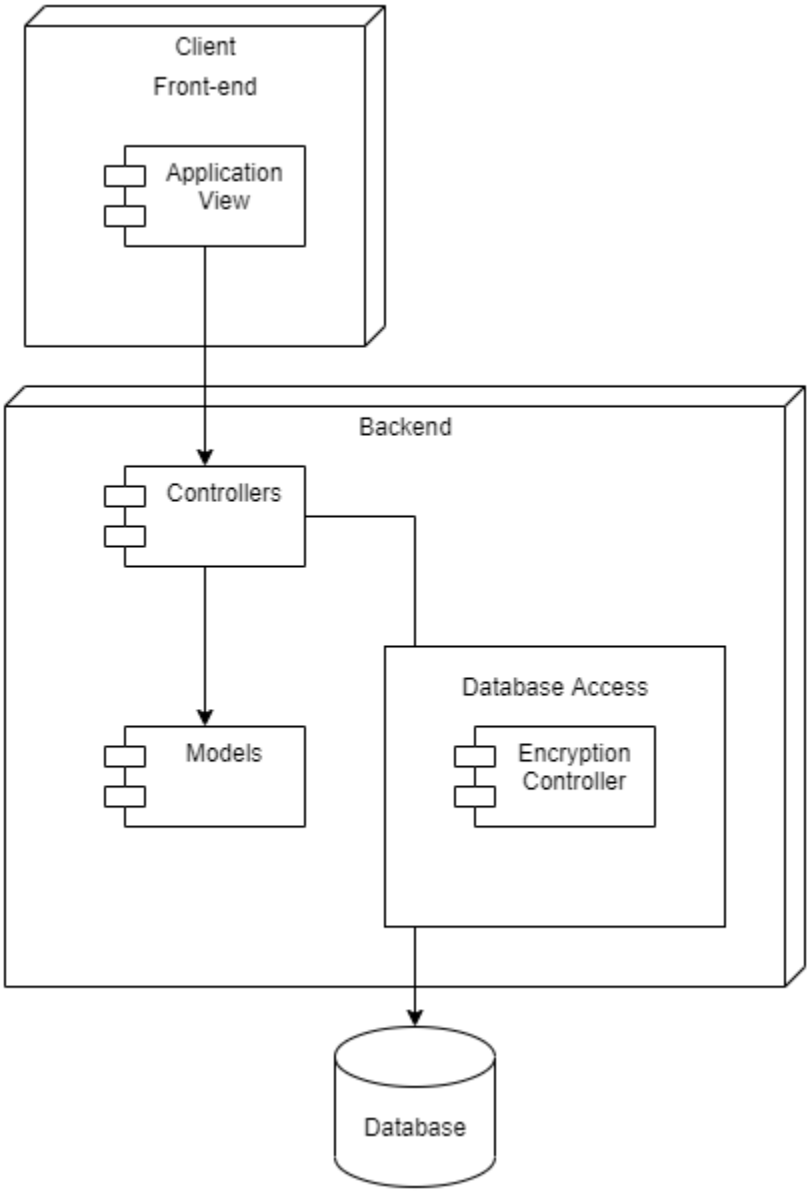

Figure 7: Software Model Implementation Diagram

4.3.3 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.

5. Consistency and Correspondences

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

5.1 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.

5.2 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.

5.3 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.