Implementation Document

for

Real-Estate Broker

Version 1.0

Prepared by

Group: 14 **Group Name: Dangling Pointers**

Chinmay Hiran Pillai	200298	chinmay20@iitk.ac.in
Shivang Pandey	200941	shivangp20@iitk.ac.in
Armeet Singh Luthra	200185	armeet20@iitk.ac.in
Arth Banka	200191	arth20@iitk.ac.in
Vaishali Rawat	201080	vaishalir20@iitk.ac.in
Utpal Dwivedi	211132	utpaldwi21@iitk.ac.in
Praveen Raj	210763	praveenr21@iitk.ac.in
Abhishek Kumar	210039	akumar21@iitk.ac.in
Pravallika Mudunuru	220814	pmudunuru22@iitk.ac.in
Gaddam Shiva Leela	220392	gaddamshiv22@iitk.ac.in

Course: CS253

Mentor TA: *Vaibhav*

Date: 13/03/2024

Index

Revisions

Version	Primary Author(s)	Description of Version	Date Completed
1.0	Chinmay Hiran Pillai, Shivang Pandey, Arth Banka	Implementation Document for Real-Estate Broker	18/03/24

1 Implementation Details

For the implementation of the software, we have opted for a ReactJS based frontend built using Javascript, a Django REST API server in Python for the backend, and a MySQL relational database for storing data.

Programming Languages Used:

1. Javascript (Frontend)
2. HTML 5 (with CSS 3) (Frontend)
3. Python (Backend)

Database System Used:

1. MySQL

Frameworks and Libraries Used:

1. Frontend:
    a. ReactJS
    b. React-Router-Dom
    c. Material UI
    d. Bootstrap
2. Backend:
    a. Django
    b. Django REST Framework
    c. MySQLClient
    d. Django CORS Headers

Reasons for choosing frameworks/databases:

1. **ReactJS**:
   It enables us to utilize the power and capabilities of Javascript to build cleaner, modular, reusable code to build the front end.

   Component-Based Architecture: ReactJS utilizes a component-based architecture, where complex UIs are built by composing reusable components. This promotes code modularity, maintainability, and easier collaboration among developers. Each component encapsulates its data and logic, leading to cleaner code organization and efficient updates to specific UI sections without affecting the entire application.

   Virtual DOM and Efficient Updates: ReactJS employs a virtual DOM, a lightweight in-memory representation of the real DOM. When changes occur in the application state, React efficiently calculates the minimal set of updates required in the real DOM. This significantly improves rendering performance, especially for complex web applications with frequent UI updates. By only updating the necessary parts of the DOM, React ensures a smooth user experience without unnecessary re-renders.

2. **Django**:
   Django comes with an inbuilt admin interface that the admin uses to add new properties to the application, and the Django REST framework allows us to build APIs easily and quickly.

   Rapid Development: Django is a high-level web development framework that promotes rapid application development (RAD) by providing boilerplate code and abstractions for common web development tasks. This allows developers to focus on implementing core functionalities and unique features specific to our project.

   Python Integration: Django is built upon Python, a widely used, general-purpose, interpreted language. Python's dynamic nature, focus on code readability and extensive standard library contribute to faster development cycles and easier project maintenance.

   Security and Maintainability: Django adheres to secure coding practices and prioritizes security updates. Additionally, it offers Long-Term Support (LTS) versions, ensuring ongoing security patches and compatibility with older codebases.

   Django Backward Compatibility: Django strives to maintain backward compatibility for reusable components within its framework. This includes interfaces, common functionalities, and data formats across different versions. A well-defined roadmap and versioning system facilitate developers in understanding potential compatibility implications when upgrading or downgrading Django versions.

3. **MySQL**:
   It provides a powerful, secure, fast, reliable and ACID-compliant relational database to store data that is easy to replicate.

   Scalability and Performance: MySQL offers robust scaling capabilities to accommodate growing web applications. It efficiently handles large datasets and concurrent user requests, ensuring optimal performance for dynamic websites.

   Open-Source and Cost-Effectiveness: MySQL is an open-source relational database management system (RDBMS), which eliminates licensing costs and fosters a large developer community. This translates to readily available resources, support, and cost-effective database management for web projects.

4. **React-Router-DOM:**

   Declarative Routing: React Router DOM promotes a declarative approach to routing. This means developers define routes and their corresponding components within their React application, enhancing readability and maintainability. The library handles the underlying URL parsing and component rendering logic, allowing developers to focus on application logic.

Component-Based Navigation: React Router DOM integrates seamlessly with the React component model. It provides components like Route, Switch, and Link that can be directly incorporated into the React component hierarchy. This enables developers to leverage the power of component composition to build reusable and well-structured navigation experiences within their web applications.

5. **Bootstrap:**

Rapid Prototyping and Development: Bootstrap provides a pre-built collection of HTML, CSS, and JavaScript components like buttons, forms, navigation bars, and more. This extensive library allows developers to quickly assemble the basic structure and functionality of a web application or website, accelerating the initial development phase. Additionally, Bootstrap's responsive design ensures these components adapt seamlessly across different screen sizes.

Consistency and Maintainability: By leveraging Bootstrap's pre-defined classes and components, developers can establish a consistent look and feel across the entire web application. This promotes a unified user experience and simplifies maintenance, as changes to a single Bootstrap class can propagate throughout the application, reducing the need to modify individual components.

2 Codebase

Link to the Github repository: https://github.com/ChinmayPillai/Real_Estate_Broker_WebApp

1. **Client Folder**: This is the code for the ReactJS frontend. The source code is present in the Client/src folder and the code for the individual components are present in the corresponding Client/src/components folders. Client/public folder contains the public images to serve to the website.
2. **Server Folder**: This is the code for the Django based API server. server/server folder contains the config files for the django REST API server. server/api folder contains the code for the REST API.
3. **README.md**: It contains the instructions to setup and run the full stack application on a system.
4. **Documents Folder**: It contains all the documents made for the software. Eg. SRS Document, Design Document etc.

3 Completeness

3.1 Completed Features:

1. **Login Page:**
   - Users can securely log in using email and password with proper verification.

2. **Register Page:**
   - A user-friendly and secure form collects user details like name, email, PAN, and phone number.

3. **Home Page:**
   - Fully implemented, providing a basic introduction to the website and sample properties.

4. **Property Page:**
   - Displays property details, images, and options for market and limit order buy/sell.
   - Users can add/delete properties from wishlist and view the order book.

5. **Limit Order Option:**
   - Users can submit bids for properties by clicking on the limit order buy or sell option.

6. **Market Order Option:**
   - Users can buy or sell properties at current market order prices.

7. **Order Book:**
   - Implemented to show the top 5 buy and sell bids on the property page.

8. **Portfolio:**
   - Displays the user's current properties in their portfolio.

9. **Watchlist:**
   - Displays the user's current properties in their watchlist.

10. **Add to Watchlist:**
    - Add a property to user’s watchlist.

11. **Funds Page:**
    - Shows current available funds and options to add/withdraw funds.
    - Not integrated with payment or bank gateway; users can add/withdraw directly.

12. **Support Page:**
    - Includes frequently asked questions and an optional form for feedback/suggestions.

13. **Navigation Bar:**
    - Implemented with options for home, portfolio, funds, register/login, and support pages.

14. **Footer:**
    - Present on all pages, displaying contact details for customer care.

15. **Responsiveness:**
    - All pages are responsive and accessible from various screen sizes.

3.2 Features to Implement in Future Versions:

1. Authorised Payment Gateway Integration:
   - Enable secure transactions for property purchases.

2. PAN and e-KYC Authentication:
   - This will enhance security and ensure user identity verification.

3. Direct Property Listing by Users:
   - Allow users to list their properties directly once verified. Due to these verification issues, currently a user cannot directly list a property. Only, the admin can list new properties (which any seller may request for through other channels)

Appendix A - Group Log

Date	Time	Description
5 March	8pm - 9pm	Met to decide the roadmap for the implementation and the structure and use of the github repository.
7 March	12am - 1am	The backend and frontend teams met separately to discuss and distribute specific tasks among the members.
12 March	8pm - 9:30pm	Met to review each member's work, and to discuss and correct any bugs or inconsistencies.
17 March	3pm - 4pm	Met to check and test the final code and fix any bugs.

Apart from these meetings, the team members were regularly in contact with each other through whatsapp and calls.

Contribution:

Frontend - Armeet, Shivang, Chinmay, Utpal, Praveen, Pravallika
Backend - Chinmay, Vaishali, Armeet, Shivang, Arth, Abhishek
Database - Chinmay
Integration - Chinmay, Armeet, Shivang