HIGH LEVEL ARCHITECTURE

P02:BAICHDAY

<TEAM MEMBER NAMES & IDS>

STUDENT ID	NAME
23100181	NASHIT IFTIKHAR
23100157	MOIZ NAFEY
23100334	SILAL ANWAR
23100089	MAHAD MUBASHIR BEG
23100225	MUHAMMAD ARSLAN ULLAH TARAR

TABLE OF CONTENTS

1. Introduction

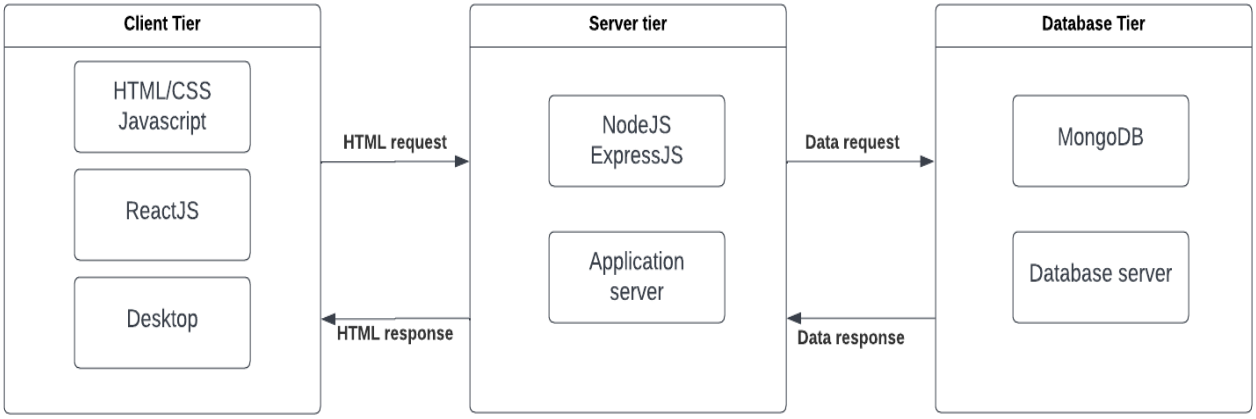
Pakistan is a huge country with a population numbering in the millions, yet the GDP of Pakistan remains rather low as compared to its massive population. We aim to facilitate the people of Pakistan by presenting our auction system. Our software will allow people to auction off their belongings to interested parties. Currently there exists no such platform that supports auctioning as a mechanism. In the status quo, people must spend valuable time and effort searching for customers to purchase their goods at a decent price. We aim to bridge this divide and bring value to the economy of Pakistan by solving this problem.

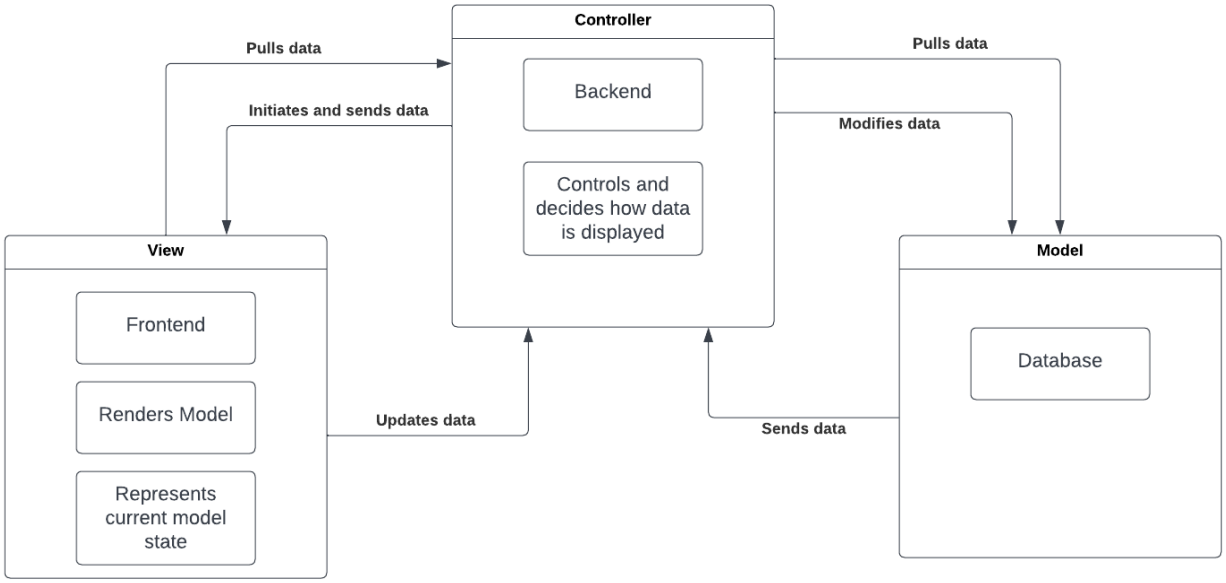
This software will target both businesses and individuals in the Pakistani context. Small scale businesses currently do not have a platform to bid for resources, and shipments. A small-scale mobile phone retailer finds it very inconvenient to currently bid for a shipment of mobile phones and similarly, repair shops face the same issue with car spare parts. Our auction system will allow wholesalers to enter their products for retail vendors to bid on. This has the potential to facilitate both wholesalers and the retail industry of Pakistan. Likewise, on a more individual scale, people with valuable assets are unable to put up their goods for the best price and must sell at the highest customer they manage to find. Our model will allow these users to list their belongings and allow bidding on it for the user to find the best price they can get from their belongings.

Our software will function similar to an ecommerce marketplace but with the added functionalities of timed biddings, scheduling of bids and other functionalities that will make the auction system a good and worthwhile experience for the users we target.

2. System Architecture

2.1 Architecture Diagram





2.2 Architecture Description

Our system is based on two architectural patterns which are the client-server and the model-view-controller.
The client server pattern is a very famous architectural model which is often used for web applications and websites. There are two types of client server models; thin client (logic heavy client) and thick client (logic heavy server). Our software will rely on the thin client server architectural model. This model in our system symbolizes the divide between the front end and backend of our software. The frontend client will be light in terms of logic and will defer to the backend server in terms of computations. On the other hand, the backend will be responsible for most of the heavyweight computations. The backend will connect to the database and retrieve data for the frontend to display.
The frontend on the other hand has the architecture of the Model-View-Controller which is a three-tier architectural pattern. The Model-View-Controller divides the program logic into three interconnected components that are responsible for handling their own responsibilities. Our frontend in this pattern is divided into three portions namely, the model, the view and the controller. The View is the components and the screens that will be shown to the user while the controller will be the segment in contact with the backend. It will receive the data and pass it on to the model for processing. The model will convert it into usable forms for the components of the view to render.

2.3 Justification of the Architecture

Pros of using Client Server Architecture and Model-View-Controller:

1. Firstly, the centralized network has complete authority over all the processes which is very necessary in our case since we are the trusted third party between the seller and the bidder.
2. The users can access the application at any time.
3. It provides a good user interface and makes it easier to navigate and find files.
4. Resources can be easily shared across various platforms.
5. The reason for using MVC is that it makes it easier to navigate between different views.
6. It allows for a faster development process since different developers can work on different views at the same time.
7. Modification in one view does not affect the whole front end.
8. MVC is a good choice for web applications as view and controller are naturally separated.

Cons of using Client Server Architecture and Model-View-Controller:

1. If the sever goes down for any reason the whole architecture is affected.
2. It has high operation cost as the view could be over burdened with update requests if the model keeps on going with frequent changes.
3. Views like graphic displays can take a longer time to make.
4. Traffic congestion can take place if there are too many users at one time which will slow the whole network.
5. Duplicated code is used between common views.

3. Risk Management

3.1 Potential Risks and Mitigation Strategies

Sr.	Risk Description	Mitigation Strategy
1.	Key staff are ill and unavailable at critical times.	Reorganize team so that there is more overlap of work and people therefore understand each other's jobs. We will ensure that at least one person working at the back end is familiar with working on the frontend while at least one person working on the front end is familiar with working on the back end
2.	The time required to develop the software is underestimated.	Investigate use of a program generator and how it can be used in our project. Websites such as Anima for Figma can be used to generate front end code for our application and it can be considered in case we are facing time constraints
3.	There will be a larger number of changes to the requirements than anticipated.	Derive traceability information to assess requirements change impact and maximize information hiding in the design. Make sure that your team is aware of the exact requirements of the project. Talk to them about the change control process and how they should only work on what is approved.
4.	The database used in the system cannot process as many transactions per second as expected.	Investigate the possibility of buying a higher-performance database.
5.	Running into technical difficulties. Data security, information privacy, software integration, and compliance are some areas where we are likely to run into unpredictable problems or dead ends.	Our team will make sure that we test and try to perform all these tasks well before the deadlines so that in case such problems do arise, we have ample time to figure out the problem. Similarly, we will also have alternatives in mind for example if we are unable to deploy our application on a particular platform, we will have a back up to cover up the mishap.

6.	Team is struggling with productivity, which can happen because of delays or burnout of individuals.	The productivity of our team can be ensured by firstly creating a well-paced project plan that will help mitigate burnout and avoid stress. We will also ensure that we are communicating effectively with each other about project details or problems. Lastly, we will try our best to help each other out with their tasks to ensure that everyone stays motivated.
7.	Underwhelming or mediocre user reception to our end product due to bugs or bad UX of the project.	It could be that the end product does not receive a good response from our users because of various reasons resulting in failure of the project. To identify and mitigate these risks, we can test our software in advance through beta testing. We can also send surveys to our users and conduct focus groups to gather more information about the users.
8.	Required training for team members is not available.	To ensure all team members are proficient in the technological platform that is being used, we will make training a priority allocating the time and resources to make sure the training is comprehensive and well-received. Weekly training meetings with the team will be held to ensure this.
9.	Poor quality code is written when completing the project making it difficult to read, review or change.	The team will ensure to maintain a high standard for their code by doing code reviews done by other team members, following clear coding standards and guides and testing all of the written code to ensure it conforms to the set standard.
10.	Failure to address priority conflicts and problems that arise during the development phase.	Failure to address conflicts that arise during development can lead to them being piled up which can cause a big problem when integrating different parts of the code. To ensure such problems are resolved at their earliest, in each sprint a team member will be assigned the responsibility to address and resolve them by the end of the sprint.

4. Tools and Technologies

Development:

- Coding Platform – VS Code (version 1.72.2)
- Front-End – Reactjs (version 18.2.0)
- Back-end – Nodejs and Expressjs (version 18.10.0)
- Database – MongoDB (version 6.0)
- API Testing – Postman (version 9.31.0)
- System Testing – Selenium (version 4.5.0)
- Version Control Git – GitHub (version 2.9.3)
- Browser for development tool and testing – Google Chrome (version 106.0.5249.103)

Deployment:

- Front-End – Netlify (version 12.0.8)
- Back-End – Heroku (version 7.60.1)
- Database – MongoDB Atlas (version 6.0)

5. Hardware Requirements

Development Machines:

- RAM – 4GB DDR4
- Processor – Intel® Core™ i3-10110U
- Processing power – Dual Core @ 2.10 GHz CPU, Max Turbo @ 4.10 GHz
- Storage and Hard drive – 20 GB available on the disk
- Architecture – 64-Bit

Deployment Servers:

- Storage – 256 GB NVMe SSD (for high performance)
- Processor – Intel® Core™ i7-10610U
- Processing power – Quad Core @ 1.80 GHz CPU, Max Turbo @ 4.90 GHz
- RAM – 16 GB DDR4
- Architecture – 64-Bit

6. Who Did What?

Name of the Team Member	Tasks done
Moiz Nafey	Tools and Technologies, Hardware Requirements
Nashit Iftikhar	Risks and Mitigation
Silal Anwar	Architectural Description, Introduction
Mahad Mubashir Beg	Architecture Justification
Muhammad Arslan Ullah Tarar	Architecture Diagram

7. 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)
Risks and Mitigation	Moiz Nafey
Tools and Technologies, Hardware Requirements	Nashit Iftikhar
Introduction, Architecture Justification	Muhammad Arslan Ullah Tarar
Architecture Description	Mahad Mubashir Beg
Architecture Diagram	Silal Anwar