**High level Architecture**

**P03:MunasibMall.PK**

**team member names & ids**

|  |  |
| --- | --- |
| **Student ID** | **Name** |
| **22100168** | **Daniyal Mumtaz** |
| **22100134** | **Abdur Rehman masood** |
| **21100034** | **Waliullah aitemad** |
| **21100161** | **Muhammad Muzammil Khan** |
| **22100199** | **Waqar Ul Haq Khatana** |

**Table of Contents**

[1.](#_gjdgxs) Introduction 3

[2.](#_30j0zll) System Architecture 4

[2.1 Architecture Diagram 4](#_1fob9te)

[2.2 Architecture Description 5](#_3znysh7)

[2.3 Justification of the Architecture 6](#_2et92p0)

[3.](#_tyjcwt) Risk Management 7

[3.1 Potential Risks and Mitigation Strategies 7](#_3dy6vkm)

[4.](#_1t3h5sf) Tools and Technologies 8

[5.](#_4d34og8) Hardware Requirements 9

[6.](#_2s8eyo1) Who Did What? 10

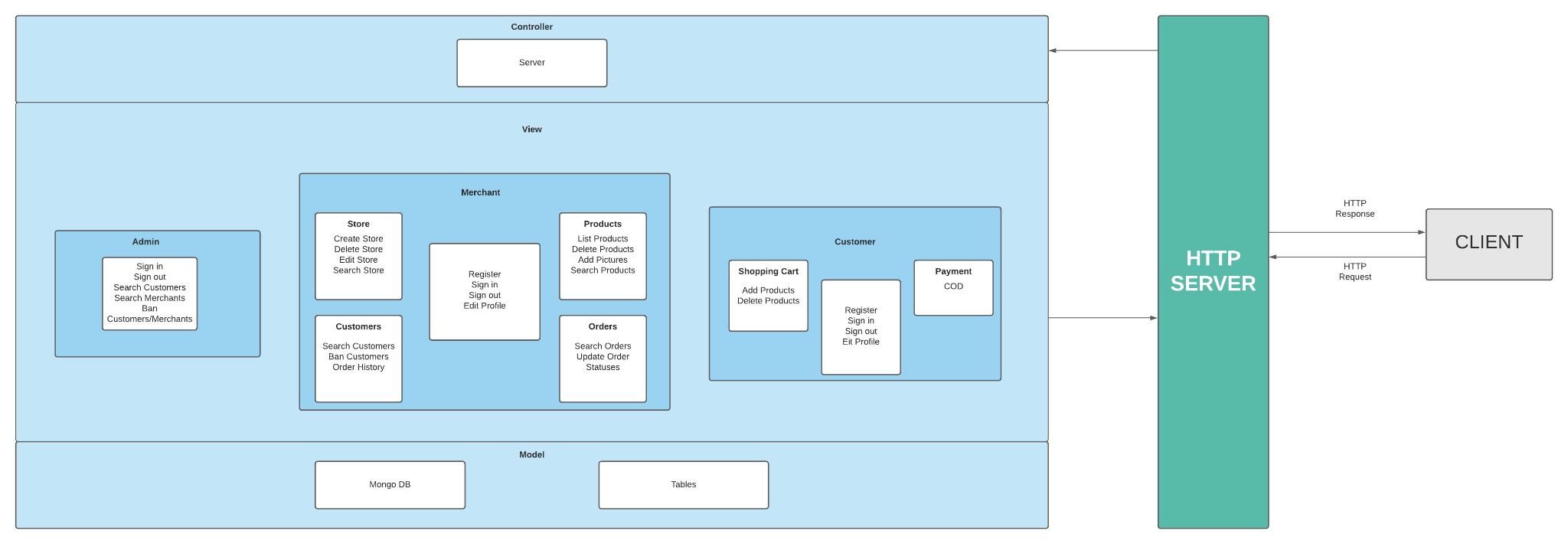
[7.](#_17dp8vu) Review checklist 10

# Introduction

Munasib Mall is a mobile e-commerce application. It will allow different merchants to have multiple dynamic web stores and customers will be able to buy from them on this platform. Merchants will have complete control over their store(listing products, removing products) and will be able to customize their store according to their wants. They will also be given some customizability in designing their store page. Customers will be able to order products of all the merchants/stores. Customers will be able to look for specific products across various stores. Orders will be sent directly to the merchant who will then handle the delivery. Payment options like easy paisa or COD will be shown to the customer who can choose their preferred method when purchasing. The system aims to provide a platform that gives more autonomy and the ability to handle orders to the merchants. Reviews and ratings will help the customer choose the right merchant.

# System Architecture

## Architecture Diagram



## Architecture Description

**Model:**

The model consists of MongoDB and Tables. This layer will store and process data. The model will interact with the controller as the controller will be able to perform CRUD operations on the model layer.

**View:**

Our view layer is the frontend. React Native will be used for development of the frontend. The view layer will interact directly with the client. This layer will provide an interface between the client and the controller. The clients will send http requests to the controller via the view layer and the controller will send the http response to the client via the same layer.

**Controller:**

Our controller is the server which is going to be developed using Node JS and Express JS. This layer will provide an interface between the model and the view layer. It will process the http request and generate the required http response.

## Justification of the Architecture

**Pros:**

* Views and controllers can be easily be added, removed, or changed.
* Views can be added or changed during execution.
* User interface components can be changed, even at runtime.
* Scalable, easier to add more servers.
* React provides optimal performance for view.
* Easier to make reusable code.
* Front-end libraries and plugins support.
* Modular architecture.
* Easier to manage development.
* Easier to update and maintain the code.

**Cons:**

* Views and controllers are often hard to separate.
* Frequent updates may slow data display and degrade user interface performance.
* MVC style makes user interface components (views, controllers) highly dependent on model components.
* Clean separation between layers is often hard to achieve.
* Harder for developers to ensure consistency.

**Justification:**

With the MVC architecture model, we feel it would be easier to work parallely on the application. We would need to ensure all the layers are consistent in use and coordinate for that. Other than that, it is easier for us to autonomously work on separate layers. The MVC model would also make it easier for us scale up in the future and offer different updates easily when needed. With thousands of potential users, load balancing is important which we will be able to implement using our architecture.We will be following the scrum model, potentially adding new features in some modules previously developed.

**How it helps non-functional requirements:**

The modifiability of the system will be modeled with Model View Controller (MVC). This will allow maintainability and more structural means for a developer to easily make changes to the system when it is required.In the case of failure a new server can be deployed within 10 minutes hence making the system up and running within the time prescribed in the non functional requirements. The low coupling along with high cohesion will allow the developers to add new features and write new code, without compromising the existing ones. This makes the system highly extensible. Since the servers can be replicated across many servers, the system will be able to manage 5000 requests. Moreover, The performance of the whole system aims to ensure ease of comfort to have the site load within 5 seconds.

# Risk Management

## Potential Risks and Mitigation Strategies

|  |  |  |
| --- | --- | --- |
| **Sr.** | **Risk Description** | **Mitigation Strategy** |
|  | Staff Illness | The work of the staff can overlap so that upon unavailability of a member, the group can reorganize and divide the work of the missing member in order to complete on time. |
|  | Product Competition | In order to increase the profits from the project and minimise the effect of competitors, better marketing strategies can be implemented, for example, more attractive discounts and concessions in the case of an e-commerce website. |
|  | Time Constraints of Deliverables | The team will manage time and track the progress of each deliverable by obtaining feedback from the members on a daily basis. |
| 4. | Requirement Change Risk | The customer will be informed that if the requirements change in the future, more time will be required to complete the project, as modifying the project requires time. |
| 5. | Economic Risks (Policy Changes of external organisations) | If there is an external policy change, for example, changes in the policies of payment applications (Easypaisa app) and methods that are recommended on the website. The customers will be informed of the updated policies and new policies for the payment option will be designed, for example if easypiasa app introduces a policy of delayed payments, the customers will be required to pay in advance of the product ordered. |
| 6. | Server Risk | If the server used for the project is low-powered and upon increase of website traffic, there are delays, the management will be suggested to purchase a high-powered server. |
| 7. | Skills Risk (Some members not familiar with the tools and technologies being used) | The tools and technologies will be discussed in detail with the staff and time and other group members with experience will be allocated to the members who are not familiar with the tools decided. |
| 8. | End-User Risk (end-users have problems while using the software) | The end-users will be surveyed on a regular basis so that if they are facing difficulties while using the application, the required changes will be made. |
| 9. | Operational Risk (If there is a problem in the working of the software e.g. unexpected crashing of software) | The problem will be identified by the team, and stakeholders will be informed of the unexpected problem. The team will work on solving the problem after giving a possible timeframe to the stakeholders. |
| 10. | Faults in Reusable Components of Software | The reusable components of the software will be kept in check by the team, and upon a fault occurring, the team will rectify it so that it can be reused when required. |

# Tools and Technologies

1. MongoDB 5.0.

MongoDB is a cross platform, no-SQL database program. MongoDB is easily scalable and supports all functions of modern database systems.

1. React Native 0.64

React Native is a React.js powered , JavaScript based mobile app framework which allows cross platform development.

1. Express(.js) 4.17.1

Express.js is a back-end web application framework for Node.js.

1. Node.js 16.7.0

Node.js is a back-end JavaScript runtime environment that executes JavaScript code outside a web browser.

1. Heroku

Application will be deployed in Heroku which is a cloud Platform as a service(PAAS).

1. Trello

Trello is a collaborative tool to manage the communication in this project.

# Hardware Requirements

Laptops for the development of the app will consist of a minimum of:

1. 8GB of Ram with 64 bits operating system
2. 512 GB of hard drive or SSD
3. GPU

5 Laptops of the team will be used, by the team of 5 members.

The app will be deployed on a server with the following specifications:

1. The CPU architecture will support 64-bit operating systems.
2. It will have a ram of 8gb
3. It will have a disk of 64gb.

# Who Did What?

|  |  |
| --- | --- |
| **Name of the Team Member** | **Tasks done** |
| Daniyal Mumtaz | Risk Management/ Hardware Requirements/System Architecture Diagram + Description |
| Abdur Rehman Masood | Risk Management/Hardware Requirements/System Architecture Justification |
| Waqar ul Haq Khatana | System Architecture Diagram + Description |
| Muzammil Khan | Introduction + Tools and Technologies |
| Waliullah Aitemad | System Architecture Justification |

# Review checklist

Before submission of this deliverable, the team must perform an internal review. Each team member will review one or more sections of the deliverable.

|  |  |
| --- | --- |
| **Section** **Title** | **Reviewer Name(s)** |
| Introduction + Tools and Technologies | Daniyal Mumtaz |
| System Architecture Diagram | Abdur Rehman Masood |
| Hardware Requirements + System Architecture Description | Muzammil Khan |
| Risk Management | Waliullah Aitemad |
| System Architecture Justification | Waqar ul Haq Khatana |