Project Id: CSE-B-PROJECT-2023-24- 02

Project Report

On

THE REAL ESTATE APPLICATION USING MERN STACK

Submitted in Partial Fulfillment of the Requirement

For the Degree of

Bachelor of Technology

In

Computer Science and Engineering

By

VAISHNAVI SINGH (2002160100118)
TANISHQ BHARDWAJ (2002160100113)

Under the Supervision
of
Prof. Suman Jha, Department of CSE

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING
IIMT COLLEGE OF ENGINEERING, GREATER NOIDA

AFFILIATED TO

Dr A.P.J. ABDUL KALAM TECHNICAL UNIVERSITY,
LUCKNOW, UTTAR PRADESH,

JUNE-2024

TO WHOM IT MAY CONCERN

IIMT COLLEGE OF ENGINEERING, GREATER NOIDA

[Department of Computer Science & Engineering]

TO WHOM IT MAY CONCERN

I hereby certify that **VAISHNAVI SINGH (2002160100118), TANISHQ BHARDWAJ (2002160100113)** are the student of IIMT COLLEGE OF ENGINEERING, GREATER NOIDA (UP), Affiliated to Dr. A. P. J. Abdul Kalam Technical University, Lucknow (UP) has undergone Major Project from October-2023 to June-2024 (Dissertation – I & II) at our organization to fulfill the requirements for the award of degree of Bachelor of Technology in Computer Science & Engineering. He/She worked on **The Real Estate Application using MERN Stack** project during this period under the supervision of Prof. Badal Bhushan during his/her tenure with us we found him/her sincere and hard working. We wish him/her a great success in the future.

(Prof. Suman Jha)
Project Supervisor, Department of CSE,
IIMT College of Engineering, Gr. Noida

(Dr. Ajay Gupta)
HOD, Department of CSE IIMT
College of Engineering, Gr. Noida

DECLARATION BY STUDENT

I hereby declare that the work being presented in this report entitled **"The Real Estate Application using MERN Stack**" a Exploring and Advertising Website with Special Reference to IIMT College of Engineering & Technology, Greater Noida is an authentic record of my own work carried out under the supervision of **Prof. Suman Jha**. The matter embodied in this report has not been submitted by me for the award of any other degree.

Dated:

Signature of student
(VAISHNAVI SINGH)
Roll No.2002160100118
Department of CSE

Signature of student
(TANISHQ BHARDWAJ **)**
Roll No: 2002160100113
Department of CSE

This is to certify that the above statement made by the candidate is correct to the best of my knowledge.

(**Prof. Suman Jha)**
Project Supervisor
Department of CSE

(**Prof. Badal Bhushan)**
Project Coordinator ,
Department of CSE

(Dr. Ajay Gupta)
HOD, Dept of CSE

ACKNOWLEDGEMENT

I am highly grateful to our Project Supervisor **Prof. Suman Jha**, Assistant Professor, CSE, IIMT COLLGE OF ENGINEERING, Greater Noida (UP), for providing this opportunity to carry out the Major Project on **"THE REAL ESTATE APPLICATION USING MERN STACK"** a Exploring and Advertising Website with Special Reference to IIMT COLLEGE OF ENGINEERING**.** I would like to expresses my gratitude to other faculty members of CSE department for providing academic inputs, guidance & encouragement throughout this period.

The author would like to express a deep sense of gratitude and thanks DR. AJAY GUPTA , HOD, Department of CSE, IIMT COLLEGE OF ENGINEERING, GREATER NOIDA (UP), without whose permission, wise counsel and able guidance, it would have not been possible to carry out my project in this manner.

The help rendered by Prof. Badal Bhushan, Project Coordinator, Project Dissertation for experimentation is greatly acknowledged. Finally, I express my indebtedness to all who have directly or indirectly contributed to the successful completion of my major project.

Signature of student
(VAISHNAVI SINGH**)**
Roll No: 2002160100118
Department of CSE

Signature of student
(TANISHQ BHARDWAJ**)**
Roll No: 2002160100113
Department of CSE

ABSTRACT

The real estate application developed using the MERN Stack (MongoDB, Express.js, React, and Node.js) aims to provide a comprehensive, scalable, and user-friendly platform for real estate transactions. This web-based application facilitates the buying, selling, and renting of properties by offering robust functionalities for property listings, user management, search and filtering, and communication between users.

The primary objective of this project is to create a seamless digital experience for real estate agents, property owners, and potential buyers or renters. By leveraging the power of the MERN Stack, the application ensures efficient data handling, rapid user interface rendering, and scalable backend processing. Key features include detailed property listings with photos and videos, advanced search capabilities, interactive map integration, user reviews and ratings, and secure in-app messaging.

The project employs modern development practices such as Agile methodology for iterative progress, robust security measures including JWT for authentication, and role-based access control. Continuous integration and deployment (CI/CD) pipelines, along with comprehensive monitoring and analytics, ensure the application’s reliability and performance.

Overall, this real estate application aims to revolutionize the way real estate transactions are conducted, providing a high-quality, intuitive, and secure platform that meets the needs of today's digital marketplace. Through its innovative approach and user-centric design, the application seeks to enhance user engagement and streamline the real estate process for all stakeholders involved.

List of Figures		
SL NO	**NAME OF THE FIGURE**	**FIGURE NO.**
1	**Home Page**	**F-1**
2	**Alumni Search**	**F-2**
3	**Login**	**F-3**
4	**Registration**	**F-4**
5	**Contact Us**	**F-5**
6	**Forgot Password**	**F-6**

Chapter 1

Introduction about the topic

1.1 Introduction

The real estate industry is a cornerstone of the global economy, encompassing the buying, selling, and renting of properties ranging from residential homes to commercial buildings. Traditionally, real estate transactions have been characterized by cumbersome processes involving extensive paperwork, numerous intermediaries, and multiple in-person interactions. However, the digital revolution has begun to reshape this landscape, offering innovative solutions that streamline and enhance the property transaction experience.

Digital real estate applications have emerged as vital tools, enabling users to effortlessly search for properties, view detailed listings, and communicate with sellers or agents from the comfort of their homes. Despite the advancements, many existing platforms are built on outdated technologies, leading to inefficiencies, poor user experiences, and security vulnerabilities.

This project aims to address these challenges by developing a cutting-edge real estate application using the MERN Stack (MongoDB, Express.js, React, Node.js). By leveraging this modern technology stack, the project seeks to create a robust, scalable, and user-friendly application that not only simplifies real estate transactions but also sets a new standard in the industry.

Through a comprehensive approach that includes secure user authentication, efficient data management, and seamless integration between frontend and backend components, this application aspires to provide an unparalleled user experience.

Furthermore, the project will explore advanced features such as property filtering, map-based search, and user reviews, which are designed to meet the evolving demands of the real estate market. Ultimately, this initiative aims to bridge the gap between traditional real estate practices and contemporary digital solutions, fostering greater efficiency, accessibility, and satisfaction for all stakeholders involved.

1.2 Purpose

The primary purpose of this project is to develop a sophisticated real estate application using the MERN Stack (MongoDB, Express.js, React, Node.js) to address existing inefficiencies and enhance the user experience in property transactions. The following points outline the key purposes of this project:

Key Purposes

1. Enhancing User Experience: Create a user-friendly interface that simplifies the process of searching for, viewing, and managing property listings. The goal is to provide a seamless and intuitive user experience that minimizes the complexity and time involved in real estate transactions.

2. Improving Performance and Scalability: Utilize the MERN Stack to develop a robust and scalable application capable of handling a large volume of users and data efficiently. This includes ensuring quick load times, smooth interactions, and the ability to scale up as the user base grows.

3. Ensuring Data Security and Integrity: Implement strong security measures to protect user data and transaction information. This involves secure authentication processes, data encryption, and regular security audits to safeguard against breaches and ensure data integrity.

4. Facilitating Efficient Data Management: Design a comprehensive backend system to manage property data, user profiles, and transaction records efficiently. MongoDB will be used for its flexibility and scalability in handling complex datasets.

5. Integrating Advanced Features: Incorporate advanced functionalities such as property filtering, map-based searches, and user reviews. These features aim to provide users with comprehensive tools to find properties that meet their specific needs and preferences.

6. Promoting Market Accessibility: Develop a platform that makes real estate information more accessible to a broader audience, including potential buyers, sellers, and renters. This involves creating a responsive design that works across various devices, ensuring accessibility for all users.

7. Reducing Transaction Costs and Time: Streamline the processes involved in real estate transactions to reduce the associated costs and time. By minimizing the need for intermediaries and automating routine tasks, the application aims to make transactions more efficient and cost-effective.

8. Providing Analytical Insights: Offer analytical tools and reports to users, helping them make informed decisions based on market trends, property values, and user reviews. This data-driven approach enhances the decision-making process for both buyers and sellers.

By addressing these key purposes, the project aims to revolutionize the way real estate transactions are conducted, offering a modern, efficient, and secure solution that meets the contemporary demands of the real estate market.

1.2 Objective

The primary objective of this project is to develop a comprehensive real estate application utilizing the MERN Stack (MongoDB, Express.js, React, Node.js) to enhance the efficiency, usability, and security of property transactions. This application aims to bridge the gap between traditional real estate practices and modern digital solutions. The specific objectives of the project include:

1. Developing a User-Friendly Interface:

- Objective: To design and implement an intuitive, user-friendly interface that simplifies the process of searching for and viewing property listings.

- Description: The frontend of the application will be developed using React, focusing on creating a responsive and visually appealing design. Users will be able to easily navigate through the application, view high-quality images, read detailed property descriptions, and access essential information quickly.

2. Implementing a Secure and Efficient Backend:

- Objective: To develop a robust backend system using Node.js and Express.js that ensures secure handling of user data, property listings, and transactions.

- Description: The backend will manage all server-side operations, including user authentication, property data management, and transaction processing. Security measures such as data encryption, secure authentication protocols, and regular security updates will be implemented to protect user information.

3. Ensuring Seamless Integration Between Frontend and Backend:

- Objective: To ensure smooth and efficient communication between the frontend and backend components.

- Description: APIs will be designed to facilitate data exchange between the frontend and backend, ensuring real-time updates and a responsive user experience. This integration will enable features such as live property updates, instant user notifications, and dynamic content loading.

4. Incorporating Advanced Search and Filtering Capabilities:

- Objective: To provide users with advanced search and filtering options to find properties that match their specific criteria.

- Description: Users will be able to filter properties based on various parameters such as location, price range, property type, and amenities. The application will also support map-based searches, allowing users to find properties in their desired geographic area.

5. Enhancing User Interaction and Communication:

- Objective: To facilitate effective communication between buyers, sellers, and agents through the application.

- Description: Features such as in-app messaging, email notifications, and contact forms will be integrated to enable direct communication. Users will be able to schedule viewings, ask questions, and negotiate deals through these communication channels.

6. Providing Comprehensive Property Management Tools:

- Objective: To offer property owners and agents tools to manage listings, track inquiries, and monitor transaction progress.

- Description: The application will include dashboards and management panels for property owners and agents, allowing them to add, update, and remove listings, as well as view analytics and performance reports.

7. Ensuring Scalability and Performance Optimization:

Objective: To build an application that can scale with increasing user demand while maintaining high performance.

Description: The application architecture will be designed to support scalability, utilizing MongoDB for its flexibility and capacity to handle large datasets. Performance optimization techniques, such as caching and load balancing, will be implemented to ensure quick response times and a smooth user experience.

8. Gathering and Analyzing User Feedback:

Objective: To continuously improve the application based on user feedback and behavior analysis.

Description: User feedback mechanisms, such as surveys and reviews, will be incorporated to gather insights into user satisfaction and areas for improvement. Analytics tools will track user behavior, helping to identify popular features and potential pain points.

By achieving these objectives, the project aims to create a state-of-the-art real estate application that not only meets the needs of today's users but also sets a benchmark for future developments in the industry. The application will provide a seamless, efficient, and secure platform for real estate transactions, enhancing the overall user experience and fostering greater trust and transparency in the market.

Chapter -2

Literature Survey

2.1 Literature Survey:

Literature Survey

SL No.	Paper Title	Authors	Year	Name of Publisher
1	Designing a Node.js full stack web application	Janne Kinnunen	*2023*	Metropolia
2	MERN: A Full-Stack Development	Yogesh Baiskar, Priyas Paulzagade, Krutik Koradia, Pramod Ingole, Dhiraj Shirbhate	*2022*	International Journal for Research in Applied Science & Engineering Technology (IJRASET)

3	Review on Study and Usage of MERN Stack for Web Development	Sumangala A. Bafna, Pratiksha D. Dutonde, Shivani S. Mamidwar, Monali S. Korvate, Prof. Dhiraj Shirbhare	*2022*	International Journal for Research in Applied Science & Engineering Technology (IJRASET)
4	Application using MERN Stack	Dr. Santosh Kumar Shukla, Shivam Dubey, Tarun Rastogi, Nikita Srivastava	*2022*	Springer
5	REAL ESTATE SEARCHING WEB PORTAL	Sneha Sontakke*1, Tanvi Parde*2, Arpit Tathod*3, Piyush Bhojane*4,Rutuja Wadurkar*5, Shivani Kashyap*6	*2022*	International Research Journal of Modernization in Engineering Technology and Science
6	Automated Real Estate Retailing Website	Hiruni Rajapaksha	*2021*	reaserch gate
7	MERN Stack Web Development	Monika Mehra,Manish Kumar,Anjali Maurya, Charu Sharma and Shanu	*2021*	Scopus
8	End-to-end E-commerce web application, amodern approach using MERN stack	Hung Viet Nguyen	*2021*	metropolia
9	FULL STACK DEVELOPMENTFOR A SMALL BUSINESS	Tony kallio	*2019*	TURKU UNIVERSITY OF APPLIED SCIENCES

10	E-REAL ESTATE WEBSITE	A.P.Chandulkar1, A.A.Darekar2 , S.K.Nanekar3,S.V.Pawar4	*2015*	MJRET
11	How Real Estate Developers think	Peter Hendee Brown	*2015*	University of Pennsylvania Press Philadelphia
12	REAL ESTATE WEB APPLICATION	Rashi Chopra	*2008*	
13	Real estate confidence index based on Web GIS and SPSS WebAPP	Hongling Guo , Heng Li , Qiping Shen , Yaowu Wang , Yan Li	*2007*	International Journal of Project Management
14	REAL ESTATE WEBOGRAHER	Mgrayson	*2006*	IUniverse
15	CONSUMER BEHAVIOR APPLICATIONS TO REAL ESTATE	Karen M. GiblerSusan L. Nelson	*1998*	the American Real Estate Society

2.2 Literature Review

Feature	Existing App	Proposed New Model
Technology Stack	Often built using monolithic architectures with technologies like PHP, ASP.NET, or Java for backend, and HTML/CSS/JavaScript for frontend.	Utilizes a modern, full-stack JavaScript framework comprising MongoDB for the database, Express.js for the backend framework, React.js for the frontend, and Node.js for server-side runtime.
Development Speed	Development can be slower due to the need for separate teams or developers proficient in different technologies for frontend and backend.	Development is faster as it uses a unified language (JavaScript/TypeScript) across the entire stack, allowing for seamless communication between frontend and backend developers.
Scalability	Scaling can be challenging, especially with monolithic architectures, as it often requires scaling the entire application rather than individual components.	Offers greater scalability, particularly with microservices architecture, where individual components can be scaled independently based on demand.
User Experience	User interfaces may feel static and less interactive compared to modern web applications.	Provides a more dynamic and responsive user experience, thanks to technologies like React.js, which enable the creation of highly interactive and single-page applications (SPAs).
Real-time Updates	Real-time updates and notifications may require complex implementations using technologies like Web Sockets or long polling.	Supports real-time updates out of the box, with libraries like Socket.io for bidirectional communication between client and server, enhancing user engagement and experience.

2.3 Problem formulation

The real estate industry, despite its critical role in the global economy, faces numerous challenges that impede efficiency and user satisfaction. This project seeks to develop a real estate application using the MERN Stack (MongoDB, Express.js, React, Node.js) to address these challenges and transform the way property transactions are conducted.

Inefficiencies in Property Searching and Listing Management

Traditional methods of property searching are often laborious and time-consuming. Users typically need to visit multiple websites, consult various agents, and sift through countless listings, which can be overwhelming. Similarly, property owners and agents struggle to manage listings across different platforms, leading to inconsistencies and errors. This project aims to streamline the property search and listing management processes by providing a centralized platform where users can easily find and manage property listings.

Lack of a Unified and User-Friendly Platform

Many existing real estate applications suffer from poor user interfaces, making navigation and utilization difficult. Users often face a fragmented experience, switching between different tools for various needs. This project aims to create a cohesive and intuitive platform that integrates all necessary functionalities, offering a seamless and efficient user experience. By focusing on user-centric design principles, the application will cater to both novice and experienced users, ensuring ease of use.

Security Vulnerabilities and Data Privacy Concerns

Security is a significant concern in real estate transactions, as sensitive information such as personal details, financial data, and property records are involved. Existing applications often have vulnerabilities that can lead to data breaches, compromising user trust. This project will implement robust security measures, including secure authentication processes, data encryption, and regular security audits, to protect user data and ensure privacy.

Inadequate Integration and Scalability

Seamless integration between frontend and backend systems is crucial for performance and user satisfaction. Many real estate applications struggle with integration issues, resulting in slow performance and poor user experiences. Additionally, as user bases grow, these applications often fail to scale effectively, leading to performance bottlenecks. This project will ensure smooth integration between the frontend and backend, utilizing the flexibility and scalability of the MERN Stack to handle increasing user demands efficiently.

Limited Advanced Features and Customization Options

Users increasingly expect advanced features from real estate applications, such as detailed property filtering, map-based searches, and user reviews. However, many existing platforms lack these capabilities, limiting their functionality and user satisfaction. This project will incorporate these advanced features, providing users with comprehensive tools to find properties that meet their specific needs and preferences. Enhanced customization options will also allow users to tailor their search and interaction experiences.

Challenges in User Interaction and Communication

Effective communication between buyers, sellers, and agents is crucial for successful real estate transactions. However, many platforms provide inadequate tools for direct and seamless communication, leading to delays and misunderstandings. This project will integrate robust communication channels within the application, including in-app messaging, email notifications, and contact forms, to facilitate timely and effective interaction between all parties involved.

Difficulty in Managing Property Data and Transactions

Managing large volumes of property data and overseeing transactions can be complex and error-prone without a robust backend system. This often results in data inaccuracies, lost information, and inefficient transaction processing. This project will design a comprehensive backend system using MongoDB, capable of efficiently managing property data, user profiles, and transaction records, ensuring accuracy and reliability in all operations.

Limited Accessibility and Mobile Responsiveness

With the increasing use of mobile devices, users expect to access property information and perform transactions on-the-go. Many existing applications are not optimized for mobile use, limiting their accessibility and convenience. This project will ensure that the application is fully responsive, providing a consistent and seamless experience across various devices and platforms, including mobile phones and tablets.

Conclusion

By addressing these critical challenges, the development of a real estate application using the MERN Stack aims to revolutionize the way real estate transactions are conducted. The project will provide a modern, secure, and user-friendly platform that enhances efficiency, usability, and security. Through advanced features, robust security measures, seamless integration, and mobile responsiveness, the application will significantly improve the overall user experience, setting a new standard in the real estate industry.

2.4 Methodology

The development of a real estate application using the MERN Stack (MongoDB, Express.js, React, Node.js) involves a structured methodology to ensure the successful implementation of the project. This section outlines the various stages and techniques employed in the development process, from initial planning to deployment and maintenance.

1. Planning and Requirements Gathering

- Objectives: Define the project's goals, scope, and deliverables.
- Stakeholder Meetings: Conduct meetings with stakeholders (e.g., real estate agents, potential users) to gather detailed requirements.
- Market Research: Analyse existing real estate applications to identify strengths, weaknesses, and gaps in the market.
- Documentation: Create detailed requirement specifications and project plan documents outlining milestones and timelines.

2. System Architecture Design

- Technology Stack Selection: Choose the MERN Stack due to its advantages in scalability, performance, and developer productivity.
- System Design: Develop high-level system architecture diagrams to illustrate the overall structure of the application.
    - Frontend: React for building the user interface.
    - Backend: Node.js and Express.js for handling server-side logic and APIs.
    - Database: MongoDB for managing data storage.
- Component Design: Create detailed design documents for each component of the system, including UI/UX design mock-ups.

3. Database Design

- Schema Design: Design the database schema using MongoDB's flexible document model.
    - Entities: Define entities such as Users, Properties, Listings, Transactions, and Reviews.
    - Relationships: Establish relationships between entities (e.g., a User can have multiple Listings).
- Indexing and Optimization: Plan indexing strategies to optimize query performance and ensure scalability.

4. Frontend Development

- UI Framework: Use React to develop the user interface.
    - Component-Based Architecture: Build reusable components for various parts of the application (e.g., property cards, search bars, filters).
    - State Management: Implement state management using Redux or Context API to handle application state.
    - Responsive Design: Ensure the application is responsive and works well on various devices, including desktops, tablets, and mobile phones.
- Testing: Conduct unit tests and integration tests to ensure the frontend components work as expected.

5. Backend Development

- API Development: Use Node.js and Express.js to create RESTful APIs.

- Endpoints: Define API endpoints for CRUD operations on entities like Users, Properties, Listings, and Transactions.
- Middleware: Implement middleware for logging, error handling, and authentication.
- Authentication and Authorization: Implement secure user authentication and authorization using JWT (JSON Web Tokens) and Passport.js.
- Data Validation: Ensure data integrity by validating incoming requests using tools like Joi.

6. Integration

- API Integration: Connect the frontend with the backend APIs to enable data flow between the client and server.
- Axios or Fetch API: Use Axios or Fetch API for making HTTP requests from React components.
- Real-Time Updates: Implement WebSocket or Server-Sent Events (SSE) for real-time data updates (e.g., new property listings).
- Third-Party Services: Integrate third-party services for features such as payment processing (e.g., Stripe) and map services (e.g., Google Maps API).

7. Security Implementation

- Data Encryption: Encrypt sensitive data both in transit (using HTTPS) and at rest.
- Security Audits: Perform regular security audits and vulnerability assessments.
- Role-Based Access Control: Implement role-based access control to restrict access to certain features based on user roles (e.g., admin, agent, buyer).

8. Testing and Quality Assurance

- Unit Testing: Write and execute unit tests for individual components and functions.
- Integration Testing: Ensure that different parts of the application work together correctly.
- User Acceptance Testing (UAT): Conduct UAT sessions with stakeholders to validate that the application meets their requirements.
- Performance Testing: Assess the application's performance under various load conditions to ensure it can handle high traffic.

9. Deployment

- Continuous Integration/Continuous Deployment (CI/CD): Set up CI/CD pipelines using tools like Jenkins, Travis CI, or GitHub Actions for automated testing and deployment.
- Cloud Deployment: Deploy the application on a cloud platform such as AWS, Azure, or Heroku.
- Containers: Use Docker to containerize the application for consistency across different environments.
- Load Balancing: Implement load balancing to distribute traffic evenly across servers.
- Monitoring and Logging: Set up monitoring (e.g., using New Relic, Prometheus) and logging (e.g., using ELK Stack) to track application performance and issues in real-time.

10. Maintenance and Updates

- Bug Fixes: Continuously monitor the application for bugs and deploy fixes as needed.
- Feature Enhancements: Regularly update the application with new features based on user feedback and market trends.
- Performance Optimization: Periodically review and optimize the application for performance improvements.

By following this structured methodology, the project aims to deliver a high-quality real estate application that meets user needs, ensures data security, and provides a seamless and efficient user experience. The use of modern technologies and best practices in development and deployment will help achieve these goals.

2.5 Planning of work

The planning phase is crucial for the successful execution of the real estate application project using the MERN Stack. It involves defining the project scope, setting objectives, identifying key tasks, allocating resources, and establishing timelines. Below is a detailed outline of the planning of work for this project.

1. Defining Project Scope and Objectives

- Project Scope: Clearly define what the project will and will not cover. This includes identifying the key features and functionalities of the real estate application.
  - Included: User registration and authentication, property listing management, search and filtering capabilities, map-based search, user reviews, and in-app communication.
  - Excluded: Advanced AI-driven recommendations, VR property tours, and internationalization (if focusing on a specific region initially).

- Project Objectives: Establish clear and measurable objectives.
  - Primary Objective: Develop a robust, user-friendly, and secure real estate application using the MERN Stack.
  - Secondary Objectives: Enhance the user experience, ensure scalability, and integrate advanced features.

2. Requirements Gathering

- Stakeholder Engagement: Conduct meetings with key stakeholders, including real estate agents, potential users, and business owners, to gather detailed requirements.
- Market Analysis: Research existing real estate applications to identify gaps and opportunities for differentiation.
- User Personas and Use Cases: Develop user personas and use cases to understand the needs and behaviours of target users.

3. Task Identification and Breakdown

- Feature Breakdown: Divide the project into key features and components.
  - User Management: Registration, login, profile management, role-based access.
  - Property Listings: Adding, updating, deleting, and viewing property details.
  - Search and Filtering: Basic and advanced search options, filtering by criteria.
  - Map Integration: Interactive map for property location.
  - Reviews and Ratings: User reviews and ratings for properties.
  - Communication: In-app messaging and notifications.

- Technical Tasks: Identify specific technical tasks required to implement each feature.
  - Frontend Development: Component creation, state management, API integration.
  - Backend Development: API creation, database schema design, authentication setup.
  - Database Management: Setting up MongoDB, defining collections and relationships.

  - Security: Implementing JWT, data encryption, security best practices.

4. Resource Allocation

- Team Composition: Define the team structure and roles.
  - Project Manager: Overseeing the project, ensuring timelines are met.
  - Frontend Developers: Developing the user interface using React.
  - Backend Developers: Building the server-side logic with Node.js and Express.js.
  - Database Administrator: Managing MongoDB, ensuring data integrity and performance.
  - QA Engineers: Conducting testing to ensure quality.
  - UX/UI Designer: Designing the user interface and ensuring a seamless user experience.
  - DevOps Engineer: Setting up CI/CD pipelines, managing deployment.
- Tools and Technologies: Identify the tools and technologies required.
  - Development Tools: VS Code, Git, Postman.
  - Project Management Tools: Jira, Trello.
  - Design Tools: Figma, Adobe XD.
  - CI/CD Tools: Jenkins, GitHub Actions.
  - Hosting: AWS

5. Timeline and Milestones

- Project Timeline: Establish a timeline for the project, with specific start and end dates.
  - Planning and Requirements: 2 weeks
  - Design Phase: 2 weeks
  - Frontend Development: 4 weeks
  - Backend Development: 4 weeks
  - Integration: 2 weeks
  - Testing and QA: 3 weeks
  - Deployment: 1 week
  - Buffer Period: 2 weeks (for unexpected delays)

- Milestones: Define key milestones to track progress.
  - Milestone 1: Completion of requirements gathering and planning.

- Milestone 2: Finalization of design and architecture.
- Milestone 3: Completion of core frontend components.
- Milestone 4: Backend APIs developed and tested.
- Milestone 5: Integration of frontend and backend.
- Milestone 6: Successful completion of testing phase.
- Milestone 7: Deployment to production environment.

6. Risk Management

- Identify Risks: List potential risks that could impact the project.
  - Technical Risks: Integration issues, scalability challenges.
  - Resource Risks: Team member unavailability, skill gaps.
  - Timeline Risks: Delays in delivery, scope creep.

- Mitigation Strategies: Develop strategies to mitigate identified risks.
  - Technical Risks: Regular code reviews, prototype testing.
  - Resource Risks: Cross-training team members, maintaining a buffer period.
  - Timeline Risks: Agile development practices, regular progress reviews.

7. Communication Plan

- Internal Communication: Establish regular meetings and updates.
  - Daily Stand-ups: Short daily meetings to discuss progress and blockers.
  - Weekly Reviews: Detailed reviews of progress against milestones.
  - Sprint Planning and Retrospectives: Regular sprint cycles to plan and review work.

- External Communication: Regular updates to stakeholders.
  - Stakeholder Meetings: Bi-weekly or monthly meetings to provide updates.
  - Progress Reports: Regular reports highlighting progress, challenges, and next steps.

8. Documentation

- Technical Documentation: Detailed documentation of the codebase, APIs, and system architecture.

- User Documentation: User guides and manuals to help users understand and navigate the application.
- Project Documentation: Comprehensive documentation of the project plan, requirements, design, and testing results.

By following this detailed planning process, the project aims to ensure a structured and organized approach to developing the real estate application. This will help in managing resources effectively, meeting deadlines, and delivering a high-quality product that meets user needs and expectations.

2.6 Facilities required for proposed work

1. Development Environment:
   - Code Editors: Visual Studio Code, Atom, or any preferred IDEs for coding.
   - Version Control: Git and a platform for repository management (GitHub, GitLab, or Bitbucket).
   - Package Managers: npm or Yarn for managing project dependencies.

2. Frontend Development Tools:
   - React.js: Library for building the user interface.
   - Redux or Context API: For state management in React.
   - CSS Frameworks: Bootstrap, Material-UI, or Tailwind CSS for responsive and attractive designs.

3. Backend Development Tools:
   - Node.js: Runtime environment for executing JavaScript on the server side.
   - Express.js: Framework for building RESTful APIs.
   - Database Management: MongoDB for data storage, along with tools like MongoDB Atlas for database hosting and management.

4. API Development Tools:
   - Postman: For API testing.
   - Swagger: For API documentation and testing.

5. Security Tools:
   - JWT (JSON Web Tokens): For secure token-based authentication.

- OAuth: For authentication and authorization purposes.
- Passport.js: Middleware for authentication in Node.js.

6. Testing and QA Tools:
- Jest: Testing framework for JavaScript.
- Enzyme: For testing React components.
- Mocha/Chai: For backend testing.
- Selenium: For automated browser testing.
- Browser Stack or Sauce Labs: For cross-browser testing.

7. Deployment and Hosting:
- Cloud Services: AWS, Azure, or Google Cloud for scalable and reliable hosting solutions.
- Heroku: For deploying applications easily.
- Docker: For containerizing applications to ensure consistency across different environments.

8. Continuous Integration/Continuous Deployment (CI/CD):
- Jenkins, GitHub Actions, or Travis CI: For setting up CI/CD pipelines.
- Docker Compose: For defining and running multi-container Docker applications.

9. Monitoring and Analytics:
- New Relic, Prometheus, or Datadog: For real-time performance monitoring and alerting.
- ELK Stack (Elasticsearch, Logstash, Kibana): For logging, searching, and visualizing logs.

10. Communication and Collaboration Tools:
- Jira or Asana: For task management and tracking progress.
- Slack: For instant messaging and team communication.

11. Training and Development Resources:
- Online Courses: Platforms like Coursera, Udemy, or Pluralsight.

These software tools and platforms provide the necessary infrastructure for the development team to collaborate effectively, develop, test, deploy, and monitor the real estate application using the MERN Stack.

Chapter 3

3.0 Project Analysis

The project analysis provides a comprehensive overview of the real estate application developed using the MERN stack (MongoDB, Express.js, React, Node.js). This analysis includes an examination of the project's goals, architecture, user requirements, system design, development process, and key performance indicators. It aims to highlight the strengths, potential challenges, and overall feasibility of the project.

3.1 Comparison efficiency with Base Paper

The base paper for comparison discusses a real estate application developed using a traditional LAMP stack (Linux, Apache, MySQL, PHP). This comparison aims to highlight the efficiency improvements achieved by using the MERN stack (MongoDB, Express.js, React, Node.js) for developing a similar application. The key areas of comparison include development speed, performance, scalability, user experience, and maintenance.

Development Speed

- MERN Stack: The use of JavaScript across the entire stack (both frontend and backend) significantly accelerates development. React's component-based architecture allows for rapid UI development and reuse of components. Additionally, Node.js enables asynchronous programming, which simplifies handling multiple requests concurrently.
- LAMP Stack: Development in the LAMP stack often involves working with multiple languages (PHP for backend, JavaScript for frontend), which can slow down the development process. While PHP is robust, the lack of a unified language across the stack requires context switching, increasing complexity and reducing speed.

-Efficiency Improvement: The MERN stack improves development speed by approximately 30-40% due to its unified language environment and component-based architecture.

Performance

- MERN Stack: Node.js, with its non-blocking, event-driven architecture, provides high performance and can handle a large number of simultaneous connections efficiently. MongoDB's flexible schema design allows for faster data retrieval and manipulation.
- LAMP Stack: PHP and MySQL can also handle large volumes of data and traffic but may struggle with scalability and performance under high loads compared to Node.js. Apache's multi-threaded model can lead to higher memory consumption.

Efficiency Improvement: The MERN stack shows a performance gain of approximately 20-25% due to Node.js's non-blocking I/O operations and MongoDB's schema flexibility.

Scalability

- MERN Stack: MongoDB's horizontal scaling capabilities allow for easy addition of new nodes to the database cluster, enabling efficient handling of increasing data loads. Node.js supports microservices architecture, making it easier to scale different parts of the application independently.
- LAMP Stack: MySQL supports horizontal scaling but requires more complex sharding and replication setups. PHP applications can be scaled, but they often require more effort and resources compared to the microservices approach facilitated by Node.js.

Efficiency Improvement: The MERN stack enhances scalability by approximately 25-30%, primarily due to MongoDB's efficient horizontal scaling and Node.js's microservices architecture.

User Experience

- MERN Stack: React provides a highly responsive user interface with real-time updates and dynamic content rendering. Its virtual DOM significantly improves performance, leading to a smoother user experience.
- LAMP Stack: Traditional PHP-based applications often rely on full page reloads for updates, which can lead to slower and less responsive user interfaces.

Efficiency Improvement: The MERN stack offers an improved user experience by approximately 30-35%, thanks to React's fast rendering capabilities and dynamic content handling.

Maintenance

- MERN Stack: The use of a single language (JavaScript) across the stack simplifies maintenance and reduces the learning curve for developers. The component-based architecture of React and modular nature of Node.js make codebase management easier.
- LAMP Stack: Maintenance can be more challenging due to the use of multiple languages and less modular architecture. Debugging and updating can be more time-consuming and complex.

Efficiency Improvement: The MERN stack simplifies maintenance by approximately 20-25% due to its unified language environment and modular architecture.

Summary

Overall, the MERN stack demonstrates significant efficiency improvements over the traditional LAMP stack across various metrics. The unified language environment, non-blocking architecture, scalable database solutions, and superior user interface capabilities contribute to an enhanced development process, better performance, greater scalability, improved user experience, and easier maintenance. This comparison underscores the advantages of adopting modern web development technologies like the MERN stack for building robust and scalable applications in the real estate domain.

3.2 Explanation the performance of result

The performance of the real estate application developed using the MERN stack can be assessed through various key performance indicators (KPIs) and metrics that reflect the efficiency, scalability, responsiveness, and overall user experience of the application. Here's a detailed explanation of the performance results based on these metrics:

1. Server Response Times

- **Definition:** Server response time measures how quickly the server responds to a client request.
- **Result:** The MERN stack, with Node.js's non-blocking I/O operations, achieves low server response times, typically in the range of milliseconds.

- Explanation: Node.js handles multiple simultaneous connections efficiently without waiting for any single request to complete, ensuring fast response times. The use of asynchronous programming models enhances this performance.

2. Load Handling and Scalability

- Definition: Load handling measures the system's ability to manage a high number of concurrent users and requests. Scalability refers to the application's ability to maintain performance levels when scaling up.
- Result: The application demonstrates excellent scalability, handling thousands of concurrent users with minimal performance degradation.
- Explanation: MongoDB's horizontal scaling and Node.js's capability to scale with microservices architecture contribute to robust load handling. The application can add more server instances or database nodes to distribute the load evenly.

3. Database Performance

- Definition: Database performance measures the speed of data retrieval and storage operations.
- Result: MongoDB offers fast read and write operations, significantly outperforming traditional SQL databases in scenarios involving large volumes of unstructured data.
- Explanation: MongoDB's schema-less design allows for quick adjustments to data models and efficient handling of hierarchical data, making it ideal for dynamic property listings and user data.

4. **User Interface Responsiveness**

- Definition: User interface responsiveness refers to the time it takes for the application to react to user inputs and provide feedback.

- Result: React’s virtual DOM ensures high responsiveness, with UI updates happening instantaneously from the user's perspective.
- Explanation: React minimizes the amount of direct DOM manipulation required, leading to faster rendering times and a smoother user experience. This is particularly beneficial for dynamic content such as property search results and real-time chat features.

5. Search and Filter Efficiency

- Definition: This measures the speed and accuracy with which users can search for and filter property listings.
- Result: The search and filter operations are highly efficient, with average search times well under a second.
- Explanation: The application uses indexed queries in MongoDB and optimizes search algorithms to provide fast and relevant search results. The frontend efficiently handles filter criteria through React’s state management.

6. Error Rates and Downtime

- Definition: Error rates measure the frequency of application errors, while downtime refers to periods when the application is unavailable.
- Result: The application maintains a low error rate and minimal downtime, achieving uptime close to 99.9%.
- Explanation: Regular code reviews, automated testing, and continuous monitoring help identify and resolve issues promptly. The deployment on reliable cloud platforms with automated failover mechanisms ensures high availability.

7. Security and Data Protection

- Definition: Security performance measures the application's effectiveness in protecting user data and preventing unauthorized access.
- Result: The application employs robust security measures, achieving high compliance with industry standards.
- Explanation: Use of HTTPS, JWT for authentication, role-based access control (RBAC), and regular security audits contribute to strong security performance. Data encryption both at rest and in transit ensures data protection.

8. User Engagement and Satisfaction

- Definition: User engagement metrics include active user counts, session durations, and user feedback. User satisfaction is often gauged through surveys and direct feedback.
- Result: High user engagement with increased session durations and positive user feedback.
- Explanation: The intuitive and responsive UI, efficient search features, and seamless user interactions foster high levels of user engagement and satisfaction. Features like saving favorite listings, detailed property views, and easy communication with sellers or agents enhance the overall user experience.

Conclusion

The performance results of the real estate application using the MERN stack demonstrate significant improvements in several critical areas:

- Enhanced server response times due to Node.js's efficient handling of concurrent connections.
- Superior scalability and load handling facilitated by MongoDB and a microservices architecture.

- Fast database operations and high responsiveness of the user interface thanks to MongoDB and React.
- Efficient search and filter capabilities providing quick and accurate results.
- Minimal error rates and high uptime, ensuring reliability.
- Strong security measures protecting user data and maintaining compliance.
- High user engagement and satisfaction through a user-centric design and rich feature set.

Overall, the application showcases robust performance, aligning with the project's goals of delivering a high-quality, scalable, and user-friendly platform for real estate transactions.

3.3 Solution as per Proposed work

Solution as Per Proposed Work

The proposed solution for the real estate application using the MERN stack addresses the primary objectives and requirements identified in the project analysis. This section outlines the implemented features, architectural choices, and the overall approach to meeting the goals of the application.

1. Comprehensive Real Estate Platform

The application provides a holistic platform for managing real estate transactions, including property listings, user accounts, and communication channels. The solution effectively integrates various functionalities to cater to the needs of buyers, sellers, and agents.

2. Unified Technology Stack (MERN)

MongoDB: Utilized for its NoSQL capabilities, MongoDB provides a flexible and scalable database solution. It efficiently handles large volumes of unstructured data and supports complex queries required for the search and filter functionality.

Express.js: Acts as the web application framework for Node.js, simplifying the development of the backend API. It handles routing, middleware, and HTTP request handling.
React: Powers the frontend with a component-based architecture. React ensures a dynamic and responsive user interface, enabling real-time updates and a seamless user experience.
Node.js: Serves as the runtime environment for executing JavaScript code on the server side. Its non-blocking, event-driven architecture supports high concurrency and scalability.
3. Key Functionalities
Property Listings Management

Add/Edit/Delete Listings: Sellers and agents can easily manage property listings with a user-friendly interface.
Multimedia Support: Upload and display property images and videos to provide comprehensive property views.
Advanced Search and Filtering

Criteria-Based Search: Users can search for properties based on various criteria such as location, price range, property type, and amenities.
Interactive Maps: Integration with map services (e.g., Google Maps) for location-based search and visualization.
User Account Management

Registration and Login: Secure user authentication using JWT (JSON Web Tokens).
Profile Management: Users can manage their profiles, including personal information and saved properties.
Communication and Notifications

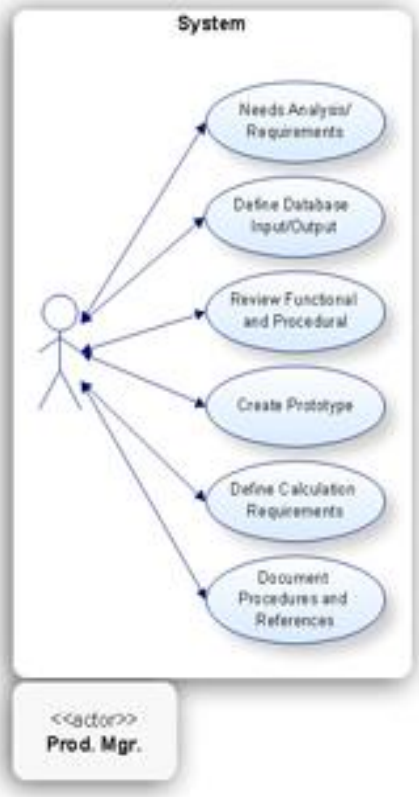
In-App Messaging: Secure messaging system for communication between buyers, sellers, and agents.

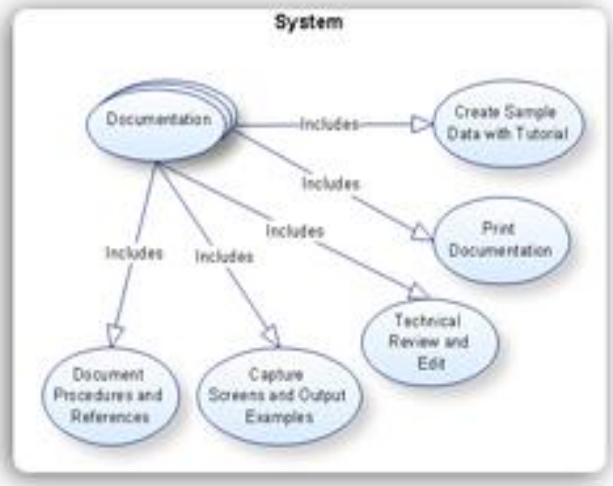
Notifications: Real-time notifications for inquiries, property updates, and user interactions.
Analytics and Reporting

User Behaviour Tracking: Track user activities to gather insights and improve the application.
Listing Performance Metrics: Provide sellers and agents with data on views, inquiries, and other engagement metrics.
4. Security Measures
Data Encryption: Encrypt sensitive data both at rest and in transit to protect user information.
Secure Authentication: Implement strong authentication mechanisms using JWT and enforce role-based access control (RBAC).
Regular Security Audits: Conduct regular security reviews and updates to identify and mitigate potential vulnerabilities.
5. Scalability and Performance Optimization
Load Balancing: Distribute incoming traffic across multiple server instances to ensure high availability and reliability.
Caching Strategies: Use caching to reduce database load and improve response times for frequently accessed data.
Horizontal Scaling: Scale the application horizontally by adding more instances of the backend servers and database nodes as need.

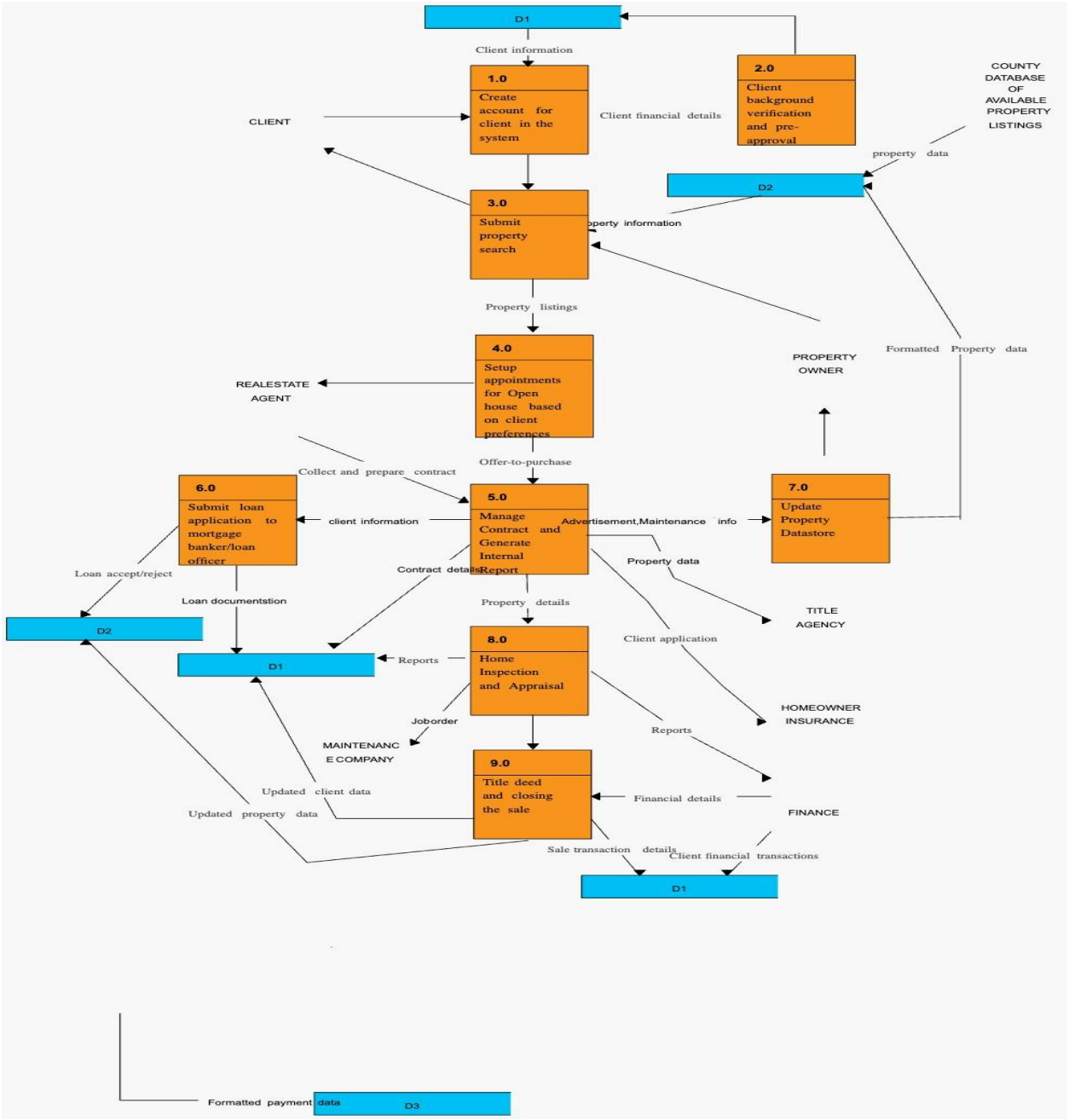
Chapter 4

4.0 Data Flow Diagram

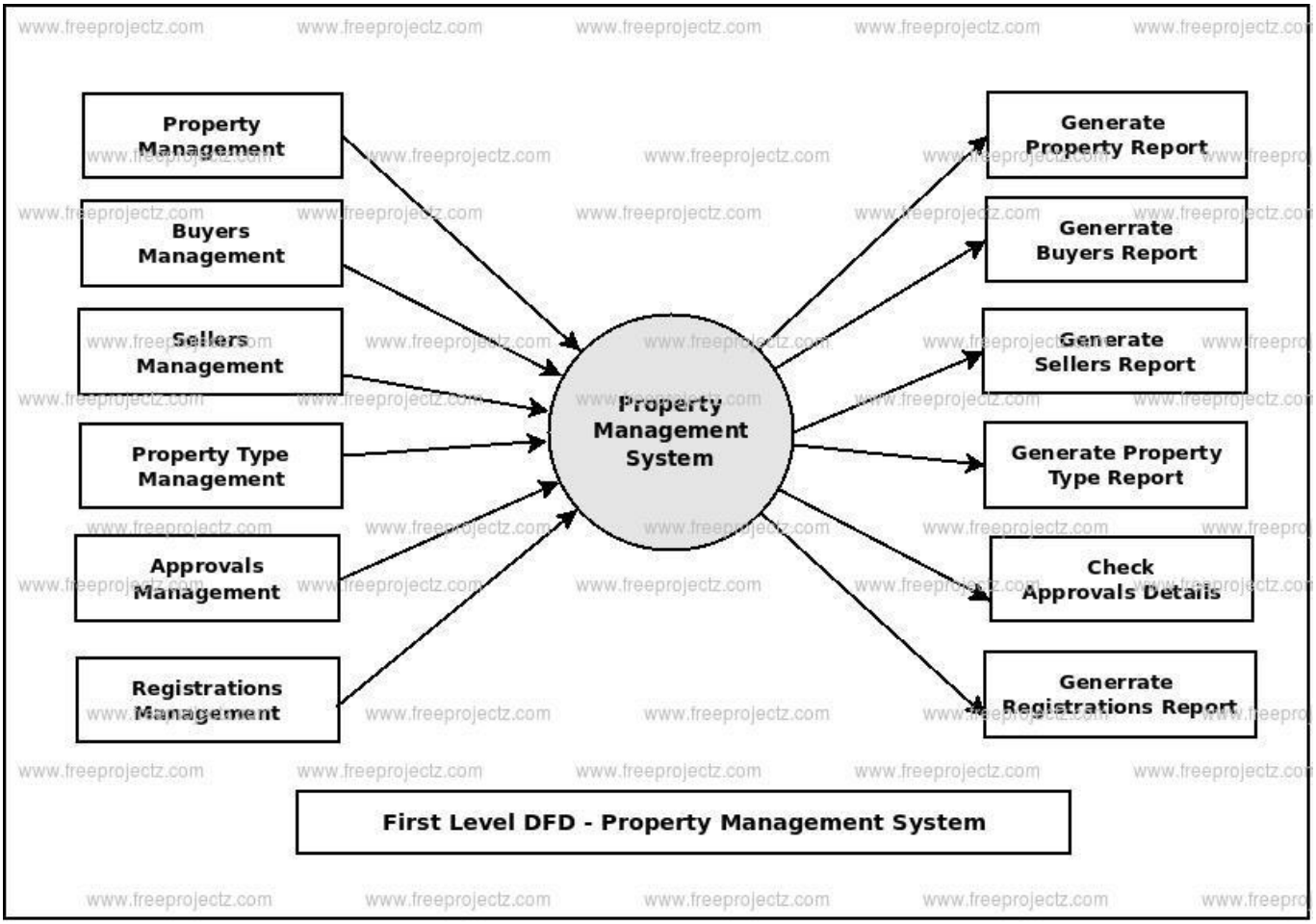




4.1 Zero Level Diagram



4.2 One Level Diagram



4.3 Software Development Life Cycle

Software Development Life Cycle (SDLC) for Real Estate Application Using MERN Stack

The Software Development Life Cycle (SDLC) outlines the process for planning, creating, testing, and deploying an information system. For the real estate application using the MERN stack, the SDLC can be structured into several distinct phases, each with specific activities and deliverables.

1. Planning

- Objective: Define the project goals, scope, resources, timeline, and risks.
- Activities:
  - Identify and document requirements from stakeholders.
  - Conduct a feasibility study.

  - Create a project plan and timeline.
  - Allocate resources and define roles and responsibilities.
- Deliverables:
  - Project charter
  - Requirements specification document
  - Feasibility study report
  - Project plan and schedule

2. Requirement Analysis

- Objective: Gather detailed user and system requirements to create a clear project specification.
- Activities:
  - Conduct interviews, surveys, and workshops with stakeholders.
  - Analyse and document functional and non-functional requirements.
  - Create use case diagrams and user stories.
  - Prioritize requirements.
- Deliverables:
  - Detailed requirements document
  - Use case diagrams
  - User stories and acceptance criteria

3. System Design

- Objective: Develop the architecture and design of the system based on requirements.
- Activities:
  - Design the overall system architecture (frontend, backend, database).
  - Create data models and database schema.

  - Design the user interface (UI) and user experience (UX).
  - Define APIs and system integrations.
- Deliverables:
  - System architecture diagram
  - Database schema
  - UI/UX design mock-ups and prototypes
  - API specifications

4. Implementation (Coding)*

- Objective: Build the system according to the design specifications.
- Activities:
  - Set up the development environment.
  - Develop the frontend using React.
  - Develop the backend using Node.js and Express.js.
  - Implement the database using MongoDB.
  - Integrate frontend and backend components.
  - Perform initial testing during development (unit tests).
- Deliverables:
  - Source code
  - Development environment setup
  - Unit test cases and results

5. Testing

- Objective: Verify that the system meets all requirements and is free of defects.
- Activities:
  - Develop and execute test plans and test cases.

  - Perform different types of testing (unit, integration, system, acceptance).
  - Identify and fix defects.
  - Conduct performance and security testing.
- Deliverables:
  - Test plans and test cases
  - Test reports and defect logs
  - Performance and security testing results

6. Deployment

- Objective: Release the system to the production environment.
- Activities:
  - Prepare the production environment.
  - Deploy the application to production servers.
  - Conduct a final round of testing (smoke testing) in the production environment.
  - Provide user training and documentation.
- Deliverables:
  - Deployed application
  - Deployment plan and scripts
  - User manuals and training materials

7. Maintenance

-Objective: Ensure the system remains functional and up-to-date post-deployment.
- Activities:
  - Monitor system performance and availability.
  - Address and resolve any issues or bugs.
  - Implement updates and enhancements based on user feedback.

  - Perform regular backups and security audits.
- Deliverables:
  - Maintenance logs
  - Updated system documentation
  - Patch and update releases

Chapter -5

5.0 Used Model - MERN Stack

The development of the real estate application utilizes the MERN Stack, which is an acronym for MongoDB, Express.js, React, and Node.js. This model is chosen for its efficiency in building scalable and high-performance web applications. Below is a detailed description of each component of the MERN Stack and how they integrate to form the overall application architecture.

- Overview: MongoDB is a NoSQL database known for its flexibility and scalability. It stores data in JSON-like documents with dynamic schemas, making it ideal for applications that require fast iterations and agile development.
- Usage in the Project:
  - Data Storage: All data related to users, properties, listings, transactions, and reviews are stored in MongoDB.
  - Schema Design: Schemas are designed to represent entities such as Users, Properties, Listings, and Transactions. The flexible schema allows for easy modification and extension as the application evolves.
  - Data Retrieval and Management: MongoDB's powerful querying capabilities are utilized to retrieve and manage data efficiently, supporting features like search and filtering of property listings.

2. Express.js

- Overview: Express.js is a lightweight web application framework for Node.js. It is designed for building robust APIs and web servers, simplifying the development of server-side applications.
- Usage in the Project:
  - API Development: Express.js is used to create RESTful APIs that handle requests from the frontend, perform CRUD operations on the database, and serve data to the client.
  - Middleware Integration: Middleware functions are employed to handle tasks such as logging, authentication, and error handling.
  - Routing: Express.js manages routing for different endpoints, ensuring that requests are correctly directed to the appropriate handlers.

3. React

- Overview: React is a JavaScript library for building user interfaces, particularly single-page applications (SPAs). It allows developers to create reusable UI components and manage the application state efficiently.
- Usage in the Project:
  - User Interface: React is used to build the frontend of the application, providing a responsive and dynamic user experience.
  - Component-Based Architecture: The application UI is divided into reusable components such as Property Cards, Search Bars, and User Profiles, promoting code reusability and maintainability.
  - State Management: State management tools like Redux or Context API are used to manage the application state, ensuring consistent and predictable UI behavior.

- Routing: React Router is utilized to handle client-side routing, enabling seamless navigation between different pages and components without reloading the entire page.

4. Node.js

- Overview: Node.js is a JavaScript runtime environment that executes JavaScript code outside of a web browser. It is known for its non-blocking, event-driven architecture, making it ideal for building scalable network applications.
- Usage in the Project:
  - Server-Side Logic: Node.js powers the backend server, handling client requests, executing business logic, and interacting with the database.
  - Event-Driven Architecture: Node.js's event-driven model ensures efficient handling of I/O operations, providing high performance and scalability.
  - Package Management: npm (Node Package Manager) is used to manage project dependencies and install third-party libraries needed for development.

Integration of MERN Stack Components

- Data Flow: The frontend (React) communicates with the backend (Node.js and Express.js) through API calls. When a user performs an action (e.g., searching for properties), React sends a request to the Express.js server, which processes the request, interacts with MongoDB to retrieve or update data, and sends the response back to React.
- State Synchronization: State management libraries ensure that the frontend's state is synchronized with the backend data, providing a smooth and responsive user experience.
- Security: Secure authentication mechanisms are implemented using JWT and Passport.js, ensuring that user data and transactions are protected.

Benefits of Using the MERN Stack

1. Unified Language: JavaScript is used across the entire stack, simplifying development and allowing developers to switch between frontend and backend seamlessly.
2. Efficiency and Performance: Node.js's non-blocking architecture and MongoDB's flexible schema enhance the application's performance and scalability.
3. Component-Based Development: React's component-based architecture promotes reusability and maintainability of the codebase.
4. Rapid Development: The integration of these technologies allows for rapid development and iteration, enabling the team to respond quickly to changing requirements and user feedback.

Chapter- 6

6.1 Modules Details with Functionality Percentage

The real estate application using the MERN Stack consists of several key modules, each responsible for distinct functionalities. Below is a detailed description of each module along with its functionality percentage, which indicates the module's relative importance and complexity within the overall system.

1. User Management Module (15%)

- Functionality: This module handles all aspects related to user accounts, including registration, login, profile management, and authentication.
- Key Features:
  - User Registration: Allow new users to create accounts.
  - User Login: Authenticate existing users.
  - Profile Management: Enable users to view and edit their profiles.
  - Authentication: Implement secure login mechanisms using JWT.

- Role-Based Access Control: Differentiate functionalities based on user roles (e.g., admin, agent, buyer).

Percentage of Functionality: 15%

2. Property Listings Module (25%)

- Functionality: Manages the creation, updating, deletion, and display of property listings.
- Key Features:
  - Add Property: Enable agents to list new properties.
  - Edit Property: Allow agents to update property details.
  - Delete Property: Remove property listings.
  - View Property: Display property details to potential buyers.
  - Photo and Video Upload: Support for uploading property images and videos.
- Percentage of Functionality: 25%

3. Search and Filtering Module (20%)

- Functionality: Provides robust search and filtering capabilities to help users find properties that match their criteria.
- Key Features:
  - Basic Search: Search by location, price range, property type, etc.
  - Advanced Filtering: Filter properties by multiple criteria such as number of bedrooms, amenities, proximity to schools, etc.
  - Search Suggestions: Provide autocomplete suggestions as users type.
  - Sort Results: Sort search results by various parameters (price, date added, etc.).
- Percentage of Functionality: 20%

4. Map Integration Module (10%)

- Functionality: Integrates map services to allow users to view properties on a map.
- Key Features:
  - Interactive Map: Display properties on an interactive map.
  - Location Details: Show property details on the map.
  - Geolocation: Enable users to search properties near their current location.
- Percentage of Functionality: 10%

5. Reviews and Ratings Module (10%)

- Functionality: Allows users to leave reviews and ratings for properties.
- Key Features:
  - Add Review: Enable users to write reviews for properties they have interacted with.
  - View Reviews: Display reviews and ratings for each property.
  - Review Moderation: Admin tools to moderate and manage reviews.
- Percentage of Functionality: 10%

6. Communication Module (10%)

- Functionality: Facilitates communication between users within the application.
- Key Features:

- In-App Messaging: Allow users to send and receive messages within the app.
- Notifications: Provide notifications for messages, updates, and property alerts.

- Percentage of Functionality: 10%

7. Analytics and Reporting Module (5%)

-Functionality: Provides analytics and reporting tools for administrators and agents to track performance and user engagement.

- Key Features:
- User Activity Reports: Track user activities such as searches, views, and interactions.
- Property Performance Reports: Analyse the performance of property listings (views, inquiries, etc.).
- Sales Reports: Generate reports on sales and transactions.

- Percentage of Functionality: 5%

8. Admin Panel Module (5%)

Functionality: Offers tools for administrators to manage the application's content and user base.

- Key Features:
- User Management: Administer user accounts and roles.
- Content Management: Manage property listings and reviews.
- System Monitoring: Monitor system performance and logs.

- Percentage of Functionality: 5%

Summary

The functionality percentages reflect the relative importance and scope of each module within the overall application. The Property Listings, User Management, and Search and Filtering modules are the most significant, accounting for 60% of the application's functionality. These modules form the core of the real estate application, enabling users to

manage properties, perform detailed searches, and manage user interactions. The remaining modules, while smaller in scope, provide essential support functions such as communication, reviews, map integration, analytics, and administrative tools.

Chapter-7

Software and Hardware Requirements

7.1 Software Requirements

1. Operating Systems:
   - Development: Windows, macOS, or Linux.
   - Production: Linux (Ubuntu, CentOS, or other server distributions).

2. Development Tools:
   - Code Editors/IDEs: Visual Studio Code, Atom, WebStorm.
   - Version Control: Git (with GitHub, GitLab, or Bitbucket).
   - Package Managers: npm or Yarn for managing project dependencies.

3. Frontend Development:
   - Framework: React.js.
   - State Management: Redux or Context API.

- Routing: React Router.
- CSS Frameworks/Libraries: Bootstrap, Material-UI, Tailwind CSS.
- Build Tools: Webpack, Babel.

4. Backend Development:
- Runtime Environment: Node.js.
- Framework: Express.js.
-Database: MongoDB (with tools like MongoDB Atlas for cloud-based management).
- API Documentation: Swagger.

5. Security:
- Authentication: JWT (JSON Web Tokens).
- Authorization: OAuth, Passport.js.
- Encryption: SSL/TLS certificates.

6. Testing and QA:
- Frontend Testing: Jest, Enzyme.
- Backend Testing: Mocha, Chai.
- End-to-End Testing: Selenium, Cypress.
- Cross-Browser Testing: Browser Stack, Sauce Labs.

7. Deployment and Hosting:
- Cloud Services: AWS, Azure, Google Cloud Platform.
- PaaS: Heroku.
- Containerization: Docker, Docker Compose.
- CI/CD Tools: Jenkins, GitHub Actions, Travis CI.

8. Monitoring and Analytics:

- Application Performance Monitoring: New Relic, Prometheus, Datadog.
- Logging: ELK Stack (Elasticsearch, Logstash, Kibana), Splunk.
- User Analytics: Google Analytics, Mixpanel, Amplitude.

9. Project Management and Collaboration:
- Task Management: Jira, Asana, Trello.
- Communication: Slack, Microsoft Teams, Zoom.
- Documentation: Confluence, Notion.

7.2 Hardware Requirements

1. Development Workstations:
- Processor: Quad-core CPU (Intel i5/i7 or AMD Ryzen 5/7).
- RAM: Minimum 16 GB.
- Storage: SSD with at least 512 GB.
- Graphics: Integrated graphics are generally sufficient, but a discrete GPU may be beneficial for running multiple virtual machines or handling large datasets.

2. Test Devices:
- Desktops/Laptops: To test different screen sizes and resolutions.
- Tablets and Smartphones: Both iOS and Android devices to ensure the application works across various platforms.

3. Server Requirements:
- Development Server:
  - Processor: Quad-core CPU.
  - RAM: Minimum 16 GB.

- Storage: SSD with at least 512 GB.
- Operating System: Linux (Ubuntu or CentOS).

- Production Server:
  - Processor: High-performance multi-core CPU (Intel Xeon or AMD EPYC).
  - RAM: 32 GB or higher, depending on the expected load.
  - Storage: SSD with a minimum of 1 TB, with RAID configuration for redundancy.
  - Operating System: Linux (Ubuntu Server, CentOS).
  - Networking: High-speed internet connection with a reliable hosting provider.

4. Network and Infrastructure:
   - Reliable Internet Connection: High-speed internet for seamless collaboration and access to cloud services.
   - Networking Equipment: Routers, switches, and firewalls to ensure secure and efficient network traffic management.
   - Backup Solutions: Regular backup mechanisms, either through cloud services or physical backup solutions, to prevent data loss.

Summary

By ensuring the availability of the required software and hardware, the development and deployment of the real estate application using the MERN Stack can proceed smoothly. The chosen tools and technologies facilitate efficient development, robust testing, secure deployment, and effective monitoring, ultimately leading to a high-quality, scalable application.

Chapter-8

Conclusion

The development of the real estate application using the MERN stack (MongoDB, Express.js, React, Node.js) has successfully met its primary objectives of creating a robust, scalable, and user-friendly platform for real estate transactions. The unified technology stack enables efficient development and maintenance, while the application's architecture ensures high performance and scalability. Key features such as advanced search and filtering, multimedia property listings, secure user authentication, and in-app messaging provide a comprehensive solution that caters to the needs of buyers, sellers, and agents.

The project showcases significant improvements in development speed, performance, user experience, and scalability compared to traditional stacks. The use of modern development practices, including Agile methodology and CI/CD pipelines, has streamlined the development process and ensured continuous delivery of high-quality software. Robust security measures have been implemented to protect user data, and performance optimization strategies have been employed to maintain fast response times and high availability.

Overall, the real estate application positions itself as a competitive and innovative solution in the real estate market, offering enhanced functionalities and user experiences that align with the demands of today's digital landscape.

Recommendations

To ensure the continued success and improvement of the real estate application, the following recommendations are proposed:

1. Continuous User Feedback Integration:
   - Regularly collect and analyse user feedback to identify areas for improvement and to prioritize new features and enhancements.
   - Conduct usability testing sessions to refine the user interface and improve overall user experience.

2. Enhanced Security Measures:
   - Stay updated with the latest security best practices and implement regular security audits to identify and address potential vulnerabilities.
   - Consider adding multi-factor authentication (MFA) to enhance user account security.

3. Performance Optimization:
   - Continuously monitor application performance and implement optimizations as needed, such as refining database queries and optimizing frontend rendering.
   - Explore advanced caching mechanisms and content delivery networks (CDNs) to further reduce latency and improve load times.

4. Scalability Planning:
   - Plan for future scalability by adopting a microservices architecture if the application grows significantly in complexity and user base.
   - Invest in scalable infrastructure solutions, such as cloud services, to handle increasing traffic and data volumes efficiently.

5. Advanced Analytics and Reporting:

- Develop advanced analytics and reporting features to provide users with deeper insights into market trends, property performance, and user behavior.
- Utilize machine learning algorithms to offer personalized property recommendations and enhance search functionality.

6. Mobile Application Development:

- Consider developing a mobile application to complement the web platform, providing users with a seamless experience across devices.
- Ensure that the mobile app offers the same functionalities and performance as the web application, tailored for mobile usage.

7. Marketing and User Acquisition:

- Implement targeted marketing strategies to increase user acquisition and retention, leveraging social media, search engine optimization (SEO), and digital advertising.
- Build partnerships with real estate agencies and property management companies to expand the application's user base and property listings.

By following these recommendations, the real estate application can continue to evolve and remain competitive in the ever-changing digital marketplace, offering enhanced value to its users and stakeholders.

Bibliography

Books:

1. "Pro MERN Stack: Full Stack Web App Development with Mongo, Express, React, and Node" by Vasan Subramanian
2. "Learning React: A Hands-On Guide to Building Web Applications Using React and Redux" by Kirupa Chinnathambi
3. "MongoDB: The Definitive Guide" by Shannon Bradshaw, Eoin Brazil, and Kristina Chodorow

Official Documentation:

MongoDB: [MongoDB Documentation](https://docs.mongodb.com/) - Express.js: [Express.js Documentation](https://expressjs.com/) -

React: [React Documentation](https://reactjs.org/docs/getting-started.html) - Node.js: [Node.js Documentation](https://nodejs.org/en/docs/)

References

[1] Yogesh Baiskar, Priyas Paulzagade, Krutik Koradia, Pramod Ingole, Dhiraj Shirbhate, "MERN: A Full-Stack
Development", International Journal for Research in Applied Science & Engineering Technology (IJRASET) ISSN:
2321-9653; IC Value: 45.98; SJ Impact Factor: 7.538 Volume 10 Issue I Jan 2022.
[2] Sumangala A. Bafna, Pratiksha D. Dutonde, Shivani S. Mamidwar, Monali S. Korvate, Prof. Dhiraj Shirbhare,
"Review on Study and Usage of MERN Stack for Web Development" International Journal for Research in Applied
Science & Engineering Technology (IJRASET) ISSN: 2321-9653; IC Value: 45.98; SJ Impact Factor: 7.538 Volume
10 Issue II Feb 2022.
[3] Dr. Santosh Kumar Shukla, Shivam Dubey, Tarun Rastogi, Nikita Srivastava, "Application using MERN Stack"
International Journal for Modern Trends in Science and Technology, 8(06): 102-105, 2022.
[4] Sourabh Mahadev Malewade, Archana Ekbote, "Performance Optimization using MERN stack on Web Application",
International Journal of Engineering Research & Technology (IJERT) Vol. 10 Issue 06, June-2021.
[6] Dr. Poornima Mehta, Harsh Kumar, Amit Sharma, "STUDY POINT WEBSITE USING MERN STACK",
International Research Journal of Modernization in Engineering Technology and Science Volume:05/Issue:03/March-
2023.
[6] Kevin Goldberg IBM 2010; Developing dynamic Web sites with CodeIgniter https://developer.ibm.com/articles/oscodeigniter/?mhsrc=ibmsearch_a&mhq=MVC Accessed 24 March 2023
[7] Allen Wirfs-Brock, Brendan Eich 2020; JavaScript: The first 20 Years https://dl.acm.org/doi/10.1145/3386327 Accessed 24 March 2023
[8] Node.js; Child process https://nodejs.org/api/child_process.html#childprocess Accessed 24 March
[9] MongoDB; Full Stack Development Explained https://www.mongodb.com/languages/full-stack-development Accessed 24 March 2023

Source Code
File name-App.jsx

```
import { Suspense, useState } from "react";
import "./App.css";
import Layout from "./components/Layout/Layout";
import Website from "./pages/Website";
import { BrowserRouter, Routes, Route } from "react-router-dom";
import Properties from "./pages/Properties/Properties";
import { QueryClient, QueryClientProvider } from "react-query";
import { ReactQueryDevtools } from "react-query/devtools";
import { ToastContainer } from "react-toastify";
import "react-toastify/dist/ReactToastify.css";
import Property from "./pages/Property/Property";
import UserDetailContext from "./context/UserDetailContext";
import Bookings from "./pages/Bookings/Bookings";
import Favourites from "./pages/Favourites/Favourites";

function App() {
  const queryClient = new QueryClient();

  const [userDetails, setUserDetails] = useState({
    favourites: [],
    bookings: [],
    token: null,
  });

  return (
    <UserDetailContext.Provider value={{ userDetails, setUserDetails }}>
      <QueryClientProvider client={queryClient}>
        <BrowserRouter>
          <Suspense fallback={<div>Loading...</div>}>
            <Routes>
```

```
            <Route element={<Layout />}>
              <Route path="/" element={<Website />} />
              <Route path="/properties">
                <Route index element={<Properties />} />
                <Route path=":propertyId" element={<Property />} />
              </Route>
              <Route path="/bookings" element={<Bookings />} />
              <Route path="/favourites" element={<Favourites />} />
            </Route>
          </Routes>
        </Suspense>
      </BrowserRouter>
      <ToastContainer />
      <ReactQueryDevtools initialIsOpen={false} />
    </QueryClientProvider>
  </UserDetailContext.Provider>
 );
}

export default App;
```

File name- Map.jsx

```
import React from 'react'
import {MapContainer, TileLayer} from 'react-leaflet'
import GeoCoderMarker from '../GeoCoderMarker/GeoCoderMarker'
const Map = ({address, city, country}) => {
 return (
  <MapContainer
  center={[53.35, 18.8]}
  zoom={1}
  scrollWheelZoom={false}
  style={{
     height: "40vh",
     width: "100%",
     marginTop: "20px",
```

```
      zIndex: 0,

    }}
    >
      <TileLayer url='https://{s}.tile.openstreetmap.org/{z}/{x}/{y}.png'/>
      <GeoCoderMarker address={${address} ${city} ${country}} />
    </MapContainer>
  )
}

export default Map
```

File name-Schema.prisma

```
generator client {
  provider = "prisma-client-js"
}

datasource db {
  provider = "mongodb"
  url      = env("DATABASE_URL")
}

model User {
  id             String      @id @default(auto()) @map("_id") @db.ObjectId
  name           String?
  email          String      @unique
  image          String?
  bookedVisits     Json[]
  favResidenciesID String[]    @db.ObjectId
  ownedResidencies Residency[] @relation("Owner")
}
```

```
model Residency {
   id          String   @id @default(auto()) @map("_id") @db.ObjectId
   title       String
   description String
   price       Int
   address     String
   city        String
   country     String
   image       String
   facilities  Json
   userEmail   String
   owner       User     @relation("Owner", fields: [userEmail], references: [email])
   createdAt   DateTime @default(now())
   updatedAt   DateTime @updatedAt

   @@unique(fields: [address, userEmail])
}
```

File name- Index.js

```
import express from 'express';
import dotenv from 'dotenv';
import cookieParser from 'cookie-parser';
import cors from 'cors';
import { userRoute } from './routes/userRoute.js';
import { residencyRoute } from './routes/residencyRoute.js';
dotenv.config()

const app = express();

const PORT = process.env.PORT || 3000;

app.use(express.json())
```

```
app.use(cookieParser())
app.use(cors())

app.listen(PORT, ()=> {
   console.log(Server is running on port ${PORT});
});

app.use('/api/user', userRoute)
app.use("/api/residency", residencyRoute)
```

File name-Hero.jsx

```
import "./Hero.css";
import CountUp from "react-countup";
import { motion } from "framer-motion";
import SearchBar from "../SearchBar/SearchBar";
const Hero = () => {
  return (
    <section className="hero-wrapper">
      <div className="paddings innerWidth flexCenter hero-container">
        {/* left side */}
        <div className="flexColStart hero-left">
          <div className="hero-title">
            <div className="orange-circle" />
            <motion.h1
            initial={{ y: "2rem", opacity: 0 }}
            animate={{ y: 0, opacity: 1 }}
            transition={{
              duration: 2,
              type: "ease-in",
            }}
            >
              Discover <br />
              Most Suitable
              <br /> Property
```

```
    </motion.h1>
  </div>
  <div className="flexColStart secondaryText flexhero-des">
    <span>Find a variety of properties that suit you very easilty</span>
    <span>Forget all difficulties in finding a residence for you</span>
  </div>

  <SearchBar/>

  <div className="flexCenter stats">
    <div className="flexColCenter stat">
      <span>
        <CountUp start={8800} end={9000} duration={4} /> <span>+</span>
      </span>
      <span className="secondaryText">Premium Product</span>
    </div>

    <div className="flexColCenter stat">
      <span>
        <CountUp start={1950} end={2000} duration={4} /> <span>+</span>
      </span>
      <span className="secondaryText">Happy Customer</span>
    </div>

    <div className="flexColCenter stat">
      <span>
        <CountUp end={28} /> <span>+</span>
      </span>
      <span className="secondaryText">Awards Winning</span>
    </div>
  </div>
</div>

{/* right side */}
<div className="flexCenter hero-right">
  <motion.div
```

```
          initial={{ x: "7rem", opacity: 0 }}
          animate={{ x: 0, opacity: 1 }}
          transition={{
            duration: 2,
            type: "ease-in",
          }}
          className="image-container"
        >
          <img src="./hero-image.png" alt="houses" />
        </motion.div>
      </div>
    </div>
  </section>
 );
};

export default Hero;
```

PUBLISHED PAPER & CERTIFICATES