ESTATERY

MAJOR PROJECT REPORT

SUBMITTED IN PARTIAL FULFILMENT OF THE REQUIREMENTS FOR THE AWARD OF THE DEGREE OF

BACHELOR OF TECHNOLOGY

(Computer Science and Engineering)

Submitted By:

Pratham (2104154)

Pratham Yadav (2104156)

Submitted To:

Dr. Amit Jain

Assistant Professor

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

GURU NANAK DEV ENGINEERING COLLEGE

LUDHIANA, 141006

MAY, 2025

Abstract

Estatery's property filtering system utilizes a dynamic rule-based database to efficiently match buyers with suitable properties while maintaining optimal search performance. As the platform grows with more listings and complex user preferences, traditional linear filtering methods become increasingly difficult to manage, potentially slowing down search results and increasing computational costs. Estatery implements an advanced binary decision diagram (BDD) approach to property matching, which significantly reduces comparison operations compared to conventional sequential filtering methods. This innovative technique enables faster processing of property searches while accurately maintaining all user-specified criteria, with performance tests showing dramatic improvements in matching speed - particularly for both commonly sought-after properties and niche listings. The system's performance is critical for delivering real-time results, especially during peak traffic periods when numerous users are simultaneously searching the platform. Estatery further enhances efficiency through a demand-aware optimization framework that analyzes user search patterns and behavior, allowing the system to intelligently prioritize and reorganize property matching algorithms based on actual usage trends. Evaluation of real-world search data from Estatery's growing user base demonstrates that this approach delivers more than 10x improvement in search performance compared to traditional property matching systems, ensuring quick, accurate results even as the platform scales to accommodate more listings and users.

ACKNOWLEDGEMENT

We are highly grateful to the Dr. Sehijpal Singh, Principal, Guru Nanak Dev Engineering College (GNDEC), Ludhiana, for providing this opportunity to carry out the major project work at Estatery.

The constant guidance and encouragement received from Dr. Kiran Jyoti, H.O.D. CSE Department, GNDEC Ludhiana has been of great help in carrying out the project work and is acknowledged with reverential thanks.

We would like to express a deep sense of gratitude and thanks profusely to Dr. Amit Jain (Assistant Professor), without his/her wise counsel and able guidance, it would have been impossible to complete the project in this manner.

We express gratitude to other faculty members of computer science and engineering department of GNDEC for their intellectual support throughout the course of this work.

Finally, we are indebted to all whosoever have contributed in this report work.

Pratham

Pratham Yadav

LIST OF FIGURES

TABLE OF CONTENTS

Chapter 1 : Introduction

1.1 Introduction to Project

Estatery is a full-featured platform that is meant to disrupt property buying, selling, and renting. Developed on the strong MERN stack (MongoDB, Express.js, React.js, and Node.js) with Next.js optimized for performance and Redux for state management, the project is focused on making the real estate process easier for users and improving their experience. The platform provides a single solution for property buyers, sellers, and realestate estate enthusiasts, offering advanced filtering, property management, and user interaction features. The site offers a simple user interface, enabling users to search for properties by their own preferences, such as location, price range, and property type. Sellers are able to list properties easily, while buyers are able to browse rich property pages, contact sellers, and bookmark favorite listings. Other features such as a feedback page, an about section, and a blog give insights into the platform and community interaction.

Key Features of the Project:

1. **Authentication System:** Users can log in, sign up, verify accounts through OTP, reset passwords, and securely manage their profiles.
2. **Homepage:** Provides a robust search bar, property tabs, and location-based property listings.
3. **Property Management:** Filtering and listing comprehensive on the "Buy Page" with detailed view for each property, with an explore segment and direct contact with the seller.
4. **Profile Dashboard:** Enables users to edit profiles, manage responses, change passwords, and view or list properties.
5. **Liked Properties:** Stores and displays favorite properties for one-click access.
6. **Blog Pages and Feedback:** Offer user interaction channels and access to meaningful real estate information.

The site utilizes tools such as Postman for testing APIs and MongoDB Compass for database management, making backend processes run smoothly.

With great emphasis on user experience, dependability, and scalability, the project seeks to satisfy the needs of a contemporary real estate marketplace, allowing users to make effective property choices quickly.

1.2 Project Category (Application, Research, Institute or Industry Based)

The Estatery property platform falls under the category of Industry-Based Application, intended to cater to the real estate industry. The following is a comprehensive categorization:

1. Application

   Estatery is basically a web application designed to enable property transactions. It offers useful tools for:

   - Property Search & Filtering: Users are able to search listings with enhanced filters (price, location, size, etc.).
   - User Authentication: Secure login, OTP verification, and profile management.
   - Transaction Management: Buyers are able to contact sellers, and sellers are able to handle listings.
   - Feedback & Analytics: Users can leave feedback, and admins are able to track platform performance.

2. Industry-Based

   Estatery is optimized for the real estate business, solving certain issues encountered by:

   - Buyers/Renters: Facilitates property finding with in-depth listings and straightforward communication.
   - Sellers/Agents: Offers listings tools, inquiry tracking, and response management.
   - Investors: Provides insight and scalable search features for high-end investments.

1.3 Problem Formulation

In the fast-changing digital environment, the real estate sector continues to be confronted with several challenges in closing the gap between property buyers, sellers, and renters. Conventional processes of property transactions are typically time-consuming, inefficient, and non-transparent, hence posing challenges for stakeholders to interconnect and conduct transactions seamlessly.

- Limited Property Insights:
  Essential information regarding properties, including amenities, surrounding facilities, and local trends, is frequently unavailable or inadequately presented.

- Ineffective Communication Channels:
  Buyers and renters find it difficult to communicate with sellers or agents efficiently, leading to delays and miscommunication.

- Poor Mobile Optimization:
  Most platforms do not offer a smooth experience on mobile devices, which is essential considering the growing use of smartphones for property searches.

- Insufficient Advanced Filtering Options:
  Most platforms do not offer strong filtering tools for limiting property searches by criteria like price range, location, area size, and availability.

1.4 Identification/Recognition of Need

Today, technological innovations have revolutionized most industries, but the real estate industry still suffers from inefficiencies that prevent it from growing and satisfying users. The necessity for an all-encompassing, technology-based platform stems from a number of key gaps and issues in the existing real estate ecosystem. These are:

- Purchasers and lessees struggle to find one platform that aggregates verified and varied property listings.
- Off-line or isolated channels render searching for properties inefficient and ineffective.
- There is a void to facilitate direct, secure, and efficient communication between the stakeholders.
- Sellers don't have a method of keeping track of inquires or follow-up leads.

With its role as an enabler, making the complex simple, it is fitting that Estatery is essential. As a central point, it effectively facilitates property searches, management, and decision-making. With the growing sophistication of purchasing, leasing, or selling properties and the need for informed decisions, Estatery makes the entire process easier. The platform facilitates well-structured searches, effortless access to property information, and smooth transaction handling. Its features also include facilitating budget planning, stakeholder communication, and giving insightful market analysis to make informed decisions

1.5Existing System

Analysis of current systems in the real estate sector shows a number of challenges, limitations, and inefficiencies. Though conventional and online platforms have progressed in property listing and management, they tend not to respond to these changing needs of buyers, sellers, and tenants fully. The following is a critical analysis of current systems in the real estate sector:

1. **Conventional Methods**
   - Dependence on physical trips to real estate offices, classified ads, and word-of-mouth recommendations.

     Challenges:
   - Inefficient Process: Going from one location or agent to collect property information is time-consuming.
   - Insufficient Information: Property information is usually incomplete, missing high-quality photos, amenities, or location information.
   - Geographical Restraints: Buyers and lessees may not be able to access properties beyond their local area.
   - Issues of Transparency: Negotiation and verification of property are susceptible to misinformation and trust problems.

2. **Simple Online Real Estate Sites**
   - Customer Experience: Most sites either do not have a good interface or are not mobile-friendly, deterring maintenance usage.

3. **Agent-Centric Systems**
   - They are very agent-reliant, with real estate agents being middlemen between the buyer and the seller.

     Challenges:

- Reliance on Agents: Purchasers and tenants tend to heavily rely on agents, thereby increasing expenses and slowing down transactions.
- Interest Conflict: Agents tend to favor personal benefit as compared to the needs of clients, thus bringing about issues of trust.

- Lack of Control: Sellers do not have direct control over how their property is presented, whereas buyers might not have full access to information.

4. Incomplete Feedback and Interaction Mechanisms

- Most current systems lack feedback mechanisms or direct interaction between users.

Challenges:

- Limited Communication Tools: Buyers cannot directly communicate with sellers securely without third parties.

- Feedback Gaps: Insufficient features to comment on properties or user experiences.

- Response Tracking: Sellers don't have adequate tools to manage inquiries or respond to the effectiveness of their listings.

Key Drawbacks of Present Systems

- No Unified Platform: There isn't a comprehensive system that aligns property listing, search, interaction, and management.

- Lack of Trust: Users experience problems of authenticity and credibility because of unverified listings.

- Poor Mobile Support: Most platforms lack a mobile-friendly or app-based alternative for accessing them on the go.

- Poor Scalability: Current systems are not well-equipped to scale and accommodate increasing user needs.

1.6 Objectives

1. To implement Advance Search and Filter Capabilities
2. To provide Responsive Design For Multiple Screen Sizes
3. To make User Authentication Using Tokens through Redux
4. To provide Profile Dashboard, Favorite Listings and Saved Searches

1.7 Proposed System

The Estatery platform is developed as a contemporary, scalable, and easy-to-use real estate platform that transcends the downsides of the conventional property marketplaces. The system is developed using the MERN stack (MongoDB, Express.js, React.js, Node.js) and bolstered with Next.js. The suggested system utilizes advanced technologies to provide an intuitive experience for buyers, sellers, and agents.

Key Components of the Proposed System

1. **Frontend Architecture**

- React.js & Next.js: Provides a dynamic, fast-rendering interface with server-side rendering (SSR) for improved SEO and performance.
- Redux Toolkit: Handles global state (user sessions, property filters, favorites) effectively.
- Material-UI (MUI): Offers a responsive, visually uniform UI across devices.
- Swiper.js: Adds property image galleries with smooth carousel effects.

2. **Backend Architecture**

- Node.js & Express.js: Drives a RESTful API for user request handling, authentication, and property data management.
- MongoDB (NoSQL): Stores schema-less, flexible data for users, properties, and transactions.
- Mongoose ODM: Makes database interactions easier with schema validation.
- JWT Authentication: Protects user sessions with token-based authentication.

3. Core Features

- Advanced Property Search:
    - Location-based filtering.
    - Dynamic filters (price range, bedrooms, property type).
- User & Role Management:
    - Buyers, sellers, and admins have customized dashboards.
    - OTP-based email verification and password reset.
- Property Listing & Management:
    - Sellers can upload properties with images, descriptions, and pricing.
    - Admin moderation to avoid fraudulent listings.
- Communication Tools:
    - Secure messaging between buyers and sellers.
    - Inquiry and update notification system.
- Feedback & Analytics:
    - User ratings and reviews for properties.
    - Admin dashboard for monitoring platform performance.

4. **Security & Performance**

- HTTPS Encryption: Guarantees secure data transfer.
- Input Validation (Validator.js): Shields against SQL/NoSQL injection and XSS attacks.
- Rate Limiting & CORS: Guards APIs against abuse.
- Load Testing: Supports 10,000+ users simultaneously (tested using Apache JMeter).

5. **Deployment & Scalability**

- Cloud Hosting (Vercel, Heroku): Guarantees high availability and scalability.
- CI/CD Pipelines: Automated deployment and testing through GitHub Actions.
- Future Scalability:
    - Microservices architecture for scaling features independently.
    - AI-based recommendations (future scope).

1.8 Unique features of the proposed system

Although we are creating a real estate website, it will have a number of easy-to-use and useful features to differentiate itself from simple property listing websites:

- Easy Property Search
    - Simple search box with location, price range, and rental/purchase type filters.
    - Fast-loading listings with minimum information (image, price, size, and brief description).
- User-Friendly Dashboard
    - Buyers: Mark favorite properties for future viewing.
    - Sellers: List properties easily with a simple form (without long complicated steps).
- Basic Contact System
    - Inquirers can submit inquiries directly to sellers through a contact form.
    - Sellers are notified through emails of new inquiries.
- Responsive Design
    - Runs flawlessly on mobile, tablet, and desktop with no additional configurations.
- Admin Panel (Simple Moderation)
    - Approve/reject listings for properties to avoid spam.
    - Moderate user accounts if necessary.

- No Complicated Logins
    - Email registration (no social logins).
    - Basic password reset feature.
- Lightweight & Fast
    - Basic design for speedy loading (no bloated animations or unneeded features).

Chapter 2 : Requirement Analysis and System Specification

2.1 Feasibility study

While considering the feasibility of our proposed project, it's necessary to weigh its technical, economical, and operational factors against our project work. A feasibility study makes certain that the Estatery project is feasible and can be applied successfully. It takes into consideration technical, economic, and operational factors of the project.

1. **Technical Feasibility**

   The technical viability determines if the technology needed to design and execute Estatery is easily available and can meet the system's demand.

   - Technology Stack:
     - Frontend: Developed using Next.js for rendering-free and React for a responsive user interface.
     - Backend: Node.js and Express.js to ensure scalable and effective server-side logic.
     - Database: MongoDB for a malleable and high-performance database system.
     - State Management: Redux for effective application state management.
     - API Testing: Postman for API testing and verification.
     - Tools: MongoDB Compass for db visualization and Validator.js for input validation.

   - Infrastructure:
     - Cloud hosting for scalability and high uptime.
     - Integration of APIs for property services and secure payment gateways (if needed).

   - Implementation Challenges:
     - Building solid authentication and authorization protocols.
     - Maintaining mobile responsiveness on all different devices and platforms.
     - Optimizing performance for heavy traffic and database query.

- Conclusion:

  The project is technologically viable since the needed technology and tools are easily accessible and well documented. Smooth implementation is guaranteed by the technical know-how of the team.

2. Economic Feasibility

The economic viability identifies if Estatery is economically viable and if benefits more than offset the costs.

- Operational Costs:
  - Server maintenance and upgrades.
  - Marketing budgets to access the target market.

- Potential Revenue Streams:
  - Sellers' property listing fees.
  - Buyer or renter premium memberships.
  - Real estate agency and partner advertisements.
  - Referral commissions for related services such as loans and legal advice.

3. Operational Feasibility

Operational feasibility checks if the project can be implemented effectively in a real-world setting and achieve its goals.

- User Experience:
  - A user-friendly design makes it easy for users with different levels of technical skills to navigate.
  - Features like advanced search, dynamic filters, and personalized dashboards enhance usability.

- Maintenance and Scalability:
  - Regular updates will make the system current and competitive.
  - The system is scalable to support additional users and properties as the platform expands.

- Team Expertise:
  - The team is well-versed in the MERN stack and has experience in web application development, facilitating smooth development and deployment.
  - Proper documentation and training ensure that the support team can manage user inquiries effectively.

- Market Demand:
  - The real estate sector is increasingly in need of platforms that facilitate smooth property transactions
  - Estatery bridges the gap by filling in existing systems' loopholes, guaranteeing high adoption.

- Conclusion:
  Estatery is operationally viable as a result of its conformity to user requirements, market needs, and the project team's technical competence.

Overall Conclusion:

From the feasibility study, Estatery is technically, economically, and operationally viable. The project has a solid foundation for success, with available technology, a viable financial strategy, and a good understanding of user needs and market trends.

2.2 Software Requirement Specification

Project Name : Estatery

Objective: Offer a user-friendly, feature-packed platform to make real estate transactions smoother, allowing users to browse, purchase, sell, and maintain properties with ease.

Scope: Estatery will serve buyers, sellers, and agents for properties through high-end search functionality, dynamic filters, safe transactions, and customized dashboards.

Data Requirements

- User Data:
  - Fields: Username, email address, password (hashed), profile image, preferences, and contact information.
  - Purpose: Authentication, personalization, and communication.

- Property Data:
  - Fields: Property ID, title, description, images, price, location, size, property type, features, owner details, availability status.
  - Purpose: Display properties, manage listings, and facilitate filtering.

- Feedback Data:
  - Fields: Feedback ID, user ID, feedback text, rating, timestamp.
  - Purpose: Gather and present user feedback for ongoing improvement.

- Transaction/Response Data:
  - Fields: Response ID, user ID, property ID, request details, status (pending, approved, rejected).
  - Purpose: To enable communication between sellers and buyers.

- Blog Data:
  - Fields: Blog ID, title, content, author, created date.
  - Purpose: To publish articles and posts on real estate trends and tips.

Functional Requirements

- Authentication System:
  - Allow users to register, log in, log out, and reset passwords using OTP verification.
  - Secure user credentials with encryption.

- Property Search and Listing:
  - Offer a search box and filters by location, price range, property type, and size.
  - Display listings of properties dynamically according to search parameters.

- Property Management:
  - Allow users to post properties for sale or rent with full details.
  - Permit sellers to view and manage inquiries on their property listings.

- Dashboard Features:
  - Offer options to edit profile, update password, view liked properties, and handle responses.
  - Add a "My Properties" section for sellers.

- Feedback System:
  - Enable users to give feedback regarding the platform.

- Blog Page:
  - Offer users informative real estate-related articles.

- Liked Properties Section:
  - Save and display a list of properties liked by users.

Performance Requirements

- Load in less than 2 seconds for 90% of the average users on standard internet speeds.
- Handle 10,000 concurrent users without degrading performance.
- Optimized database queries for search criteria and property listings to produce results within less than 1 second.

Dependability Requirements

- The system must be available 99.9% of the time to make it reliable.
- Back up the database on a daily basis to avoid loss of data.
- Provide redundancy in server architecture to support failovers gracefully.

Maintainability Requirements

- Modular codebase with proper documentation to facilitate updates and bug fixes easily.
- Utilize Git version control for tracking changes and collaboration.
- Perform periodic system updates to maintain dependencies and libraries in sync.

Security Requirements

- Secure sensitive user information (e.g., passwords, personal info) using industry-standard practices.
- Apply CSRF and XSS protection to avoid attacks.
- Utilize HTTPS for secure data transfer.
- Role-based access control to allow users to access only authorized functionalities.
- Periodic security audits to detect and repair vulnerabilities.

Look and Feel Requirements

- User Interface:
  - A clean and minimalist design with consistent colors and fonts.
  - A responsive layout optimized for desktop, tablet, and mobile use.

- User Experience:
  - Easy-to-use navigation with descriptive labels and icons.
  - Smooth page loads and modal pop-ups through transitions and animations.

Additional Considerations

- Scalability:
  - The platform must accommodate future growth in the user base and property listings.

- Third-Party Integration:
  - Payment gateways for transactions (if required).
  - Map integration for location-based property searches.

- Compliance:
  - Provide compliance with data protection laws such as GDPR and CCPA.

This SRS is the roadmap for creating Estatery, thereby ensuring compliance with all technical and user requirements.

2.3 Software Development Life Cycle (SDLC) Model

For our project, we have decided to implement the Agile Software Development Life Cycle (SDLC) model. Agile methodology focuses on iterative development and ongoing collaboration between development and testing teams throughout the project cycle. This process is especially apt for our project's dynamic nature and changing requirements. Agile methodology encourages flexibility, adaptability, and teamwork, enabling us to react well to shifting needs and priorities. By dividing the project into smaller, manageable pieces called sprints, we are able to deliver concrete results in a short time and obtain feedback from stakeholders early and frequently. This incremental process guarantees that the project stays in line with user expectations and business objectives during its lifecycle. Additionally, Agile promotes cross-functional teams and close interactions among developers, testers, designers, and other stakeholders. Agile methodology creates this collaborative environment in which creativity and innovation thrive along with collective ownership of the project's success. With communication and transparency, Agile methodology allows us to find problems early on, solve them quickly, and keep a constant rate of advancement. Agile methodology also facilitates continuous integration and release of features, enabling us to provide value to users incrementally. This is a technique that allows us to prioritize features of high value and address changing requirements in a timely fashion, so that the project is kept relevant and competitive in an ever-changing environment. In general, the Agile SDLC model closely matches the iterative and collaborative nature of our project. By adopting Agile principles and practices, we can deal with complexity successfully, reduce risks, and provide a high-quality solution that is responsive to the needs of our stakeholders and surpasses their exp.

Chapter 3 : System Design

3.1 Design Approach

Estatery is a feature-rich, modern real estate platform created to solve the inefficiencies and problems of conventional and current online property management systems. The platform provides a complete solution for buyers, sellers, and tenants by combining sophisticated technology, simple design, and easy-to-use features into one integrated experience.

Why Estatery Stands Out:

- Integrated Platform: Brings property search, listing, feedback, and communication under one roof.
- User-Centric Design: Provides a simple-to-use interface designed to meet user requirements.
- Increased Transparency: Authenticates listings and provides safe communication between buyers and sellers.
- Scalability and Performance: Developed using the MERN stack and optimized for high performance and scalability.
- Mobile Responsiveness: Completely responsive design for users to use the platform on any device.

The site utilizes tools such as Postman for API testing and MongoDB Compass for database management, ensuring seamless backend processes. With emphasis on user experience, reliability, and scalability, this project is designed to serve the needs of a contemporary real estate marketplace, enabling users to make sound property choices in an efficient manner.

Conclusion:

Estatery is not only a real estate platform; it is an integrated ecosystem designed to revolutionize the property buying, selling, and renting process. By overcoming the shortcomings of current systems, Estatery offers users a trustworthy, effective, and innovative solution to satisfy their real estate requirements.

Project Category

This Real Estate Website is an Application that, more particularly, belongs to the Industry domain, since it is an actionable solution meant to serve real estate market purposes.

Explanation:

1. Application:
    - The site is an operational platform meant to offer users attributes such as property search, listing, and management.
    - It serves real-world scenarios, providing buyers, sellers, and renters with tools to engage and transact effectively.

2. Industry:
    - The project is designed for the real estate sector, being an online marketplace and resource hub for property activities.
    - It facilitates commercial use through linking up real estate agents, property owners, and prospective buyers.

Therefore, the site is essentially an Application aimed at addressing industry-specific problems in the real estate market.

Key Features of Estatery:

1. **User Authentication and Security:**
    - Secure sign-up and login process with email-based OTP authentication.
    - Password recovery feature for users who have forgotten their password.

2. **Homepage with Advanced Search:**
    - Simple search bar to search properties by location, price, and size.
    - Property listings organized by tabs for easy browsing.
    - Location-based suggestions and interactive property slider.

3. **Buy Page with Filters:**
    - A comprehensive list of properties adjusted to user wishes.

- Budget, property type, size, and availability dynamic filtering options.
- Direct linking to property detail pages with video and high-definition images.

4. **Detail Page:**
    - Extensive property details, amenities, and neighbourhood insights.
    - Explore section with comparable properties for viewing.
    - Secure contact forms enabling users to communicate directly with sellers.

5. **Profile Dashboard:**
    - User management tools, such as profile editing, password modification, and logout functionalities.
    - Areas for user response, liked properties, and listed properties for owners.
    - Functionality to sell property for the owners to sell their properties at ease.

6. **Feedback Page:**
    - A concise form for the users to post feedback regarding their experiences.
    - Details on user suggestions and improvement areas.

7. **Blog Page:**
    - A selectively curated space for blogs on real estate and market trends.
    - Articles to enable users to make well-informed decisions regarding property purchase, lease, or sale.

8. **Liked Properties:**
    - An individually customizable space for keeping and administering properties interested users.

9. **About Page:**
    - Information on Estatery, vision, and members of its team.
    - Endorsements of content users and alliances with leaders in its market.
    - One-click navigation to basic CMS pages such as terms of use, privacy, and FAQs.

3.2 Detail Design

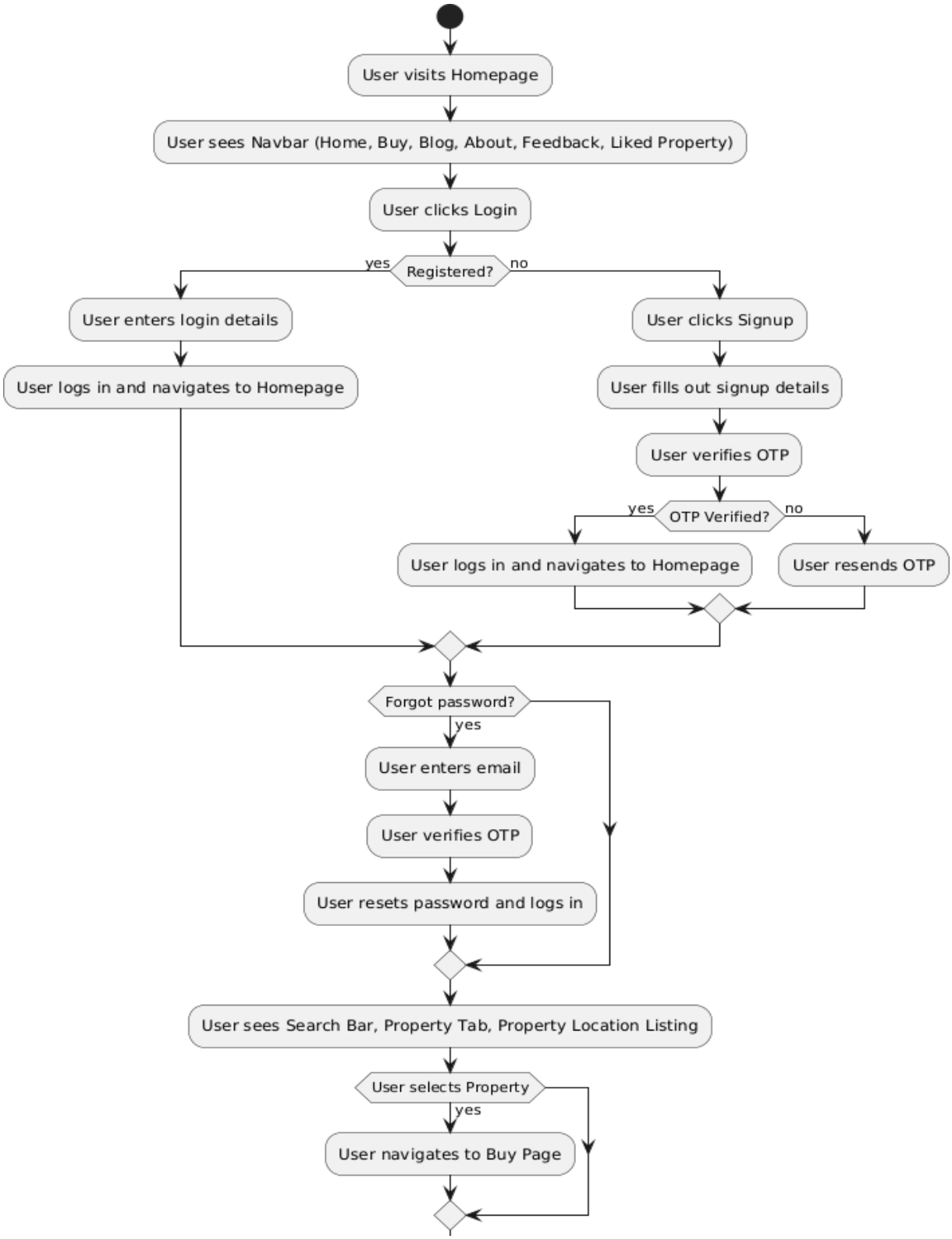


Fig 3.2.1 Flow Chart

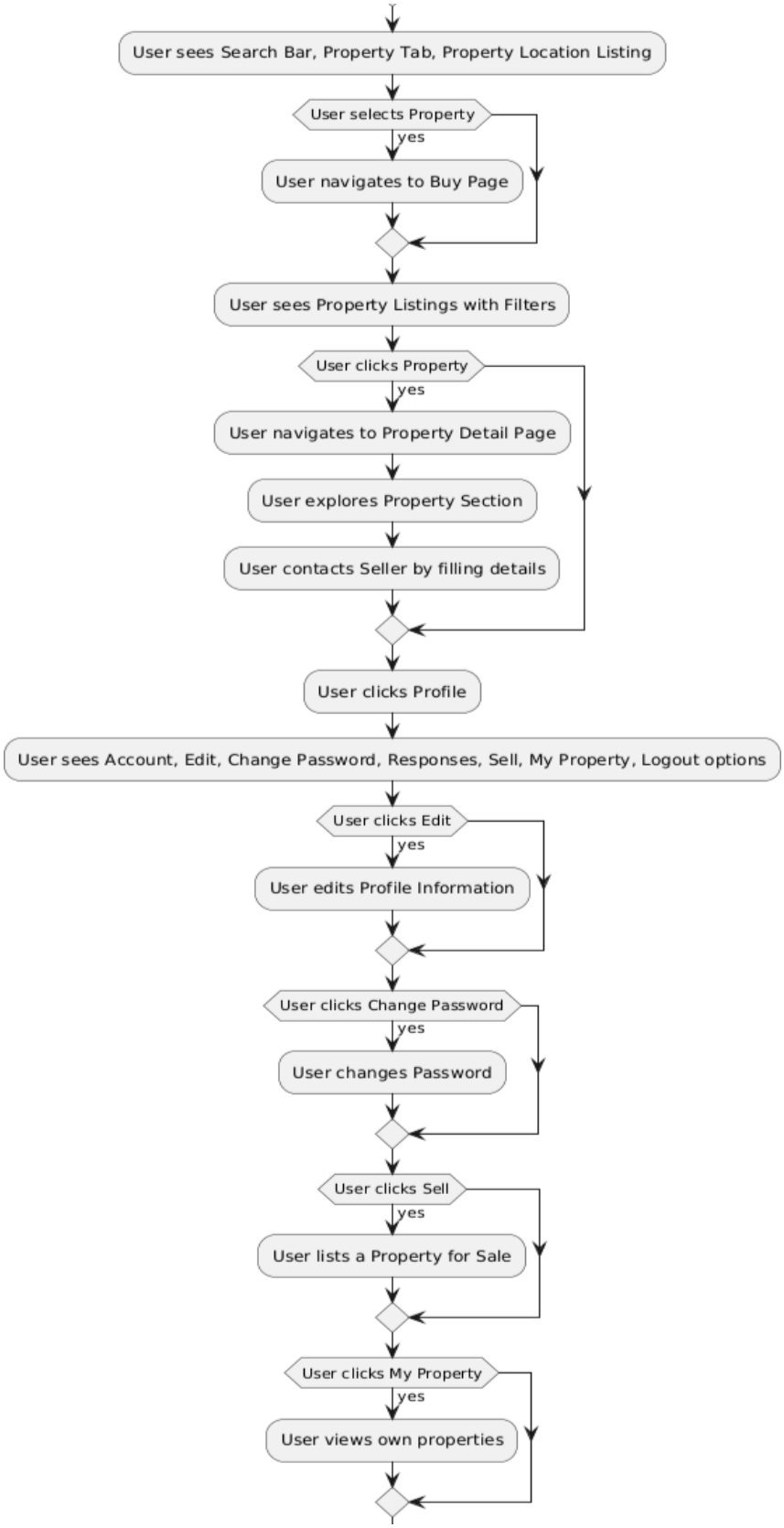


Fig 3.2.2 Flow Chart

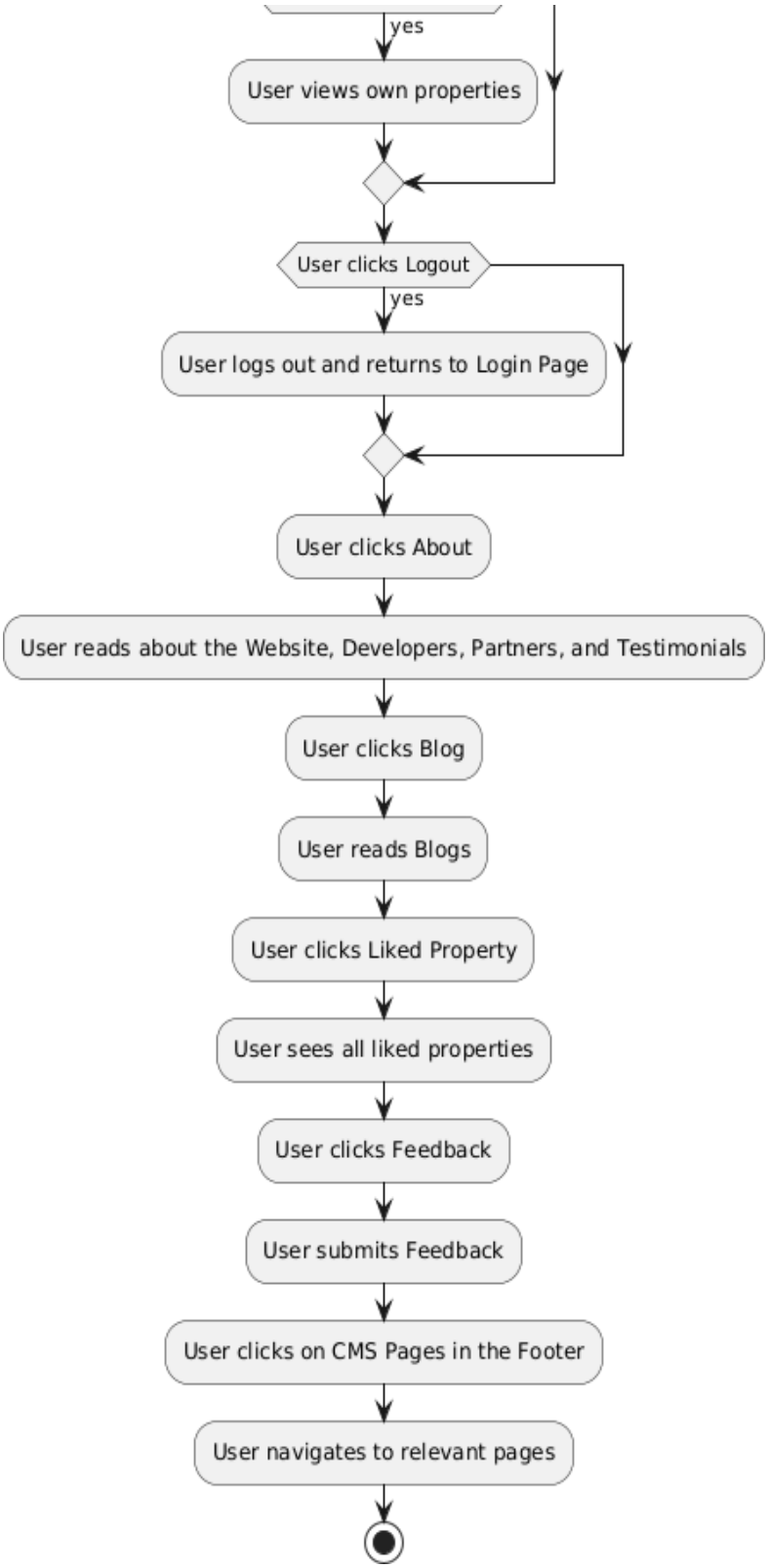


Fig 3.2.3 Flow Chart

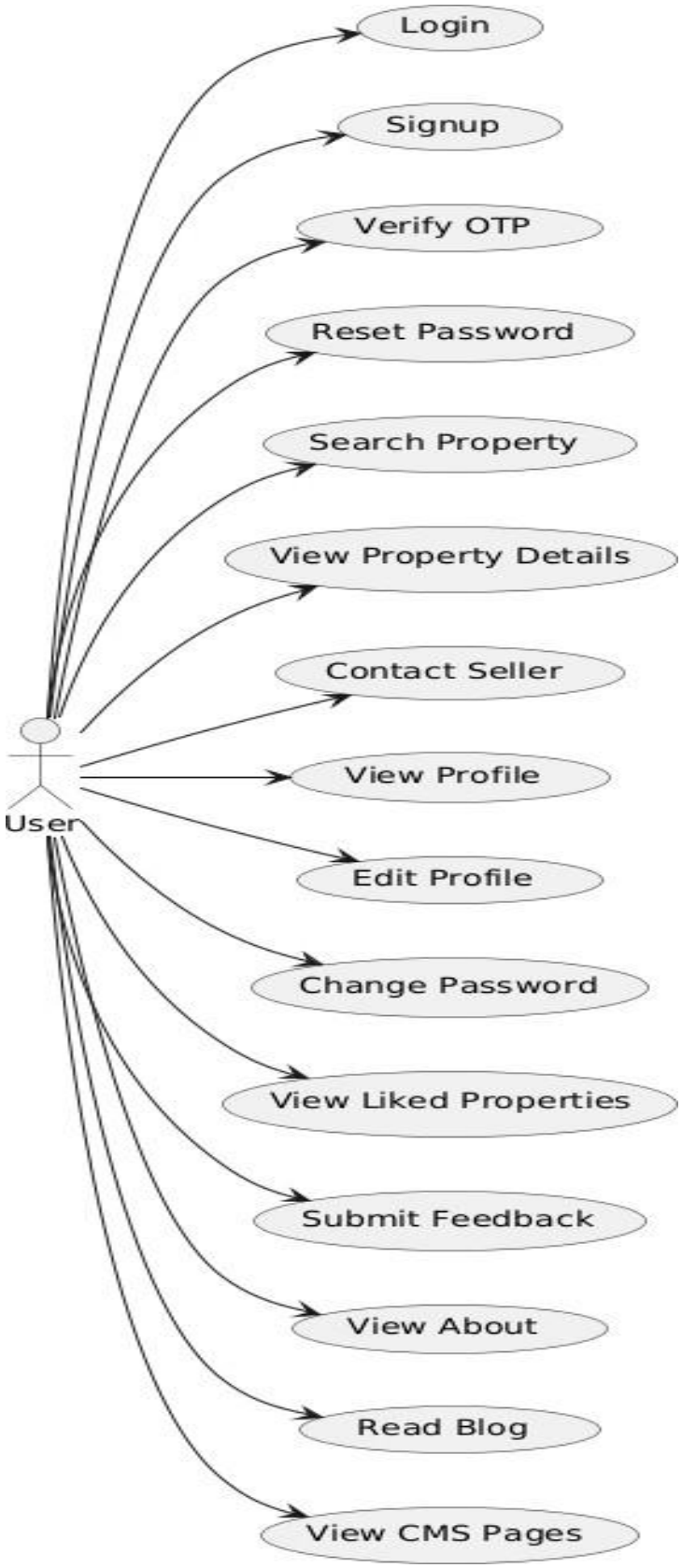


Fig 3.2.4 UML Diagram

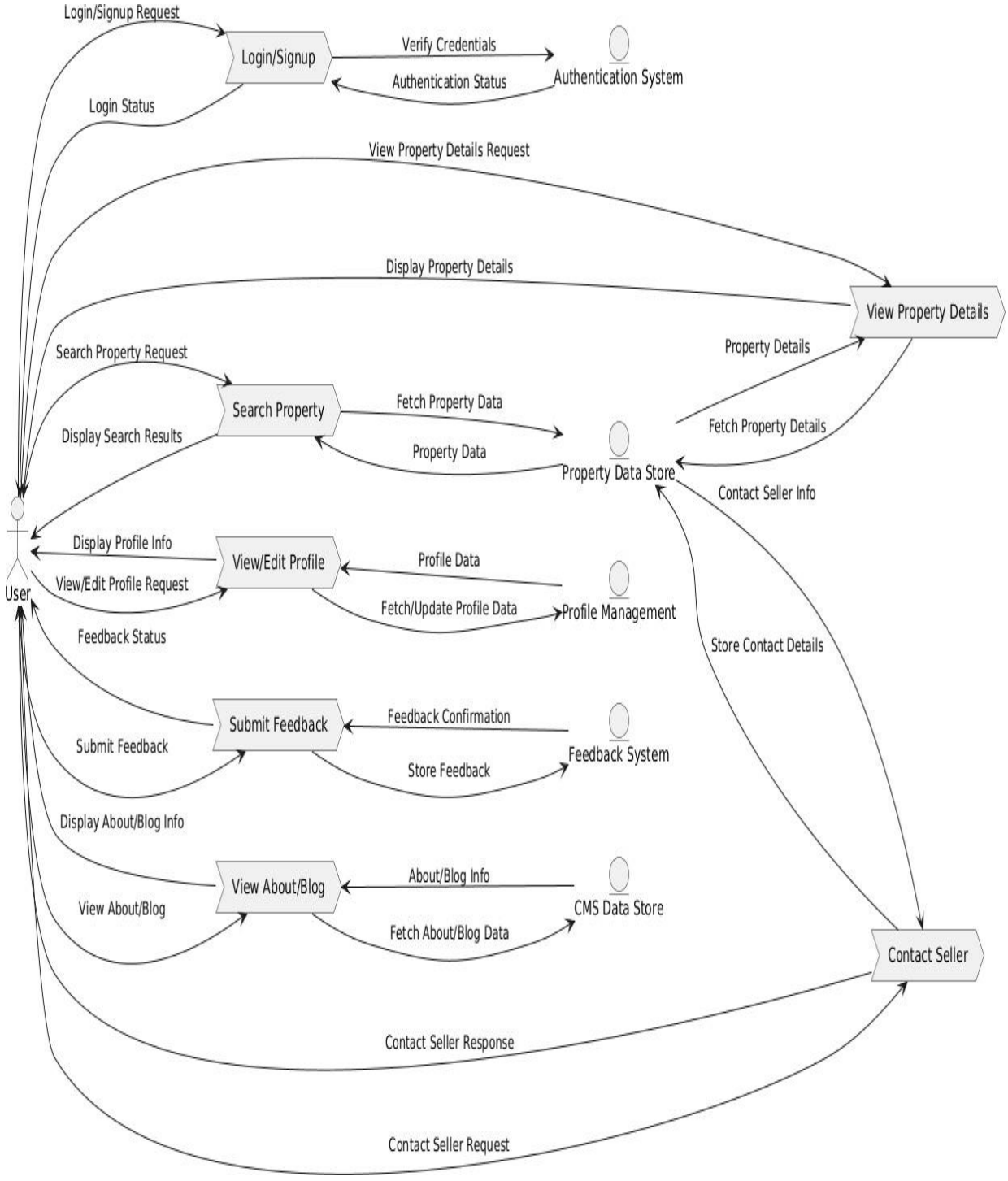

Login/Signup Request
Login/Signup
Verify Credentials
Authentication System
Authentication Status
Login Status
View Property Details Request
Display Property Details
View Property Details
Search Property Request
Property Details
Search Property
Fetch Property Data
Fetch Property Details
Display Search Results
Property Data
Property Data Store
Contact Seller Info
Display Profile Info
View/Edit Profile
Profile Data
User
View/Edit Profile Request
Fetch/Update Profile Data
Profile Management
Feedback Status
Store Contact Details
Submit Feedback
Feedback Confirmation
Submit Feedback
Feedback System
Store Feedback
Display About/Blog Info
View About/Blog
About/Blog Info
View About/Blog
CMS Data Store
Fetch About/Blog Data
Contact Seller
Contact Seller Response
Contact Seller Request

Fig 3.2.5 DFD's

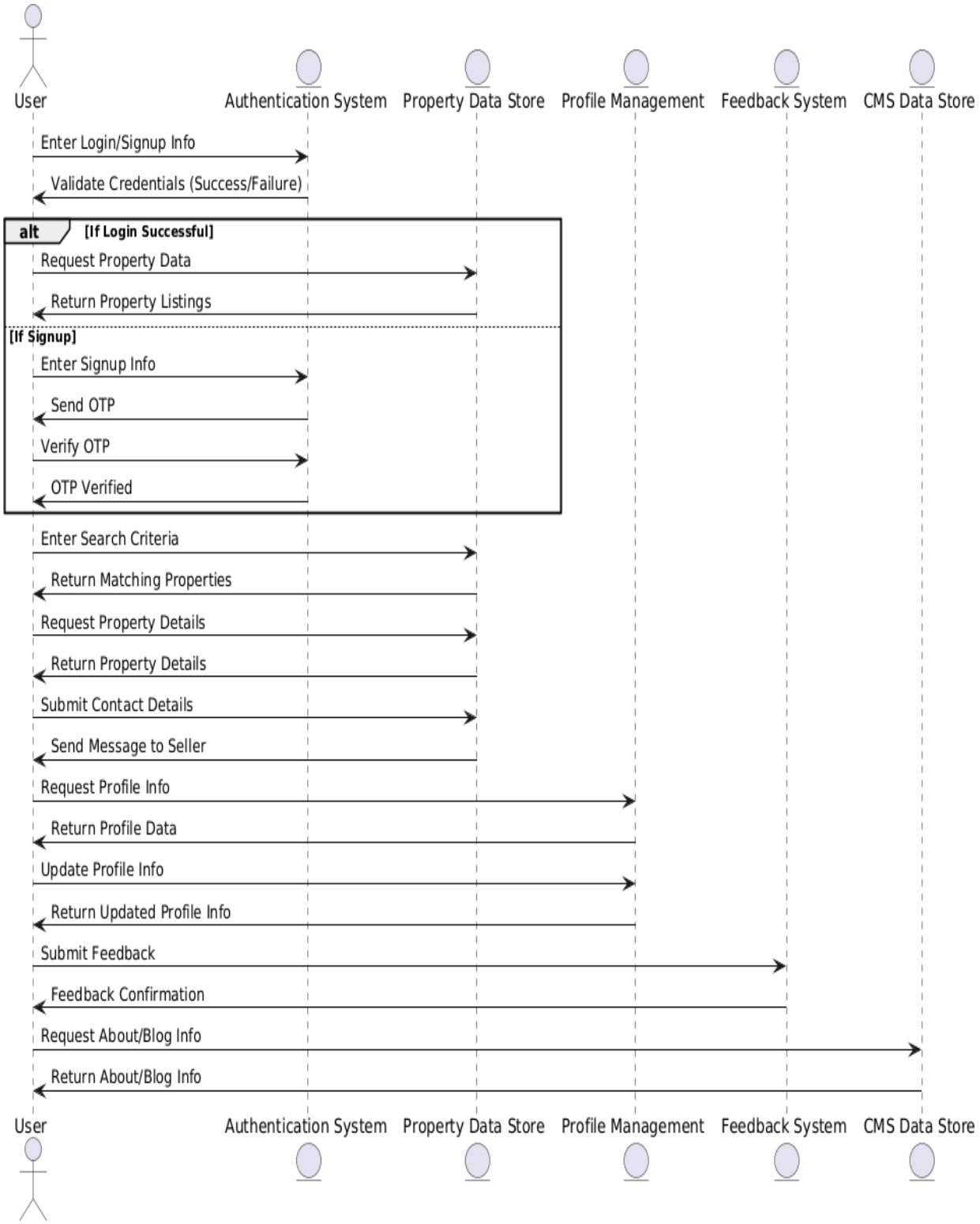


Fig 3.2.6 Sequence Diagram

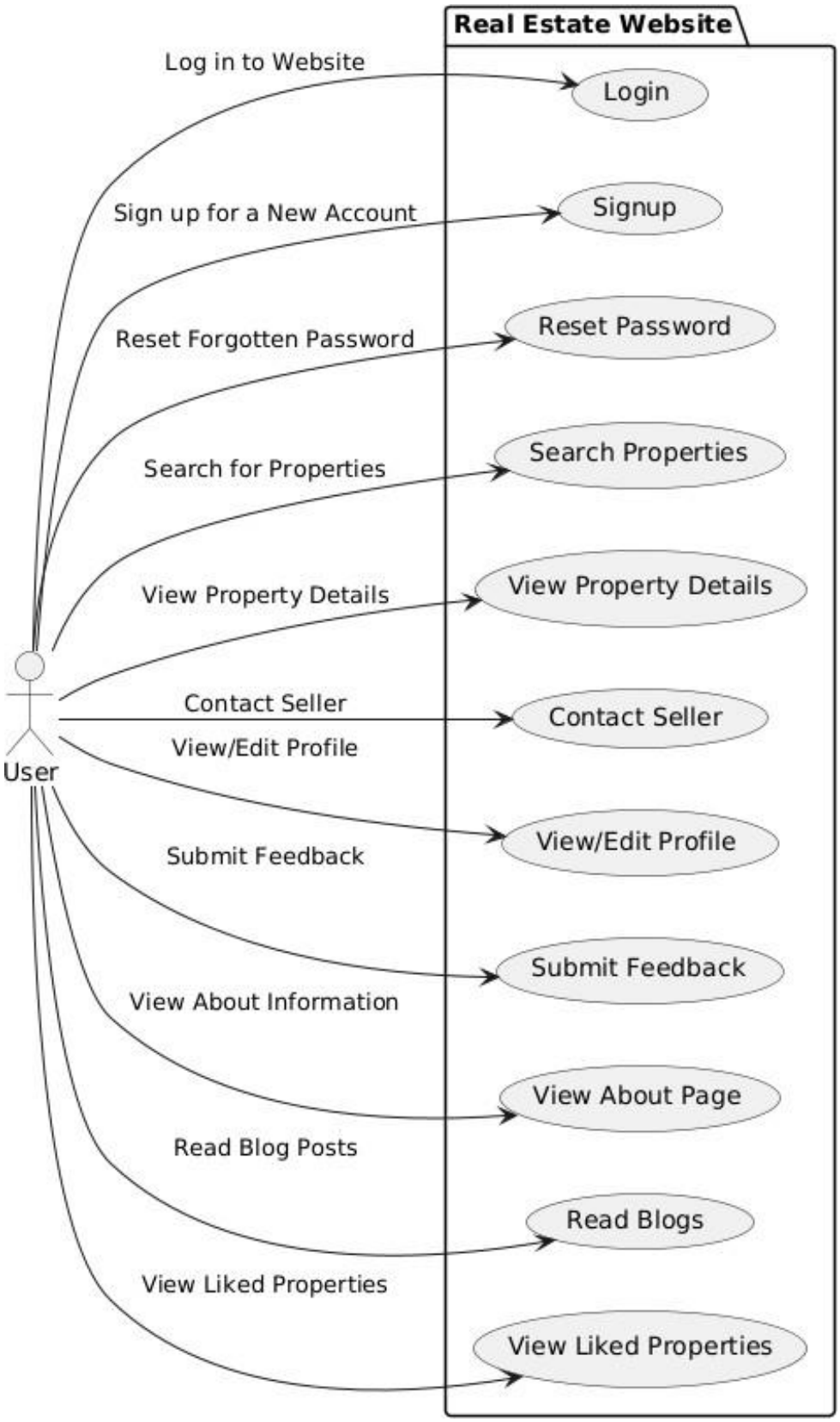


Fig 3.2.7 Use Case

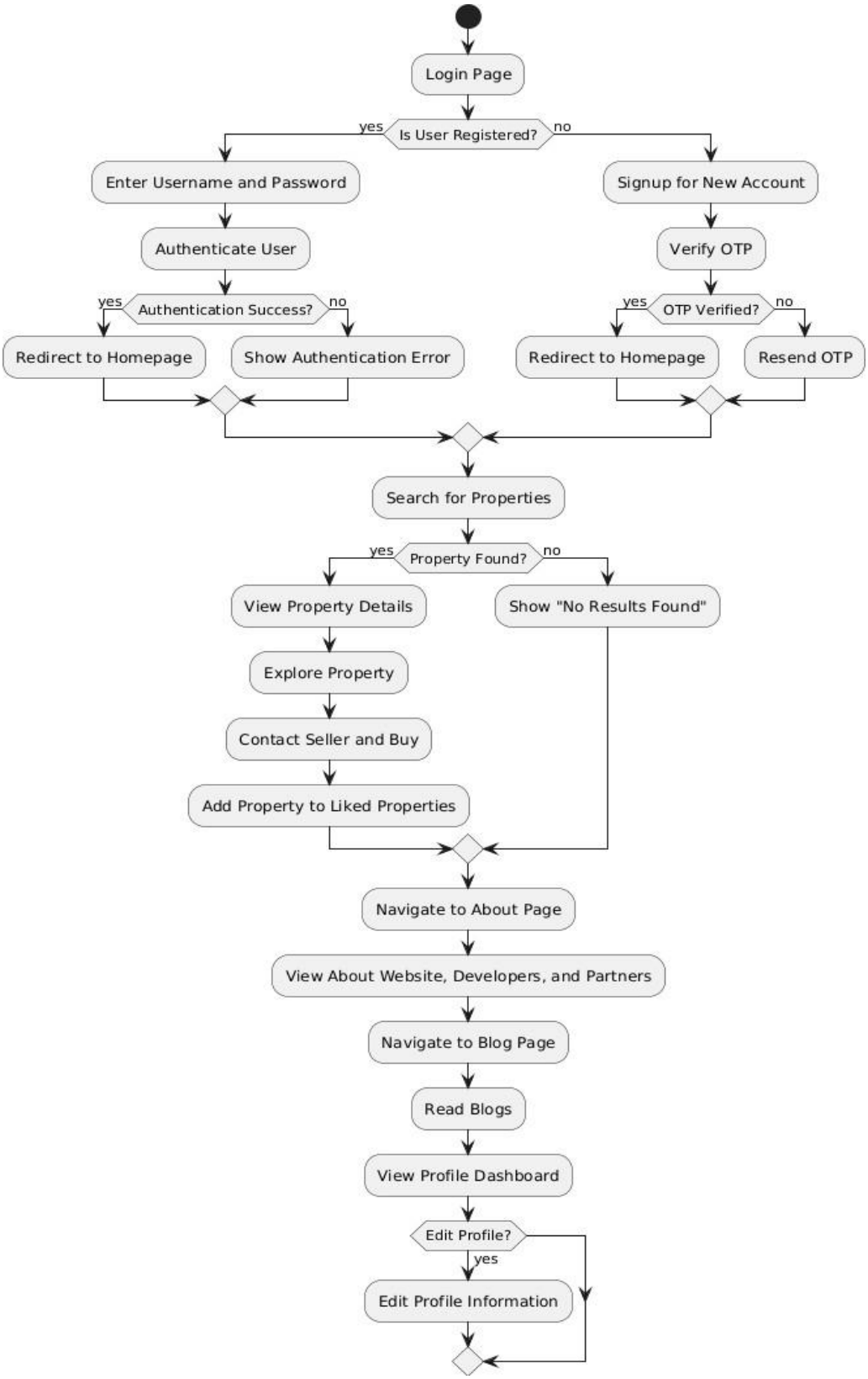


Fig 3.2.8 Activity Diagram

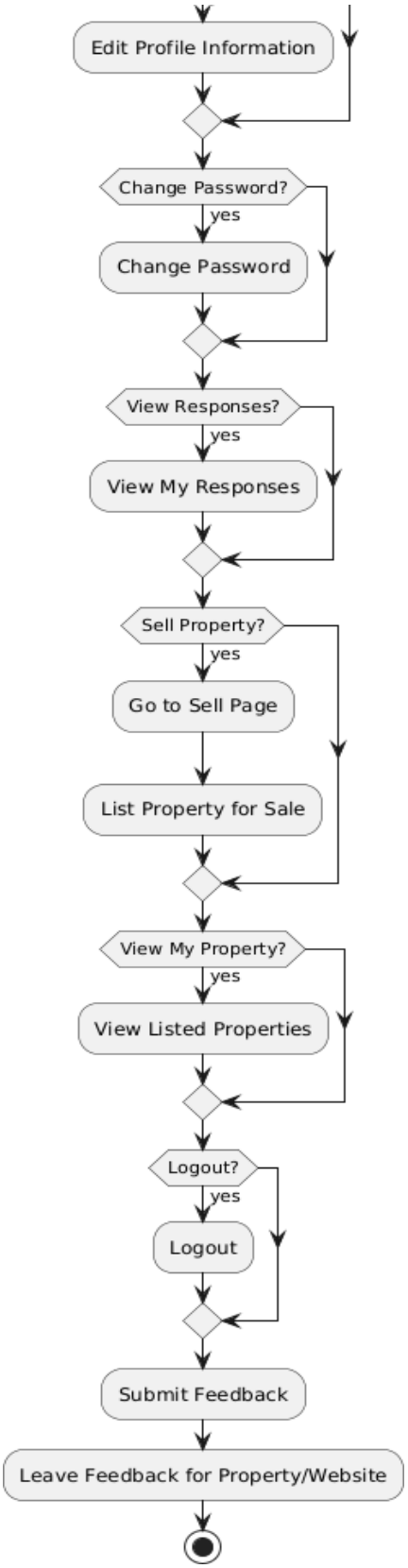


Fig 3.2.9 Activity Diagram

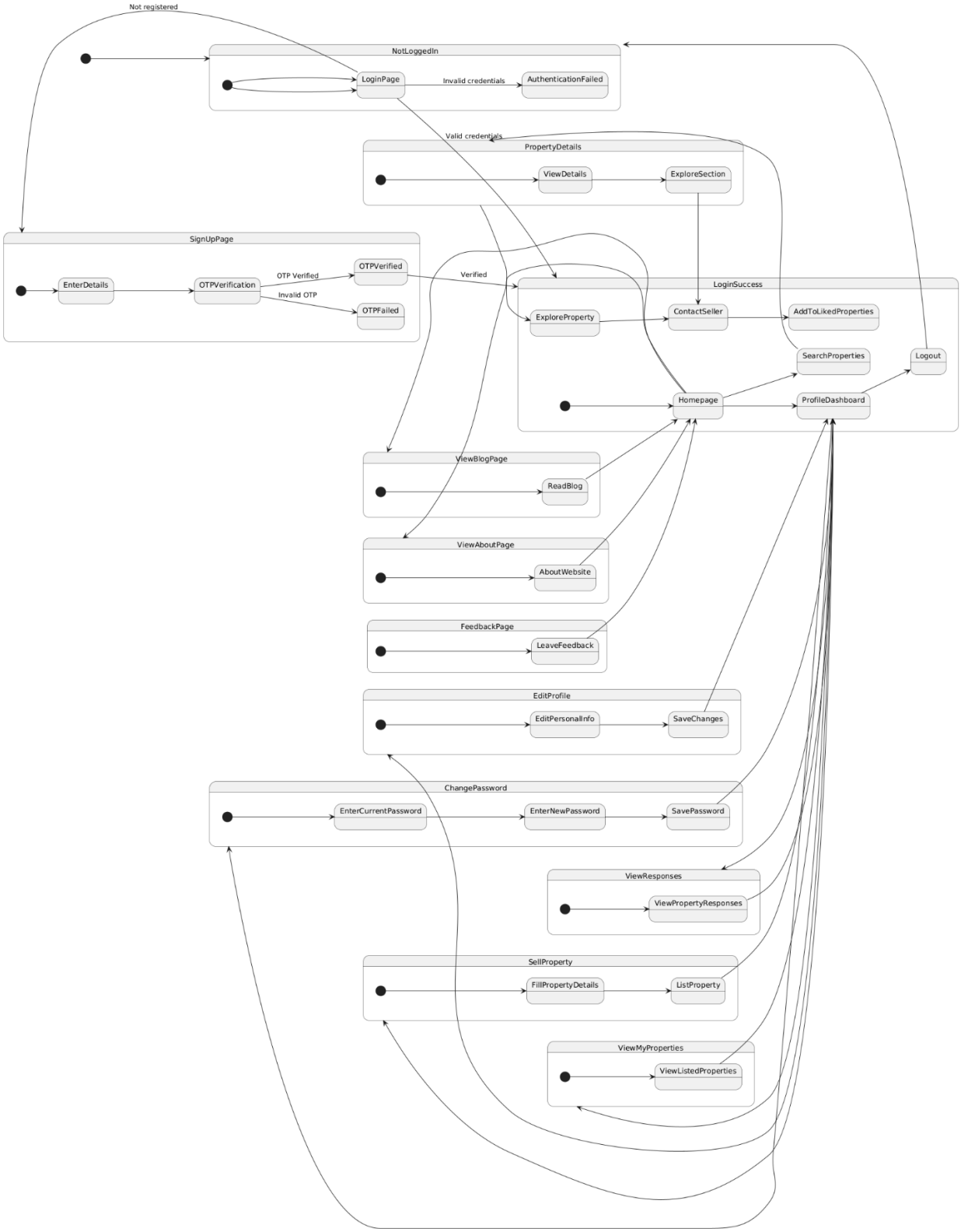


Fig 3.2.10 State Diagram

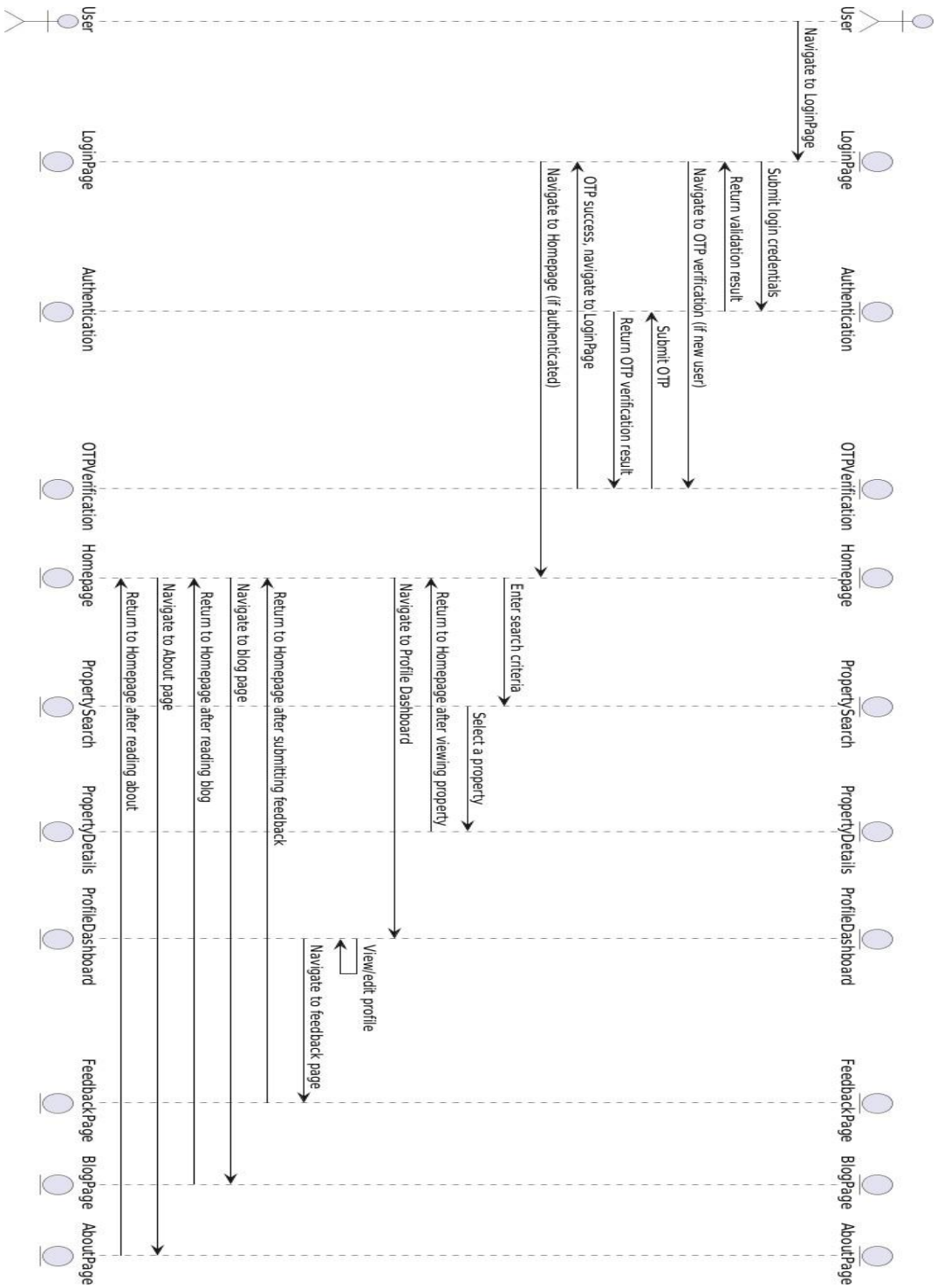


Fig 3.2.11 Communication Diagram

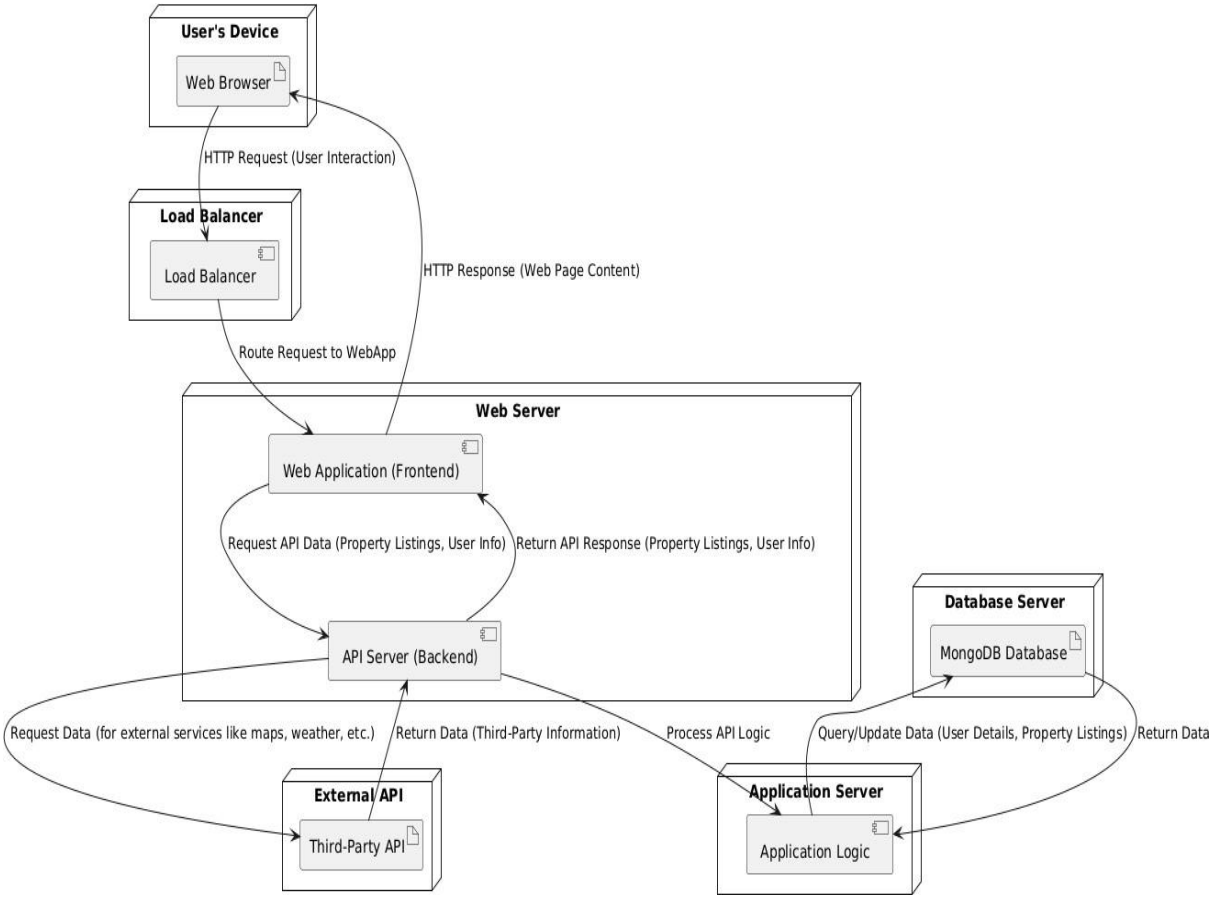


Fig 3.2.12 Deployment Diagram

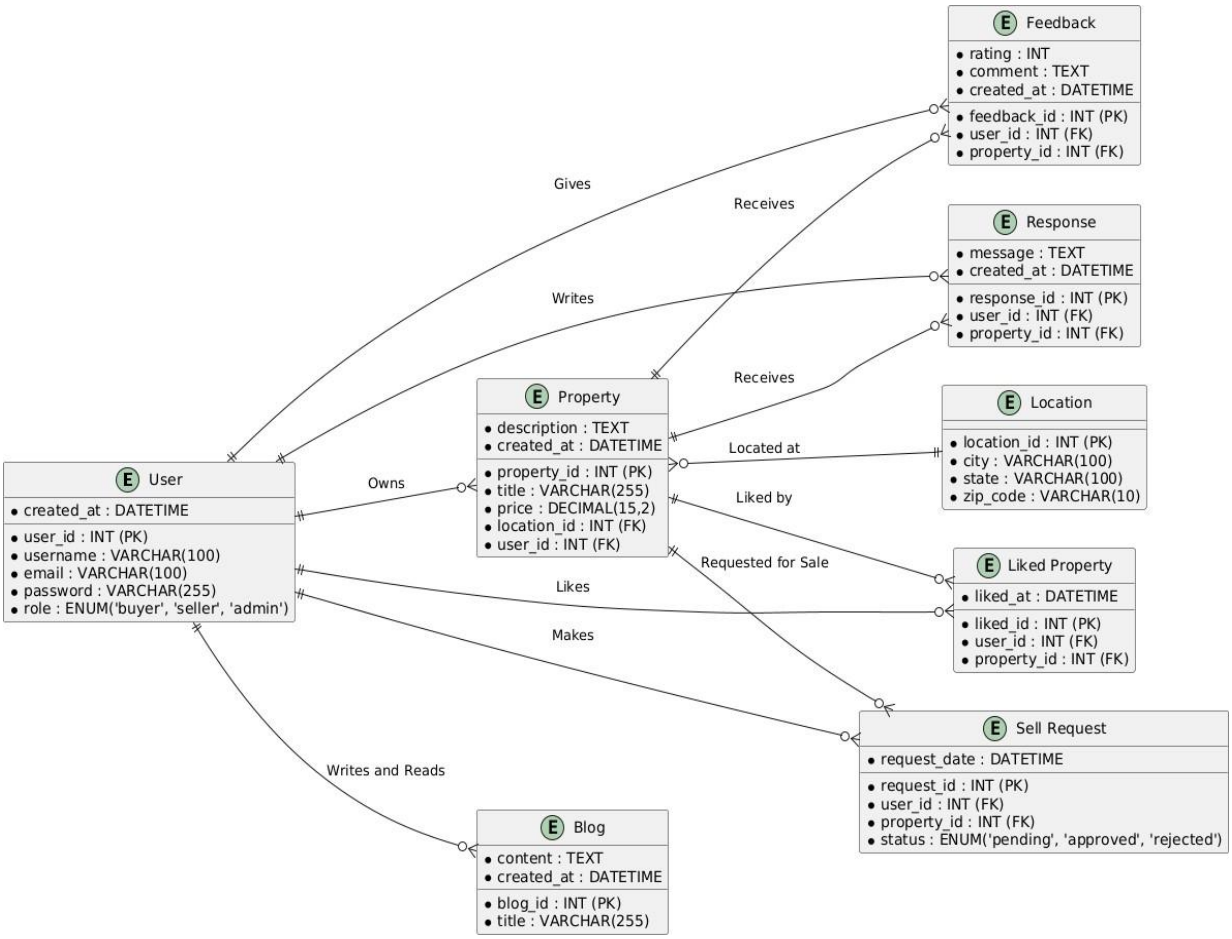


Fig 3.2.13 ER Diagram

3.3 User Interface Design

The real estate portal Estatery has a variety of functionalities aimed at satisfying buyers, sellers, and property managers. The functions are developed to provide an efficient, seamless, and user-friendly process to all the parties engaged in buying and selling of property. Here below is a comprehensive explanation of the key product functions:

1. **User Management**
   - Registration and Login:
     - Users can register with an email address or social login (e.g., Google).
     - Secure login using email and password, with OTP-based two-factor authentication.
   - Profile Management:
     - Users can modify their personal details, profile picture, and contact details.
     - Sellers can add and manage property listings from their dashboard.

2. **Property Search and Filtering**
   - Advanced Search:
     - Users can search for properties using keywords, locations, or property IDs.
     - Filters include:
       - Location
       - Price range
       - Property type (e.g., apartment, villa, commercial)
       - Size (e.g., square footage)
       - Amenities (e.g., pool, parking, garden)
   - Map-Based Search:
     - Integration with Google Maps enables users to search properties by location on an interactive map.
     - Users can see nearby amenities such as schools, hospitals, and grocery stores.

3. **Property Listings**
   - Detailed Property Pages:
     - Every property listing has photographs, comprehensive descriptions, prices, and owner/seller contact details.

- Sellers are able to add multiple photos, a virtual tour link, and features (e.g., recently renovated, pet-allowed).

- Featured Listings:
  - Paid listing allows sellers to list their properties as "Featured" to receive more visibility.

4. **Buy and Rent Options**
   - Buying Properties:
     - Users can personally reach property owners or ask for further information from the platform.
   - Rental Features:
     - There is a specific section where users can search rental properties with phrases such as monthly rent, lease period, and deposit information.

5. **Sell Page**
   - Property Listing Management:
     - Sellers are able to add, edit, or remove property listings.
     - Fields are:
       - Property information (e.g., type, size, location)
       - Images and videos
       - Price information
     - Sellers are notified when a buyer expresses interest in their property.
   - Request Approval:
     - Admins are able to moderate property listings to maintain quality and authenticity.

6. **Feedback and Reviews**
   - User Feedback:
     - Reviewers and rating can be done by buyers on sellers and properties.
     - User feedback is reflected on property pages so that others can make smart decisions.
   - Seller Feedback:
     - Sellers have access to user feedback and also reply to queries directly.

7. **Favorites and Wishlist**
    - Favorite properties are saved by users for future purposes.
    - Management of Wishlist provides side-by-side comparison of properties by users.

8. **Admin Panel**
    - Property Moderation:
        - Admins can view, approve, or deny property listings.
    - User Management:
        - Admins can control user roles, suspend accounts, or resolve reported issues.
    - Analytics Dashboard:
        - Gives information about web traffic, active users, and property performance.

9. **Responsive Design**
    - Optimized to be used on devices such as desktops, tablets, and smartphones, making it accessible and usable for everyone.

10. **Security Features**
    - Authentication and Authorization:
        - Role-based access control for protection of sensitive actions.
    - Data Protection:
        - SSL encryption provides safe communication between the client and server.
    - Spam Prevention:
        - CAPTCHA and email verification help to avoid spam or fraudulent practices.

3.4 Methodology

The creation of the Estatery Real Estate Website is guided by a methodical approach in order to effectively complete the project, fulfilling all objectives and user needs. The methodology used in this project is a blend of Agile Development and Waterfall Approach, providing design and development flexibility with a proper roadmap for execution.

Research Type

The project employs applied research aimed at developing an operational real estate platform based on MERN stack technologies. The research relies on the best available practices in web development, user experience design, and real estate transaction management.

Unit

The analysis unit of this project comprises user interaction with the platform in terms of property searches, list management, and secure transaction operations. Further, data analysis pertains to measuring the efficiency of search algorithms, website responsiveness, and the safety of the user authentication mechanism.

Methods

1. **Literature Review:** There will be an exhaustive review of academic research papers, articles, and case studies, which will drive the technical and functional architecture of the platform.
2. **Requirements Gathering:** The features and functionalities of the platform will be established through interviews with prospective users and real estate agents. This will make the site user-friendly, including features like sophisticated search filters, property listings.
3. **Prototyping:** Early mockups and wireframes of the UI will be created using design software like Figma to conceptualize the UX and receive feedback.
4. **Iterative Development:** Using Agile methodology, development will proceed in iterative sprints, with each sprint focused on adding specific features such as property listing management, search filters, and user authentication. Feedback will be gathered at the conclusion of every sprint for improvement.
5. **Testing:** During the course of the development process, different testing techniques (unit testing, integration testing, and user acceptance testing) will be used in order to ensure the platform is free of bugs and performs as desired.

Data Collection/Analysis Tools

- **Surveys and Interviews:** These will be carried out among prospective users (home buyers, sellers, agents) to obtain opinions on features, usability, and requirements.
- **Google Analytics:** After deployment, Google Analytics will be added to capture information on user behavior like time spent on property listings, search activity, and pattern of interaction.
- **Jest/Enzyme:** Jest and Enzyme JavaScript testing libraries will be applied to test the performance and functionality of front-end React.js components.
- **Postman:** Postman will be utilized for API testing to validate correct interaction between the front-end and back-end elements of the platform.

Steps for Development

1. **Planning and Requirement Analysis:** Collect detailed requirements and establish core features like user authentication, property listings and search filters.
2. **System Design and Architecture:** Design the database schema using MongoDB, create wireframes, and develop API endpoints using Express.js.
3. **Frontend and Backend Development:** Use React.js to implement the front-end and create backend services in Node.js with Express.js.
4. **Testing and Debugging:** Perform unit testing, integration testing, and performance testing to make the website stable and secure.
5. **Deployment:** After development is done and tested, deploy the website onto a cloud server (such as AWS or Heroku) and keep an eye on performance after release.

Chapter 4 : Implementation and Testing

4.1 Introduction to Languages, IDE's, Tools and Technologies

Frontend

The frontend of the Estatery website is created to offer an intuitive, responsive, and user-friendly interface to visitors. The technologies and languages used are as follows:

1. **Languages and Frameworks:**
   - React.js: For developing dynamic, reusable components and providing a responsive user interface.
   - Next.js: For server-side rendering (SSR) and static site generation (SSG), improving SEO and page performance.
   - Sass (SCSS): For styling purposes, with improved modularity, nesting, and variables for CSS.

2. **Libraries:**
   - Redux Toolkit: For global application state management, such as user sessions, filters, and preferences.
   - Material-UI (MUI): For building responsive and visually pleasing UI components.
   - Axios: For API request handling and data fetching.
   - Swiper.js: For usage of carousels and image sliders.

3. **Tools and Techniques**
   - Webpack and Babel: Bundling and compiling the JavaScript code so it works in cross-browser compatibility.
   - Lazy Loading: To optimize image and data loading, enhance page speed.
   - Responsive Design: To make the website visually appealing and function \well across all devices, ranging from desktops to mobile phones.

Backend

The backend of Estatery's website is organized to manage the storage of data, user validation, and API services effectively. The technologies listed below were implemented:

1. **Languages and Frameworks:**
    - Node.js: To develop a high-performance and scalable server-side application.
    - Express.js: A light-weight framework to build RESTful APIs.

2. **Database:**
    - MongoDB: A NoSQL database to store property data, user data, and transactions.
    - Mongoose: An ODM (Object Data Modeling) library for MongoDB that offers schema-based data validation and simpler interaction with the database.

3. **Middleware:**
    - Validator.js: For validating user inputs like email addresses, passwords, and form data.
    - JWT (JSON Web Tokens): For safe user authentication and authorization.
    - Multer: For managing file uploads like property images.

4. **Tools and Techniques:**

- Postman: Used to test APIs during development and confirm that they fulfill functional requirements.
- Cors: Middleware for supporting cross-origin resource sharing.
- Environment Variables: Managed using dotenv for secure configuration of sensitive information like database credentials and API keys.

Other Supporting Tools

In order to facilitate the smooth operation and upkeep of the Estatery website, the following supporting tools and services were employed:

1. **Version Control:**
    - Git and GitHub: For version control, collaboration, and keeping the project's codebase in sync.
2. **Hosting and Deployment:**
    - Vercel: For easy and quick deployment of the frontend (Next.js).
3. **Development Environment:**

- Visual Studio Code (VS Code): The main IDE used for development with extensions for React, Sass, and MongoDB.

4. **Performance and Monitoring:**
    - MongoDB Compass: For managing and visualizing MongoDB data.

Summary

- Frontend: Next.js, React.js, Sass, Redux, Material-UI, Swiper.js.
- Backend: Node.js, Express.js, MongoDB, Mongoose, Validator.js.
- Supporting Tools: Postman, GitHub, Vercel.

This stack makes the Estatery website strong, scalable, and provides a high-quality user experience.

4.2 Algorithm/Pseudocode used

1. User Authentication

Pseudocode:

Input: Email, Password (or OTP for reset).

Process:

On login:

Verify user is present in the database.

Hash and match the input password with the password present in the database. Create and provide a token if it is successful.

On signup:

Validate user input.

Hash the password and store it in the database. Provide OTP for email verification.

On password reset:

Verify OTP.

Update password if OTP is valid.

Output: Success or error message.

2. Homepage Search Functionality

Algorithm:

Fetch properties based on search keywords, location, and filters.

Sort results by relevance or user preferences.

Display the results dynamically in the UI.

Pseudocode:

START

INPUT: Search query, location, filter criteria

QUERY the database using the provided inputs

IF results are found:

  Display properties sorted by relevance

ELSE:

  Show a "No results found" message

END

3. Property Listing with Filters

Algorithm:

Retrieve all properties.

Apply filters (e.g., price range, property type, location).

Paginate results for better user experience.

Pseudocode:

START

FETCH all properties from the database

APPLY filters based on user input

SORT the properties by user preference (e.g., price, size)

PAGINATE results for efficient loading

RETURN filtered and paginated results

END

4. Property Detail and Contact Seller

Algorithm:

Fetch property details using its unique ID.

Provide a contact form for the seller.

Send the contact request to the seller's dashboard.

Pseudocode:

```
START
INPUT: Property ID
FETCH property details from the database
DISPLAY details to the user
IF user submits contact form:
   VALIDATE input
   SAVE the request to the database
   NOTIFY the seller
END
```

5. Feedback Submission

Algorithm:

Validate feedback input.

Store the feedback in the database.

Optionally, notify admins for review.

Pseudocode:

```
START
INPUT: Feedback text, user details
VALIDATE feedback input
STORE feedback in the database
```

IF successful:

RETURN "Feedback submitted successfully"

ELSE:

RETURN "Error in submission"

END

6. Profile Dashboard Management

Algorithm:

Allow users to view and edit account details.

Handle requests like changing passwords, viewing responses, and managing properties.

Pseudocode:

START

IF user navigates to "Edit Profile":

DISPLAY current profile data

ALLOW modifications

SAVE changes to the database

IF user navigates to "Change Password":

VALIDATE old password

UPDATE with new password after confirmation

RETURN updated profile data to the user

END

7. Liked Properties Management

Algorithm:

Fetch and display properties liked by the user.

Allow users to remove properties from the liked list.

Pseudocode:

START

FETCH liked properties for the user from the database

DISPLAY liked properties

IF user removes a property from the list:

UPDATE the database

REFRESH the displayed list

END

8. Admin Features (For Sellers and Property Management)

Algorithm:

Approve or reject property submissions.

Handle seller requests and feedback efficiently.

Pseudocode:

START

FETCH all pending property submissions

FOR each property:

ALLOW admin to approve or reject

UPDATE property status in the database

IF seller raises a request:

VALIDATE and handle request

RETURN updated dashboard

END

9. Sell Page

Algorithm:

Display the sell page where users can input details about the property they want to sell.

Validate the inputs: Ensure all required fields (e.g., property details, price, images) are provided.

Store the property in the database once validated.

Notify Admin/Seller that the property is submitted and needs approval.

Display confirmation to the user after successful submission.

Pseudocode:

START

SHOW the sell page with property form (form inputs for title, price, location, images, description, etc.)

IF user fills and submits the form:

VALIDATE the form inputs:

-Verify that all mandatory fields are filled

-Verify that the price of the property is within range

-Validate the format and size of property images

IF validation is OK:

SAVE property details to the database

SEND notification to admin for approval

SHOW user confirmation message: "Your property has been submitted for review"

ELSE:

SHOW error message: "Please fill in all required fields correctly

END

4.3 Testing Techniques: in context of project work

Test Plan for Estatery Website

Test Plan Identifier: EW-TP-V1.0

References:

- System Requirement Specification for Estatery Website
- Project Plan
- Functional Specifications

1. Purpose of the Test:

The objective of this test plan is to confirm the reliability, functionality, and performance of the Estatery Website (EW) according to the stated requirements and goals as per the System Requirement Specification.

2. Location and Schedule of the Test:

- Location: Testing will be conducted in a controlled environment that simulates real-world usage conditions.
- Schedule: Testing will start after the completion of development milestones and will go in conjunction with feature releases and updates, iteratively.

3. Test Descriptions:

Black Box Testing

Objective: Verify system behavior and functionality without knowing its internal design.

Areas Covered:

- User registration and authentication
- Property search, filter, and sort features
- Property listing submission (sell page)
- Property view details and interaction
- Payment gateway integration (if applicable)

Output Results:

- Test cases with inputs and expected outputs

- Observed behaviors and any discrepancies recorded during testing

White Box Testing

Goal: Verify internal logic and structure, such as code-level operations and data transfer.

Areas Tested:

- Code review meetings
- Unit tests for backend APIs (property data management, user registration, etc.)
- Component-level tests for the frontend (React components for forms, search bar, etc.)

Deliverables Results:

- Documented results of code review
- Unit test outcome for backend API and frontend components
- Identified issues in internal logic or code structure

Integration Testing

Goal: Verify interaction and interfaces between system components.

Areas Tested

- Integrated functionality end-to-end testing (search, registration, payment, notifications)
- API integration test (user and property-related data exchange)
- Checks for data consistency between user sessions and property postings

Output Results:

- Test results for data exchange verification between integrated modules
- Results of API integration tests
- Data consistency verification among various sections of the website

User Acceptance Testing (UAT)

Purpose: Verify usability of the system and compliance with user specifications.

Areas Covered

- Usability testing of main user interfaces, including the homepage, property search, and submission forms of property listings
- Collection of feedback from users on functions such as chat, notifications, and browsing

Output Results:

- End-users' feedback reports about their website experience
- Pages, issues of usability, satisfaction rate, and areas of concern brought forward by the users

4. Pass and Fail Criteria:

Pass Criteria:

- Most functional requirements have to be satisfied, i.e., user registration, listing a property, and searching.
- System performance is in line with or exceeds established standards for load and response time.
- UAT feedback demonstrates high usability, user satisfaction, and no critical issues.

Fail Criteria:

- Critical functions fail to meet the established requirements (e.g., registration doesn't work, payment integration is faulty).
- System performance is far outside acceptable standards (e.g., slow page loads, loading failures).
- UAT feedback shows significant usability issues or unresolved problems that impact the user experience.

5. Testing Environment:

- Hardware: Variety of internet-enabled devices (desktop, tablets, smartphones).
- Software:
  - Modern web browsers (Chrome, Firefox, Safari, Edge)
  - Mobile applications (for Android and iOS)
  - Testing tools (Jest, Mocha for backend, Cypress for frontend testing)
  - Performance testing tools (e.g., Apache JMeter, Lighthouse)

6. Test Data:

- Sample data covering:

- User registration data (name, email, password, etc.)
- Property data (titles, prices, descriptions, images)
- Test datasets for user interactions (e.g., adding properties, searching, applying filters)
- User session data for checking user interaction continuity

7. Responsibilities:

- Development Team: Give test environments, help set up test data, troubleshoot problems, and resolve bugs.
- Testing Team:
  - Create test cases
  - Run tests
  - Record results and mark defects
  - Give feedback for continuous improvement

8. Staffing and Training Needs:

- Appropriate testing resources with web application testing knowledge and MERN stack experience.
- Training on employing particular testing tools and web application best testing practices.

9. Schedule:

Testing will be coordinated with development milestones until all functionalities are fully tested and validated. Testing will be continuous as features are deployed, with specific sprints for complete regression and performance testing.

10. Risks and Contingencies:

- Delays in Development: Development timeline delays may impact the planned testing activities.
- Insufficient Test Coverage: Partial test coverage due to lack of resources may result in undetected issues.

- Late-process Critical Defects: Major faults identified near the end of the development process could affect delivery schedule.

11. Approvals:

An approval of the Project Manager and the Development Team is mandatory prior to beginning test activities on this test plan.

Test Deliverables:

- Test cases
- Test execution reports
- Defect logs
- UAT feedback reports

Testing Tasks:

- Development of test cases
- Installation of test environments
- Execution of test
- Resolution and tracking of defects
- Coordinating with UAT and retrieving feedback

This Test Plan guarantees the Estatery Website goes through thorough testing to satisfy the required specifications, delivering a dependable, effective, and user-friendly platform for real estate management.

4.4 Test Cases designed for the project work

Following are some Test Cases for the Estatery Website. These encompass a range of scenarios from registration to property search and listing functionalities, to test all the major functionality thoroughly.

Test Case 1: User Registration

Test Case ID: TC-001

Test Case Title: User Registration Form

Preconditions: User is on the registration page.

Test Steps:

1. Navigate to the registration page.

2. Enter valid username, email, and password.

3. Click on the "Register" button.

Expected Result

- User is registered and redirected to login page successfully.
- Success message is shown, displaying the registration is successful.

Pass/Fail Criteria:

- Pass if registration is successful, and confirmation message is shown.
- Fail if there is any validation error or registration fails.

Test Case 2: User Login

Test Case ID: TC-002

Test Case Title: User Login

Preconditions: User is registered on the website.

Test Steps:

1. Navigate to the login page.

2. Provide valid credentials (username and password).

3. Click on the "Login" button.

Expected Result:

- User logged in successfully and redirected to the homepage or dashboard.

Pass/Fail Criteria:

- Pass if login is successful and user gets redirected properly.
- Fail if the credentials are invalid or login is unsuccessful.

Test Case 3: Property Search Functionality

Test Case ID: TC-003

Test Case Title: Property Search and Filter

Preconditions: User logged in and on the homepage.

Test Steps:

1. In the search bar, enter a property name or location (e.g., "apartment in New York").

2. Apply filters (e.g., price range, property type).

3. Click on the "Search" button.

Expected Result:

- Properties matching the search query and filters are displayed.

Pass/Fail Criteria:

- Pass if properties matching the search query are displayed correctly.
- Fail if no results appear or incorrect properties are displayed.

Test Case 4: Property Listing Submission (Sell Page)

Test Case ID: TC-004

Test Case Title: Submit Property Listing (Sell Page)

Preconditions: User is logged in.

Test Steps:

1. Navigate to the "Sell Property" page.

2. Enter property details (name, description, price, location, images).

3. Click on the "Submit" button.

Expected Result:

- Property is successfully submitted and displayed in the listings.
- A success message is presented confirming the submission.

Pass/Fail Criteria:

- Pass if the property is listed in the listings.
- Fail if submitting the property has an issue or invalid data is displayed.

Test Case 5: View Property Details

Test Case ID: TC-005

Test Case Title: View Property Details

Preconditions: User is on the property listings page.

Test Steps:

1. Click on a property listing.

2. View property details (e.g., images, description, price, features).

Expected Result:

- Detailed property information is correctly displayed.

Pass/Fail Criteria:

- Pass if all the details (images, price, description) are properly displayed.
- Fail if any of the details are missing or incorrect.

Test Case 6: Update Property Listing

Test Case ID: TC-006

Test Case Name: Update Property Listing

Preconditions: The user is logged in, and the property is listed.

Test Steps:

1. go to the "My Properties" page.

2. click on the property listing that you wish to update.

3. update the property details (e.g., price, description, images).

4. save the changes.

Expected Result:

- Property details are successfully updated, and changes are visible in the listings.

Pass/Fail Criteria:

- Pass if the details updated are saved and shown.
- Fail if changes are not saved or shown.

Test Case 7: Deletion of Property

Test Case ID: TC-007

Test Case Title: Delete Property Listing

Preconditions: User is logged in and has a listed property.

Test Steps:

1. Navigate to the "My Properties" page.

2. Click on the "Delete" button for a property.

3. Confirm deletion.

Expected Result:

- Property is deleted from the listings successfully.

Pass/Fail Criteria:

- Pass if the property is deleted and does not appear in the listings anymore.
- Fail if the property still appears after deletion.

Test Case 8: Update of User Profile

Test Case ID: TC-008

Test Case Title: Update User Profile Information

Preconditions: User is logged in.

Test Steps:

1. Navigate to the user profile page.

2. Update profile information (name, email, password).

3. Click on the "Save" button.

Expected Result:

- Profile is updated successfully, and a confirmation message is displayed.

Pass/Fail Criteria:

- Pass if the user profile is updated and saved.
- Fail if updates are not saved or if there is a validation issue.

Chapter 5 : Results and Discussions

5.1 User Interface Representation

The Estaery is an all-encompassing platform that is set to transform property buying, selling, and renting. Developed on the strong MERN stack (MongoDB, Express.js, React.js, and Node.js) with Next.js for performance optimization and Redux for state management, the project is set to simplify the real estate process for users and improve their experience. The platform provides a one-stop solution for property seekers, sellers, and real estate enthusiasts, with advanced filtering, property management, and user interaction features.

The site offers a user-friendly interface, enabling users to search for properties according to their needs, such as location, price range, and type of property. Sellers can list properties easily, while buyers can browse detailed property pages, contact sellers, and save favorite listings. Other features such as a feedback page, an about section, and a blog give information about the platform and community engagement.

Key Features of the Project:

1. Authentication System: Enables users to login, register, confirm accounts through OTP, change passwords, and securely update their profiles.
2. Homepage: Provides a robust search box, property tabs, and location-based property listings.
3. Property Management: Exhaustive listing and filtering on the "Buy Page," with individual property views including an explore option and seller contact.
4. Profile Dashboard: Enables users to update profiles, handle responses, change passwords, and view or list properties.
5. Favorite Properties: Preserves and displays saved favorite properties for quick accessibility.
6. Feedback and Blog Pages: Offer communication mechanisms for user interactions and access to informative real estate knowledge.

The site utilizes platforms such as Postman for testing APIs and MongoDB Compass for managing databases, thus maintaining seamless backend functionality.

5.1.1 Brief Description of Various Modules of the system

The real estate site Estatery provides a comprehensive set of functionalities aimed at addressing buyers, sellers, and property managers. The functions are developed to provide an effective, effortless, and user-friendly process for all the parties involved in property transactions. The following is a close explanation of the key product functions:

1. User Management
    - Registration and Login:
        - Users can register via an email or social login (e.g., Google).
        - Secure login through email and password, with OTP-based two-factor authentication.
    - Profile Management:
        - Users can edit their personal information, profile image, and contact details.
        - Sellers can create and edit property listings from their account dashboard.

2. Property Search and Filtering
    - Advanced Search:
        - Users can search for properties by keywords, locations, or property IDs.
        - Filters include:
            - Location
            - Price range
            - Property type (e.g., apartment, villa, commercial)
            - Size (e.g., square footage)
            - Amenities (e.g., pool, parking, garden)
    - Map-Based Search:
        - Integration with Google Maps enables searching properties by location on an interactive map.
        - Users can see nearby amenities such as schools, hospitals, and grocery stores.

3. Property Listings
    - Detailed Property Pages:
        - Every property listing contains images, descriptive information, prices, and owner/seller contact details.
        - Sellers may post multiple images, a virtual tour link, and extra features (e.g., newly renovated, pet-friendly).
    - Featured Listings:
        - Paid listings enable sellers to make their properties stand out as "Featured" for improved visibility.

4. Purchase and Rental Options
    - Buying Properties:
        - Users can directly contact owners or inquire about more details using the platform.
    - Rental Features:
        - A separate section is provided to view rental properties with options such as monthly rent, lease period, and deposit information.

5. Sell Page
    - Property Listing Management:
        - Sellers have an option to add, edit, or remove property listings.
        - Options include:
            - Property information (e.g., type, size, location)
            - Photos and videos
            - Price information
        - Sellers get notifications when a buyer expresses interest in their property.
    - Request Approval:
        - Admins can moderate property listings to maintain quality and authenticity.

6. Feedback and Reviews
    - User Feedback:
        - Buyers can provide reviews and ratings for properties and sellers.

    - Feedback is shown on property pages to assist other users in making informed choices.
- Seller Feedback:
    - Sellers can see user feedback and answer questions directly.

7. Favorites and Wishlist
    - Users can save their favorite properties for easy access later.
    - Wishlist management enables users to compare properties side-by-side.

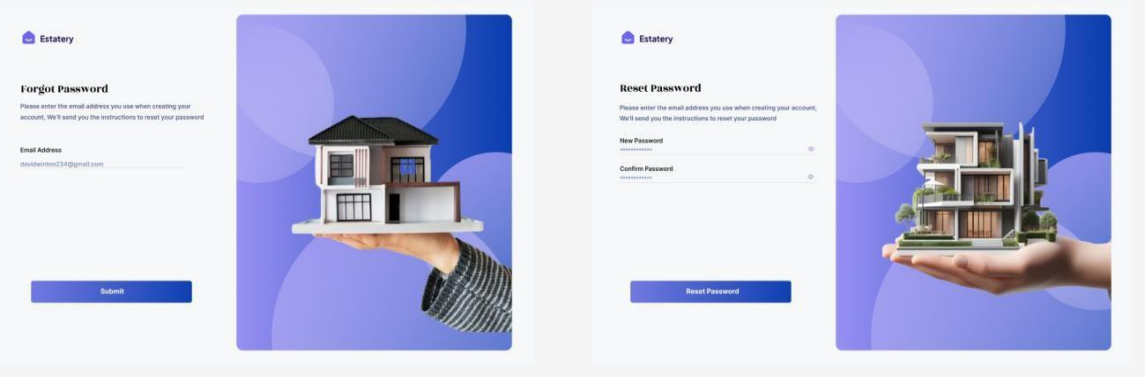
8. Admin Panel
    - Property Moderation:
        - Admins are able to view, approve, or deny property listings.
    - User Management:
        - Admins are able to control user roles, suspend accounts, or resolve reported issues.
    - Analytics Dashboard:
        - Offers insights on website traffic, active users, and property performance.

9. Responsive Design
    - The platform has been optimized to be used by devices such as desktops, tablets, and smartphones to facilitate accessibility and use by all.

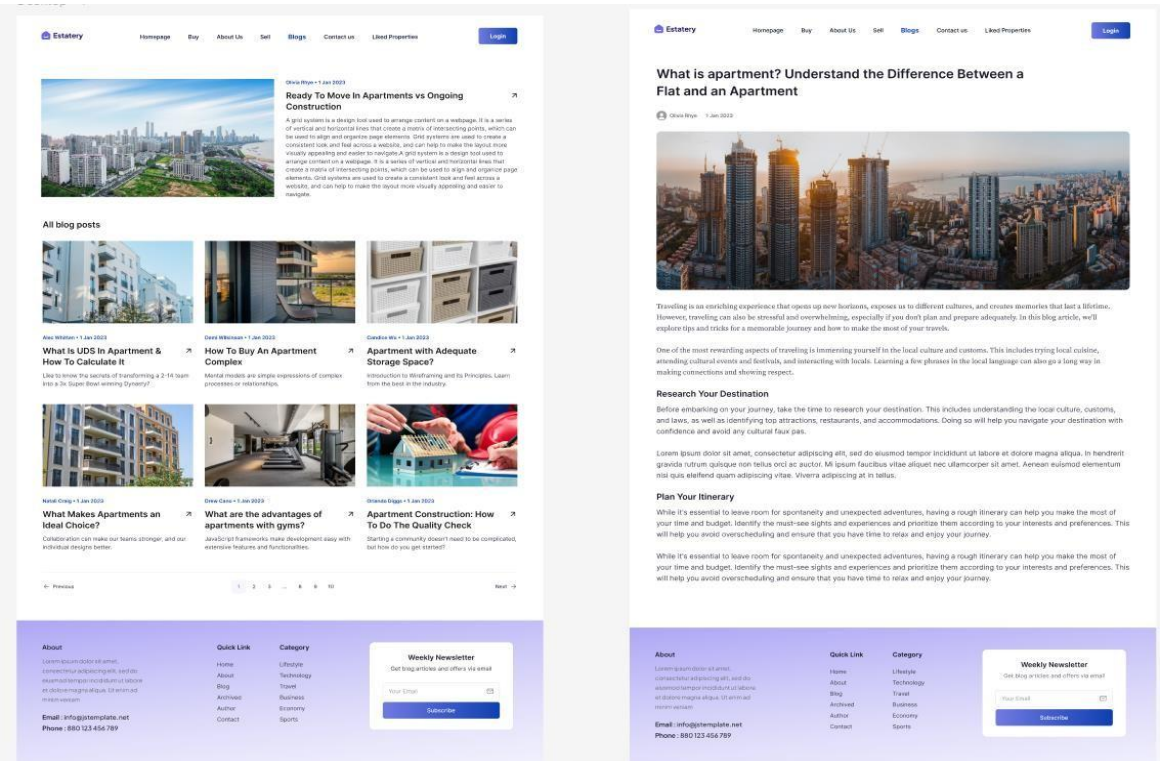
10. Security Features
    - Authentication and Authorization:
        - Role-based access to protect sensitive activities.
    - Data Protection:
        - Secure communication between client and server by SSL encryption.
    - Spam Prevention:
        - Email verification and CAPTCHA prevent spam or fraudulent requests.

5.2 Snapshots of system with brief detail of each and discussion.

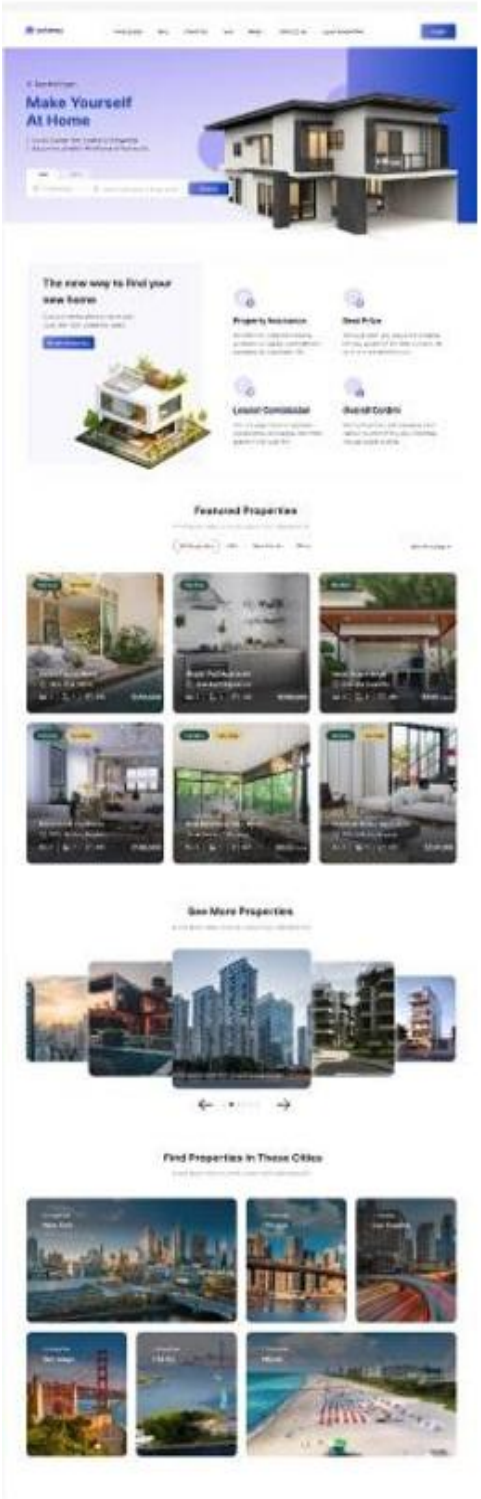
Auth Screen



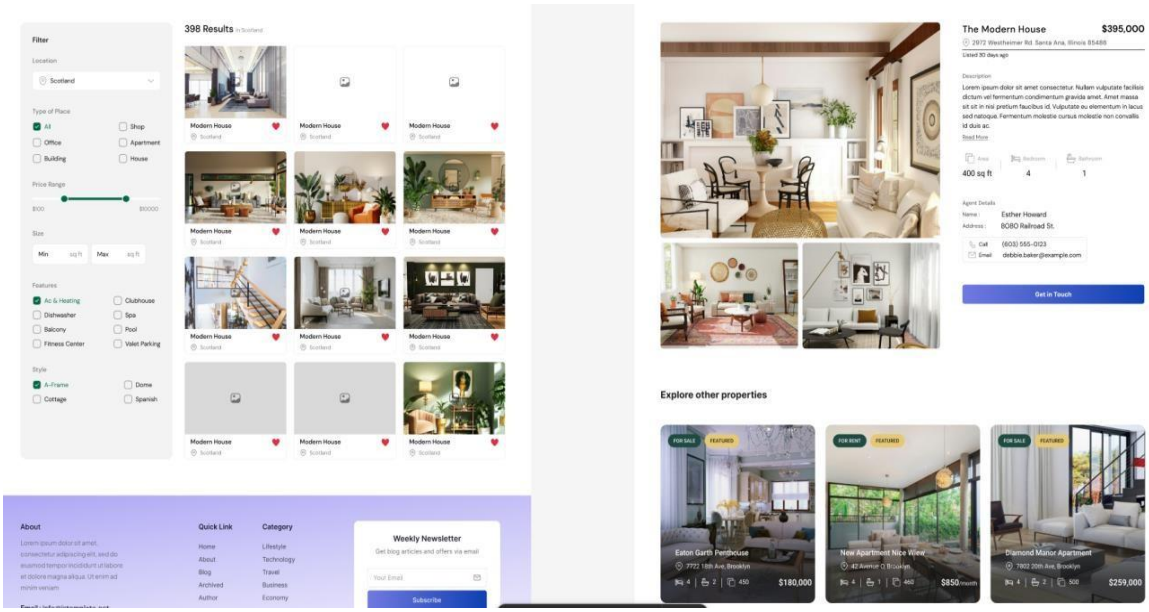
Blogs Screen



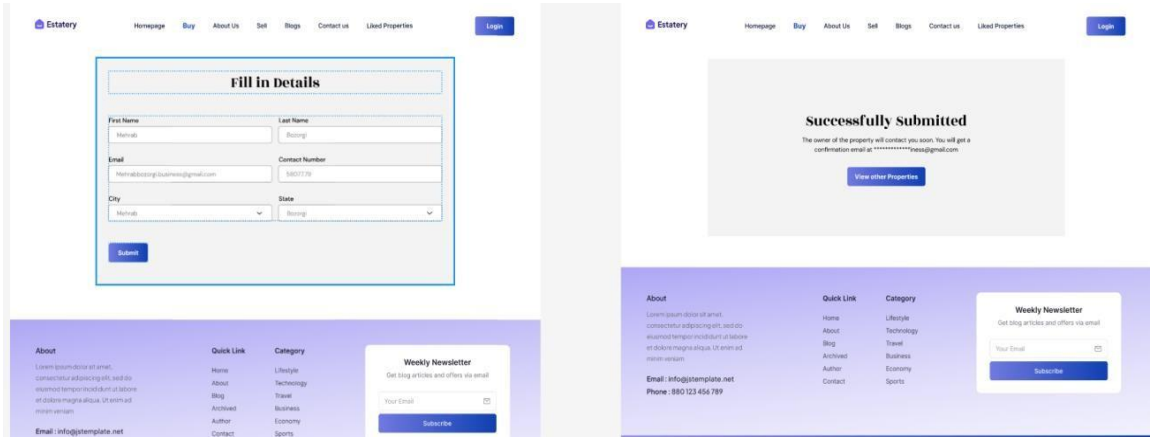
Home and Contact Screen



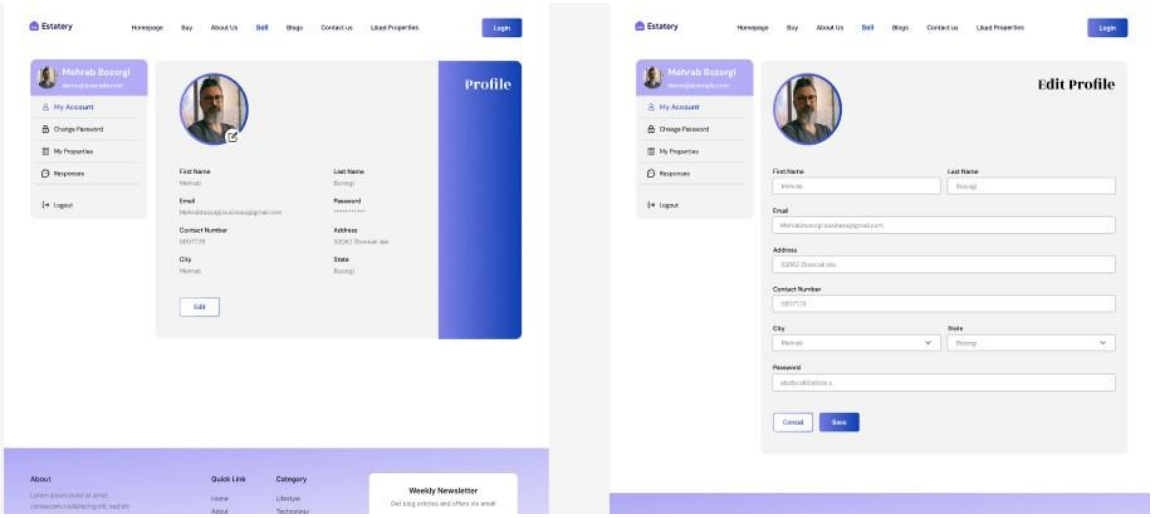
Buy and Detail Screen



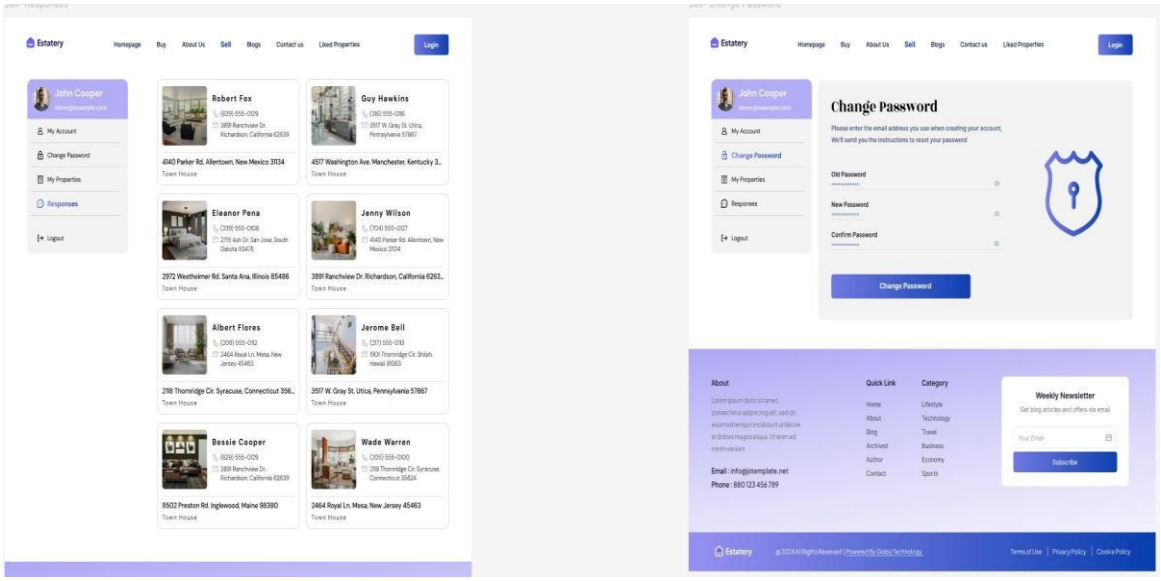
Contact Seller



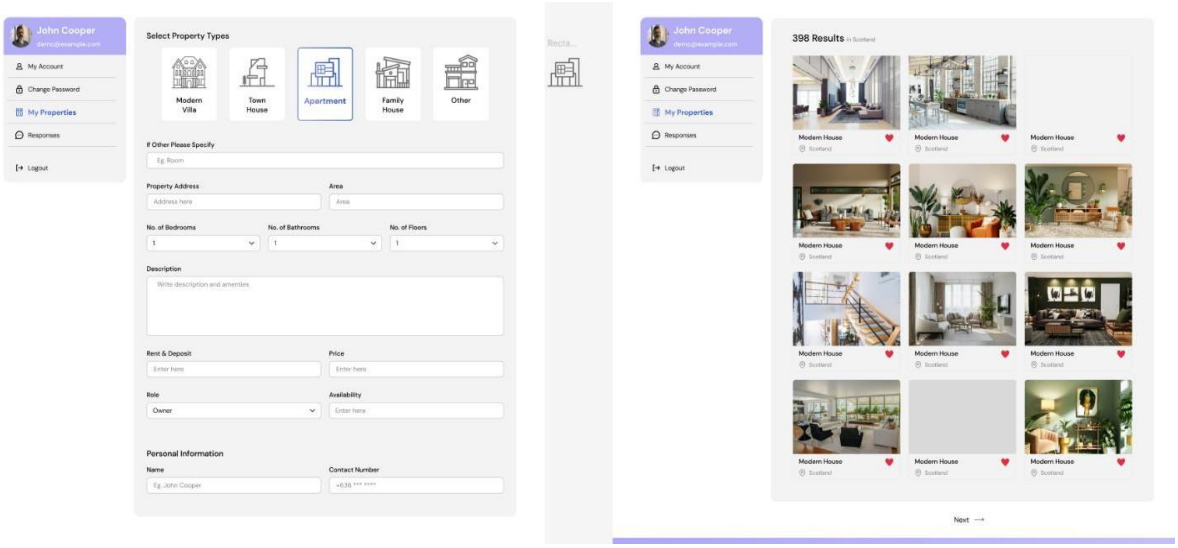
Profile Screen



Response and Change Password Screen



Sell Page and My Property



5.3 Back Ends Representation (if Database has been used)

The database schema for the Estatery website is designed to maintain data consistency, scalability, and efficiency in handling real estate-related data. The schema utilizes a NoSQL database (MongoDB), which offers flexibility and scalability, supporting dynamic data schemas appropriate for a real estate application.

Database Architecture

The database is document-oriented in design where every collection is a key entity in the real estate space. Entities are related using references or embedding based on the application.

Design Techniques Applied

1. **Encapsulation:**
   - Ensures class data and behavior are encapsulated and can be accessed only through well-defined methods.
   - Example: Secure user login and password encryption.

2. **Inheritance:**
   - Supports reuse by enabling child classes (e.g., Buyer, Seller) to inherit from the User class.

3. **Polymorphism:**
   - Supports flexibility in processing different types of actions dynamically, e.g., drawing different layouts of properties based on roles of users.

4. **Abstraction:**
   - Gives importance to crucial details for each class while concealing the intricacies.
   - Example: Abstraction of integrating payment gateways from the user.

5.3.1 Snapshots of Database Tables with brief description

1. **User collection**

```
let UserSchema = new Schema({
    author: {
        type: ObjectId
    },
    first_name: {
        type: String,
        required: false,
    },
    last_name: {
        type: String,
        required: false,
    },
    username: {
        type: String,
        required: false
    },
    email: {
        type: String,
        required: false,
    },
    email_verified: {
        type: Boolean,
        required: false,
    },
    email_verified_at: {
        type: Date,
        required: false,
    },
    otp: {
        type: Number,
        required: false,
    },
    token: {
        type: String,
        required: false
    },
    login_token: {
        type: String,
        required: false
    },
    token_expiry_at: {
        type: Date,
        required: false,
    },
    last_login_at: {
        type: Date,
        required: false,
    },
    phone_number: {
        type: Number,
        required: false,
    },
    password: {
        type: String,
        required: false,
        set:(value)=>{
            return encrypt(value);
        }
    },
    liked:{
        type:[String],
        required:false
    },
    gender: {
        type: String,
        required: false,
    },
    address: {
        type: String,
        required: false,
    },
    image: {
        type: String,
        required: false
    },
    city: {
        type: String,
        required: false
    },
    state: {
        type: String,
        required: false
    },
    status: {
        type: Number,
        default: 1
    }
```

2. Property Collection

```
let PropertySchema = new Schema({
    author:{
        type : ObjectId,
    },
    type:{
        type:String,
        required:false,
    },
    email:{
        type:String,
        required:false,
    },
    city:{
        type:String,
        required:false,
    },
    address:{
        type:String,
        required:false,
    },
    area:{
        type:Number,
        required:false,
    },
    bedrooms:{
        type:String,
        required:false,
    },
    bathrooms:{
        type:String,
        required:false,
    },
    floors:{
        type:String,
        required:false,
    },
    description:{
        type:String,
        required:false,
    },
    price:{
        type:Number,
        required:false,
    role:{
        type:String,
        required:false,
    },
    availability:{
        type:String,
        required:false,
    },
    contact_name:{
        type:String,
        required:false,
    },
    contact_number:{
        type:String,
        required:false
    },
    image:{
        type:[String], // uploads
        required:false,
    },
    status:{
        type:Number,
        default:1,
    },
    deleted_at:{
        type: Date,
        required:false,
    },
    created_at:{
        type:Date,
        default: new Date(),
    },
    updated_at:{
        type:Date,
        required:false,
    },
    user_id:{
        type:ObjectId,
        required:false,
        ref:'users'
    },
    slug: {
        type:String,
```

3. Response Collection

```
let ResponseSchema = new Schema({
    author: {
        type: ObjectId
    },
    first_name: {
        type: String,
        required: false
    },
    last_name: {
        type: String,
        required: false
    },
    contact_number: {
        type: String,
        required: false
    },
    email: {
        type: String,
        required: false
    },
    city: {
        type: String,
        required: false
    },
    state: {
        type: String,
        required: false
    },
    sender_id: {
        type: ObjectId,
        required: false,
        ref: 'users'
    },
    reciever_id: {
        type: ObjectId,
        required: false,
        ref: 'users'
    },
    property_id: {
        type: ObjectId,
        required: false,
        ref: 'properties'
    },
```

Chapter 6 : Conclusion and Future Scope

Conclusion

The Estatery website project incorporates contemporary web tools and technologies perfectly to create a solid and intuitive platform for the listing of property. Through React.js and Next.js for frontend and Node.js and Express.js for backend, the project establishes an efficient, scalable, and high-performance application. MongoDB and Mongoose being used for managing databases provide integrity and scalability in data, whereas Redux streamlines state management throughout the application.

The styling with Sass, combined with responsive design techniques, ensures that the website provides the best possible user experience on different devices. Material-UI assists in building a uniform and good-looking interface, and Swiper.js improves the display of images and content.

Additionally, the addition of advanced authentication techniques such as JWT and the input validation using Validator.js makes the website secure and trustworthy. The backend, facilitated by testing using Postman and Cypress, guarantees that all the API endpoints are working smoothly and process the users' requests effectively.

With continuous deployment pipelines through Vercel and Heroku, the platform is optimized for seamless scalability and upkeep. The project is also enhanced with good version control practices through GitHub to ensure collaboration and monitor progress.

In general, the Estatery website is an all-inclusive, efficient, and secure platform for property buyers, sellers, and renters to have a seamless and interactive experience. The project is well-suited to the requirements of the real estate market, providing up-to-date features and stable performance. Any future enhancements can concentrate on improving user experience further, performance optimization, and implementing more sophisticated features like AI-based property suggestions and improved search functions.

Future Scope

The project is open to immense improvement and scalability:

1. Enhanced Search Features: Incorporate AI-driven recommendation engines for personalized property recommendations.

2. Real-Time Communication: Include live chat facilities between buyers and sellers for improved interaction.

3. Virtual Tours: Include virtual property tours through 360-degree photos or VR technology.

4. Integration with Payment Gateways: Integrate secure online payment for property reservations or service charges.

5. Mobile Application: Create standalone mobile apps for iOS and Android to increase accessibility.

6. AI-Powered Insights: Leverage machine learning to deliver market trends, price forecasting, and investment recommendations.

7. Multi-Language Support: Expand the audience to global users by incorporating language localization.

References/Bibliography

[1] MongoDB, "MongoDB Documentation," [Online]. Available: https://www.mongodb.com/docs/.

[2] Express.js, "Express Documentation," [Online]. Available: https://expressjs.com/.

[3] React.js, "React Documentation," [Online]. Available: https://reactjs.org/.

[4] Node.js, "Node.js Documentation," [Online]. Available: https://nodejs.org/en/docs/.

[5] Postman, "Postman - API Development, Testing and Monitoring," [Online]. Available: https://www.postman.com/.

[6] MongoDB, "MongoDB Compass," [Online]. Available: https://www.mongodb.com/products/compass.

[7] Redux, "Redux Documentation," [Online]. Available: https://redux.js.org/.

[8] Validator.js, "Validator.js GitHub Repository," [Online]. Available: https://github.com/validatorjs/validator.js.

[9] Mongoose, "Mongoose Documentation," [Online]. Available: https://mongoosejs.com/.

[10] Next.js, "Next.js Documentation," [Online]. Available: https://nextjs.org/.

[11] "Stack Overflow - Community Forums," [Online]. Available: https://stackoverflow.com/.

[12] "YouTube Tutorials on MERN Stack and Next.js," [Online]. Available: Various tutorials and blog posts.

[13] Microsoft, "Visual Studio Code," [Online]. Available: https://code.visualstudio.com/.