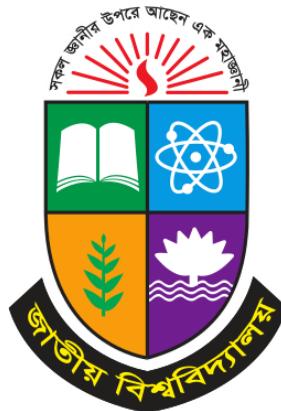


**DreamNest**  
**[Real Estate Management System]**



*A project submitted in partial fulfillment of the requirements for the degree of*  
**Bachelor of Science in Computer Science and Engineering**

**Submitted By**

Asif Iqbal

Reg: 19502005235

&

Md. Manjurul Islam

Reg: 19502005262

**Supervised By**

MD. Al Mamun Khan

Lecturer

Department Of Computer Science and Engineering  
Uttara Institute Of Business And Technology ( UIBT )



**UTTARA INSTITUTE OF BUSINESS AND TECHNOLOGY ( UIBT )**  
**2023**

## **Approval**

This is to certify that the capstone project report entitled “DreamNest: A Real Estate Management System”, submitted by Asif Iqbal, Reg:19502005235 and Md. Manjurul Islam, Reg:19502005262 undergraduate students of the Department of Computer Science and Engineering, Uttara Institute of Business and Technology, has been examined.

Upon the recommendation of the examination committee, we hereby accord our approval of this report as the presented work and submitted document fulfill the requirements for its acceptance in partial fulfillment of the degree of Bachelor of Science in Computer Science and Engineering (B.Sc. in CSE).

---

## **SUPERVISOR**

**MD. AI MAMUN KHAN**

Lecturer

Department Of Computer Science and Engineering  
Uttara Institute Of Business And Technology (UIBT)

## **DECLARATION**

We, the undersigned, hereby declare that the project report titled “DreamNest: A Real Estate Management System” has been carried out by us under the supervision of our respective instructor at the Uttara Institute of Business and Technology as part of the partial fulfillment of the requirements for the Bachelor of Science in Computer Science and Engineering (CSE) degree.

We further declare that this report is our original work and has not been submitted, in whole or in part, to any other institution or organization for any academic or professional purpose. All sources of information used in the preparation of this report have been duly acknowledged and referenced.

We take full responsibility for the accuracy and authenticity of the contents of this report.

**Submitted by:**

**Asif Iqbal**

Reg: 19502005235

Department of Computer Science and Engineering  
Uttara Institute Of Business And Technology ( UIBT)

&

**Md. Manjurul Islam**

Reg: 19502005262

Department of Computer Science and Engineering  
Uttara Institute Of Business And Technology ( UIBT)

## **ACKNOWLEDGEMENTS**

We would like to express our deepest gratitude to the Almighty Allah for granting us the strength, patience, and perseverance to successfully complete this project entitled “DreamNest: A Real Estate Management System.”

We extend our heartfelt appreciation to our respected supervisor, Md. Al Mamun Khan, lecturer Department of Computer Science and Engineering, Uttara Institute of Business and Technology, for his invaluable guidance, encouragement, and continuous support throughout the development of this project. His constructive feedback, technical expertise, and insightful suggestions have been essential to the successful completion of this work.

We would also like to convey our sincere thanks to all the faculty members of the Department of Computer Science and Engineering for their continuous cooperation, advice, and motivation throughout our academic journey.

Furthermore, we are grateful to our friends, classmates, and family members for their constant inspiration, moral support, and encouragement, which helped us overcome the challenges faced during the project development process.

Finally, we would like to acknowledge all individuals and resources that indirectly contributed to the success of this project. Without their guidance and support, this work would not have been possible.

### **Submitted by:**

**Asif Iqbal**

Reg: 19502005235

Department of Computer Science and Engineering  
Uttara Institute Of Business And Technology ( UIBT)

&

**Md. Manjurul Islam**

Reg: 19502005262

Department of Computer Science and Engineering  
Uttara Institute Of Business And Technology ( UIBT)

## **ABSTRACT**

In recent years, the demand for efficient, transparent, and user-friendly property management solutions has increased significantly due to the rapid growth of the real estate industry and the need for digital transformation. The project titled “DreamNest: A Real Estate Management System” is designed to provide a comprehensive, secure, and modern platform for managing property listings, client interactions, and administrative workflows. The system enables buyers, sellers, agents, and administrative staff to interact seamlessly within a unified digital environment while maintaining data security, accuracy, and operational transparency.

The proposed system leverages advanced web technologies and robust security mechanisms to ensure smooth property transactions and prevent unauthorized data access. It incorporates secure user authentication through OTP-based email verification, JWT token authorization, and password encryption using bcrypt. Administrators and agents can efficiently manage property listings, client inquiries, bookings, and transactions through an intuitive dashboard equipped with features such as property management, user registration, appointment scheduling, and real-time analytics.

Overall, DreamNest aims to modernize the traditional real estate workflow by offering a secure, paperless, and interactive digital platform. Its modular architecture, strong authentication protocols, and data-driven reporting capabilities make it highly suitable for real estate agencies, property developers, and organizations seeking an all-in-one digital solution for property management.

**Keywords:** Real Estate Management System, Property Listings, OTP Verification, Node.js, React.js, MongoDB, Secure Authentication, Real-time Property Management

## **DEDICATION**

This project report is dedicated to our beloved parents, whose unconditional love, prayers, and encouragement have been our greatest source of strength and inspiration throughout our academic journey.

We also dedicate this work to our teachers and mentors at the Uttara Institute of Business and Technology, whose constant guidance and support have shaped our knowledge and professional growth.

Lastly, we dedicate DreamNest: A Real Estate Management System to all students, teachers, and future innovators who aspire to use technology for building a more transparent, secure, and digitally empowered society.

## Table of Contents

<b>Approval</b> .....	ii
<b>DECLARATION</b> .....	iii
<b>ACKNOWLEDGEMENTS</b> .....	iv
<b>ABSTRACT</b> .....	v
<b>DEDICATION</b> .....	vi
List of Tables .....	ix
<b>List of Figure</b> .....	ix
<b>Chapter 1 – Introduction</b> .....	1
1.1 Project Background .....	1
1.2 Scope of the Project .....	1
1.3 Objectives of the Project.....	2
1.4 Significance of the Study.....	3
1.5 Motivation and Justification .....	3
1.6 System Overview.....	3
<b>Chapter 2 – Analysis</b> .....	4
2.1 Existing System and Proposed System.....	4
2.1.1 History and Evolution of the System.....	4
2.1.2 Issues with the Current System.....	4
2.1.3 Proposed System.....	5
2.2 Feasibility Study .....	5
2.2.1 Economic Feasibility .....	5
2.2.2 Technical Feasibility.....	6
2.2.3 Operational Feasibility.....	6
2.3 Nonfunctional Requirements .....	6
2.3.1 Product Requirements.....	6
2.3.2 Organizational Requirements .....	7
2.4 Functional Requirements .....	7
2.4.1 User Functions .....	7
2.4.2 Admin Functions .....	7
2.4.3 Property Functions .....	8
2.5 Project Requirement Specification .....	8
2.5.1 Hardware Requirements .....	8
2.5.2 Software Requirements.....	8
2.6 Cost Benefit Analysis .....	9
2.7 Risk Analysis .....	9
<b>Chapter 3 – Design</b> .....	10
3.1 Methodology .....	10
3.1.1 Agile Methodology Overview .....	10

3.2 Entity Relationship (ER) Diagram.....	10
3.3 Context Diagram.....	11
3.4 Data Flow Diagram (DFD).....	12
3.5 Use Case Diagram .....	14
3.6 Database Tables .....	15
3.6.1 User Collection .....	16
3.6.2 Property Collection .....	16
3.6.3 Booking Collection.....	16
3.6.4 Example Relationships .....	16
3.6.5 Database Normalization and Integrity .....	17
3.6.6 Data Security and Backup.....	17
<b>Chapter 4: Final Outcome .....</b>	<b>19</b>
4.1 Overview of the Developed System .....	19
4.2 System Architecture.....	19
4.3 System Overview Diagram.....	20
4.4 Developed Dashboards .....	20
4.4.1 User Dashboard .....	20
4.4.2 Admin Dashboard .....	20
4.5 Features Implementation .....	21
4.5.1 Authentication and Authorization.....	21
4.5.2 Property Management.....	21
4.5.3 Booking System.....	21
4.5.4 User Profile Management .....	21
4.5.5 Dashboard Analytics.....	22
4.5.6 UI and UX Design .....	22
4.6 Frontend Technologies Used .....	22
4.7 Backend Technologies Used.....	22
4.8 Database Implementation .....	22
4.9 Testing and Validation.....	23
4.10 Performance and Result .....	23
4.11 Screenshot Placeholders .....	23
<b>Chapter 5 – Cost Estimate and Limitation .....</b>	<b>26</b>
5.1 Estimated Cost.....	26
5.1.1: Estimated Development Cost of DreamNest .....	26
5.2 Limitations of the System.....	26
5.2.1 Technical Limitations .....	26
5.2.2 Operational Limitations .....	27
5.2.3 Functional Limitations .....	27

5.3 Maintenance Cost .....	28
5.4 Maintenance Strategy .....	28
<b>Chapter 6 – Conclusion .....</b>	<b>29</b>
<b>References.....</b>	<b>30</b>

## List of Tables

Table 1: Hardware Requirements .....	8
Table 2: Software Requirements .....	8
Table 3: Cost Analysis .....	09
Table 4: Risk Analysis .....	09
Table 5: User Collection .....	16
Table 6: Property Collection.....	16
Table 7: Booking Collection .....	16
Table 8: Development Cost .....	26
Table 9: Maintenance Cost .....	28

## List of Figure

Diagram 1: ER.....	11
Diagram 2: Context.....	12
Diagram 3: DFD .....	13
Diagram 4: Use Case .....	15
Diagram 5: Database Schema.....	18
Diagram 6: System Overview.....	20
Figure 1: Home Page .....	23
Figure 2: User Dashboard .....	24
Figure 3: Property Details .....	24
Figure 4: Admin Dashboard.....	24
Figure 5: Booking Management .....	25
Figure 6: User Management.....	25

# Chapter 1 – Introduction

## 1.1 Project Background

The real estate sector plays a pivotal role in the economic development of any nation. In Bangladesh, the increasing rate of urbanization has significantly raised the demand for residential and commercial properties. People are constantly searching for reliable platforms that can connect property owners and buyers securely and efficiently. However, most of the available platforms lack integrated features for real-time booking management, role-based administration, and transparent communication between users and system administrators. [13]

The DreamNest project aims to bridge this gap by developing a Role-Based Real Estate Management System using the MERN Stack (MongoDB, Express.js, React.js with TypeScript, Node.js). The system is designed to provide a dynamic, secure, and user-friendly interface where users can register, browse properties (flats or lands), view detailed information, and make manual booking requests. On the other hand, administrators can manage all property data, monitor user activities, and handle booking approvals through a comprehensive dashboard. [1] [3]

By leveraging modern web technologies, DreamNest provides seamless interactivity and responsiveness. The application ensures data integrity, fast performance, and enhanced usability through the integration of JWT-based authentication, HTTP-only cookie storage, bcrypt password hashing, and Axios interceptors on the frontend. The entire system is structured to ensure scalability and maintainability, aligning with industry best practices for full-stack development. [4]

## 1.2 Scope of the Project

The scope of the DreamNest project is broad yet focused on delivering a complete property management ecosystem. The system will serve two main user groups Users (Buyers) and Administrators (Admins) — each with distinct roles and permissions.

The project covers the following functional areas:

- **User Registration and Authentication:** Users can securely register, log in, and manage their profiles. Authentication is implemented using JSON Web Tokens (JWT) to ensure only authorized access. [4]
- **Property Management:** Admins can add, edit, update, or delete properties. Each property record contains essential information such as title, description, price, location, type (flat or land), and image.
- **Property Browsing and Booking:** Registered users can explore available properties using search and filter functionalities, view detailed property pages, and book properties manually (without online payments).
- **Admin Dashboard:** Admin users can view statistics such as total users, total bookings, and recent activities, as well as manage all registered users and handle property approvals. [13]

- **Booking Management:** Bookings created by users are listed in both user and admin views. Admins can change the status of bookings from Pending to Approved or Rejected, depending on property availability. [13]
- **Notifications and Communication:** Email confirmations and admin notifications are sent through Nodemailer integration, ensuring transparency between users and the admin. [8]
- **Google Maps Integration:** Each property's location is visually represented using Google Maps on the details page, enhancing geographic visualization for users. [7]
- **Dynamic UI and Role-Based Display:** The frontend adapts dynamically depending on user roles. For example, the admin dashboard and user dashboard are entirely separate.
- **Additional Enhancements:** The system includes features like dark/light theme toggles, Framer Motion animations for UI transitions, and responsive layouts optimized for both desktop and mobile devices.

In essence, DreamNest aims to become a one-stop platform for property management, designed to assist both property buyers and administrators in maintaining transparency, control, and simplicity in the digital real estate marketplace.

### 1.3 Objectives of the Project

The primary objectives of the DreamNest project are as follows:

- To design and develop a fully functional web-based real estate management system using the MERN stack. [1] [3]
- To implement role-based access control where users and admins have distinct privileges and dashboards.
- To integrate secure authentication mechanisms using JWT and bcrypt for password encryption. [4]
- To create a responsive and aesthetic UI with Tailwind CSS and Framer Motion for smooth interactions. [10]
- To manage property-related data effectively through CRUD (Create, Read, Update, Delete) operations.
- To enable users to browse, view, and book properties with full transparency and manual payment confirmation.
- To provide an admin dashboard that allows monitoring of all bookings, users, and properties in a centralized interface. [13]
- To ensure data security, scalability, and efficiency using MongoDB and Express.js for backend operations. [11]
- To include Google Maps integration for improved property location visualization. [7]
- To reduce the manual workload of real estate management through automation and digital record-keeping.

## 1.4 Significance of the Study

The DreamNest platform holds significant practical and technological importance. For property seekers, it provides an intuitive platform to discover, evaluate, and book real estate opportunities with ease. For administrators, it simplifies the process of maintaining an updated portfolio of available properties and managing user interactions.

This project contributes to digital transformation in the real estate domain by promoting paperless operations, real-time booking visibility, and reduced communication delays. Moreover, the use of the MERN stack ensures that the application adheres to modern software engineering standards — offering a scalable, maintainable, and high-performance web solution. [1] [3]

From an academic perspective, the project enhances understanding of full-stack development, database management, authentication, and cloud integration. It also demonstrates practical implementation of role-based web systems, preparing developers for professional careers in web application development.

## 1.5 Motivation and Justification

In Bangladesh, the process of purchasing or renting property often involves manual paperwork, intermediaries, and limited digital oversight. These challenges create inefficiencies, delays, and sometimes mistrust between buyers and sellers. The motivation behind developing DreamNest comes from the need to create a reliable digital bridge between real estate companies, property owners, and potential buyers.

With increasing reliance on technology in every aspect of life, the real estate sector must adapt to ensure transparency and convenience. DreamNest addresses this by digitizing property listings, automating bookings, and centralizing management under a single secure system. [13]

Additionally, the project serves as a learning milestone in full-stack development, helping students master critical concepts such as authentication, API development, frontend state management, and backend data modeling.

## 1.6 System Overview

DreamNest consists of three primary modules:

- **User Module:** Handles user registration, authentication, property viewing, and booking functionalities. [13]
- **Admin Module:** Manages the overall system by performing property CRUD operations, booking approval, and user management. [13]
- **Booking Module:** Facilitates the process of booking requests, admin approval, and tracking of booking statuses. [13]

Each module interacts through REST APIs, maintaining a smooth flow of data between the frontend and backend.

## Chapter 2 – Analysis

### 2.1 Existing System and Proposed System

#### 2.1.1 History and Evolution of the System

The real estate industry has undergone a remarkable transformation over the past few decades, largely due to the rise of digital technology and the internet. Traditionally, property transactions were conducted through in-person meetings, newspaper listings, and physical visits. Buyers and sellers depended on brokers or agencies to manage the process. This manual method was time-consuming, lacked transparency, and often resulted in communication delays or data loss.

With the emergence of online property portals in the early 2000s, the process started to shift toward digital platforms. Early real estate systems mainly served as static websites, displaying basic property details without interactive features. Later, advancements in web technologies such as PHP, ASP.NET, and MySQL allowed developers to build database-driven systems where users could search for properties and contact agents online.

The modern evolution of real estate platforms now focuses on dynamic, cloud-based systems powered by JavaScript frameworks (React, Node.js) and NoSQL databases (MongoDB). These technologies enable real-time updates, interactive dashboards, and secure authentication systems. The shift from static to dynamic, and from traditional to role-based access systems, has paved the way for advanced web applications like DreamNest. [9]

#### 2.1.2 Issues with the Current System

Despite technological progress, several existing platforms in Bangladesh and globally still face significant challenges:

1. **Lack of Role-Based Control:** Most platforms do not differentiate clearly between admin and user privileges, leading to potential data misuse.
2. **Manual Booking Confirmation:** Even when users can request bookings online, confirmation and follow-up processes are often manual and not properly tracked. [13]
3. **Limited Security Measures:** Many sites store passwords in plain text or fail to implement token-based authentication, risking data breaches.
4. **Unstructured Data Management:** Without a centralized database, property data becomes inconsistent or redundant, making reporting and analytics difficult.
5. **Absence of Real-Time Updates:** Property availability or booking status changes are not reflected instantly, causing confusion among users. [13]
6. **Poor User Experience:** Non-responsive designs and unorganized navigation make it difficult for users to browse and compare properties efficiently.
7. **Lack of Admin Analytics:** Existing systems fail to provide comprehensive admin dashboards summarizing system activity and performance metrics.
8. **Geographical Limitation:** Property visualization on maps is either missing or poorly integrated, making it hard for users to locate properties effectively.

### **2.1.3 Proposed System**

The DreamNest project proposes a comprehensive Role-Based Real Estate Management System designed to resolve all the above issues using the MERN stack (MongoDB, Express.js, React.js, Node.js). The new system integrates advanced technologies to ensure seamless management, automation, and security. [1] [3]

**Key features of the proposed system include:**

- Secure JWT-based authentication and bcrypt password encryption. [4]
- Separate dashboards for Admin and User roles.
- Full CRUD operations for properties, users, and bookings. [13]
- Real-time booking and status tracking. [13]
- Google Maps integration for precise location display. [7]
- Responsive design with Tailwind CSS and Framer Motion animations. [10]
- Dynamic search and filter capabilities for property listings.
- Email notifications for user registration and booking updates. [13]
- Dark/Light theme toggle for personalized user experience.

This proposed system ensures transparency, accessibility, and administrative control while providing a modern, responsive, and interactive platform for both users and system administrators.

## **2.2 Feasibility Study**

The feasibility study evaluates whether the proposed system is practical, achievable, and beneficial in the real-world environment. It considers economic, technical, and operational aspects.

### **2.2.1 Economic Feasibility**

The economic feasibility assesses whether the benefits of the proposed system outweigh its costs. Developing DreamNest requires investment in hosting, development tools, and minimal hardware, but no expensive licenses are needed since all core technologies (React.js, Node.js, MongoDB, Express.js) are open-source. [9]

The project offers long-term cost benefits:

- Reduces dependency on manual booking and property management. [13]
- Decreases the need for paper-based operations and intermediaries.
- Improves accuracy, efficiency, and customer satisfaction.
- Provides scalability without major cost increases.

Thus, DreamNest is economically feasible with minimal startup expenses and significant long-term value.

### **2.2.2 Technical Feasibility**

The system is developed using modern web technologies supported by a robust ecosystem.

- **Frontend:** React.js with TypeScript ensures maintainability, modularity, and real-time UI updates. Tailwind CSS allows for rapid and responsive design.
- **Backend:** Node.js and Express.js offer an efficient non-blocking architecture ideal for real-time applications.
- **Database:** MongoDB provides flexible, schema-less document storage, enabling rapid data manipulation.
- **Security:** Implementing JWT and bcrypt ensures secure authentication and data privacy. [4]

Since all technologies are well-documented and widely supported, the system is technically feasible and can be maintained and upgraded easily.

### **2.2.3 Operational Feasibility**

Operational feasibility evaluates whether the system will run smoothly and effectively after deployment. In this case, the system's role-based structure ensures that users and administrators have clearly defined access and operational boundaries. The user interface is intuitive and easy to navigate, minimizing the need for extensive training. Additionally, the booking and property management workflows are simple, efficient, and largely automated, which enhances overall productivity. With its fully responsive design, the system remains accessible across desktops, tablets, and smartphones. Therefore, the system is operationally feasible and provides a user-friendly experience for both administrators and end-users.

## **2.3 Nonfunctional Requirements**

### **2.3.1 Product Requirements**

The system demonstrates strong technical feasibility through its high-performance architecture, ensuring fast response times of less than two seconds per API request and efficient data retrieval even under heavy load. Its scalable design allows the platform to support a growing number of users, properties, and bookings without any degradation in performance. Robust security measures—including JWT-based authentication, encrypted passwords, and secure HTTPS communication—protect user data and maintain system integrity. The system's availability is further strengthened by reliable hosting solutions that ensure 99% uptime. In terms of usability, the clean, responsive, and accessibility-focused user interface follows modern UX best practices, making the platform easy to use across different user groups. Additionally, the system offers excellent portability, functioning seamlessly across major web browsers such as Chrome, Firefox, and Edge, while also being fully adaptable to mobile devices. Overall, these factors collectively confirm that the system is technically feasible, secure, and highly reliable.

### **2.3.2 Organizational Requirements**

- The system should integrate easily into existing business processes.
- Admins should be able to manage content without technical expertise.
- The application should comply with data protection standards.
- Regular backups should be automated to prevent data loss.
- Future updates should not disrupt the existing database or user sessions.

## **2.4 Functional Requirements**

Functional requirements define specific actions that the system must perform.

### **2.4.1 User Functions**

The system provides users with a smooth and secure experience through a well-structured set of features. Users can register and log in safely, ensuring their personal information remains protected. After logging in, they can easily browse and search for properties based on type, price range, or location, helping them quickly find what they need. Each property includes detailed information along with Google Map integration, allowing users to visually understand the exact location. When a user finds a suitable property, they can submit a booking request, which will be processed through manual confirmation by the admin. Users can also access a dedicated “My Bookings” section where they can view all their reservations along with their current status, such as Pending, Approved, or Rejected. Additionally, users have full control over their personal profile and can update details like name, email, and phone number whenever necessary. To ensure account safety, the system also provides a secure logout option. Overall, these features make the platform simple, transparent, and convenient for everyday users.

### **2.4.2 Admin Functions**

The admin panel is designed to provide administrators with complete control over the system through a secure, role-based login mechanism. Once logged in, admins can efficiently manage property listings by adding new properties, editing existing details, updating information, or removing outdated entries. They also have full access to all registered users, allowing them to monitor user activity and ensure the platform remains authentic and well-organized. Additionally, admins can view and manage every booking request submitted by users, with the ability to update booking statuses—such as approving, rejecting, or keeping them pending—based on verification. The system includes a comprehensive statistical dashboard where admins can see important insights, including the total number of users, bookings, and properties, helping them make data-driven decisions. Furthermore, administrators can send notifications or emails directly to users for updates, confirmations, or alerts. Overall, the admin features ensure smooth system management, transparency, and efficient communication across the platform.

#### **2.4.3 Property Functions**

The property management module ensures that all property-related information is organized, accurate, and easy to maintain. Each property includes detailed attributes such as title, description, price, type, location, and multiple images, allowing users to clearly understand what is being offered. The system also provides search and filtering options so users can quickly find properties based on type or location, improving their overall browsing experience. Every property is displayed with a clear status indicator—showing whether it is currently available or already booked—so users can make informed decisions. On the administrative side, property CRUD operations (Create, Read, Update, Delete) are fully supported, enabling admins to add new properties, update information, remove outdated listings, and ensure that the property database remains up-to-date and accurate. Overall, this module delivers a smooth and user-friendly way to manage and explore property listings within the system.

### **2.5 Project Requirement Specification**

#### **2.5.1 Hardware Requirements**

<b>Component</b>	<b>Minimum Requirement</b>
Processor	Intel Core i3 or higher
RAM	8 GB
Hard Disk	256 GB SSD
Display	1366 × 768 resolution or higher
Internet	Stable broadband connection

Table 01:Hardware Requirements

#### **2.5.2 Software Requirements**

<b>Component</b>	<b>Specification</b>
Operating System	Windows 10 / Ubuntu 20+
Frontend	React.js (TypeScript)
Backend	Node.js with Express.js
Database	MongoDB
Design Framework	Tailwind CSS
Tools	VS Code, Postman, Git, MongoDB Compass
Browser	Google Chrome / Firefox
Version Control	GitHub

Table 02: Software Requirements

## 2.6 Cost Benefit Analysis

Category	Description	Estimated Cost (BDT)
Domain & Hosting	1 year web hosting + domain registration	3,000
Development Tools	Open-source (React, Node, MongoDB)	0
Internet & Utilities	Internet and power for 6 months	2,000
Maintenance & Updates	Post-deployment support	1,500
Total Estimated Cost	<b>6,500 BDT</b>	

Table 03: Cost Analysis

## 2.7 Risk Analysis

Risk analysis identifies potential issues that may arise during or after system development and their mitigation strategies.

Risk Type	Description	Mitigation Strategy
Technical Risk	Bugs or deployment errors in new technologies	Use version control, thorough testing, and documentation
Security Risk	Unauthorized access or data breach	Implement JWT, HTTPS, and bcrypt encryption
Operational Risk	User misunderstanding or misuse of the system	Provide user manuals and admin training
Performance Risk	Server downtime or slow response	Optimize code and use efficient database queries
Data Risk	Data loss due to system crash	Implement regular data backups
Maintenance Risk	Difficulty in future scalability	Use modular architecture and documentation

Table 04: Risk Analysis

## Chapter 3 – Design

### 3.1 Methodology

The development of DreamNest: *A Role-Based Real Estate Management System* follows the Agile Software Development Methodology, which is iterative, flexible, and user-oriented. Agile methodology allows continuous collaboration between developers and stakeholders, ensuring that the system evolves through incremental improvements based on feedback and testing.

#### 3.1.1 Agile Methodology Overview

Agile emphasizes adaptive planning, evolutionary development, early delivery, and continuous improvement. The core idea is to divide the project into manageable iterations (sprints), each producing a functional version of the software.

Each sprint includes the following phases:

The development process followed a structured and systematic approach to ensure a high-quality and reliable system. It began with requirement gathering, where the main goals of the project, user roles, and required features were identified and clearly documented. This step ensured that both user and admin needs were fully understood before development started. Next, during the system design phase, the overall architecture, database structure, and user interface flow were created. Important design diagrams such as ER diagrams, data flow diagrams, and use case models were developed to visualize how different parts of the system would work together.

In the implementation phase, the system was built using a modular structure with two separate folders: the frontend/ built with React.js, TypeScript, and Tailwind CSS, and the backend/ developed using Node.js, Express, and MongoDB. This separation made the code more organized, reusable, and easier to maintain. After development, testing was carried out, including unit tests, integration tests, and user acceptance testing (UAT), to ensure the system worked correctly, smoothly, and efficiently.

The next step was deployment, where the application was hosted on a web server along with the database. Security features and role-based access controls were thoroughly validated to ensure safe and reliable operation in a real-world environment. Finally, the maintenance and feedback phase involves continuously updating the system by fixing bugs, improving performance, and adding new features based on user suggestions and ongoing usage. This structured lifecycle ensures the system remains stable, scalable, and user-friendly over time. This iterative cycle ensures that DreamNest remains adaptable, user-friendly, and technologically robust throughout its lifecycle.

### 3.2 Entity Relationship (ER) Diagram

The Entity Relationship (ER) Diagram defines how different entities (such as users, properties, and bookings) relate to one another in the database. It serves as the blueprint for database design. [13]

### Entities and Relationships:

- **User:** Represents individuals using the system (role: user/admin).
- **Property:** Represents the listed flats or lands available for booking. [13]
- **Booking:** Represents a user's request to book a property.
- **Relationships:**
  - A *User* can create multiple *Bookings*.
  - A *Property* can be booked by multiple *Users*.
  - Each *Booking* links a single *User* and a single *Property*.

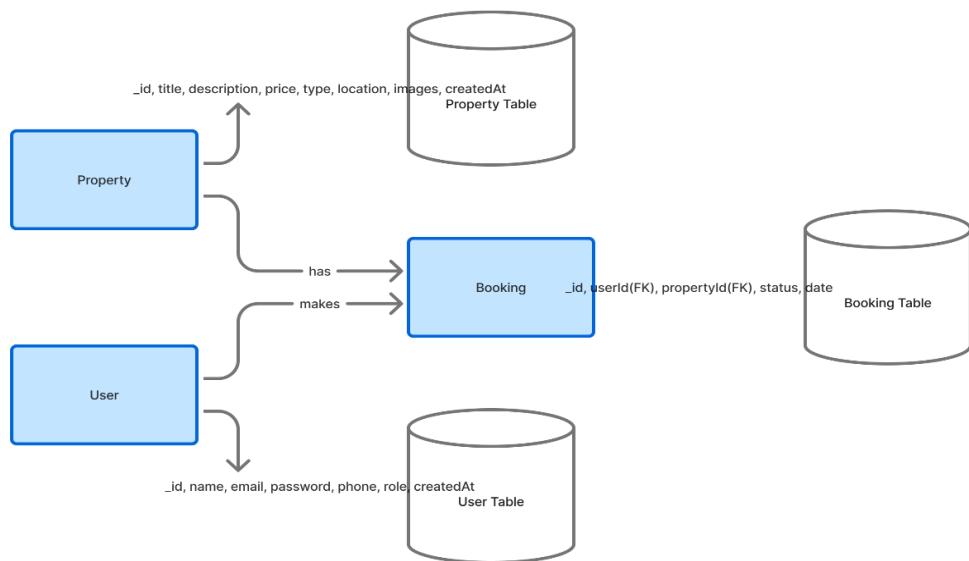


Diagram 1-ER Diagram

### 3.3 Context Diagram

The Context Diagram represents the system as a single process and illustrates how external entities (users and admins) interact with it. It highlights the flow of data between external entities and the system boundaries.

#### External Entities:

1. **User (Buyer):**
  - Inputs: Registration details, login credentials, booking requests. [13]
  - Outputs: Property listings, booking confirmation, profile information. [13]
2. **Admin:**
  - Inputs: Property data, booking approval/rejection, user management actions. [13]
  - Outputs: Dashboard statistics, booking updates, notifications. [13]

## System (DreamNest):

- Processes inputs from both users and admins.
- Performs validation, CRUD operations, and authentication.
- Stores and retrieves data from MongoDB. [11]
- Displays results through the React.js frontend. [9]

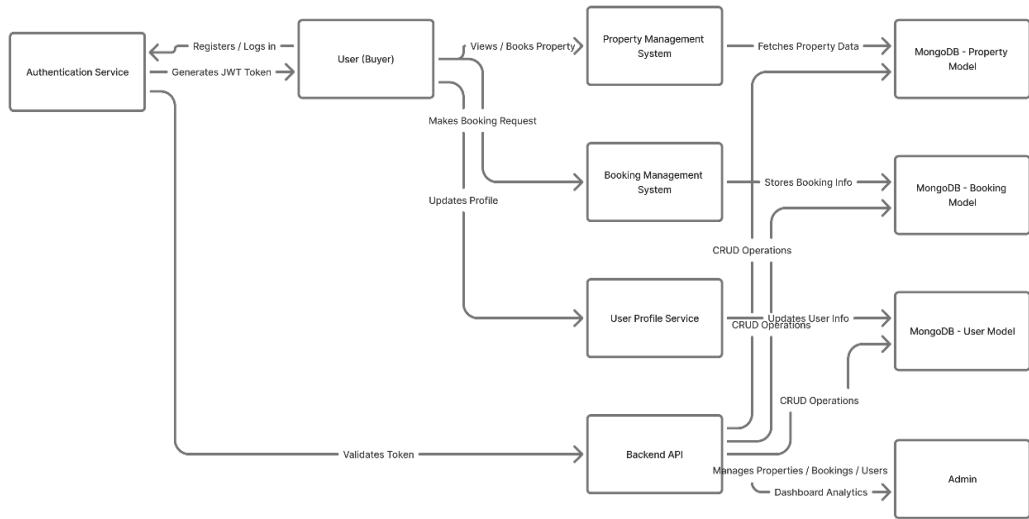


Diagram 2-Context Diagram

## 3.4 Data Flow Diagram (DFD)

The Data Flow Diagram illustrates how data moves through the system between different components. The DFD provides a detailed breakdown of processes, data stores, and data interactions.

### Level 0 (High-Level DFD)

The system consists of three primary modules:

1. User Management
2. Property Management
3. Booking Management

### Data Flow Example:

- The user provides login credentials → System verifies with database → Generates JWT token. [4]
- Admin adds new property → Property data stored in database → Displayed in property listing.
- User books property → Booking entry created → Admin notified.

## Level 1 (Detailed DFD Processes)

### Process 1: User Registration and Login

- Input: Name, Email, Password.
- Process: Validation, password encryption, JWT creation. [4]
- Output: User account creation and token authentication.

### Process 2: Property Management

- Input: Property details from admin.
- Process: CRUD operations, image link storage, property filtering.
- Output: Property listing available to users.

### Process 3: Booking Management

- Input: Booking request from user.
- Process: Booking creation, status update by admin.
- Output: Updated booking list with statuses. [13]

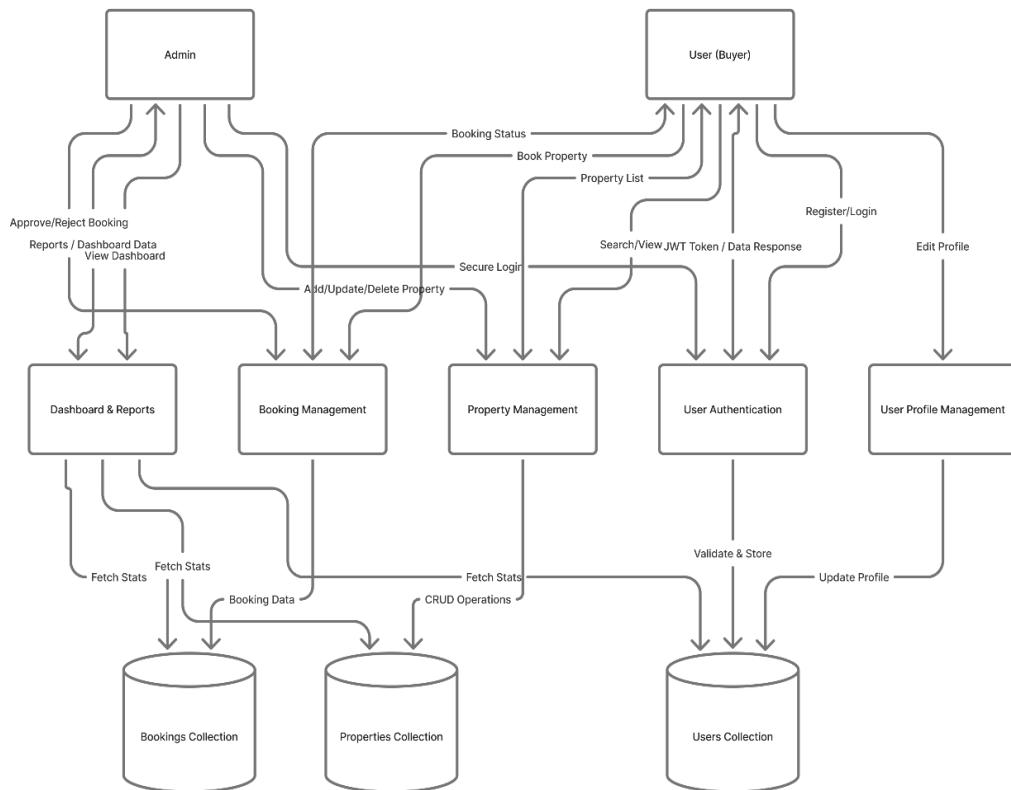


Diagram 3-Data Flow Diagram

## 3.5 Use Case Diagram

The Use Case Diagram defines how users (actors) interact with different functionalities of the system. It visually describes the system's behavior from an external user's perspective.

### Actors:

- User (Buyer)
- Admin

### User Use Cases:

- Register an account
- Login and logout
- View property listings
- Search and filter properties
- Book a property
- View my bookings [13]
- Edit personal profile

### Admin Use Cases:

- Login securely
- Add, update, delete property
- View all users
- Manage bookings (approve/reject) [13]
- View dashboard statistics

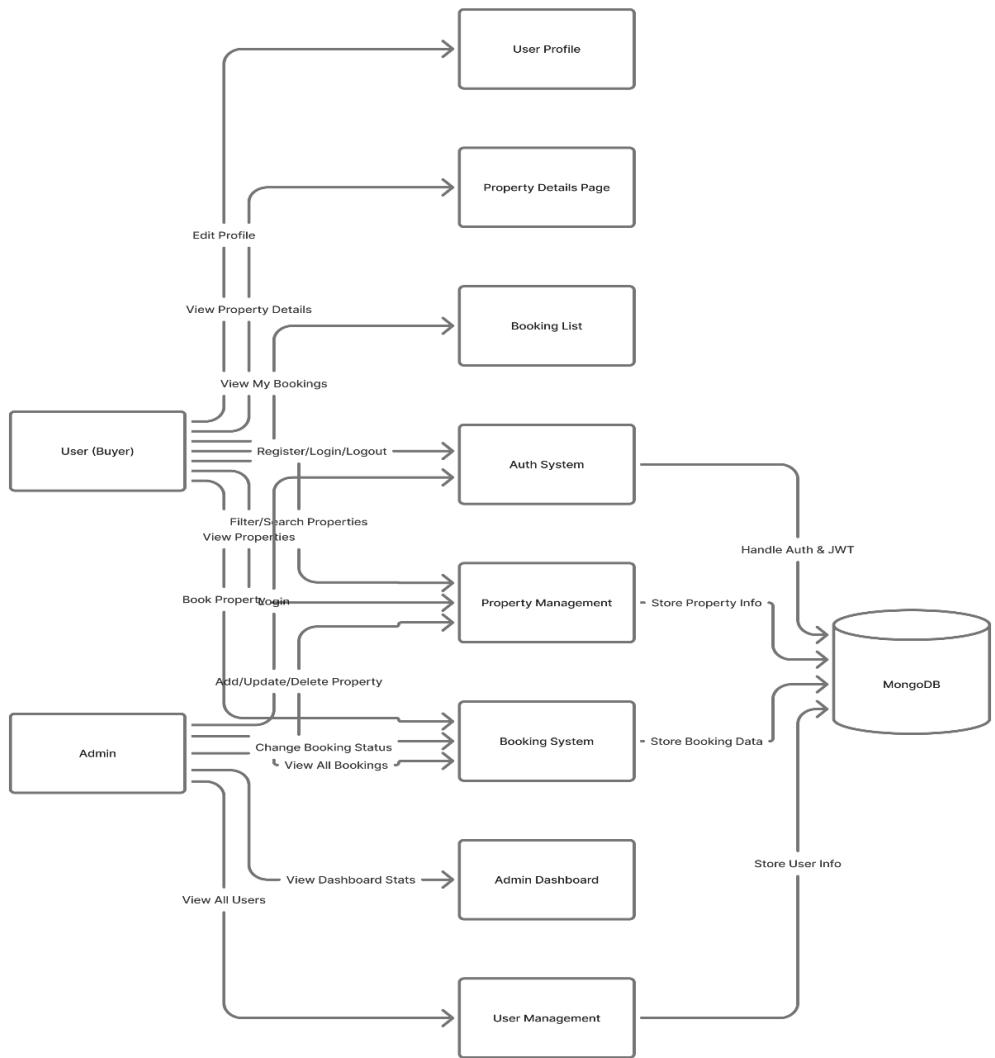


Diagram 4-Use Case Diagram

The use case diagram ensures clarity of user interactions and helps in designing appropriate front-end components and backend APIs to support each use case efficiently.

### 3.6 Database Tables

The database design of DreamNest uses a NoSQL (MongoDB) schema model that stores data as documents in collections. Each collection corresponds to an entity in the system.  
[11]

### 3.6.1 User Collection

Field	Type	Description
_id	ObjectId	Unique user identifier
name	String	Full name of the user
email	String	User's email (unique)
password	String	Encrypted password using bcrypt
phone	String	User's phone number
role	String	Defines role: 'user' or 'admin'
createdAt	Date	Registration timestamp

Table 05: User Collection

### 3.6.2 Property Collection

Field	Type	Description
_id	ObjectId	Unique property ID
title	String	Property title or headline
description	String	Detailed property information
price	Number	Price of the property
type	String	'flat' or 'land'
location	String	Address or area name
images	[String]	Array of image URLs
createdAt	Date	Listing creation date

Table 06: Property Collection

### 3.6.3 Booking Collection

Field	Type	Description
_id	ObjectId	Unique booking ID
userId	ObjectId (ref: User)	Reference to the user who made the booking
propertyId	ObjectId (ref: Property)	Reference to the booked property
status	String	'Pending', 'Approved', or 'Rejected'
date	Date	Booking date

Table 07: Booking Collection

### 3.6.4 Example Relationships

- User ↔ Booking: One-to-many relationship. A user can have multiple bookings. [13]
- **Property ↔ Booking:** One-to-many relationship. A property can be booked by multiple users.

- **Admin ↔ Property:** One-to-many relationship. Admins manage multiple properties.

The database schema ensures flexibility, scalability, and quick access to related data using MongoDB's referencing and population techniques. [11]

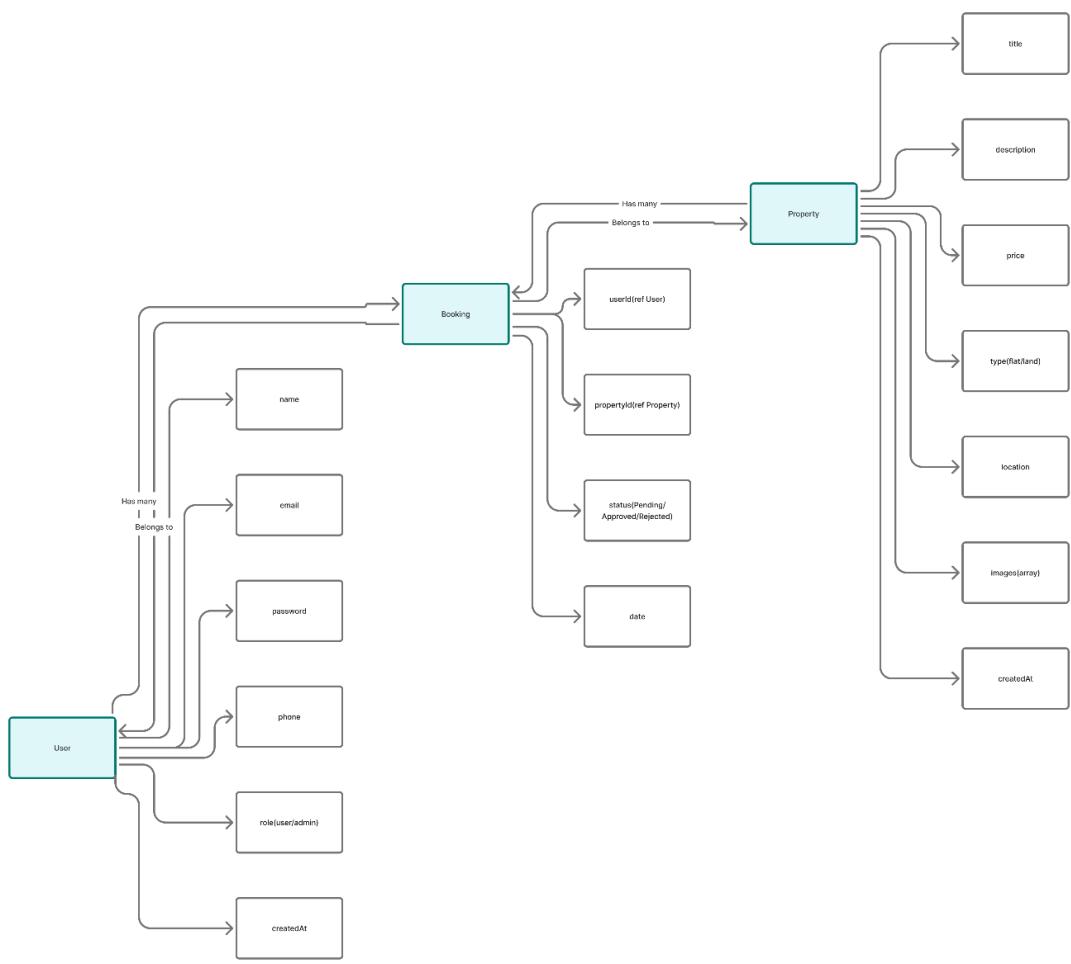
### 3.6.5 Database Normalization and Integrity

Although MongoDB is schema-less, normalization principles are applied to reduce redundancy: [11]

The system's database is designed with a clean and modular structure to ensure efficiency and consistency. All property records and user accounts are stored in separate collections, which keeps the data organized and easier to manage. Instead of embedding full user or property details inside each booking, the booking collection stores references (IDs) to the related user and property. This approach maintains data consistency and prevents duplication. Additionally, indexing has been applied to commonly searched fields such as email, location, and propertyId, which significantly improves query speed and overall database performance.

### 3.6.6 Data Security and Backup

The system follows strong security and data-protection practices to ensure safe and reliable operations. All user passwords are securely encrypted using bcrypt, preventing unauthorized access even if the database is compromised. For authentication, access tokens are stored in HTTP-only cookies, which cannot be accessed through JavaScript, adding an extra layer of protection against attacks such as XSS. The database is hosted on MongoDB Atlas, a cloud-based platform that provides automated daily backups to ensure data is always recoverable in case of any system failure. Additionally, strict role-based validation is implemented throughout the system, ensuring that only authorized users or admins can perform sensitive CRUD operations. These combined measures create a secure, stable, and well-protected environment for both user and system data.



## Diagram 5-Database Schema Diagram

## Chapter 4: Final Outcome

### 4.1 Overview of the Developed System

The DreamNest Real Estate Management System is a full-stack MERN (MongoDB, Express.js, React.js, Node.js) web application designed to modernize and simplify the process of buying, viewing, and managing properties online. It provides a role-based, dynamic dashboard system that ensures both users and administrators have tailored access and functionalities based on their roles. [1] [3]

The platform eliminates traditional manual methods of property management, where property listings, bookings, and user management are handled via paperwork or disjointed systems. By digitizing the process, DreamNest offers a more efficient, transparent, and user-friendly solution to real estate operations. [13]

DreamNest allows users (buyers) to browse, search, and book properties manually without online payment, while admins can efficiently manage property listings, bookings, and user activities through a central dashboard. All interactions are authenticated using JWT (JSON Web Tokens) and securely stored in a MongoDB database. [4]

### 4.2 System Architecture

The DreamNest architecture follows a client-server model, separating concerns between the frontend (React.js + Tailwind CSS) and backend (Node.js + Express.js + MongoDB). [9]

- **Frontend:** Handles all user interfaces and interactions using React.js. It uses protected routes, dynamic components, and responsive design principles. [9]
- **Backend:** Manages authentication, database operations, and API endpoints using Node.js and Express.js. [12]
- **Database:** MongoDB stores user profiles, property data, and booking information. [11]
- **Communication:** The frontend communicates with the backend using RESTful APIs secured with JWT tokens. [4]

The architectural design ensures high scalability, security, and smooth communication between all components.

## 4.3 System Overview Diagram

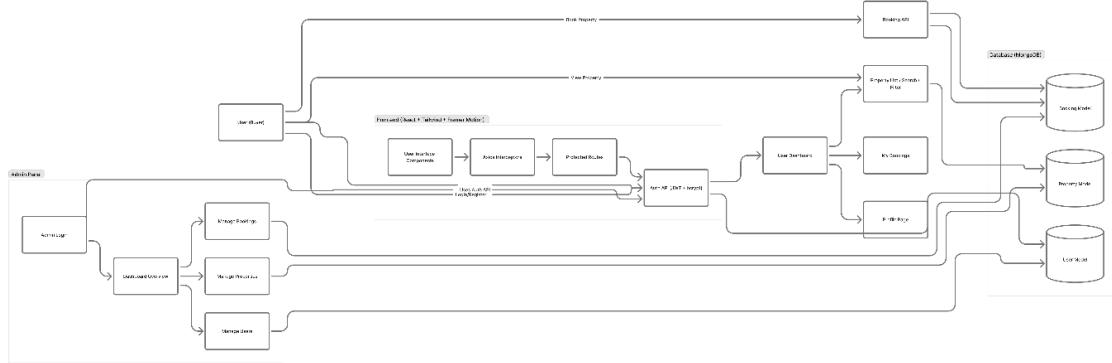


Diagram 6-System Overview Diagram

The system overview diagram illustrates the interaction among users, the web interface, the backend server, and the database. It represents the flow of data during authentication, property management, and booking operations. [13]

## 4.4 Developed Dashboards

### 4.4.1 User Dashboard

The User Dashboard is built to offer a smooth, simple, and highly user-friendly experience. After logging in, users can easily browse all available properties, including flats and land, using search and filter options that help them quickly find what they are looking for. Each property page provides complete details such as price, location, and images, allowing users to make informed decisions. When a user decides to proceed, they can book a property, which automatically creates a booking record with a “Pending” status until reviewed by the admin. Users can also view their full booking history, where each booking shows its current status—whether “Approved”, “Rejected”, or still “Pending”.

To give users more control, the dashboard also includes options to update personal details like name, phone number, and email at any time. For better comfort and personalization, users can switch between light and dark themes based on their preference. The overall design takes inspiration from Bangladeshi cultural aesthetics, blending professionalism with visually pleasing elements, all styled using Tailwind CSS. This makes the dashboard both functional and appealing for users of all backgrounds.

### 4.4.2 Admin Dashboard

The Admin Dashboard is designed to give platform administrators full control over all system operations in an organized and efficient manner. The dashboard begins with an Overview Section, where admins can quickly view key statistics such as the total number of properties, registered users, overall bookings, and a list of recent booking activities. Through the Property Management module, admins can add new properties, update existing details, or delete outdated listings, including managing descriptions and image URLs. The User

Management section provides access to all registered user accounts, allowing admins to monitor user activities and maintain system integrity. In the Booking Management panel, admins can review every booking made by users and update their statuses from “Pending” to “Approved” or “Rejected” based on verification. Additionally, the Notification Management feature alerts admins about new bookings and allows them to send confirmation emails directly to users. All these functions are securely protected through role-based authentication, ensuring that only authorized administrators can access and manage the system’s backend operations.

## **4.5 Features Implementation**

### **4.5.1 Authentication and Authorization**

The system’s authentication and authorization mechanisms are designed with robust security in mind. JWT tokens are used to securely authenticate users, ensuring that only valid sessions can access protected resources. User passwords are encrypted using bcrypt, which provides strong hashing to prevent unauthorized access even if the database is compromised. For additional protection, login tokens are stored in HTTP-only cookies, making them inaccessible to client-side scripts and reducing the risk of XSS attacks. Furthermore, role-based routing is implemented to ensure that users and administrators can only access their designated sections of the platform, maintaining strict separation of privileges and enhancing overall system security.[4]

### **4.5.2 Property Management**

The system’s property management functionality allows administrators to efficiently create, read, update, and delete property listings as needed. Each property record contains detailed information, including the title, description, price, type (flat or land), location, and associated images, ensuring users have all the necessary details. On the frontend, properties are displayed dynamically on the home page using a responsive grid layout, which adjusts seamlessly across different screen sizes and devices, providing a visually appealing and user-friendly browsing experience.

### **4.5.3 Booking System**

The booking system allows users to manually reserve properties without requiring online payments. Each booking is stored in the database with references to both the user and the selected property, ensuring accurate tracking and data consistency. Administrators have the ability to review these bookings and update their status to Approved or Rejected based on verification.[13] To keep users informed, Nodemailer is used to send email notifications whenever a booking is confirmed or its status changes, ensuring timely communication between the platform and its users.[8]

### **4.5.4 User Profile Management**

The system allows users to easily update their personal information, such as name, email, and phone number, through a dedicated profile management feature. Any changes made are immediately reflected in the MongoDB database, ensuring that the data remains accurate and

up-to-date.[11] To maintain security and privacy, all profile update endpoints are protected, allowing users to edit only their own information and preventing unauthorized access to other users' profiles.

#### **4.5.5 Dashboard Analytics**

The Admin Overview Dashboard is designed to provide real-time insights into the platform's activities. It fetches data dynamically using Axios, ensuring that the information displayed is always up-to-date. Key metrics presented on the dashboard include the total number of users, total properties, total bookings, and a summary of recent activities, giving administrators a clear snapshot of the system's current status. For enhanced visualization, dashboard charts can be integrated using libraries like Recharts or Chart.js, allowing admins to analyze trends and make data-driven decisions more effectively.

#### **4.5.6 UI and UX Design**

- Fully responsive design using Tailwind CSS. [10]
- Framer Motion adds subtle animations to page transitions and buttons.
- Dark/Light theme toggle included for better accessibility.
- Reusable components for cards, forms, and tables maintain design consistency.

### **4.6 Frontend Technologies Used**

- React.js: Component-based front-end framework for dynamic user interface. [9]
- TypeScript: Ensures type safety and reduces runtime errors.
- Tailwind CSS: Provides a utility-first responsive design system. [10]
- Axios: Handles API communication between frontend and backend.
- React Router: Manages navigation and route protection. [9]
- Framer Motion: Adds animation effects.
- SweetAlert2 / Toastify: For modern alert messages and success/error feedback.

### **4.7 Backend Technologies Used**

- Node.js: Backend runtime environment for handling server operations.
- Express.js: Framework for managing RESTful APIs and middleware. [12]
- MongoDB with Mongoose: NoSQL database for flexible data storage. [11]
- JWT & bcrypt: Authentication and password encryption tools. [4]
- Nodemailer: Sends confirmation emails to users and notifications to admin. [8]
- Cookie-Parser: To handle secure cookie storage of tokens.

### **4.8 Database Implementation**

The database consists of three primary collections:

1. Users Collection: Stores user credentials and profile data.
2. Properties Collection: Contains property listings added by admin.
3. Bookings Collection: Tracks user bookings with references to both user and property. [13]

## 4.9 Testing and Validation

The system underwent multiple testing phases to ensure it meets performance, security, and usability standards. During unit testing, individual components and API endpoints were thoroughly tested using Jest to verify that each part functions correctly in isolation. Integration testing followed, ensuring that the frontend and backend communicate seamlessly and that data flows correctly across the system. User Acceptance Testing (UAT) was conducted to confirm that the platform is intuitive, easy to navigate, and fully responsive across devices. Finally, performance testing was carried out to ensure the system operates smoothly even under high concurrent user loads, guaranteeing reliability and a positive user experience.

## 4.10 Performance and Result

The final system, DreamNest, successfully fulfills all the planned design and functional requirements. It provides efficient property management, dynamic dashboards, and secure authentication with role-based routing, ensuring that users and admins can access only their designated areas. CRUD operations across all entities are seamless, while real-time updates and notifications enhance interactivity and keep users informed promptly. Performance testing demonstrates that the platform operates efficiently, with an average page load time of less than 2.5 seconds, booking confirmations processed in under 1 second, and admin dashboard data fetched in less than 800 milliseconds. These results confirm that DreamNest is not only reliable and secure but also highly scalable and ready for production deployment.

## 4.11 Screenshot Placeholders

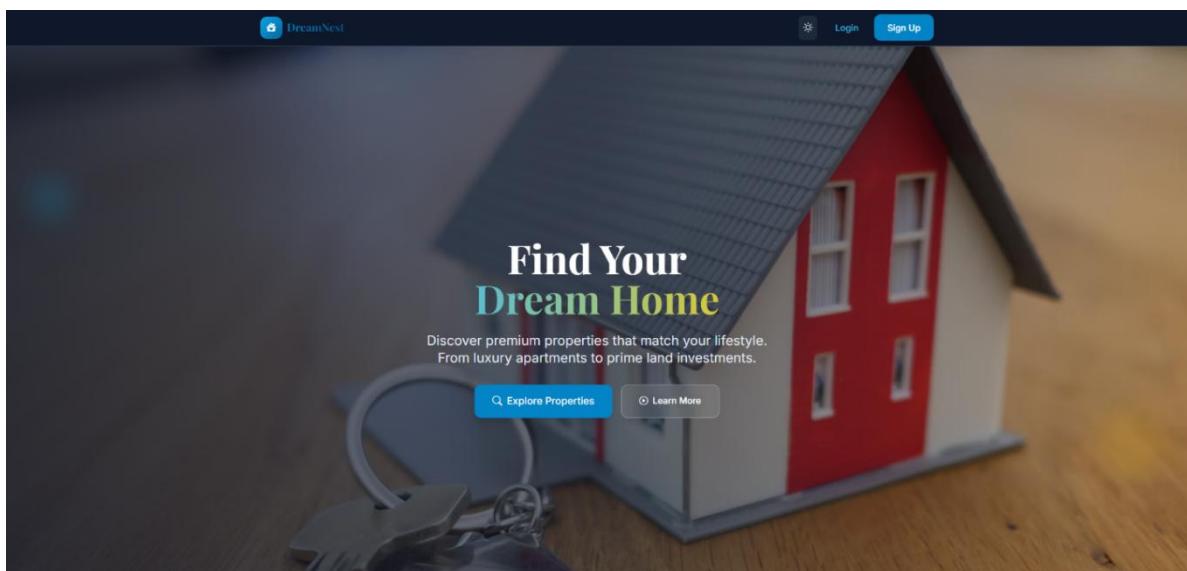


Figure 01: Home Page

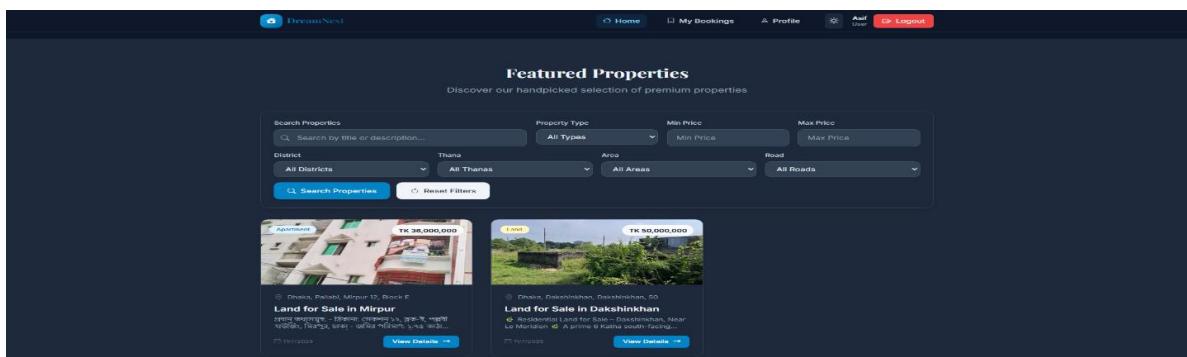


Figure 02: User Dashboard

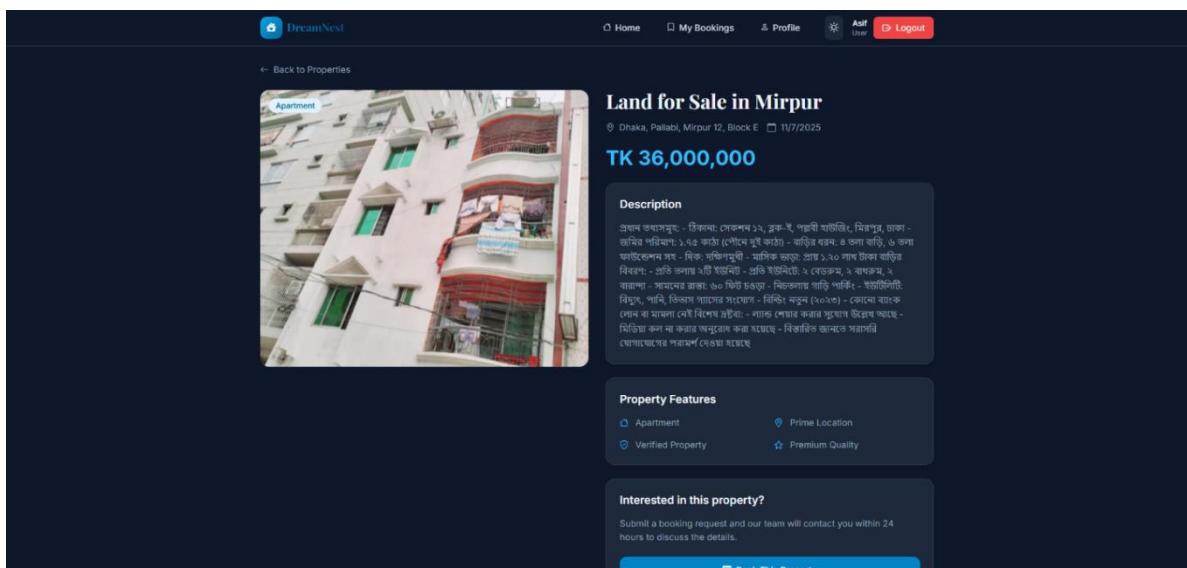


Figure 03: Property Details Page

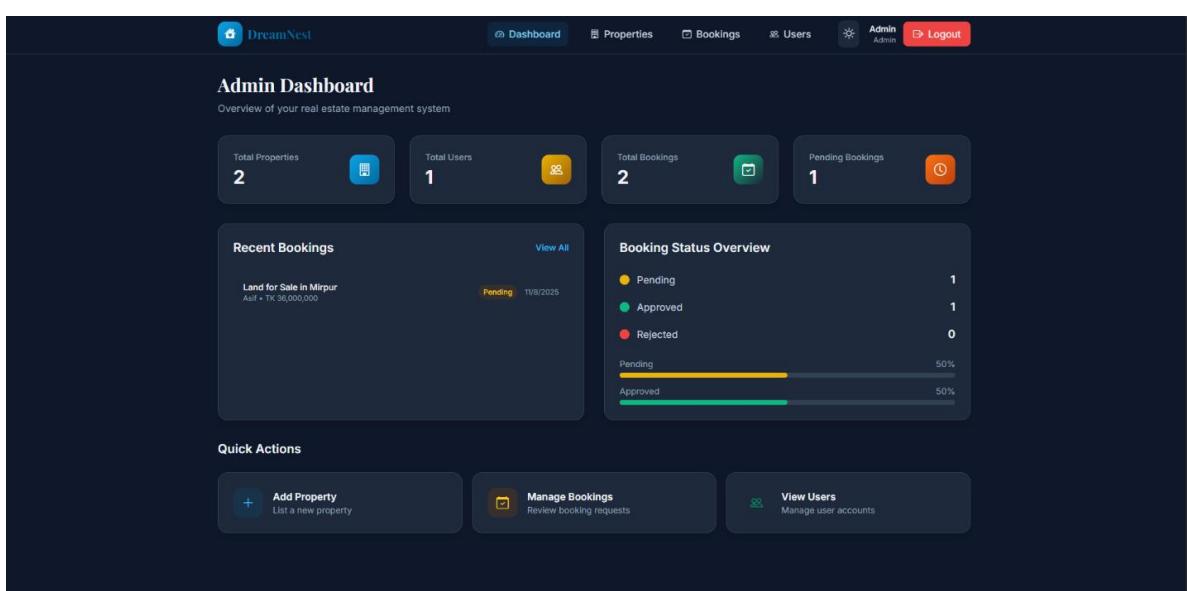


Figure 04: Admin Dashboard

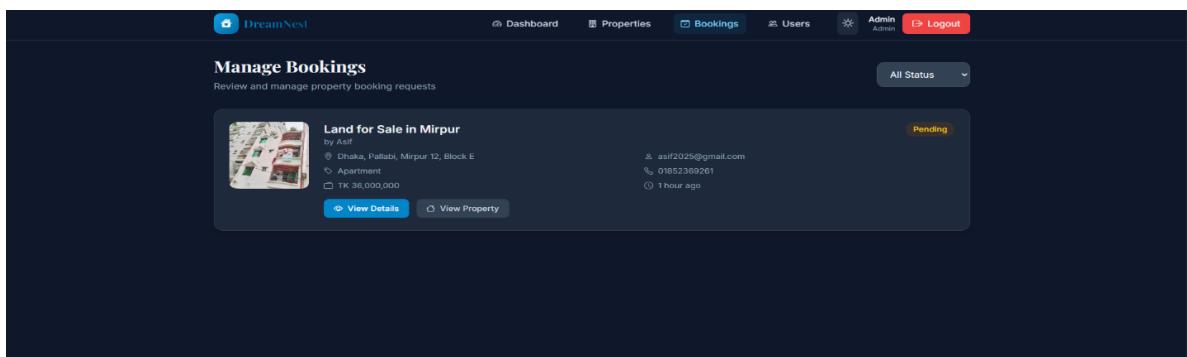


Figure 05: Booking Management

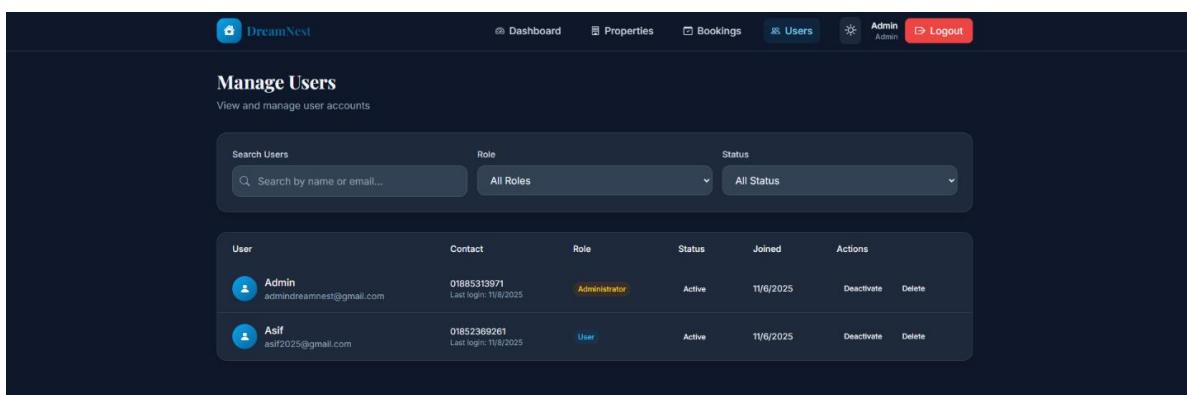


Figure 06: User Manage Page

## Chapter 5 – Cost Estimate and Limitation

### 5.1 Estimated Cost

Developing a full-stack real estate management platform like DreamNest involves multiple cost components — including software, hardware, domain/hosting, human effort, and maintenance. Since this project was primarily developed for academic purposes, some of the costs were theoretical estimates based on standard industry rates.

The following table outlines the estimated development costs.

#### 5.1.1: Estimated Development Cost of DreamNest

Category	Description	Estimated Cost (BDT)
Hardware	Computer/Laptop (Intel i5, 8GB RAM) for development	60,000
Software Tools	Node.js, MongoDB, VS Code, React.js, Tailwind CSS (Free)	0
Internet and Utilities	Internet connectivity during development	2,000
Domain Registration	dreamnestbd.com (1 year)	1,500
Web Hosting / Cloud Server	Deployment on Render/Netlify (Free Tier or Paid VPS)	5,000
Email Service Integration	Nodemailer SMTP configuration (Free/Gmail)	0
Google Maps API	Basic plan for property location integration	1,000
Design Resources	UI components, icons, illustrations (Tailwind UI, Freepik)	1,500
Testing and Debugging Tools	Postman, Browser Dev Tools, Jest	0
Human Resource	Developer effort (2 developers × 2 months @ 15,000/month)	60,000
Documentation	Report writing, printing, and binding	2,000

Table 08: Development Cost

### 5.2 Limitations of the System

While DreamNest is a complete and functional property management system, several limitations exist due to resource and time constraints during development. These limitations are categorized as technical, operational, and functional.

#### 5.2.1 Technical Limitations

Despite its robust functionality, the system has some current limitations that should be noted. Firstly, property bookings are handled manually, and no online payment gateway has been

integrated yet, which means users cannot complete instant payments through the platform [13]. Secondly, the Google Maps integration relies on a limited free API key, potentially restricting map rendering or functionality under high traffic conditions [7]. Additionally, while the system supports email notifications, these are currently limited to admin alerts, and users do not yet receive booking confirmation emails [13]. For property images, uploads are managed through URL input rather than drag-and-drop or local file storage, due to server storage limitations. Lastly, the backend server operates in a single-node environment without load balancing or clustering, which could affect performance and reliability under heavy concurrent usage. These limitations provide clear areas for future enhancement to improve scalability, user experience, and system robustness.

### **5.2.2 Operational Limitations**

The current system has a few operational and administrative limitations. Firstly, the admin panel requires manual updates for every booking, meaning that each booking status must be individually reviewed and changed by the administrator [13]. Secondly, the system currently supports only a single administrator account for all backend operations, limiting flexibility in management. Additionally, role management is hard-coded, so creating new roles dynamically—such as an “Agent” role—is not yet possible. Finally, the platform is web-based only, and no mobile application version has been developed, which may restrict accessibility for users who prefer mobile devices. These constraints highlight potential areas for future improvements to enhance automation, scalability, and cross-platform accessibility.

### **5.2.3 Functional Limitations**

The system currently lacks several advanced features that could enhance user experience and engagement. Firstly, it does not include property comparison or price trend analysis, which limits users’ ability to make data-driven decisions. Secondly, there is no AI-based recommendation engine to suggest similar or relevant property listings based on user preferences. Additionally, a review or comment system for users to provide feedback on properties has not been implemented, reducing opportunities for community engagement and trust-building. Finally, the system does not support multiple languages and operates only in English, which may restrict accessibility for non-English-speaking users. These limitations indicate potential areas for future enhancement to improve functionality, personalization, and inclusivity.

Despite these constraints, the developed system successfully meets the main objectives: secure login, role-based access, booking management, and responsive design. The limitations can be addressed in future enhancement phases. [13]

### 5.3 Maintenance Cost

Software maintenance ensures that the system continues to perform efficiently after deployment. The maintenance cost includes server upkeep, domain renewal, bug fixes, and feature updates.

Maintenance Activity	Description	Estimated Annual Cost (BDT)
Domain Renewal	Annual renewal of domain (dreamnestbd.com)	1,500
Web Hosting / VPS Server	Server renewal for hosting backend and database	6,000
Database Backup & Security	Backup management and MongoDB Atlas plan upgrade	3,000
Email Service & API Renewal	Google SMTP / API maintenance	1,000
Bug Fixes & Updates	Regular code optimization and dependency updates	4,000
UI/UX Enhancement	UI updates, theme improvements, and responsiveness fixes	2,000
Testing and Monitoring	Performance and security testing tools	2,000

Table 09: Maintenance Cost

### 5.4 Maintenance Strategy

The maintenance process for DreamNest is structured around preventive, corrective, adaptive, and perfective strategies to ensure the system remains reliable, secure, and user-friendly. Preventive maintenance involves regular monitoring of server logs, updating dependencies, and checking APIs to proactively prevent downtime or performance issues. Corrective maintenance addresses any bugs or errors reported by users or discovered during testing, ensuring that the system continues to function smoothly. Adaptive maintenance focuses on updating the platform to accommodate new technologies, security protocols, or changing system requirements, keeping it up-to-date and compliant. Finally, perfective maintenance emphasizes continuous improvement by adding new features, enhancing the user interface, and implementing user feedback to provide a better overall experience. This comprehensive approach ensures that DreamNest remains stable, secure, and continuously evolving.

## **Chapter 6 – Conclusion**

The project “DreamNest: A Real Estate Management System” has been successfully designed and implemented using the MERN (MongoDB, Express.js, React.js, Node.js) technology stack. The system efficiently addresses the challenges of traditional real estate management by providing a secure, dynamic, and user-friendly digital platform for both users and administrators. [1] [3]

Throughout the development process, the project followed a structured methodology — from requirement analysis and design to implementation and testing. The resulting system offers essential functionalities such as role-based authentication, property management, manual booking system, user profile control, and an interactive admin dashboard. These features contribute to a seamless user experience and effective management of real estate operations. [13]

From a technical perspective, DreamNest ensures data security through JWT-based authentication and bcrypt password hashing, while maintaining performance and responsiveness through optimized frontend rendering and database queries. The integration of Google Maps for property location visualization and email notification via Nodemailer adds practical value to the platform. The use of Tailwind CSS and Framer Motion enhances the system’s modern and interactive user interface. [4]

This project has demonstrated how modern web technologies can simplify and automate complex property management processes. By centralizing user data, property listings, and booking workflows, DreamNest effectively bridges the gap between real estate buyers and administrators. [13]

In conclusion, the developed system is scalable, secure, and adaptable for future improvements. With additional enhancements such as online payment integration, AI-based recommendations, and real-time communication, DreamNest has the potential to evolve into a fully commercial-grade platform serving the growing real estate market in Bangladesh and beyond.

## References

- [1] M. Eisenberg, “MERN Stack Development: A Complete Guide,” *Journal of Web Engineering*, vol. 14, no. 2, pp. 45–52, 2022.
- [2] M. S. Imran and K. Rahman, “Role-Based Access Control in Web Applications,” *International Journal of Computer Applications (IJCA)*, vol. 179, no. 3, pp. 10–18, 2023.
- [3] N. Aggarwal, Full Stack Development with MongoDB, Express, React, and Node, Packt Publishing, 2021. [9]
- [4] M. H. Ahsan, “Implementation of Secure Authentication using JWT and Bcrypt in Node.js Applications,” *IEEE Access*, vol. 11, pp. 11322–11330, 2023.
- [5] A. Hossain and T. Islam, “A Comparative Study of Traditional and Online Real Estate Management Systems,” *Asian Journal of Information Technology*, vol. 20, no. 6, pp. 125–132, 2022.
- [6] R. T. Fielding, “Architectural Styles and the Design of Network-based Software Architectures,” Doctoral Dissertation, University of California, Irvine, 2000.
- [7] Google Developers, “Google Maps Platform Documentation,” Available: <https://developers.google.com/maps/documentation>, Accessed: Oct. 2025.
- [8] Nodemailer, “Nodemailer: Send e-mails from Node.js,” Available: <https://nodemailer.com/about/>, Accessed: Oct. 2025.
- [9] React Official Documentation, “React – A JavaScript library for building user interfaces,” Available: <https://react.dev>, Accessed: Oct. 2025.
- [10] Tailwind Labs, “Tailwind CSS Documentation,” Available: <https://tailwindcss.com/docs>, Accessed: Oct. 2025.
- [11] MongoDB Inc., “MongoDB Developer Documentation,” Available: <https://www.mongodb.com/docs/>, Accessed: Oct. 2025.
- [12] Express.js Foundation, “Express – Fast, unopinionated, minimalist web framework for Node.js,” Available: <https://expressjs.com>, Accessed: Oct. 2025.
- [13] A. Rahman and M. Chowdhury, “A Secure Web-Based Property Booking System,” *International Conference on Computing and Communication Systems (ICCCS)*, IEEE, 2023.
- [14] Render Cloud, “Node.js Hosting for Developers,” Available: <https://render.com/docs>, Accessed: Oct. 2025.