Reuse Plan

System reuse

The system is tailored to meet the client's specific requirements, however with a few modification the application could be reused by other parties with similar process.

With the vast employment of standardized components, the system is easy to deploy and re-deploy.

Component reuse

Application Frameworks

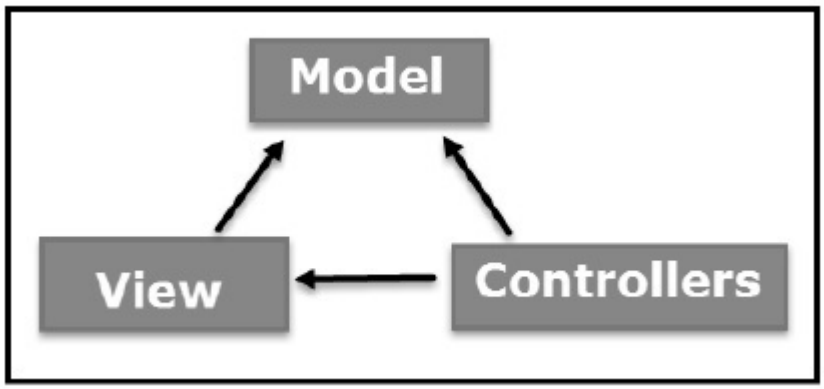
Component	Technology	Advantage
Back-end	HTTP Framework: Spring Cloud & Spring Boot	• Quick development • Tested • Team have experience
Back-end	ORM Framework: MyBatis	• Simplicity • Team have experience
Back-end	Authentication Framework: Shiro	• Opensource • Team have experience
Database	MySQL	• free • opensource • widely used • Team have experience
Front-end	Framework: React Component Library: Bootstrap	• Declarative • Component-Based • Team have experience
Middle-ware	RESTful API	• the message format for exchanging information between front-end and back-end)

High level design patterns

Model-View-Controller

The **Model-View-Controller (MVC)** is an architectural pattern that separates an application into three main logical components: the **model**, the view, and the controller. Each of these components are built to handle specific development aspects of an application. MVC is one of the most frequently used industry-standard web development framework to create scalable and extensible projects.

MVC Components



Model

The Model component corresponds to all the data-related logic that the user works with. This can represent either the data that is being transferred between the View and Controller components or any other business logic-related data. For example, a Customer object will retrieve the customer information from the database, manipulate it and update it data back to the database or use it to render data.

View

The View component is used for all the UI logic of the application. For example, the Customer view will include all the UI components such as text boxes, dropdowns, etc. that the final user interacts with.

Controller

Controllers act as an interface between Model and View components to process all the business logic and incoming requests, manipulate data using the Model component and interact with the Views to render the final output. For example, the Customer controller will handle all the interactions and inputs from the Customer View and update the database using the Customer Model. The same controller will be used to view the Customer data.

Resource

Software Engineering, 7th edition. Chapter 18

Spring Cloud

https://spring.io/projects/spring-cloud

Spring Boot

https://spring.io/projects/spring-boot

MyBatis

https://mybatis.org/mybatis-3/getting-started.html

Shiro

https://shiro.apache.org/

MySQL

https://www.mysql.com/

React

https://reactjs.org/

RESTful API

https://restfulapi.net/

MVC

https://www.tutorialspoint.com/mvc_framework/mvc_framework_introduction.htm