

**VISVESVARAYA TECHNOLOGICAL UNIVERSITY**  
JNANASANGAMA, BELAGAVI - 590018



**Web Technology and Application (21CSE155) Project Report**  
**on**

**ESTATECRAFT**

*Submitted in partial fulfillment for the award of degree of*

**Bachelor of Engineering**  
**in**  
**COMPUTER SCIENCE AND ENGINEERING**

Submitted by  
**ABHIRAM BS (1BG21CS001)**  
**ROHITH B (1BG21CS060)**

Under the Guidance of  
**Mrs. Geetha L S**  
**Assistant Professor**  
**Department of CSE, BNMIT**



*Vidyaya Amrutam Ahnuthe*

***B.N.M. Institute of Technology***

**An Autonomous Institute Under VTU**

**Department of Computer Science and Engineering**

**2023– 2024**

# *B.N.M. Institute of Technology*

An Autonomous Institute Under VTU

## DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING



*Vidyaya Amrutham Ashnatho*

## **CERTIFICATE**

Certified that the Web Technology and Application (21CSE155) project report entitled **ESTATECRAFT** carried out by Mr. **ABHIRAM B S (1BG21CS001)** and Mr. **ROHITH B (1BG21CS060)** are the bonafide students of V Semester B.E., **B.N.M Institute of Technology** in partial fulfillment for the Bachelor of Engineering in COMPUTER SCIENCE AND ENGINEERING of the **Visvesvaraya Technological University**, Belagavi during the year 2023-24. It is certified that all corrections / suggestions indicated for internal Assessment have been incorporated in the report. The project report has been approved as it satisfies the academic requirements in respect of Web Technology and Application prescribed for the said degree.

**Mrs. Geetha L S**  
**Assistant Professor**  
**Department of CSE**  
**BNMIT, Bengaluru**

**Dr. Chayadevi M L**  
**Professor and HOD**  
**Department of CSE**  
**BNMIT, Bengaluru**

Name and Signature

1. Examiner 1:

2. Examiner 2:

## ACKNOWLEDGMENT

The success and final outcome of this project required a lot of guidance and assistance from many people and I am extremely privileged to have got this all along the completion of my project.

I would like to thank **Shri. Narayan Rao R Maanay**, Secretary, BNMEI, Bengaluru for providing the excellent environment and infrastructure in the college.

I would like to sincerely thank **Prof. T J Rama Murthy**, Director, BNMIT, Bengaluru for having extended his constant support and encouragement during the course of this project.

I would like to sincerely thank **Dr. S Y Kulkarni**, Additional Director, BNMIT, Bengaluru for having extended his constant support and encouragement during the course of this project.

I would like to express my gratitude to **Prof. Eishwar N Maanay**, Dean, BNMIT, Bengaluru for his relentless support and encouragement.

I would like to thank **Dr. Krishnamurthy G N**, Principal, BNMIT, Bengaluru for his constant encouragement.

I would like to thank, **Dr. Chayadevi M L**, Professor & Head of the Department of Computer Science and Engineering for the encouragement and motivation she provides.

I would also like to thank **Mrs. Geetha L S**, Assistant Professor, Department of Computer Science and Engineering for providing me with her valuable insight and guidance wherever required throughout the course of the project and its successful completion.

**ABHIRAM B S – 1BG21CS001**

**ROHITH B – 1BG21CS060**

## ABSTRACT

EstateCraft is a cutting-edge real estate platform designed to revolutionize the traditional property search and management process. Developed using the MERN (MongoDB, Express.js, React.js, Node.js) stack, EstateCraft integrates advanced technology and user-centric design principles to provide a seamless and efficient real estate experience for buyers, sellers, landlords, and tenants.

At the core of EstateCraft is its intuitive user interface, which offers an array of powerful features to simplify property discovery and management. With sophisticated search filters, users can effortlessly narrow down their options based on location, price, property type, and amenities, ensuring they find properties that align with their specific preferences and requirements.

EstateCraft sets itself apart with its immersive virtual tour and 3D walkthrough capabilities, allowing users to explore properties remotely and gain a comprehensive understanding of their layout and features. Through integrated communication tools, such as built-in messaging systems and chatbots, users can easily connect with agents, landlords, or potential buyers, facilitating swift and transparent interactions throughout the real estate transaction process.

Furthermore, EstateCraft prioritizes security and convenience by offering secure online transaction processing, enabling users to make payments, deposits, and purchases directly through the platform. Leveraging data analytics and insights, EstateCraft empowers users with valuable market trends and property values, facilitating informed decision-making when it comes to buying, selling, or investing in real estate.

With mobile accessibility through its dedicated app, EstateCraft ensures that users can manage their property search and transactions on the go, providing flexibility and convenience in today's fast-paced world. By combining innovative technology with a user-friendly interface, EstateCraft aims to redefine the real estate landscape, making property search and management more efficient, transparent, and enjoyable for all stakeholders.

# TABLE OF CONTENTS

<b>CONTENTS</b>	<b>Page No.</b>
ACKNOWLEDGEMENT	I
ABSTRACT	II
<b>Chapter 1 – INTRODUCTION</b>	<b>1</b>
1.1 Overview	1
1.2 Motivation	2
1.3 Web Technology	3
1.4 MERN Stack	5
1.5 Applications of Web Technology	6
<b>Chapter 2 – PROBLEM STATEMENT AND OBJECTIVES</b>	<b>9</b>
2.1 Problem Statement	9
2.2 Objectives	10
<b>Chapter 3 – SYSTEM REQUIRMENTS SPECIFICATION</b>	<b>12</b>
3.1 Visual Studio Code	12
3.2 Web Suite	13
3.3 React.js	14
3.4 MongoDB and MongoDB Atlas	14
3.5 Node.js and its frameworks	15
3.6 Auth0 for Authentication	18
3.7 Cloudinary - Image Service	18
3.8 Mantine	19
3.9 Leaflet	19

<b>Chapter 4 – SYSTEM DESIGN</b>	20
4.1 MERN Architecture	20
4.2 Architectural Overview	22
4.3 Schema Design	23
4.4 Flowchart	25
<b>Chapter 5 – IMPLEMENTATION</b>	27
5.1 Install Node Dependencies	27
5.2 Backend Connection	28
5.3 Database Integration and Schema	28
5.4 Frontend Creation and Routes	29
5.5 Backend Functionality	31
5.6 Testing and Debugging	34
5.7 Responsiveness and User Experience	34
5.8 Auth0 and Cloudinary Setup	35
<b>Chapter 6 – RESULTS</b>	38
6.1 Landing Page	38
6.2 Properties Page	39
6.3 Login and Authentication	42
6.4 Add Property Feature	43
6.5 Booking Feature	43
6.6 Favourites Feature	43
6.7 Add Property Feature	43
<b>Chapter 7 – CONCLUSION AND FUTURE SCOPE</b>	46

7.1 Conclusion	46
7.2 Future Scope	47
REFERENCES	III

# LIST OF FIGURES

Figure No.	Name of Figure	Page No.
1.1	EstateCraft Logo	1
3.1	Visual Studio Code	13
3.2	HTML	13
3.3	CSS	13
3.4	React.js	14
3.5	Mongo DB	15
3.6	Node.js	15
3.7	Express.js	16
3.8	Prisma	16
3.9	JWT	16
3.10	Nodemon	17
3.11	ThunderClient	17
3.12	Yarn	17
3.13	Auth0	18
3.14	Cloudinary	18
3.15	Mantine	19
3.16	Leaflet	19
4.1	MERN Architecture	20
4.2	Schema Diagram	23
4.3	Flowchart	25
5.1	Node Dependencies	27
5.2	Backend Connection	28
5.3	Database Integration and Schema	29



5.4	Frontend Creation and Routes	30
5.5	Backend Functionality	33
5.6	Auth0 Setup	35
5.7	Cloudinary Setup	35
6.1	Landing Page	36
6.2	Properties Page	37
6.3	Login and Authentication	38
6.4	Add Property Procedure	39
6.5	Property Page	40
6.6	Booking Feature	41
6.7	Favourites Feature	42
6.8	Database Operation	43

# CHAPTER – 1

## INTRODUCTION

### 1.1 Overview

In an era where technology permeates every aspect of our lives, the real estate industry stands as a testament to both the opportunities and challenges presented by digital innovation. Traditionally characterized by cumbersome processes, fragmented information, and limited accessibility, the journey of buying, selling, or renting property has long been in need of a transformative solution. Enter EstateCraft – a visionary platform poised to revolutionize the real estate experience through the seamless integration of modern technology and user-centric design.

At its core, EstateCraft embodies the ethos of simplification, aiming to alleviate the complexities that have long plagued the real estate journey for individuals and businesses alike. From the moment users embark on their property search to the final transaction, EstateCraft strives to provide a frictionless experience that empowers users with choice, transparency, and efficiency.

EstateCraft represents more than just a technological solution; it embodies a promise of transformation – a commitment to reshaping the real estate landscape for the better. By harnessing the power of the MERN stack, EstateCraft endeavors to deliver a holistic platform that not only simplifies the property search process but also fosters trust, transparency, and efficiency at every touchpoint.

With intuitive search algorithms, immersive virtual tours, and seamless communication channels, EstateCraft empowers users to navigate the real estate market with confidence and clarity. Whether it's finding the perfect home, listing a property for sale, or managing rental agreements, EstateCraft streamlines the process, enabling users to focus on what truly matters – making informed decisions that align with their goals and aspirations.



**Figure 1.1: EstateCraft Logo**

## **1.2 Motivation:**

Estatecraft's motivation is fueled by a combination of factors aimed at addressing the shortcomings of traditional real estate practices and leveraging the potential of web technology:

- **Streamlining Operations:** Estatecraft recognizes the inefficiencies inherent in traditional real estate practices, such as manual paperwork, fragmented processes, and lengthy turnaround times. By digitizing and automating various aspects of property management, booking, and transaction processing, Estatecraft aims to streamline operations and reduce administrative burdens for all stakeholders involved. Through features like automated listing creation, centralized property management dashboards, and integrated communication channels, Estatecraft empowers property owners and agents to manage their portfolios more efficiently, saving time and resources while ensuring smoother transactions.
- **Enhancing User Experience:** At the core of Estatecraft's mission is the commitment to delivering an exceptional user experience. The platform is designed with the end user in mind, prioritizing simplicity, intuitiveness, and accessibility at every touchpoint. Whether it's browsing property listings, scheduling viewings, or negotiating terms, Estatecraft strives to provide a seamless and enjoyable experience for property owners, tenants, and buyers alike. User-centric design principles, intuitive navigation structures, and responsive interfaces ensure that users can easily find what they need and accomplish their goals with minimal friction.
- **Bridging Stakeholder Gaps:** Estatecraft serves as a bridge between various stakeholders in the real estate ecosystem, facilitating transparent communication, collaboration, and relationship-building. By providing centralized platforms for property listings, messaging, and transaction tracking, Estatecraft fosters smoother interactions between property owners, agents, tenants, and buyers. Real-time updates, secure document sharing, and integrated feedback mechanisms ensure that all parties are kept informed and engaged throughout the transaction process. By fostering trust, transparency, and accountability, Estatecraft helps bridge the gaps between stakeholders, paving the way for more successful transactions and lasting relationships.
- **Embracing Innovation:** Innovation is at the heart of Estatecraft's approach to real estate technology. The platform continually seeks to push the boundaries of what's possible, leveraging cutting-edge web technologies, data analytics, and machine learning algorithms to redefine the real estate experience. From personalized property recommendations and predictive pricing models to virtual property tours and blockchain-based smart contracts, Estatecraft embraces innovation in all its forms to

enhance value for users and drive industry-wide transformation. By staying ahead of the curve and embracing emerging trends, Estatecraft aims to set new standards of excellence and innovation in the real estate sector.

- **Driving Positive Change:** Ultimately, Estatecraft's overarching goal is to make a positive impact on the real estate industry and the communities it serves. By democratizing access to property information, promoting fairness and transparency in transactions, and empowering individuals to make informed decisions, Estatecraft seeks to enrich lives and foster sustainable growth. Through initiatives such as affordable housing initiatives, green building initiatives, and community development programs, Estatecraft aims to contribute to the betterment of society while achieving commercial success. By aligning its business goals with broader social and environmental objectives, Estatecraft strives to drive positive change and create a legacy of lasting impact in the real estate domain.

### **1.3 Web Technology**

Web technology serves as the foundation for countless digital innovations across industries, offering a versatile toolkit for creating interactive, accessible, and scalable solutions. In the context of Estatecraft, web technology plays a pivotal role in driving various aspects of the platform's functionality and user experience:

- **Connectivity and Accessibility:** Web technology enables Estatecraft to reach a broad audience of users by providing a platform that is accessible from any device with an internet connection. Through web-based interfaces, users can browse property listings, communicate with agents, and complete transactions from the comfort of their own homes or on the go. This level of connectivity breaks down geographical barriers and ensures that Estatecraft's services are available to anyone, anywhere, at any time.
- **Scalability and Performance:** As Estatecraft grows and expands its user base, web technology allows the platform to scale its infrastructure and accommodate increasing demand without sacrificing performance or reliability. Cloud-based hosting solutions, distributed computing architectures, and scalable database systems ensure that Estatecraft can handle spikes in traffic, growing data volumes, and evolving user needs while maintaining a seamless user experience.
- **User Experience Enhancement:** The user experience is a key focus for Estatecraft, and web technology enables the platform to deliver intuitive, engaging, and personalized experiences to its users. Through responsive web design, interactive features, and seamless navigation, Estatecraft ensures that users can easily find the

information they need, engage with property listings, and complete transactions with minimal friction. By prioritizing user-centric design principles, Estatecraft enhances user satisfaction and loyalty, driving long-term engagement and success.

- **Data Management and Analysis:** Data is a valuable asset for Estatecraft, and web technology facilitates the collection, storage, analysis, and utilization of data to drive informed decision-making and optimize performance. With robust database management systems, data analytics tools, and machine learning algorithms, Estatecraft can derive actionable insights from user behavior, market trends, and property performance metrics. This data-driven approach enables Estatecraft to personalize user experiences, tailor marketing strategies, and optimize operations to meet user needs and preferences effectively.
- **Security and Compliance:** Security is paramount in Estatecraft, and web technology enables the platform to implement robust security measures to protect user data and ensure compliance with industry regulations and standards. Through encryption, authentication, access control, and regular security audits, Estatecraft safeguards user information, financial transactions, and sensitive data from unauthorized access, breaches, and cyber threats. By prioritizing security and compliance, Estatecraft maintains the trust and confidence of its users and stakeholders, fostering long-term relationships and loyalty.
- **Innovation and Differentiation:** Web technology is synonymous with innovation, and Estatecraft leverages this innovation to differentiate itself in the competitive real estate market. By exploring emerging technologies such as artificial intelligence, augmented reality, and blockchain, Estatecraft can introduce novel features and services that enhance the real estate experience and set it apart from competitors. Whether it's using AI-powered chatbots for customer support, AR-based virtual tours for property visualization, or blockchain-based smart contracts for secure transactions, Estatecraft continuously pushes the boundaries of what's possible to deliver value and innovation to its users.

In summary, web technology serves as a catalyst for innovation and growth in Estatecraft, enabling connectivity, scalability, user experience enhancement, data management, security, compliance, and innovation. By leveraging the power of web technology, Estatecraft can deliver a modern, efficient, and user-centric real estate platform that meets the needs of its users and stakeholders while driving long-term success and sustainability.

## 1.4 MERN Stack

Estatecraft relies on the MERN stack—a robust combination of MongoDB, Express.js, React.js, and Node.js—to build a versatile, scalable, and efficient web application. Each component of the MERN stack contributes unique capabilities that collectively enable Estatecraft to deliver a seamless user experience and robust functionality:

- **MongoDB:** Flexible and Scalable Database Solution: MongoDB serves as Estatecraft's database solution, offering flexibility, scalability, and performance for managing diverse data sets. As a NoSQL database, MongoDB stores data in flexible, schema-less JSON-like documents, allowing Estatecraft to adapt to changing requirements and accommodate evolving data structures seamlessly. Its distributed architecture enables horizontal scaling, allowing Estatecraft to handle large volumes of data and high concurrency levels efficiently. With features such as indexing, sharding, and replication, MongoDB ensures reliable data storage, fast query performance, and high availability for Estatecraft's real-time applications.
- **Express.js:** A Simplified Server-Side Development, Express.js is Estatecraft's server-side web application framework, built on top of Node.js, that simplifies the development of robust, scalable APIs and web servers. With its minimalist, unopinionated design, Express.js provides a lightweight framework for handling HTTP requests, routing, middleware integration, and server-side logic. Estatecraft leverages Express.js to create RESTful APIs for managing property listings, user authentication, booking transactions, and other backend functionalities. Its modular architecture and extensive middleware ecosystem enable Estatecraft to streamline development, improve code organization, and enhance performance.
- **React.js:** A Dynamic and Interactive User Interfaces, React.js powers Estatecraft's client-side user interface, enabling the creation of dynamic, responsive, and interactive web applications. As a declarative JavaScript library, React.js facilitates the creation of reusable UI components that encapsulate presentation logic and state management. Estatecraft leverages React.js to build intuitive user interfaces for browsing property listings, filtering search results, viewing property details, and interacting with booking forms. Its virtual DOM reconciliation algorithm ensures efficient UI updates, minimizing rendering overhead and enhancing performance. With features such as component lifecycle methods, context API, and hooks, React.js enables Estatecraft to create rich, interactive user experiences that adapt seamlessly to user interactions and device capabilities.

- **Node.js:** A Scalable and Efficient Server-Side Runtime, Node.js serves as Estatecraft's server-side JavaScript runtime environment, enabling non-blocking, event-driven I/O operations for building scalable, real-time web applications. By leveraging JavaScript on both the client and server sides, Estatecraft achieves code reuse, consistency, and productivity throughout the development lifecycle. Node.js's asynchronous, single-threaded architecture ensures high concurrency and responsiveness for handling concurrent requests, I/O operations, and data processing tasks. Estatecraft utilizes Node.js to implement server-side logic, database interactions, authentication middleware, and WebSocket communication for real-time updates. Its extensive ecosystem of npm modules, built-in APIs, and community support accelerates development, enhances scalability, and fosters innovation for Estatecraft's backend infrastructure.

In summary, the MERN stack provides Estatecraft with a powerful foundation for building modern, efficient, and scalable web applications. MongoDB offers flexibility and scalability for data storage, Express.js simplifies server-side development, React.js enables dynamic and interactive user interfaces, and Node.js provides a scalable and efficient runtime environment. Together, these technologies empower Estatecraft to deliver a seamless user experience, robust functionality, and high performance for its real estate platform.

## **1.5 Applications of Web Technology**

Web technology encompasses a wide range of tools, frameworks, and protocols that enable the creation of dynamic, interactive, and accessible web applications. Here are some general applications of web technology across various industries and sectors:

- **E-Commerce:** Web technology revolutionizes the way businesses conduct online commerce, enabling the creation of e-commerce platforms that facilitate buying and selling goods and services over the internet. From online marketplaces and retail websites to digital storefronts and payment gateways, web technology powers every aspect of the e-commerce experience. Features such as product catalogs, shopping carts, secure payment processing, and personalized recommendations enhance the shopping experience for consumers while providing merchants with tools for inventory management, order fulfillment, and customer relationship management.
- **Social Networking:** Social networking platforms leverage web technology to connect users, facilitate communication, and enable social interactions in virtual communities. From social media giants like Facebook, Twitter, and Instagram to professional networking sites like LinkedIn, web technology powers features such as user profiles,

news feeds, messaging, photo sharing, and group collaboration. Real-time updates, notifications, and content moderation tools ensure a dynamic and engaging user experience while enabling platform owners to monetize user-generated content through advertising and subscription-based models.

- **Education and E-Learning:** Web technology revolutionizes education by providing access to online learning resources, virtual classrooms, and collaborative tools that enable distance learning and skill development. E-learning platforms leverage web technology to deliver multimedia content, interactive quizzes, discussion forums, and live streaming sessions to students of all ages and backgrounds. Features such as learning management systems (LMS), video conferencing, screen sharing, and online assessments empower educators to deliver engaging and personalized learning experiences while enabling students to learn at their own pace and convenience.
- **Entertainment and Media:** Web technology transforms the entertainment industry by providing on-demand access to digital content such as music, movies, TV shows, games, and live events. Streaming platforms like Netflix, Spotify, and Twitch leverage web technology to deliver high-quality audio and video content to users across devices, including smartphones, tablets, smart TVs, and gaming consoles. Features such as content recommendation algorithms, personalized playlists, and social sharing integrations enhance the entertainment experience while enabling content creators and rights holders to monetize their creations through subscriptions, advertisements, and digital sales.
- **Healthcare and Telemedicine:** Web technology enables the delivery of healthcare services remotely through telemedicine platforms, virtual consultations, and remote monitoring solutions. Telehealth platforms leverage web technology to connect patients with healthcare providers, enabling video consultations, electronic prescriptions, and secure medical record management. Features such as appointment scheduling, symptom checkers, and medication reminders empower patients to manage their health proactively while improving access to healthcare services for underserved populations, rural communities, and patients with mobility limitations.
- **Travel and Hospitality:** Web technology revolutionizes the travel and hospitality industry by providing online booking platforms, travel aggregators, and destination guides that empower travelers to plan and book trips with ease. Travel websites and mobile apps leverage web technology to offer features such as flight and hotel search, price comparison, itinerary planning, and customer reviews. Integration with mapping



services, weather forecasts, and local attractions enhances the travel experience while enabling businesses to attract customers, optimize inventory, and streamline operations through automation and data analytics.

In summary, web technology permeates every aspect of modern life, enabling businesses, organizations, and individuals to connect, communicate, collaborate, and transact in virtual environments. From e-commerce and social networking to education and healthcare, the applications of web technology are vast and diverse, shaping the way we work, learn, shop, entertain, and interact in the digital age.

## CHAPTER – 2

### PROBLEM STATEMENT AND OBJECTIVES

#### 2.1 Problem Statement:

The primary objectives of our project, "EstateCraft - Revolutionizing Real Estate," are to create a modern and user-friendly platform for property search and management. This platform aims to streamline the entire real estate process by simplifying property discovery, enhancing viewing experiences, and introducing innovative features that elevate the overall user experience. The Objective of the proposed solution is:

- **Enhance User Experience:** The primary objective of EstateCraft is to elevate the user experience within the real estate domain. By leveraging intuitive design principles and seamless navigation, EstateCraft aims to simplify the property search process, making it more accessible and enjoyable for buyers, sellers, landlords, and tenants alike.
- **Streamline Property Management:** EstateCraft seeks to streamline property management tasks for landlords, property managers, and agents. Through centralized listing management, automated communication tools, and integrated transaction processing, EstateCraft aims to reduce administrative burden and optimize efficiency in property management workflows.
- **Facilitate Transparent Transactions:** Transparency is paramount in real estate transactions. EstateCraft aims to foster trust and transparency by providing comprehensive property information, facilitating seamless communication between stakeholders, and ensuring secure online transactions. By establishing a transparent ecosystem, EstateCraft endeavors to instill confidence and mitigate risks for all parties involved.
- **Promote Accessibility and Inclusivity:** EstateCraft is committed to promoting accessibility and inclusivity within the real estate industry. Whether it's through mobile optimization, multilingual support, or accessibility features for users with disabilities, EstateCraft aims to ensure that its platform is accessible to users from diverse backgrounds and demographics.
- **Drive Innovation and Continuous Improvement:** Innovation is at the core of EstateCraft's ethos. As technology evolves and user needs evolve, EstateCraft is committed to driving continuous improvement and innovation within its platform.

By aligning its objectives with the needs and expectations of its users, EstateCraft is poised to redefine the real estate experience, setting new standards of excellence in accessibility, transparency, and efficiency. Through its unwavering commitment to innovation and user-

centric design, EstateCraft aims to empower individuals and businesses to navigate the complexities of the real estate market with confidence and ease.

## **2.2 Objectives**

The implementation of the Estatecraft project is guided by several key objectives aimed at delivering a robust, user-friendly, and innovative real estate platform. These objectives serve as the foundation for the development process, ensuring that the final product meets the needs and expectations of its users while achieving business goals effectively.

- **Create a Seamless User Experience:** One of the primary objectives of the project implementation is to create a seamless and intuitive user experience across all aspects of the Estatecraft platform. This includes designing user interfaces that are easy to navigate, visually appealing, and responsive across devices. By prioritizing user-centric design principles and incorporating feedback from user testing sessions, the project aims to ensure that users can effortlessly browse property listings, schedule viewings, and complete transactions with minimal friction.
- **Implement Robust Backend Functionality:** Behind the scenes, the project focuses on implementing robust backend functionality to support the core features of the Estatecraft platform. This includes developing secure authentication mechanisms, efficient data storage solutions, and scalable APIs for managing property listings, user accounts, bookings, and transactions. By leveraging technologies such as the MERN stack and integrating best practices for backend development, the project aims to build a reliable and scalable infrastructure that can handle the demands of a growing user base.
- **Enhance Property Discovery and Exploration:** Another key objective of the project is to enhance the property discovery and exploration process for users. This involves implementing advanced search functionalities, personalized recommendations, and interactive property tours to help users find their ideal properties more efficiently. By leveraging technologies such as data analytics and machine learning, the project aims to provide users with relevant and personalized property suggestions based on their preferences, search history, and behavior on the platform.
- **Ensure Security and Privacy:** Security and privacy are paramount considerations in the implementation of the Estatecraft platform. The project focuses on implementing robust security measures, such as encryption, authentication, and authorization, to protect user data and transactions from unauthorized access and cyber threats.
- **Drive Innovation and Differentiation:** The project aims to drive innovation and differentiation in the real estate market by leveraging cutting-edge technologies and

introducing novel features and services. This includes exploring emerging technologies such as artificial intelligence, augmented reality, and blockchain to enhance the real estate experience and differentiate Estatecraft from competitors. By continuously pushing the boundaries of what's possible and staying ahead of industry trends, the project aims to position Estatecraft as a leader in the real estate technology space.

In summary, the implementation of the Estatecraft project is guided by objectives focused on creating a seamless user experience, implementing robust backend functionality, enhancing property discovery and exploration, ensuring security and privacy, and driving innovation and differentiation. These objectives serve as the roadmap for the development process, ensuring that the final product meets the needs and expectations of its users while achieving business objectives effectively.

## CHAPTER – 3

### SYSTEM REQUIREMENTS SPECIFICATION

This chapter presents the software and hardware requirements for the whole project. System requirements are essential in building a project. The system requirements help in identifying the resources required to execute the application. These requirements include hardware, software and other dependencies. By identifying the necessary resources, developers can ensure that the application runs smoothly, without any performance issues. System requirements also help in ensuring that the application is compatible with the intended hardware and software environment.

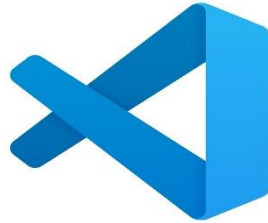
#### 3.1 Visual Studio Code

Visual Studio Code (VS Code) emerges as a robust and highly beneficial tool for constructing web applications with the MERN stack. This section delves into the ways in which VS Code facilitates the development process in this context.

- **Integrated Development Environment (IDE):** VS Code stands as a comprehensive IDE that offers an array of features like IntelliSense, code debugging, and Git integration, all of which are highly advantageous when dealing with the complexities of a web project powered by the MERN stack.
- **Live Collaboration:** VS Code's Live Share extension empowers seamless real-time collaboration among developers. This functionality enables multiple team members to collaborate on the same codebase simultaneously, regardless of geographical constraints.
- **JavaScript and React Support:** VS Code provides extensive support for JavaScript and React development, including code-highlighting, autocompletion, and debugging. This capability proves invaluable when crafting the user interface and front-end components of the MERN stack application.
- **Multi-Platform Compatibility:** Being a cross-platform code editor, VS Code ensures that the development and testing of the MERN stack application can be seamlessly carried out on various operating systems, including Windows, macOS, and Linux.
- **Access to Libraries and Modules:** VS Code offers access to a wealth of JavaScript libraries and modules, streamlining the development process. This feature reduces the need to write every piece of code from scratch, enhancing productivity and code quality.
- **Integration with Other Tools:** VS Code seamlessly integrates with a variety of development tools, such as version control systems, project management platforms, and

- build tools. This integration fosters efficient project management and collaboration among team members, promoting a cohesive and coordinated workflow.

By harnessing the capabilities of Visual Studio Code within the MERN stack context, developers can create dynamic and scalable web applications while benefiting from a feature-rich environment that enhances productivity, collaboration, and code quality.



**Figure 3.1: Visual Studio Code**

### 3.2 Web Suite

- **HTML:** HTML is a markup language used to structure the content of web pages. It uses a system of tags to define the various elements within a web page, such as headings, paragraphs, lists, images, links, and more. These tags provide a logical framework for organizing information, making it possible for browsers to render content correctly. HTML forms the skeleton of a webpage, determining how content is laid out and structured.



**Figure 3.2: HTML**

- **CSS:** CSS is a style sheet language used to control the visual presentation of HTML elements. It defines how elements should be displayed, specifying attributes like colors, fonts, spacing, and layout. By using CSS, web designers and developers can create consistent and aesthetically pleasing designs for their websites. CSS allows for the separation of content and presentation, making it easier to maintain and update the appearance of a website across multiple pages.



**Figure 3.3: CSS**

### 3.3 React.js

React is an open-source JavaScript library widely used for building user interfaces (UIs) in web applications. Developed and maintained by Facebook, React allows developers to create dynamic and responsive UI components that efficiently update and render based on changes in data. React employs a declarative approach, where developers describe how a UI should look based on the application's state, and it automatically handles the efficient rendering of components as the data changes. One of React's key features is its virtual DOM (Document Object Model), which optimizes the process of updating the actual DOM, resulting in improved performance. React can be used in combination with other libraries and frameworks to create robust and interactive web applications.[6]

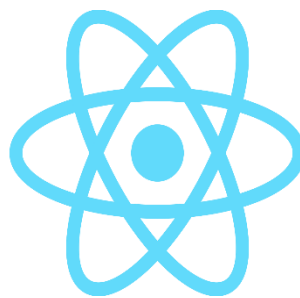


Figure 3.4: React.js

### 3.4 MongoDB and MongoDB Atlas

MongoDB stands as a prevalent open-source NoSQL database management system that adopts a flexible, scalable, and document-oriented approach to data storage. This approach is particularly advantageous for projects rooted in the MERN (MongoDB, Express.js, React, Node.js) stack, where dynamic data handling and evolving schemas are imperative. MongoDB thrives on collections of JSON-like documents, a design that enables agile data modeling and accommodates unstructured or semi-structured data.

Complementing MongoDB is MongoDB Atlas—an all-encompassing cloud-based database service provided by MongoDB Inc. This service revolutionizes the process of deploying, scaling, and managing MongoDB databases by automating manual tasks and simplifying complex configurations. MongoDB Atlas extends an array of features that enhance the overall database deployment experience.

The cloud-based infrastructure of MongoDB Atlas empowers the application to seamlessly adapt to increased data volumes and surging user activity. This cloud-centric architecture, fortified by MongoDB Atlas's robust security mechanisms, assures the safeguarding of sensitive dining and financial data.

The high availability and automated backup mechanisms integral to MongoDB Atlas further reinforce data integrity, ensuring that the application remains resilient in the face of unforeseen events. This amalgamation of MongoDB and MongoDB Atlas not only furnishes a reliable and scalable data management framework but also aligns seamlessly with the requirements of the MERN stack's dynamic and user-centric development approach.



**Figure 3.5: Mongo DB**

### **3.5 Node.js and its frameworks**

Node.js, is an open-source, server-side JavaScript runtime environment that allows developers to build scalable and high-performance network applications. It's based on the V8 JavaScript engine and uses an event-driven, non-blocking I/O model, making it well-suited for building real-time applications like chat applications, streaming services, and APIs. Node.js enables developers to use JavaScript on both the client and server sides, which streamlines the development process and promotes code reuse. It has a vast ecosystem of libraries and modules available through the Node Package Manager (npm), making it a popular choice for building various types of applications.



**Figure 3.6: Node.js**

- **Express.js:** Express is a fast and minimalist web application framework for Node.js. It provides a set of tools and features that simplify the process of building web applications and APIs by offering a lightweight and flexible structure. Express enables developers to easily handle routing, middleware integration, and HTTP request/response handling. It's commonly used to create server-side applications, RESTful APIs, and other backend services in Node.js, allowing developers to focus on their application's logic rather than dealing with low-level HTTP intricacies.[8]





**Figure 3.7: Express.js**

- **Prisma:** Prisma is a modern database toolkit and ORM (Object-Relational Mapping) designed for Node.js and TypeScript applications. It offers developers an intuitive API for interacting with databases, simplifying tasks like schema definition, migration management, and query building. With Prisma, developers can leverage a declarative modeling syntax to define database schemas and automatically generate type-safe TypeScript interfaces based on those schemas. Additionally, Prisma supports automatic schema migrations, ensuring seamless evolution of the database structure over time. It also provides a powerful query builder for constructing complex database queries with ease. [7]



**Figure 3.8: Prisma**

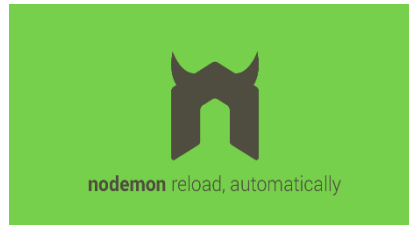
- **JWT:** A JWT (JSON Web Token) is a compact way to securely send information between parties. It includes a header, a payload, and a signature. JWTs are often used for secure authentication and authorization in web applications. The header specifies how the token is encrypted, the payload holds the data, and the signature ensures the token's integrity. They are useful for sharing data without constant database queries.



**Figure 3.9: JWT**

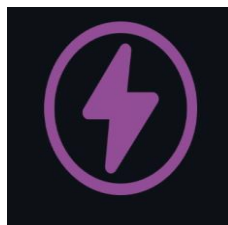
- **Nodemon:** Nodemon is a development tool that monitors changes in files within a Node.js application and automatically restarts the server when changes are detected.

This eliminates the need for manual server restarts during development, saving time and streamlining the testing process. Nodemon is particularly valuable for rapidly iterating and debugging applications, as it provides a seamless way to observe code changes and immediately see the effects without manual intervention. It's commonly used during the development phase to enhance productivity and improve the development experience.[9]



**Figure 3.10: Nodemon**

- **ThunderClient:** Thunder Client is a Visual Studio Code extension designed for efficient API testing. It empowers developers to create, manage, and execute HTTP requests within the VS Code environment. With a user-friendly interface, ThunderClient simplifies the process of configuring headers, parameters, and request bodies, while also facilitating response inspection and request debugging. It's a valuable tool for ensuring the functionality and reliability of APIs during development.



**Figure 3.11: ThunderClient**

- **Yarn:** Yarn is a package manager for JavaScript projects, serving as an alternative to npm. Yarn offers advantages such as faster dependency resolution, improved caching mechanisms, and deterministic installations. Developers can utilize Yarn commands to manage project dependencies efficiently, including adding, removing, and upgrading packages.



**Figure 3.12: Yarn**

### 3.6 Auth0 for Authentication

Auth0 serves as Estatecraft's authentication and authorization system, facilitating secure access to user accounts and features. Integrated with Google authentication, Auth0 streamlines the login process, allowing users to sign in using their Google credentials. This removes the need for separate Estatecraft login details, enhancing convenience and reducing friction. Auth0 offers robust security measures like multi-factor authentication and passwordless login, safeguarding user accounts against unauthorized access. Additionally, Auth0 provides comprehensive user management tools, enabling administrators to efficiently handle user accounts, permissions, and roles. Overall, Auth0 ensures a secure and user-friendly authentication experience, fostering trust and confidence in Estatecraft's platform.[1]



Figure 3.13: Auth0

### 3.7 Cloudinary - Image Service

Cloudinary functions as Estatecraft's image service provider, delivering a dependable and scalable solution for managing and optimizing images throughout the platform. Seamlessly integrated into Estatecraft's property listings, Cloudinary simplifies the process of uploading, storing, and accessing property images. Leveraging advanced image processing features like resizing and compression, Cloudinary ensures optimal image presentation across various devices and screen sizes, prioritizing fast loading times. Moreover, Cloudinary offers automatic image transformation, responsive delivery, and seamless integration with content delivery networks (CDNs), augmenting Estatecraft's interface performance and user experience. Estatecraft utilizes Cloudinary to showcase property images in high quality, while efficiently managing bandwidth usage and load times. Additionally, Cloudinary's robust security measures, such as access control and encryption, bolster the protection of property images, fortifying Estatecraft's overall security framework.[2]



Figure 3.14: Cloudinary

### 3.8 Mantine

Mantine is a modern React component library designed to simplify the development of UI components and provide a consistent and accessible user experience across web applications. It offers a comprehensive collection of pre-built components, hooks, and utilities for building responsive and visually appealing interfaces.

One of the standout features of Mantine is its extensive component library, which includes components for menus, forms, hooks, dates, and more. These components are fully customizable and come with built-in accessibility features, ensuring compliance with web accessibility standards such as WCAG.[3]

Mantine also provides developers with a set of hooks and utilities for common tasks such as form handling, styling, and state management. These hooks streamline development workflows and improve code maintainability by abstracting away complex logic into reusable functions.



**Figure 3.15: Mantine**

### 3.9 Leaflet

Leaflet is a JavaScript library commonly used for integrating interactive maps into web applications. It provides developers with a lightweight and flexible solution for displaying geographical data in a user-friendly manner. Leaflet offers a wide range of features, including customizable markers, layers, and overlays, allowing developers to create visually appealing and informative maps tailored to their specific needs.

One of the key advantages of Leaflet is its simplicity and ease of use. Developers can quickly integrate Leaflet into their projects with minimal setup, thanks to its intuitive API and comprehensive documentation. Leaflet also supports various map providers, including OpenStreetMap, Mapbox, and Google Maps, giving developers the flexibility to choose the most suitable mapping solution for their application.[5]

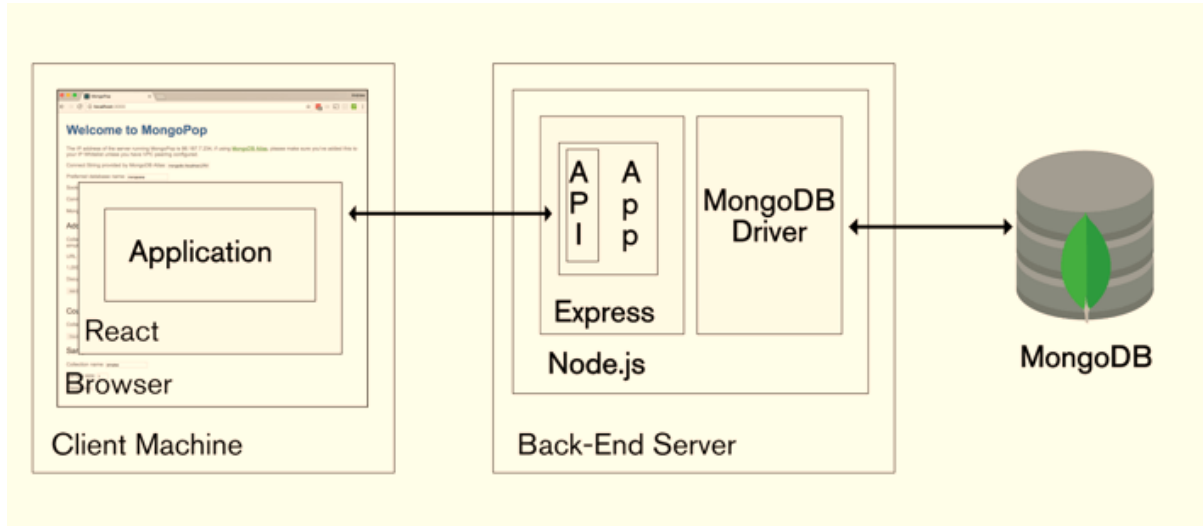


**Figure 3.16: Leaflet**

## CHAPTER – 4

### SYSTEM DESIGN

#### 4.1 MERN Architecture



**Figure 4.1: MERN Architecture**

MERN stack is a popular JavaScript framework used for building web applications. It is an acronym for MongoDB, Express.js, React, and Node.js. Each component of the stack plays a specific role in the development process. MERN stack, known for its versatility and efficiency, allows developers to build robust and scalable applications using JavaScript.

- **MongoDB:** MongoDB is a NoSQL database that stores data in a flexible, JSON-like format known as BSON (Binary JSON). It offers scalability, high performance, and flexibility, making it suitable for handling large volumes of data and accommodating dynamic schemas. MongoDB's document-based storage allows developers to store related data together in a single document, eliminating the need for complex joins commonly associated with relational databases. It is particularly well-suited for applications requiring fast iteration and frequent schema changes.
- **Express.js:** Express.js is a lightweight web application framework for Node.js. It provides a robust set of features for building web servers and APIs, including middleware support, routing, and request handling. Express.js simplifies the process of building server-side applications by providing a minimal and flexible structure that allows developers to create scalable and efficient backend systems. With its minimalist design philosophy, Express.js allows developers to build RESTful APIs and handle HTTP requests and responses with ease.[4]

- **React.js:** React.js is a JavaScript library for building user interfaces. Developed by Facebook, React.js allows developers to create reusable UI components that update dynamically based on changes to data. React's component-based architecture promotes code reusability, modularity, and maintainability, making it ideal for building complex web applications with rich user experiences. React utilizes a virtual DOM (Document Object Model) to efficiently update the UI, resulting in fast rendering and improved performance. It also supports server-side rendering, enabling developers to render React components on the server for improved SEO and initial load times.
- **Node.js:** Node.js is a runtime environment that allows developers to run JavaScript code outside the browser. Built on Chrome's V8 JavaScript engine, Node.js enables developers to build scalable and high-performance server-side applications using JavaScript. Node.js utilizes an event-driven, non-blocking I/O model, making it well-suited for building real-time, data-intensive applications and APIs. With its extensive package ecosystem, known as npm (Node Package Manager), Node.js provides access to a vast array of libraries and modules that simplify development tasks and accelerate the development process.

In a MERN stack application, the front end uses React, which handles the UI and user interactions. It's in charge of developing user-facing web and mobile applications. You manage failures effectively at this stage and have the ability to reuse code with React. It assists in creating and managing lists, forms, events, and functions effectively.

The server layer is the second tier of the MERN stack, below the React layer, and comprises Express.js and Node.js. This is where URL redirecting and HTTP requests take place. The client tier and database tier are connected by this layer, creating a seamless transition from the top to the bottom.

The final tier of the MERN stack, the database layer, stores your app's data with MongoDB. This layer efficiently stores information until it's retrieved.

Together, the MERN Stack provides a comprehensive solution for building full-stack web applications. MongoDB offers a scalable and flexible database solution, Express.js provides a robust backend framework, React.js enables the creation of dynamic and interactive user interfaces, and Node.js allows developers to build scalable and high-performance server-side applications. By leveraging the strengths of each component, developers can build modern web applications that are efficient, scalable, and maintainable.

## 4.2 Architectural Overview

- **Backend Services:** The MERN stack (MongoDB, Express.js, React.js, Node.js) forms the backbone of Estatecraft's backend services. MongoDB serves as the database solution, offering flexibility and scalability for storing property listings, user data, and transaction records. Express.js is utilized to develop RESTful APIs for handling CRUD operations, authentication, and data validation. Node.js powers the server-side runtime environment, enabling non-blocking I/O operations and real-time communication through WebSocket protocols. The MERN stack enables seamless integration, high performance, and scalability, providing a robust foundation for Estatecraft's backend infrastructure.
- **Authentication and Authorization:** Auth0 is integrated into Estatecraft to provide authentication and authorization services. Users can sign in to Estatecraft using their existing Google credentials, leveraging Auth0's support for Google authentication. Auth0 offers robust security features such as multi-factor authentication, passwordless login, and identity verification to protect user accounts from unauthorized access and cyber threats. Comprehensive user management capabilities provided by Auth0 allow administrators to manage user accounts, permissions, and roles effectively. Integration with Auth0 ensures a secure, reliable, and user-friendly authentication experience for Estatecraft users, enhancing trust and confidence in the platform.
- **Image Storage and Optimization:** Cloudinary is chosen as the image service provider for Estatecraft, offering a reliable and scalable solution for storing, managing, and optimizing images used across the platform. Property images are uploaded, stored, and retrieved through Cloudinary, which provides advanced image processing capabilities such as resizing, cropping, and compression. Cloudinary ensures that images are optimized for fast loading and optimal display on various devices and screen sizes, enhancing the performance and user experience of Estatecraft's image-rich interface. Additional features such as automatic image transformation, responsive delivery, and CDN integration further improve image performance and reliability. Robust security features provided by Cloudinary, including access control and encryption, protect property images against unauthorized access and data breaches, ensuring the integrity and confidentiality of user data.
- **Frontend Architecture:** Estatecraft's frontend architecture is built using React.js, a declarative JavaScript library for building user interfaces. React.js enables the creation of dynamic, responsive, and interactive UI components that adapt seamlessly to user interactions and device capabilities. Component-based architecture allows for

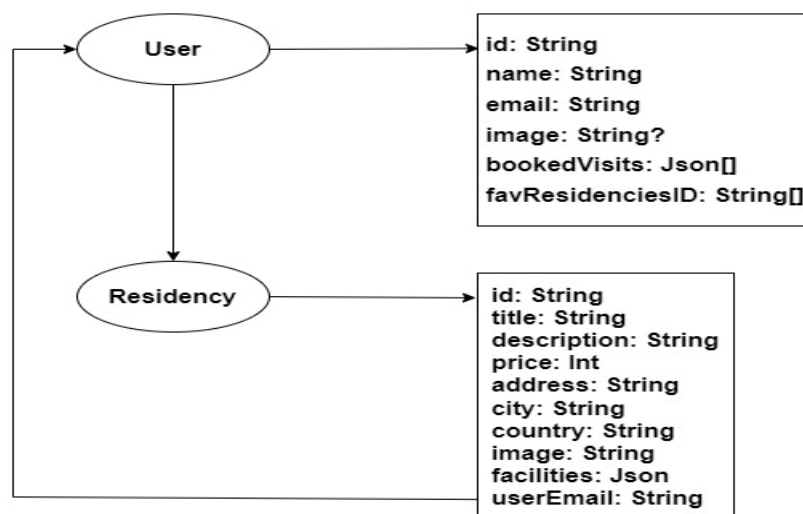
modularity, reusability, and maintainability of frontend code, facilitating rapid development and iteration. Real-time data updates and interactive features are enabled through integration with backend APIs, providing users with a smooth and intuitive browsing experience. React.js integrates seamlessly with Estatecraft's backend services, ensuring efficient data flow and synchronization between the frontend and backend components.

- **Scalability and Performance Optimization:** Estatecraft's system design prioritizes scalability and performance optimization to handle increasing user traffic and data volumes effectively. Continuous monitoring, performance testing, and optimization efforts ensure that Estatecraft remains responsive, reliable, and performant under varying load conditions and usage patterns.

By following this comprehensive system design, Estatecraft ensures a robust, scalable, and user-friendly real estate platform that meets the needs of its users while achieving business objectives effectively.

### 4.3 Schema Design

The Schema Diagram is a visual representation of the database structure used in the EstateCraft project. It provides a comprehensive view of the collections, documents, fields, and their relationships within the MongoDB database. The SchemaDiagram is a crucial tool for understanding the organization and hierarchy of data in the application, assisting developers and stakeholders in grasping the data architecture. The Schema Diagram of the project is as shown in Figure 4.2



**Figure 4.2: Schema Diagram**



The schema design of the database consists of two main entities: User and Residency. Each entity has a set of attributes that define its properties and characteristics.

### 1. User Entity

The User entity represents the users of the application, who can be either travelers or hosts. Each user has the following attributes:

- **id:** A unique identifier for the user, generated by MongoDB.
- **name:** The name of the user, entered by the user during registration.
- **email:** The email address of the user, entered by the user during registration and used for authentication and communication.
- **image:** An optional attribute that stores the URL of the user's profile picture, uploaded by the user.
- **bookedVisits:** An array of JSON objects that store the information of the residencies that the user has booked or visited. Each object contains the id, title, image, and date of the residency.
- **favResidenciesID:** An array of strings that store the ids of the residencies that the user has marked as favorites.

### 2. Residency Entity

The Residency entity represents the residencies that are available for booking on the application. Each residency has the following attributes:

- **id:** A unique identifier for the residency, generated by MongoDB.
- **title:** The title of the residency, entered by the host during listing.
- **description:** The description of the residency, entered by the host during listing and providing details such as amenities, rules, and nearby attractions.
- **price:** The price of the residency, entered by the host during listing and expressed in Indian rupees (INR).
- **address:** The address of the residency, entered by the host during listing and used for displaying the location on a map.
- **city:** The city where the residency is located, entered by the host during listing and used for filtering the search results.
- **country:** The country where the residency is located, entered by the host during listing and used for filtering the search results.
- **image:** The URL of the image of the residency, uploaded by the host during listing and used for displaying the residency on the application.

- **facilities:** A JSON object that stores the facilities that the residency offers, such as wifi, parking, air conditioning, etc. Each facility has a name and a boolean value indicating whether it is available or not.
- **userEmail:** The email address of the user who listed the residency, used for linking the residency to the host and enabling communication between the host.

## 4.4 Flowchart

The flowchart serves as a visual representation of the sequential steps and interactions involved in navigating the system. It provides a comprehensive overview of how users interact with the platform, starting from the landing page and progressing through various functionalities such as property search, viewing popular residencies, contacting options, login procedures, property booking, and adding properties for sale.

The flowchart depicts the logical flow of actions, decisions, and outcomes within the system, helping stakeholders understand the user journey and system behavior at a glance. By visualizing these steps and interactions, the flowchart aids in identifying potential bottlenecks, inefficiencies, or areas for improvement in the user experience.

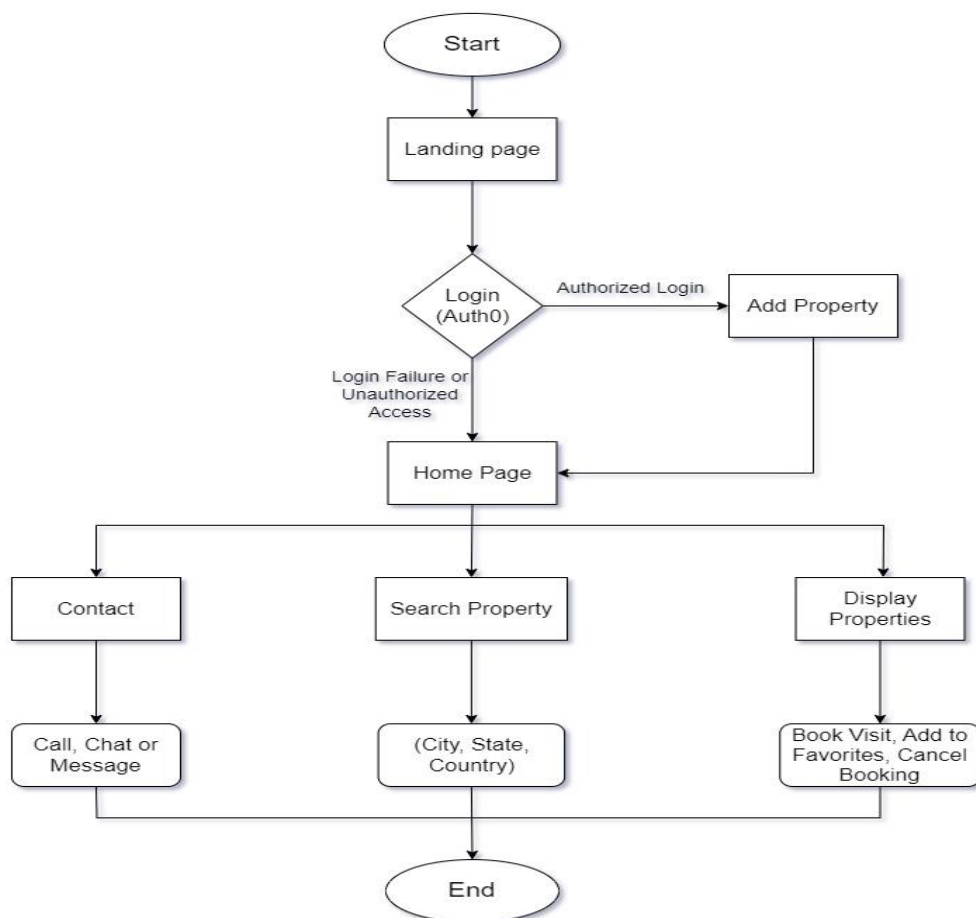


Figure 4.3: Flowchart

- **Landing Page:** This is the first point of contact for users. It's designed to be visually appealing and user-friendly, providing a snapshot of what the website offers. It may include featured properties, recent listings, and a brief introduction to the platform.
- **Search Functionality:** This is a crucial feature for any real estate website. Users can search for properties using various criteria such as title, city, or country. The search results are then displayed in an organized manner, allowing users to easily browse through the listings.
- **Display Popular Residencies:** This feature showcases popular residencies based on recent additions or user ratings. It helps users discover new properties that they might not have found through the search function.
- **Contact Options:** Communication is key in real estate transactions. The website provides various contact options such as call, chat, or message, enabling users to easily get in touch with property owners or the platform's support team.
- **Login Pop-up:** User authentication is handled through a pop-up window from Auth0. This ensures the security of user data and allows the platform to provide personalized services.
- **Booking Property:** Once a user is interested in a property, they can book it for a selected date. The booking process is designed to be straightforward and user-friendly.
- **Add Property Functionality:** Users who want to sell their property can add it to the platform. They can provide all the necessary details such as location, price, size, and photos. The platform then makes the listing available for potential buyers to view and book.
- **End of Flow:** This represents the completion of a user's journey on the website, whether it's booking a property, adding a new listing, or simply browsing through the available options.

The MERN stack (MongoDB, Express.js, React.js, Node.js) is a powerful combination of technologies that makes all these features possible. MongoDB is used for the database, Express.js and Node.js handle the backend, and React.js is used for the frontend. This stack allows for efficient data handling, fast rendering, and a seamless user experience.

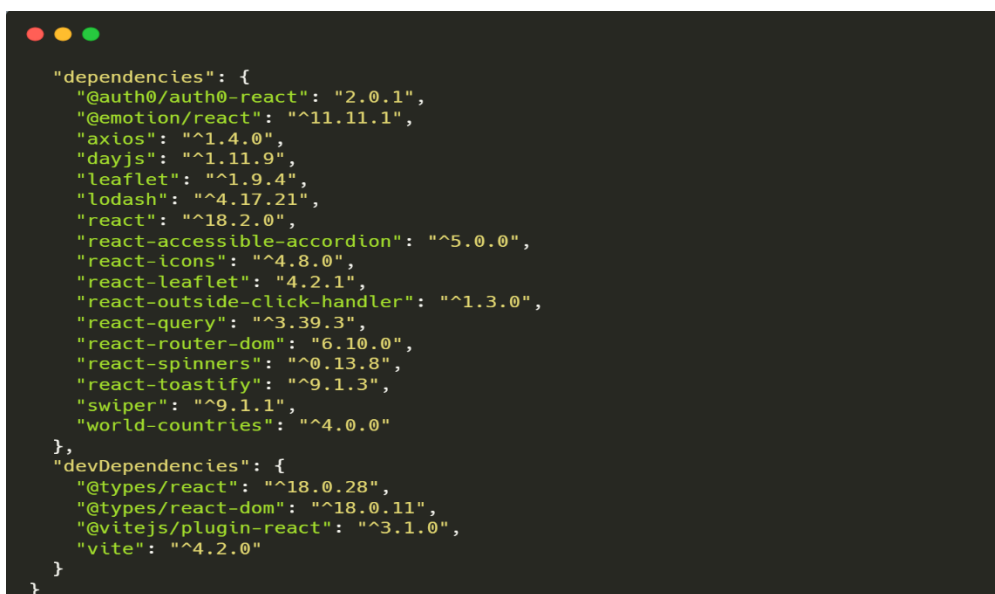
## CHAPTER-5

### IMPLEMENTATION

In the implementation phase, Estatecraft is structured into several modules, each tasked with distinct responsibilities in real estate management. The residency creation module handles the process of adding new properties to the platform, including details such as title, description, and image. User registration is managed separately, ensuring a streamlined process for individuals to create accounts and access the platform's features securely. Booking management facilitates the reservation of properties by users, while favorites handling allows users to mark preferred residencies for easy access. These modules operate independently yet interact seamlessly to ensure a cohesive user experience. By encapsulating specific functionality within each module and enabling inter-module communication, Estatecraft achieves efficient and comprehensive real estate management capabilities.

#### 5.1 Install Node dependencies

The EstateCraft project's dependencies encompass a comprehensive set of libraries and tools crucial for frontend and backend development. On the frontend, React, React Router, and various UI libraries such as Mantine and React Icons facilitate user interface creation and navigation. Backend operations are managed by Express and Mongoose, while Axios and JWT enhance data handling and security. Additionally, testing tools like Testing Library ensure quality assurance. Noteworthy dependencies include Auth0 for authentication and various utility libraries such as lodash for enhanced functionality. These dependencies collectively empower efficient development, ensuring seamless communication between frontend and backend components while facilitating robust security measures and quality testing.



```
{
  "dependencies": {
    "@auth0/auth0-react": "2.0.1",
    "@emotion/react": "^11.11.1",
    "axios": "^1.4.0",
    "dayjs": "^1.11.9",
    "leaflet": "^1.9.4",
    "lodash": "^4.17.21",
    "react": "^18.2.0",
    "react-accessible-accordion": "^5.0.0",
    "react-icons": "^4.8.0",
    "react-leaflet": "4.2.1",
    "react-outside-click-handler": "^1.3.0",
    "react-query": "^3.39.3",
    "react-router-dom": "6.10.0",
    "react-spinners": "^0.13.8",
    "react-toastify": "^9.1.3",
    "swiper": "^9.1.1",
    "world-countries": "^4.0.0"
  },
  "devDependencies": {
    "@types/react": "^18.0.28",
    "@types/react-dom": "^18.0.11",
    "@vitejs/plugin-react": "^3.1.0",
    "vite": "^4.2.0"
  }
}
```

Figure 5.1: Node Dependencies

## 5.2 Backend Connection

The backend connection code for Estatecraft is built using Express, a popular web application framework for Node.js. It utilizes various middleware such as dotenv for environment variable management, cookie-parser for handling cookies, and cors for enabling cross-origin resource sharing. The code initializes an Express application, listens on a specified port (or defaults to port 3000), and establishes routes for user and residency management. User routes handle authentication, registration, and booking functionality, while residency routes manage property creation and retrieval. By structuring the backend with Express and employing middleware for essential functionalities, Estatecraft ensures robust API handling, security, and scalability. Additionally, the use of environment variables enhances configuration management, enabling seamless deployment across different environments.



```
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);
```

**Figure 5.2: Backend Connection**

## 5.3 Database Integration and Schema

The provided code outlines the database integration and schema design for Estatecraft using Prisma as the client provider and MongoDB as the data source. The integration leverages Prisma's capabilities to interact with MongoDB seamlessly, abstracting away the complexities of database operations.

The schema defines two main models: User and Residency. The User model represents individuals accessing the platform, storing essential information such as name, email, image, booked visits, and favorite residencies. On the other hand, the Residency model encapsulates details related to properties listed on the platform, including title, description, price, address, city, country, facilities, and owner information.

By structuring the schema with Prisma and MongoDB, Estatecraft ensures efficient data management, flexibility in schema design, and scalability to accommodate growing user and property data. Additionally, the defined relationships between User and Residency models facilitate seamless navigation and querying of related data entities.



```
generator client {
  provider = "prisma-client-js"
}
datasource db {
  provider = "mongodb"
  url      = env("DATABASE_URL")
}
model User { ... }
model Residency { ... }
```

**Figure 5.3: Database Integration and Schema**

## 5.4 Frontend Creation and Routes

### i. React Application Structure:

- The provided code establishes the foundation of a React application structured with components and pages for frontend development.
- The App component serves as the entry point, wrapping the entire application with essential providers and setting up routing for navigation.
- Components such as Layout, Website, Properties, Property, Bookings, and Favourites represent different sections and functionalities within the application.

### ii. Routing Configuration:

- React Router is utilized for declarative routing, allowing different components to be rendered based on the URL path.
- The `<BrowserRouter>` component establishes a router context, enabling navigation through the browser's URL history.
- Nested `<Route>` components define the mapping between URL paths and corresponding components to be rendered.
- Dynamic routing is achieved using route parameters (`:propertyId`) to display specific property details on the Property page.
- Additional routes are defined for pages like Bookings and Favourites, providing users access to their bookings and favorite properties.

### iii. Context Management:

- The `UserDetailContext` is employed for managing user-related data such as favorites, bookings, and authentication token across different components.

- useState hook is used to initialize and update user details within the App component, ensuring centralized state management for user-related data.
- Context providers wrap the application, enabling components to access and modify user details as needed, promoting a more cohesive and efficient data flow within the application.

iv. **Integration with React Query and Toast Notifications:**

- React Query and Toastify libraries are integrated to enhance data fetching and display notifications to users, respectively.
- The QueryClientProvider wraps the application, providing a context for managing data fetching and caching with React Query.
- Toast notifications are displayed using the <ToastContainer> component, offering real-time feedback to users on actions such as property bookings or authentication status changes.[10]

```
import { Suspense, useState } from "react";
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 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;
```

Figure 5.4: Frontend Creation and Routes

Overall, the frontend creation and routing in the provided code demonstrate a structured approach to building a React application with clear navigation paths and efficient state management. Integration with additional libraries enhances functionality and user experience, making the application more robust and user-friendly.

## **5.5 Backend Functionality**

### **i. Axios Integration:**

- The code utilizes Axios, a popular HTTP client for making asynchronous requests to the backend server.
- Axios allows for seamless communication between the frontend and backend, enabling data exchange and retrieval.

### **ii. API Configuration:**

- The api object is configured with a base URL pointing to the backend server's API endpoint (`http://localhost:8000/api`).
- This configuration ensures that all subsequent API requests are directed to the appropriate backend routes.

### **iii. Fetching Properties:**

- The `getAllProperties` function retrieves all available properties from the backend server.
- It makes a GET request to the `/residency/allresd` endpoint and handles the response accordingly.
- If the request is successful, the function returns the property data; otherwise, it displays an error toast message.

### **iv. Fetching Property Details:**

- The `getProperty` function retrieves details of a specific property identified by its ID.
- It makes a GET request to the `/residency/:id` endpoint, passing the property ID as a parameter.
- Upon successful retrieval, the function returns the property details; otherwise, it displays an error toast.

### **v. User Registration:**

- The `createUser` function facilitates user registration by sending a POST request to the `/user/register` endpoint.
- It requires the user's email and authorization token for authentication.
- Upon successful registration, the function logs the user's email; otherwise, it displays an error toast.



**vi. Booking Visits:**

- The bookVisit function enables users to book visits to specific properties on selected dates.
- It sends a POST request to the /user/bookVisit/:propertyId endpoint, providing the visit details such as date and property ID.
- The function requires the user's email and token for authentication and handles any errors with toast notifications.

**vii. Removing Bookings:**

- The removeBooking function allows users to cancel their booked visits.
- It sends a POST request to the /user/removeBooking/:id endpoint, passing the booking ID as a parameter.
- The function requires the user's email and token for authentication and displays error messages if necessary.

**viii. Adding to Favorites:**

- The toFav function enables users to add properties to their list of favorites.
- It sends a POST request to the /user/toFav/:id endpoint, providing the property ID.
- The function requires the user's email and token for authentication and handles any errors gracefully.

**ix. Fetching Favorites and Bookings:**

- The getAllFav and getAllBookings functions retrieve the user's favorite properties and booked visits, respectively.
- They send POST requests to the /user/allFav and /user/allBookings endpoints, requiring the user's email and token for authentication.
- The functions handle errors and display appropriate toast messages if necessary.

**x. Creating Residencies:**

- The createResidency function facilitates the addition of new properties to the database.
- It sends a POST request to the /residency/create endpoint, providing residency data.
- The function requires the user's authorization token for authentication and handles any errors gracefully.

The provided backend code facilitates seamless frontend-backend interaction for property retrieval, user registration, booking management, and favorites handling. Utilizing Axios or HTTP requests with integrated error handling ensures reliable communication and a smooth user experience.

```

export const getAllProperties = async () => {
  try {
    const response = await api.get("/residency/allresd", {
      timeout: 10 * 1000,
    });

    if (response.status === 400 || response.status === 500) {
      throw response.data;
    }
    return response.data;
  } catch (error) {
    toast.error("Something went wrong");
    throw error;
  }
};

export const getProperty = async (id) => {
  try {
    const response = await api.get(`/residency/${id}`, {
      timeout: 10 * 1000,
    });

    if (response.status === 400 || response.status === 500) {
      throw response.data;
    }
    return response.data;
  } catch (error) {
    toast.error("Something went wrong");
    throw error;
  }
};

export const createUser = async (email, token) => {
  try {
    await api.post(
      "/user/register",
      { email },
      {
        headers: {
          Authorization: `Bearer ${token}`,
        },
      }
    );
    console.log(email);
  } catch (error) {
    toast.error("Something went wrong, Please try again");
    throw error;
  }
};

export const removeBooking = async (id, email, token) => {
  try {
    await api.post(
      "/user/removeBooking/${id}",
      {
        email,
      },
      {
        headers: {
          Authorization: `Bearer ${token}`,
        },
      }
    );
  } catch (error) {
    toast.error("Something went wrong, Please try again");
    throw error;
  }
};

export const getAllBookings = async (email, token) => {
  if (!token) return;
  try {
    const res = await api.post(
      "/user/allBookings",
      {
        email,
      },
      {
        headers: {
          Authorization: `Bearer ${token}`,
        },
      }
    );
    return res.data["bookedVisits"];
  } catch (error) {
    toast.error("Something went wrong while fetching bookings");
    throw error;
  }
};

export const bookVisit = async (date, propertyId, email, token) => {
  try {
    await api.post(
      "/user/bookVisit/${propertyId}",
      {
        email,
        id: propertyId,
        date: dayjs(date).format("DD/MM/YYYY"),
      },
      {
        headers: {
          Authorization: `Bearer ${token}`,
        },
      }
    );
  } catch (error) {
    toast.error("Something went wrong, Please try again");
    throw error;
  }
};

export const toFav = async (id, email, token) => {
  try {
    await api.post(
      "/user/toFav/${id}",
      {
        email,
      },
      {
        headers: {
          Authorization: `Bearer ${token}`,
        },
      }
    );
  } catch (e) {
    throw e;
  }
};

export const createResidency = async (data, token) => {
  console.log(data);
  try {
    const res = await api.post(
      "/residency/create",
      {
        data,
      },
      {
        headers: {
          Authorization: `Bearer ${token}`,
        },
      }
    );
  } catch (error) {
    throw error;
  }
};

```

Figure 5.5: Backend Functionality

## 5.6 Testing and Debugging

Testing and debugging using ThunderClient, a versatile API client extension for Visual Studio Code, involves a streamlined and efficient process to ensure the functionality and performance of APIs. ThunderClient provides a user-friendly interface to create, send, and analyze HTTP requests, making it an excellent tool for developers.

In the testing phase, ThunderClient simplifies the creation of requests by offering intuitive input fields for various parameters like headers, query parameters, and request bodies. This facilitates comprehensive testing of API endpoints with different inputs. It supports various HTTP methods, aiding in testing different scenarios such as GET, POST, PUT, DELETE, etc. Test suites can be organized, enabling the execution of multiple requests in a sequence, emulating real-world scenarios.

The debugging process is made more efficient by ThunderClient's real-time response visualization. Responses are presented in a well-structured format, making it easier to spot errors or discrepancies. Debugging is further enhanced through ThunderClient's ability to display headers, response time, status codes, and response body in a clear manner.

ThunderClient's interactive interface allows developers to modify requests on the fly and observe how changes impact responses. This feature expedites debugging by enabling immediate adjustments and iterations to ensure proper functionality. Additionally, ThunderClient supports exporting and importing requests, facilitating collaboration and documentation.

In summary, ThunderClient simplifies testing and debugging by providing an intuitive interface, real-time response visualization, and on-the-fly request modifications. Its capabilities enhance the efficiency and accuracy of identifying and rectifying issues within APIs, contributing to the overall quality and reliability of web applications.[11]

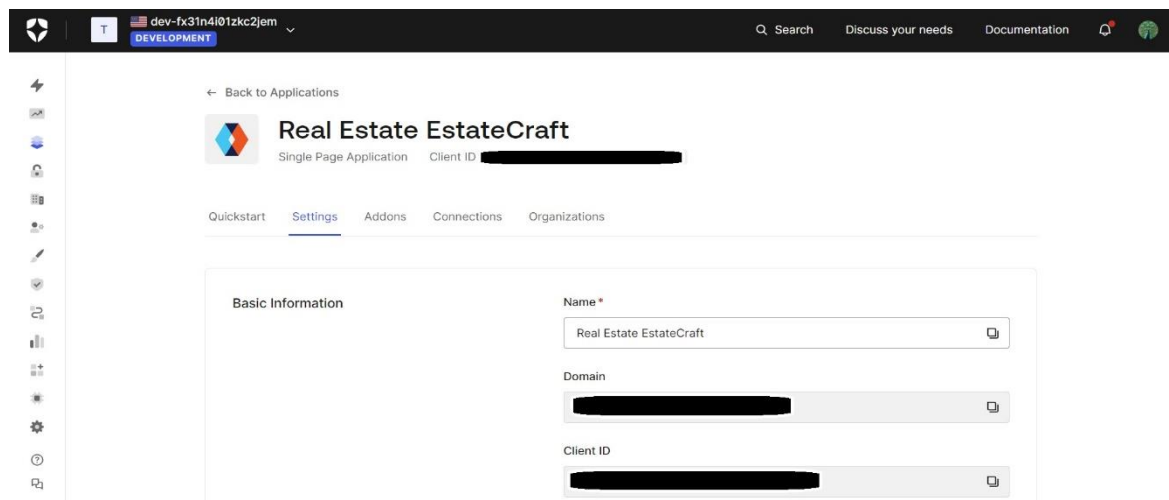
## 5.7 Responsiveness and User Experience

The "EstateCraft" project is designed to elevate user experience by integrating the MongoDB, Express.js, React, and Node.js technologies. It emphasizes creating a seamless and engaging platform for users. Leveraging React's dynamic components and state management, the interface offers intuitive navigation and real-time updates. Express.js enables efficient data handling and API integration, enhancing responsiveness. MongoDB stores and retrieves data, ensuring scalability and robustness. The project optimizes user interactions through responsive design, adapting content to diverse screen sizes. User-centric features, like forms and personalized recommendations, enrich engagement. Comprehensive testing and debugging ensure a smooth user journey. Overall, the MERN project prioritizes user satisfaction by

seamlessly combining these technologies, resulting in a sophisticated, dynamic, and user-friendly web application that delivers a highly satisfying and immersive user experience.[12]

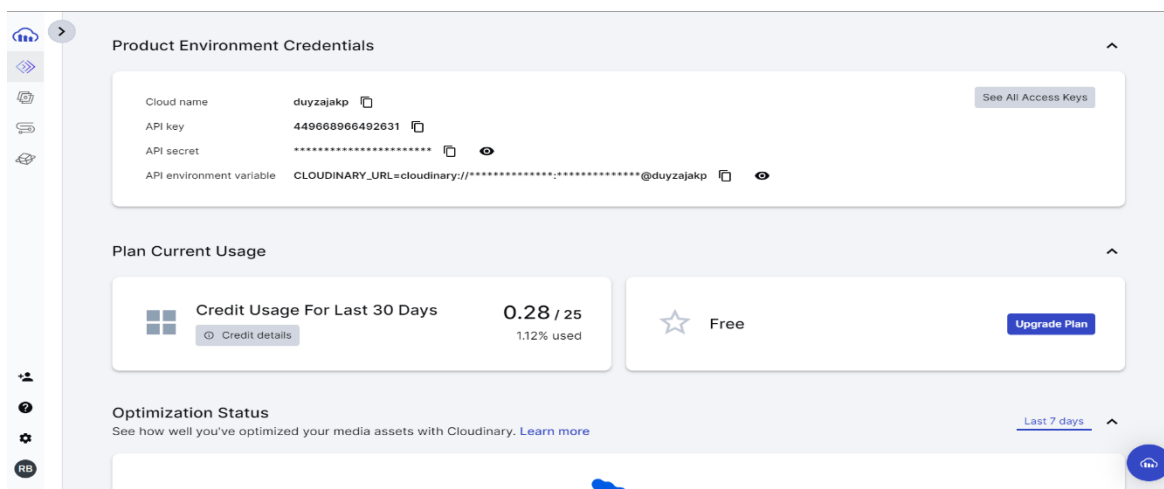
## 5.8 Auth0 and Cloudinary Setup

**Auth0:** Auth0 is a powerful authentication and authorization platform that simplifies user authentication in applications. During the initial setup, developers configure Auth0 by creating an account, setting up applications, and integrating authentication mechanisms such as social login or username/password authentication. This process involves obtaining client IDs and secrets, configuring callback URLs, and implementing Auth0 SDKs or libraries in the application codebase.



**Figure 5.6: Auth0 Setup**

**Cloudinary:** Cloudinary is a cloud-based image and video management service that streamlines media handling in applications. The initial setup of Cloudinary involves creating an account, configuring settings such as upload presets and transformations, and integrating the Cloudinary SDK or library into the application code. Developers generate API keys and secrets to authenticate requests to Cloudinary's API endpoints, enabling functionalities like uploading, storing, manipulating, and delivering images and videos efficiently.

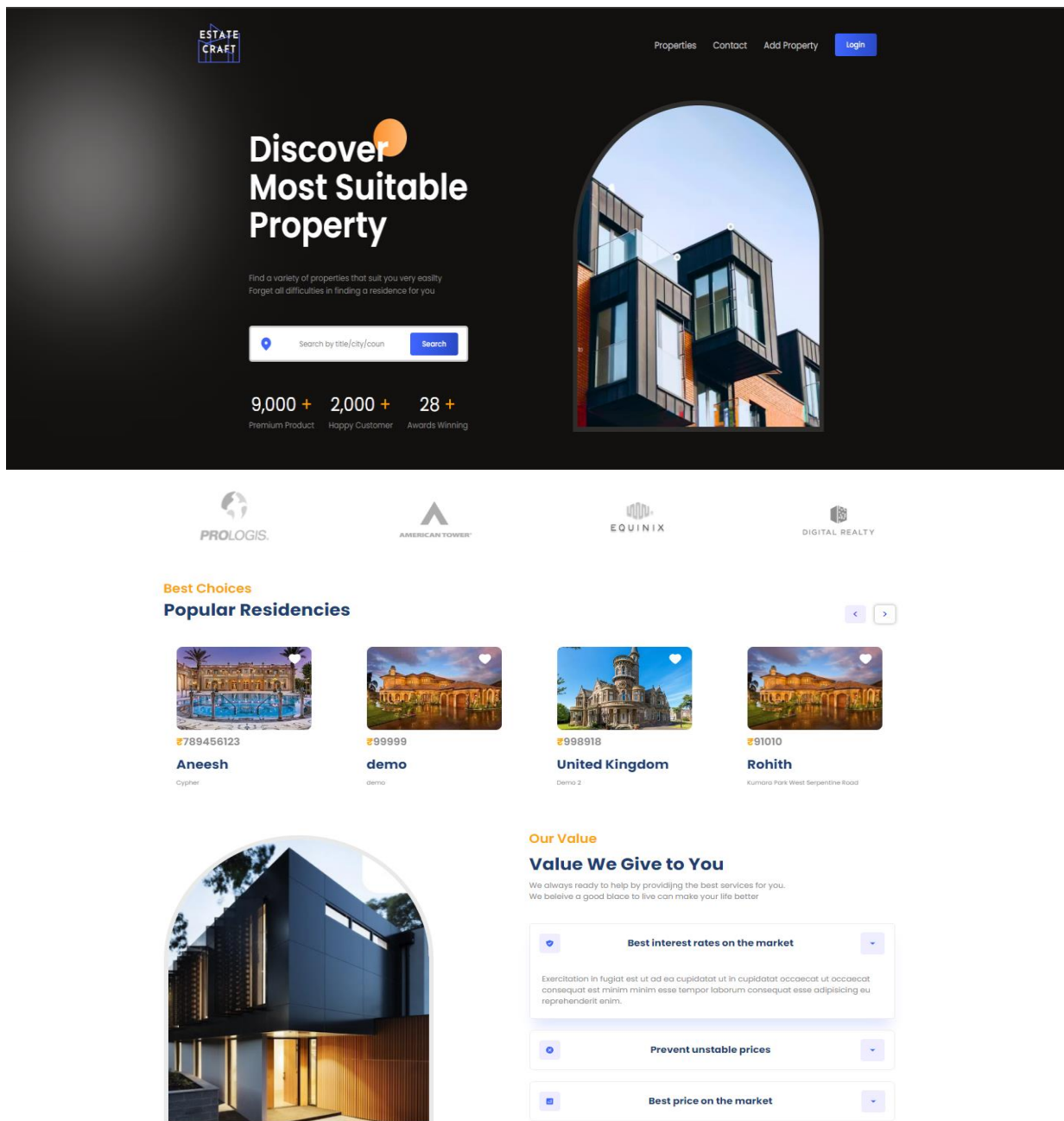


**Figure 5.7: Cloudinary Setup**

## CHAPTER-6

# RESULTS

### 6.1 Landing Page



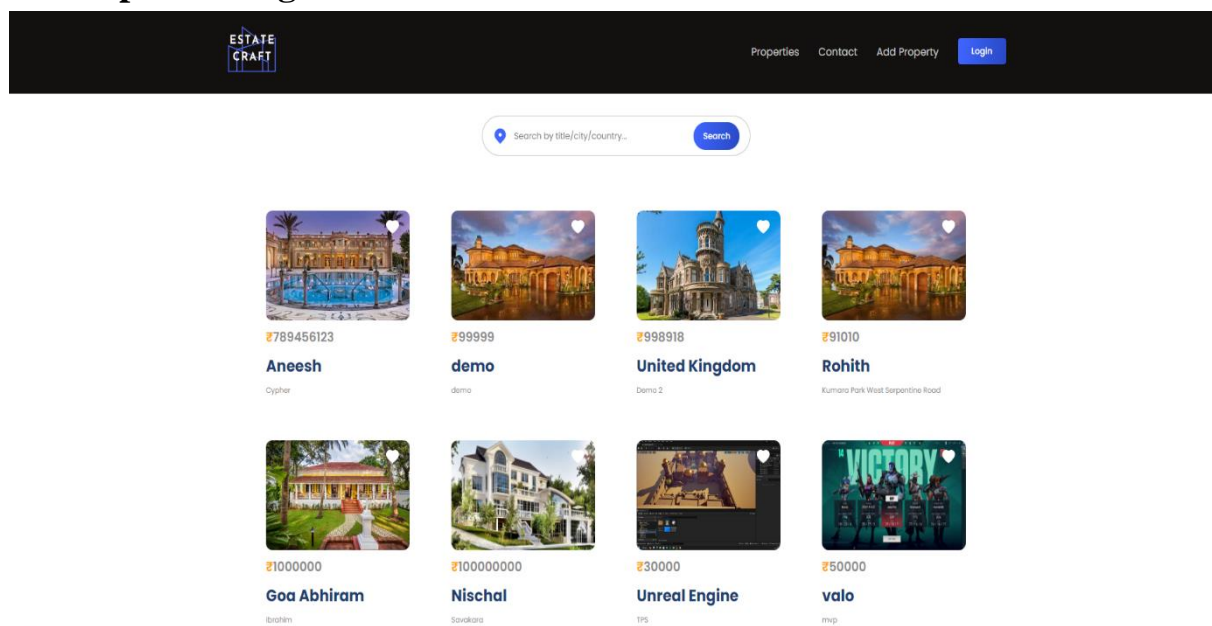
**Figure 6.1: Landing Page**

The “ESTATE CRAFT” website is a comprehensive platform designed to cater to the diverse needs of property buyers and sellers. The landing page immediately captures the attention of visitors with its clean design and intuitive navigation. The header provides easy access to various sections of the website, including “Properties”, “Contact”, “Add Property”, and “Login”, ensuring a seamless user experience.

The main content area of the landing page showcases a variety of properties, reflecting the website's extensive portfolio. The properties are presented in a visually appealing manner, with high-quality images and detailed information. This not only enhances the user experience but also helps potential buyers make informed decisions. The website also features a powerful search function, allowing users to find properties that match their specific requirements. Whether it's a city apartment or a countryside house, users can easily find properties that suit their lifestyle and budget.

Overall, the "ESTATE CRAFT" website stands out as a reliable and user-friendly platform in the real estate market. It successfully combines functionality with aesthetics, offering a valuable service to its users. Its modern design, extensive property listings, and commitment to providing the best market prices make it a go-to platform for all real estate needs. The website reflects a strong brand image, assuring users of its reliability and service quality. It is a testament to the potential of digital platforms in transforming the real estate industry.

### 6.2 Properties Page

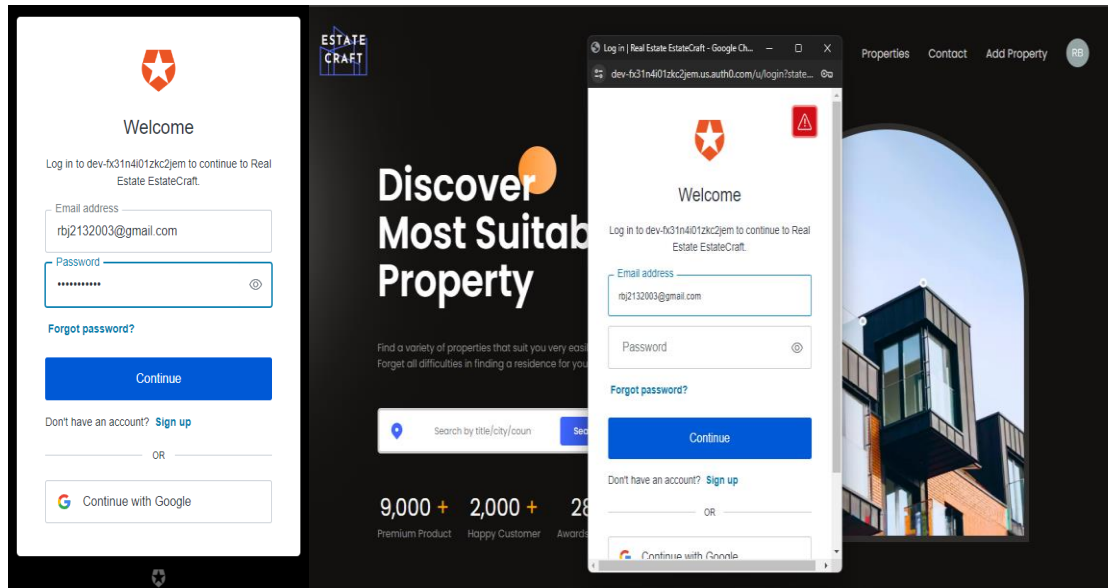


**Figure 6.2: Properties Page**

The "ESTATE CRAFT" property page is a well-organized platform showcasing various property listings. Each listing is presented in a detailed card format, featuring an image, identification number, and additional text. The user-friendly interface, complete with a search bar, ensures easy navigation. The page effectively caters to diverse property needs, offering a wide range of options from city apartments to countryside houses. Overall, it provides a seamless property browsing experience, making it a reliable platform for real estate transactions.



## 6.3 Login and Authentication



**Figure 6.3: Login and Authentication**

The “ESTATE CRAFT” website provides a seamless login experience for both first-time and returning users. The login interface is user-friendly, featuring fields for email and password. For new users, there’s an option to sign up, ensuring a smooth onboarding process. In case users forget their password, a ‘Forgot password?’ feature is available for quick account recovery. Notably, the website also incorporates Google authentication, offering users an alternative and convenient way to access their accounts using their Google credentials. This feature enhances the user experience by providing a secure and efficient login method. The login popup appears over the homepage, allowing users to access their accounts without navigating away from the main page. Overall, “ESTATE CRAFT” prioritizes user convenience and security in its login process.

## 6.4 Add Property Feature

The “Add Property” feature on the “ESTATE CRAFT” website is a meticulously designed, four-phase process that ensures a comprehensive and detailed property listing. This process is not only user-friendly but also efficient, making it easy for users to list their properties on the platform.

In the first phase, users are asked to provide the location details of the property. This includes the country, city, and specific address. The interface is intuitive, with fields for each detail. A map is provided to help users ensure they have entered the correct location. This phase is crucial as it helps potential buyers or renters understand the geographical context of the property.

The second phase involves uploading images of the property. Users can easily upload images directly from their device. This phase enhances the property listing with visual details, giving

potential buyers or renters a visual understanding of the property. It allows users to showcase their property in the best possible light.

The third phase requires users to enter basic details about the property. This includes the title, description, and price. This phase ensures that all necessary information about the property is collected. The title and description allow users to provide additional details about the property that may not be evident from the images. The price field is particularly important as it gives potential buyers or renters an idea of the cost of the property.

The final phase involves entering detailed amenities about the property. This includes the number of bedrooms, parking spaces, and bathrooms. This information helps potential buyers or renters understand the features of the property. It provides a comprehensive overview of the property, ensuring that users have all the information they need to make an informed decision. Overall, the “Add Property” feature on the “ESTATE CRAFT” website ensures a comprehensive and detailed property listing. It enhances the user experience and the effectiveness of the platform, making it a reliable platform for real estate transactions. By guiding users through each phase, it ensures that all necessary information is collected, contributing to a comprehensive property listing. This process reflects the website’s commitment to providing a user-friendly and efficient platform for real estate transactions.

The figure illustrates the 'Add Property' procedure through four sequential steps:

- Step 1: Location (Address)** - Users select a location on a map of India. Input fields for Country (India), City (Bangalore), and Address (Jaynagar) are provided. A 'Next Step' button is at the bottom.
- Step 2: Images (Upload)** - A large image of a villa is displayed. 'Back' and 'Next' buttons are at the bottom.
- Step 3: Basics (Details)** - Users enter property details: Title (Villa), Description (Big House), and Price (100000000). 'Back' and 'Next step' buttons are at the bottom.
- Step 4: Amenities** - Users specify property features: No of Bedrooms (5), No of Parkings (3), and No of Bathrooms (4). 'Back' and 'Add Property' buttons are at the bottom.

**Figure 6.4: Add Property Procedure**



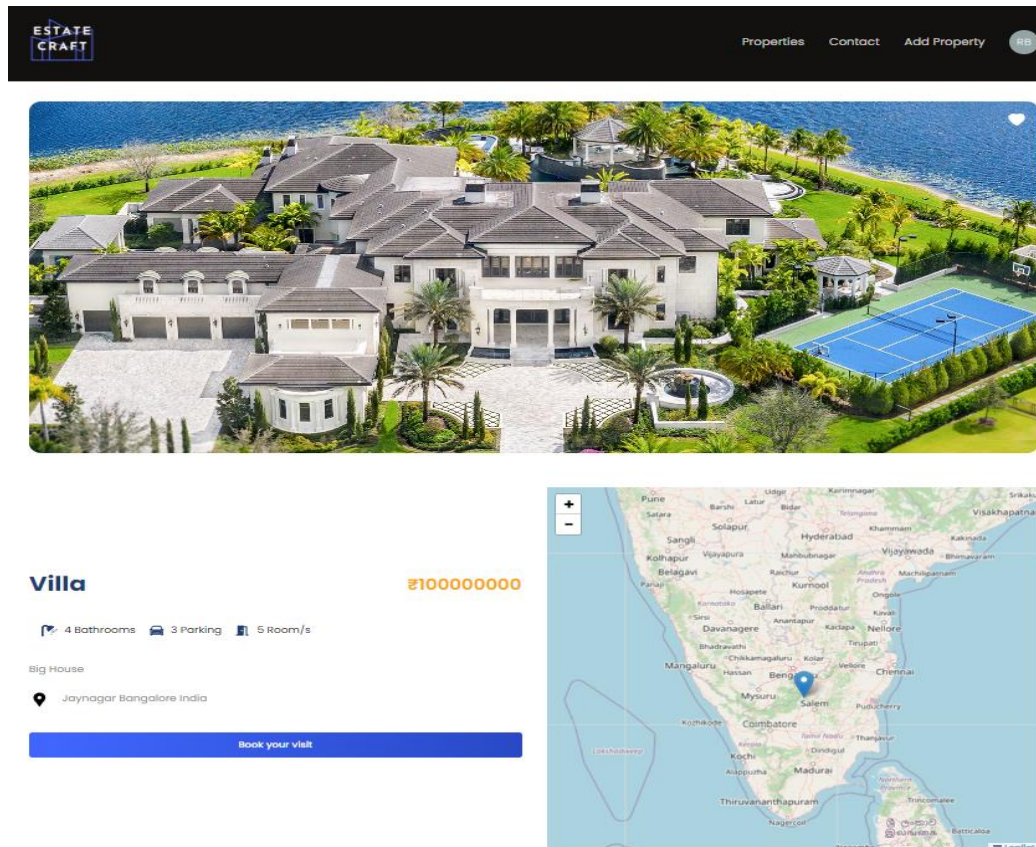


Figure 6.5: Property Page

## 6.5 Booking Feature

The website "ESTATE CRAFT" streamlines the property booking process with meticulous attention to detail, ensuring user-friendliness at every step. Initially, users are greeted with a plethora of property details including vivid images, room and bathroom counts, parking availability, and a succinct description. Additionally, the property's geographical location is prominently displayed, aiding users in gauging its context. An inviting "Book your visit" button beckons users to proceed seamlessly.

Moving forward, the appointment scheduling phase offers users an intuitive calendar interface, allowing them to effortlessly select their preferred visit date. Dates that are unavailable are clearly marked, ensuring users can choose a convenient time for their visit, thereby enhancing their overall experience.

The subsequent phase requires users to input fundamental property details such as the title, description, and price. These details are crucial for potential buyers or renters as they provide a comprehensive understanding of the property beyond just visuals. The title and description enable users to impart additional insights that may not be evident from the images alone, while the price field offers transparency regarding the property's cost.

In the final phase, users delve into the specifics of the property's amenities, encompassing factors such as bedroom count, parking spaces, and bathrooms. This information is pivotal in aiding potential buyers or renters in comprehending the property's features comprehensively, thereby facilitating an informed decision-making process.

Upon furnishing all requisite details, users are promptly furnished with a booking confirmation, assuring them of the successful processing of their booking. Furthermore, users are endowed with the flexibility to cancel their booking should their plans undergo a change, thereby instilling a sense of control and reassurance.

In summation, the booking process on the "ESTATE CRAFT" website epitomizes simplicity and efficiency, with each phase meticulously crafted to offer clear instructions and user-friendly interfaces. From perusing property details to scheduling a visit and receiving a booking confirmation, every aspect is thoughtfully designed to furnish users with a seamless experience. The provision for booking cancellation further accentuates the platform's user-centric design, ensuring users have the flexibility they require when planning property visits. This refined process, coupled with the comprehensive property listings, underscores the website's commitment to providing a reliable and convenient platform for property bookings, thereby exemplifying the transformative potential of digital platforms within the real estate industry.

Select your date of visit

<

March 2024

>

Mo	Tu	We	Th	Fr	Sa	Su
26	27	28	29	1	2	3
4	5	6	7	8	9	10
11	12	13	14	15	16	17
18	19	20	21	22	23	24
25	26	27	28	29	30	31

Book visit

Villa

₹100000000

4 Bathrooms

3 Parking

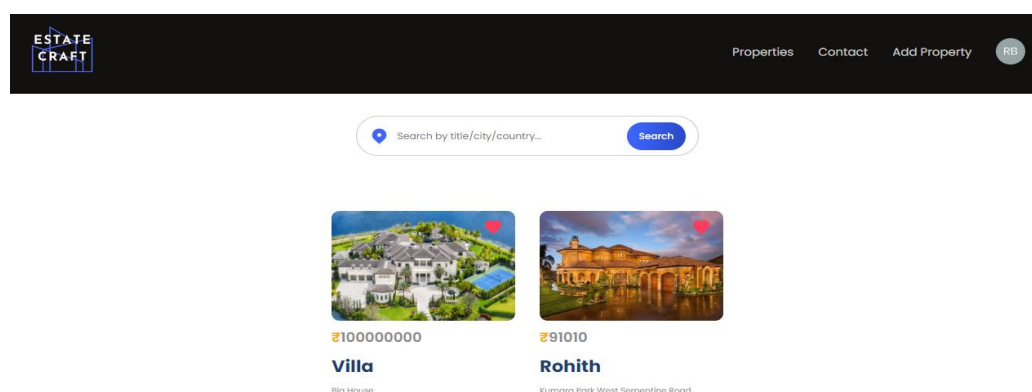
5 Room/s

Big House

Jaynagar Bangalore India

Cancel booking

Your visit already booked for date 29/03/2024



**Figure 6.6: Booking Feature**

## 6.6 Favourites Feature

The “Favorites” tab on the “ESTATE CRAFT” website provides a personalized space for users to save and quickly access properties they are interested in. Users can mark properties as favorites, and they will be stored in this section for easy retrieval. This feature enhances the user experience by allowing potential buyers to compare, review details, or show them to others without having to search for them again. It reflects the website’s commitment to providing a user-friendly and efficient platform for its users.

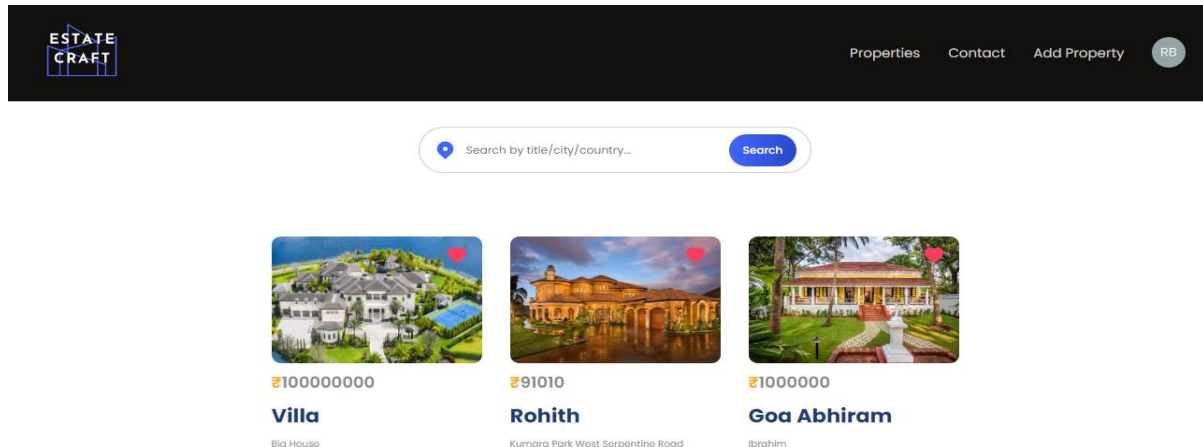


Figure 6.7: Favourites Feature

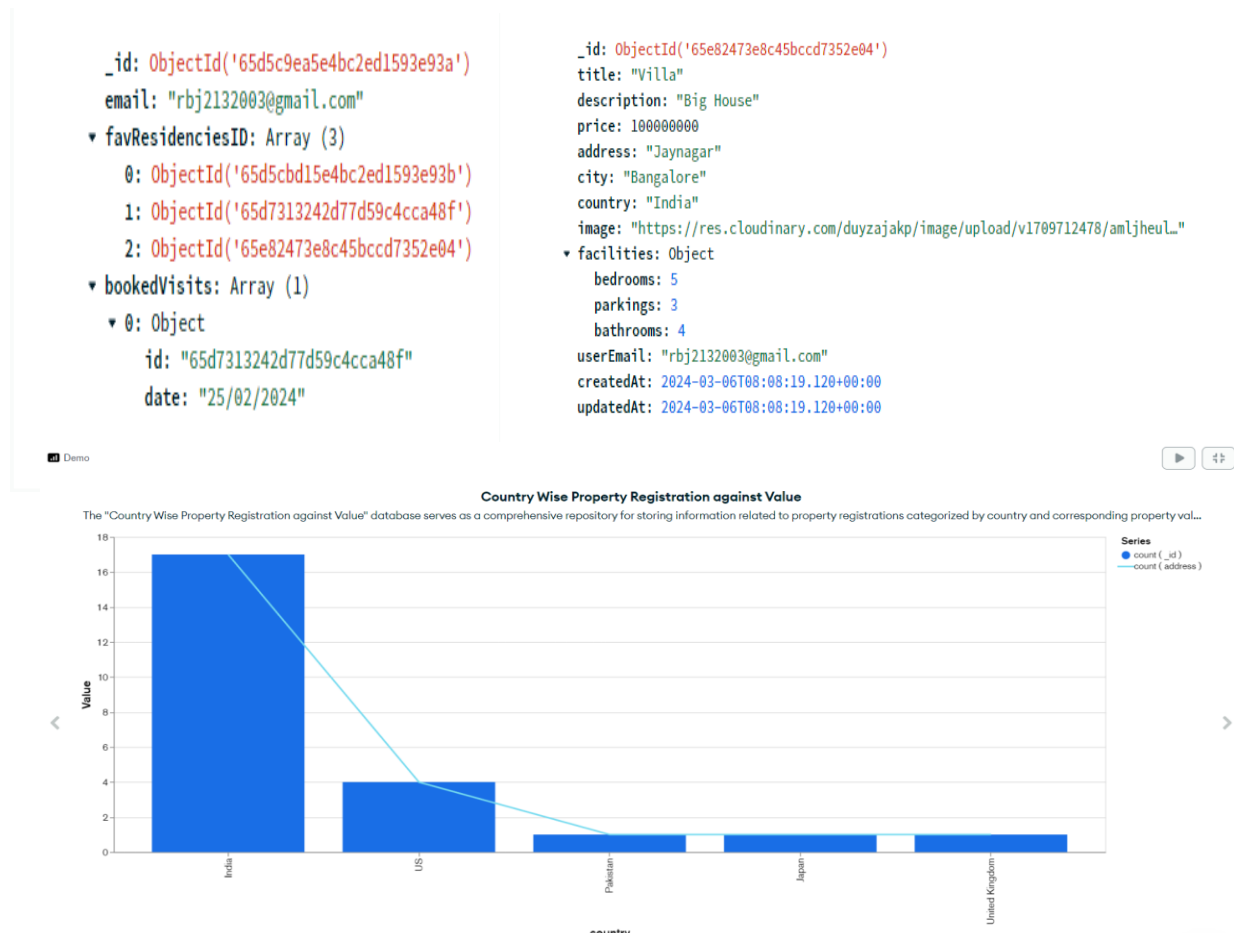
## 6.7 Database Operation

The Figure, which represents a property listing on the “ESTATE CRAFT” website. This entry is likely the result of the “Add Property” feature you mentioned earlier. Let’s break down the CRUD (Create, Read, Update, Delete) operations in this context:

- **Create:** When you add a new property listing through the “Add Property” feature, a new document is created in the MongoDB Atlas database. This document contains key-value pairs that store information about the property, such as its title, description, price, location, and facilities.
- **Read:** When you or other users view the property listings on the website, the application reads the documents from the database and displays the information. For instance, when viewing the details of the “Villa” property, the application reads the corresponding document in the database.
- **Update:** If you need to change any information about the property, such as updating the price or changing the description, the corresponding document in the database is updated. For example, if the price of the “Villa” property changes, the ‘price’ field in the corresponding document is updated.

- Delete: If a property is removed from the listings, the corresponding document is deleted from the database.

In the database entry shown in your image, we can see various fields like 'title', 'description', 'price', 'location', and 'facilities', which store the respective details of the property. This structured way of storing data allows for efficient CRUD operations, ensuring that the "ESTATE CRAFT" website can effectively manage its property listings. It's a testament to the flexibility and efficiency of using MongoDB Atlas for handling data in real estate applications.



**Figure 6.8: Database Operation**

## CHAPTER-7

# Conclusion and Future Scope

### 7.1 Conclusion

In conclusion, the development journey of EstateCraft has been an enriching and educational experience, providing valuable insights into various aspects of modern web development. Throughout the project, we delved into the MERN (MongoDB, Express, React, Node.js) stack, a popular technology stack known for its flexibility and scalability. Working with MongoDB Atlas, the cloud-hosted database service, offered valuable hands-on experience in managing data storage and retrieval in a cloud environment. We gained proficiency in schema design, data modeling, and querying techniques, enabling us to efficiently structure and manage the property and user data within the application.

Moreover, integrating authentication and authorization functionalities using Auth0 proved to be a crucial aspect of the project. Understanding the authentication flow, implementing secure authentication mechanisms, and managing user sessions were key learning points that contributed to enhancing the application's security and user experience. Additionally, leveraging Cloudinary for image management further expanded our skillset, allowing us to efficiently handle image uploads, transformations, and optimizations within the application.

On the frontend side, exploring various React.js components and libraries, including Leaflet for map integration and Mantine for UI components, provided valuable insights into building responsive and visually appealing user interfaces. Understanding how to utilize these libraries effectively, customize components, and handle state management enhanced our proficiency in frontend development and UI/UX design principles.

Throughout the project, we encountered and overcame various challenges, including debugging issues, optimizing performance, and ensuring compatibility across different devices and browsers. These challenges served as valuable learning opportunities, helping us develop problem-solving skills and gain a deeper understanding of the intricacies involved in web application development.

Overall, the EstateCraft project has been an invaluable learning experience, allowing us to gain practical knowledge and hands-on experience with a wide range of tools, technologies, and best practices in web development. Moving forward, the insights and skills acquired during this project will continue to serve us well in our future endeavors, empowering us to tackle new challenges and create innovative solutions in the dynamic field of web development.

## **7.2 Future Scope**

Looking ahead, EstateCraft holds significant potential for further enhancement and expansion to meet the evolving needs of users and capitalize on emerging trends in the real estate industry.

Some future scope areas include:

- **Advanced Search and Filtering:** Implementing advanced search and filtering capabilities based on various parameters such as price range, property type, amenities, and location can improve the user experience by helping users find properties that match their specific criteria more efficiently.
- **Enhanced User Interactions:** Introducing features like real-time chat support, virtual property tours, and interactive property maps can enhance user engagement and facilitate seamless communication between property owners, tenants, and buyers.
- **Integration with External APIs:** Integrating with external APIs such as property listing services, mortgage calculators, and real-time market data providers can provide users with valuable insights and streamline decision-making processes related to property buying, selling, and renting.
- **Personalized Recommendations:** Implementing machine learning algorithms to analyze user preferences, browsing history, and interactions can enable the platform to offer personalized property recommendations tailored to individual user preferences, enhancing user satisfaction and retention.
- **Multi-language Support:** Adding support for multiple languages can enhance the accessibility and usability of the platform for users from diverse linguistic backgrounds, broadening its reach and appeal to a global audience.
- **Enhanced Security Features:** Strengthening security measures such as two-factor authentication, encrypted data storage, and regular security audits can bolster user trust and confidence in the platform, safeguarding sensitive user information from potential security threats.
- **Mobile Application Development:** Developing dedicated mobile applications for iOS and Android platforms can provide users with greater convenience and flexibility in accessing the platform's features and services on their mobile devices, further enhancing the platform's accessibility and user experience.

By focusing on these future scope areas and continually iterating and improving the platform based on user feedback and market trends, EstateCraft can solidify its position as a leading destination for real estate solutions, offering innovative features and unparalleled value to its users.

## REFERENCES

- [1] [\*https://auth0.com/\*](https://auth0.com/)
- [2] [\*https://cloudinary.com/\*](https://cloudinary.com/)
- [3] [\*https://mantine.dev/\*](https://mantine.dev/)
- [4] [\*https://axios-http.com/docs/intro\*](https://axios-http.com/docs/intro)
- [5] [\*https://leafletjs.com/reference.html\*](https://leafletjs.com/reference.html)
- [6] [\*https://legacy.reactjs.org/docs/getting-started.html\*](https://legacy.reactjs.org/docs/getting-started.html)
- [7] [\*https://www.prisma.io/docs\*](https://www.prisma.io/docs)
- [8] [\*https://expressjs.com/\*](https://expressjs.com/)
- [9] [\*https://www.npmjs.com/package/nodemon\*](https://www.npmjs.com/package/nodemon)
- [10] [\*https://reactrouter.com/en/main/components/routes\*](https://reactrouter.com/en/main/components/routes)
- [11] [\*https://www.youtube.com/watch?v=edBx-fjgh4k&list=WL&index=7&pp=gAQBiAQB\*](https://www.youtube.com/watch?v=edBx-fjgh4k&list=WL&index=7&pp=gAQBiAQB)
- [12] [\*https://www.youtube.com/watch?v=rNIfNO-ATb0&list=WL&index=9&t=7350s&pp=gAQBiAQB\*](https://www.youtube.com/watch?v=rNIfNO-ATb0&list=WL&index=9&t=7350s&pp=gAQBiAQB)