Contents

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.

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.

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.

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.
- **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.

- **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.

- **Notifications and Communication:**
  Email confirmations and admin notifications are sent through **Nodemailer** integration, ensuring transparency between users and the admin.

- **Google Maps Integration:**
  Each property's location is visually represented using Google Maps on the details page, enhancing geographic visualization for users.

- **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:

1. **To design and develop** a fully functional web-based real estate management system using the MERN stack.

2. **To implement role-based access control** where users and admins have distinct privileges and dashboards.

3. **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.
5. **To manage property-related data** effectively through CRUD (Create, Read, Update, Delete) operations.
6. **To enable users** to browse, view, and book properties with full transparency and manual payment confirmation.
7. **To provide an admin dashboard** that allows monitoring of all bookings, users, and properties in a centralized interface.
8. **To ensure data security, scalability, and efficiency** using MongoDB and Express.js for backend operations.
9. **To include Google Maps integration** for improved property location visualization.
10. **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.

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.

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:

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

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*.

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.
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.

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.

Key features of the proposed system include:

- Secure **JWT-based authentication** and **bcrypt password encryption**.
- Separate **dashboards for Admin and User** roles.
- Full **CRUD operations** for properties, users, and bookings.
- Real-time booking and status tracking.
- **Google Maps integration** for precise location display.
- **Responsive design** with Tailwind CSS and Framer Motion animations.
- Dynamic **search and filter** capabilities for property listings.
- **Email notifications** for user registration and booking updates.
- **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.

The project offers long-term cost benefits:

- Reduces dependency on manual booking and property management.
- 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.

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 focuses on whether the system will function effectively once deployed.

- The system's role-based design ensures that users and admins have clear operational boundaries.
- User interfaces are intuitive, reducing the need for extensive training.

- Booking and property management workflows are simple and automated.
- The responsive design ensures accessibility across desktops, tablets, and smartphones.

Hence, the system is **operationally feasible** and user-friendly for both administrators and end-users.

2.3 Nonfunctional Requirements

2.3.1 Product Requirements

- **Performance:** Fast response time (<2 seconds per API request) and efficient data retrieval.
- **Scalability:** Supports increasing numbers of users, properties, and bookings without performance degradation.
- **Security:** JWT tokens, encrypted passwords, and HTTPS ensure confidentiality and integrity.
- **Availability:** 99% uptime through reliable hosting solutions.
- **Usability:** Clean, responsive, and accessible UI following UX best practices.
- **Portability:** Compatible with major browsers (Chrome, Firefox, Edge) and adaptable to mobile devices.

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

- Register and log in securely.
- Browse and search properties by type, price, and location.

- View detailed property information with Google Map integration.
- Book a property (manual confirmation).
- View “My Bookings” list with statuses (Pending, Approved, Rejected).
- Edit personal profile (name, email, phone).
- Log out securely.

2.4.2 Admin Functions

- Secure admin login (role-based access).
- Add, edit, delete, and update property information.
- View and manage all registered users.
- Monitor all bookings and update booking status.
- View statistical dashboard (total users, bookings, and properties).
- Send notifications or emails to users.

2.4.3 Property Functions

- Maintain property details (title, description, price, type, location, images).
- Allow search and filter by property type or location.
- Display property status (available/booked).
- Handle property CRUD operations by admin.

2.5 Project Requirement Specification

2.5.1 Hardware Requirements

Component	Minimum Requirement
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

Component	Specification
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	**6,500 BDT**	

Table 03: Cost Analysis

Benefits:

- Reduced operational cost by automating manual processes.
- Increased system efficiency and transparency.
- Improved accessibility and faster decision-making for users and admins.

The benefits of *DreamNest* significantly outweigh its minor development and maintenance costs.

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:

1. **Requirement Gathering:**
   The initial phase focuses on identifying and understanding the project goals, user roles, and functionalities. Both user and admin requirements were analyzed and documented.

2. **System Design:**
   Based on the requirements, system architecture, database structure, and UI flow were designed. The ER diagrams, data flow diagrams, and use case models were prepared to visualize system interactions.

3. **Implementation (Development):**
   The frontend and backend were developed separately under two folders —

   - **frontend/** for React.js + TypeScript + Tailwind CSS
   - **backend/** for Node.js + Express + MongoDB
     The modular approach ensures better maintainability and reusability of code.

4. **Testing:**
   Unit testing, integration testing, and user acceptance testing (UAT) were conducted to ensure the functionality, reliability, and performance of the system.

5. **Deployment:**
   The system was deployed on a web server with database hosting. Security and role-based access controls were validated in the production environment.

6. **Maintenance and Feedback:**
   Regular updates are applied to fix bugs, enhance performance, and add new features based on user feedback.

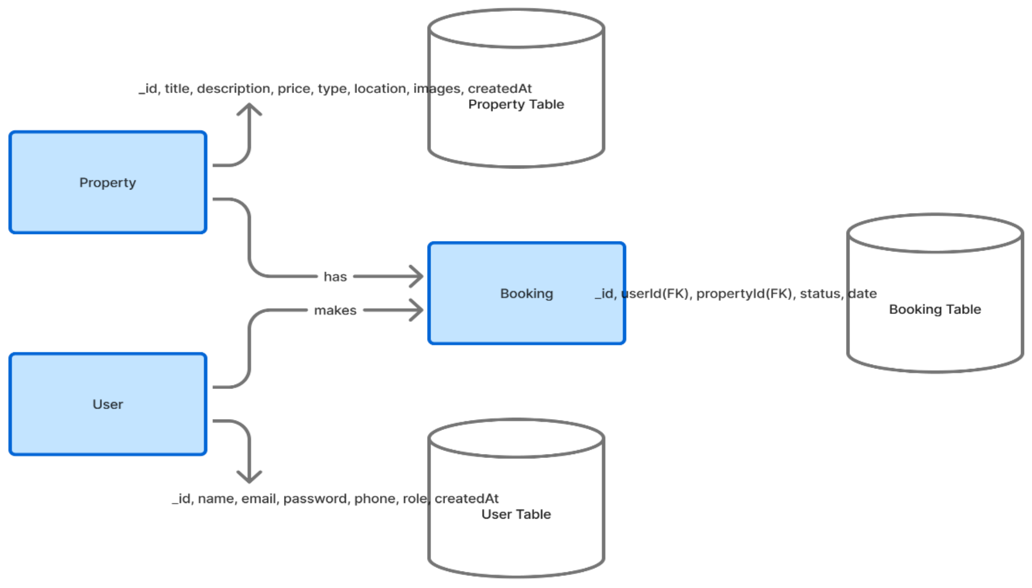
This iterative cycle ensures that *DreamNest* remains adaptable, user-friendly, and technologically robust throughout its lifecycle.

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.

Entities and Relationships:

- **User:** Represents individuals using the system (role: user/admin).
- **Property:** Represents the listed flats or lands available for booking.
- **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*.



_id, title, description, price, type, location, images, createdAt
Property Table
Property
has
makes
Booking
_id, userId(FK), propertyId(FK), status, date
Booking Table
User
_id, name, email, password, phone, role, createdAt
User Table

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.
   - Outputs: Property listings, booking confirmation, profile information.
2. **Admin:**
   - Inputs: Property data, booking approval/rejection, user management actions.
   - Outputs: Dashboard statistics, booking updates, notifications.

System (DreamNest):

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

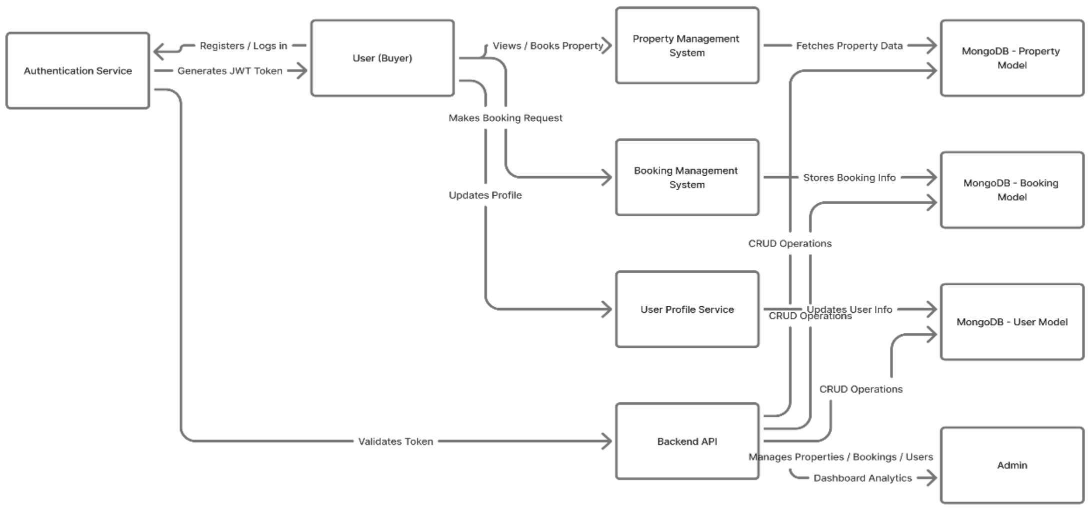


Diagram 2-Context Diagram

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.
- 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.
- 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.

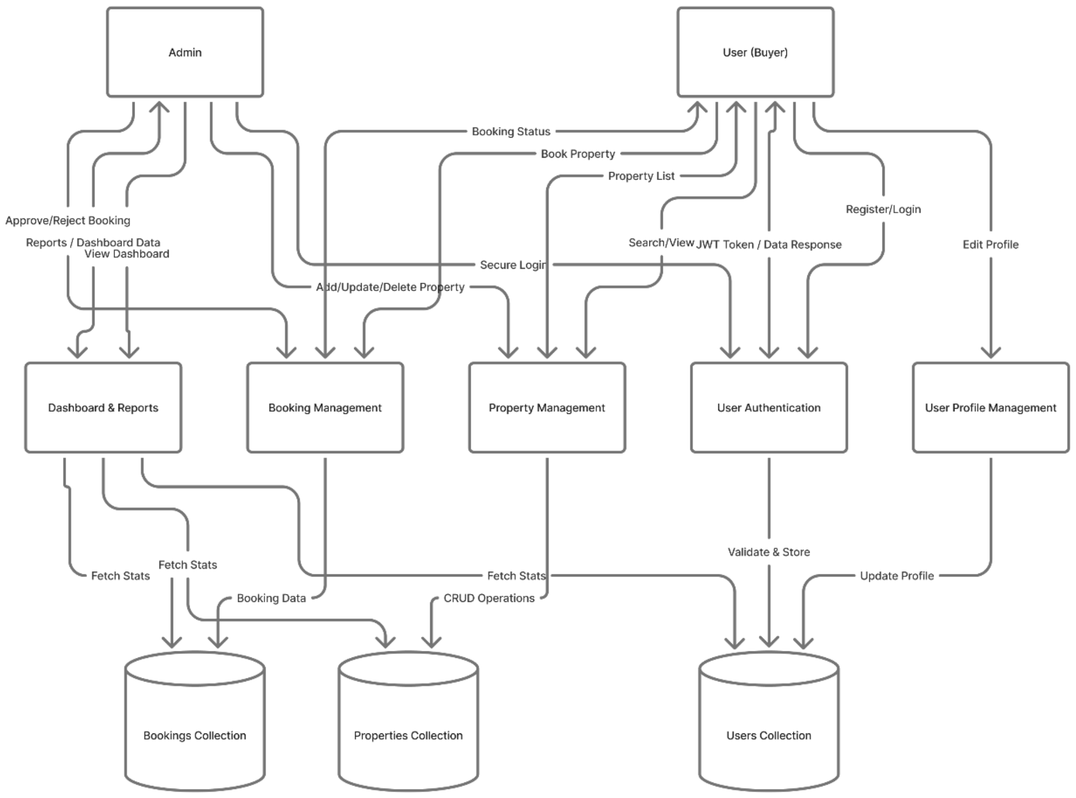


Diagram 3-Data Flow Diagram

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
- Edit personal profile

Admin Use Cases:

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

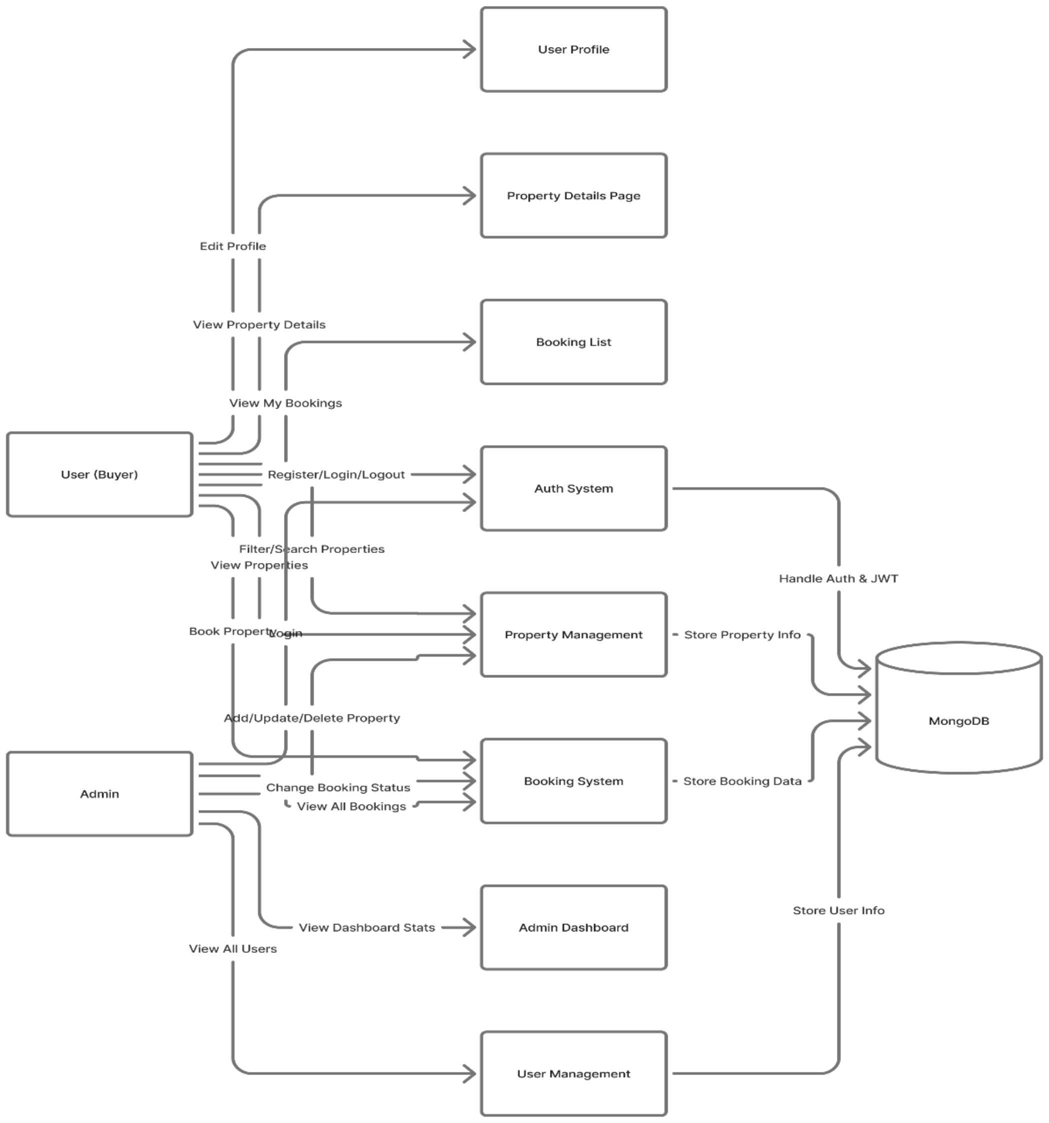


Diagram 4-Use Case Diagram

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.

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.
- **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.

3.6.5 Database Normalization and Integrity

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

- Properties and users are stored separately to maintain modularity.
- Bookings contain references rather than embedded data, ensuring consistency.
- Indexing is applied on frequently queried fields such as email, location, and propertyId.

3.6.6 Data Security and Backup

- All passwords are encrypted using **bcrypt**.
- Access tokens are stored in **HTTP-only cookies** for enhanced security.
- MongoDB Atlas (cloud-based) is used for database hosting with automated daily backups.

- Proper role-based validation prevents unauthorized CRUD operations.

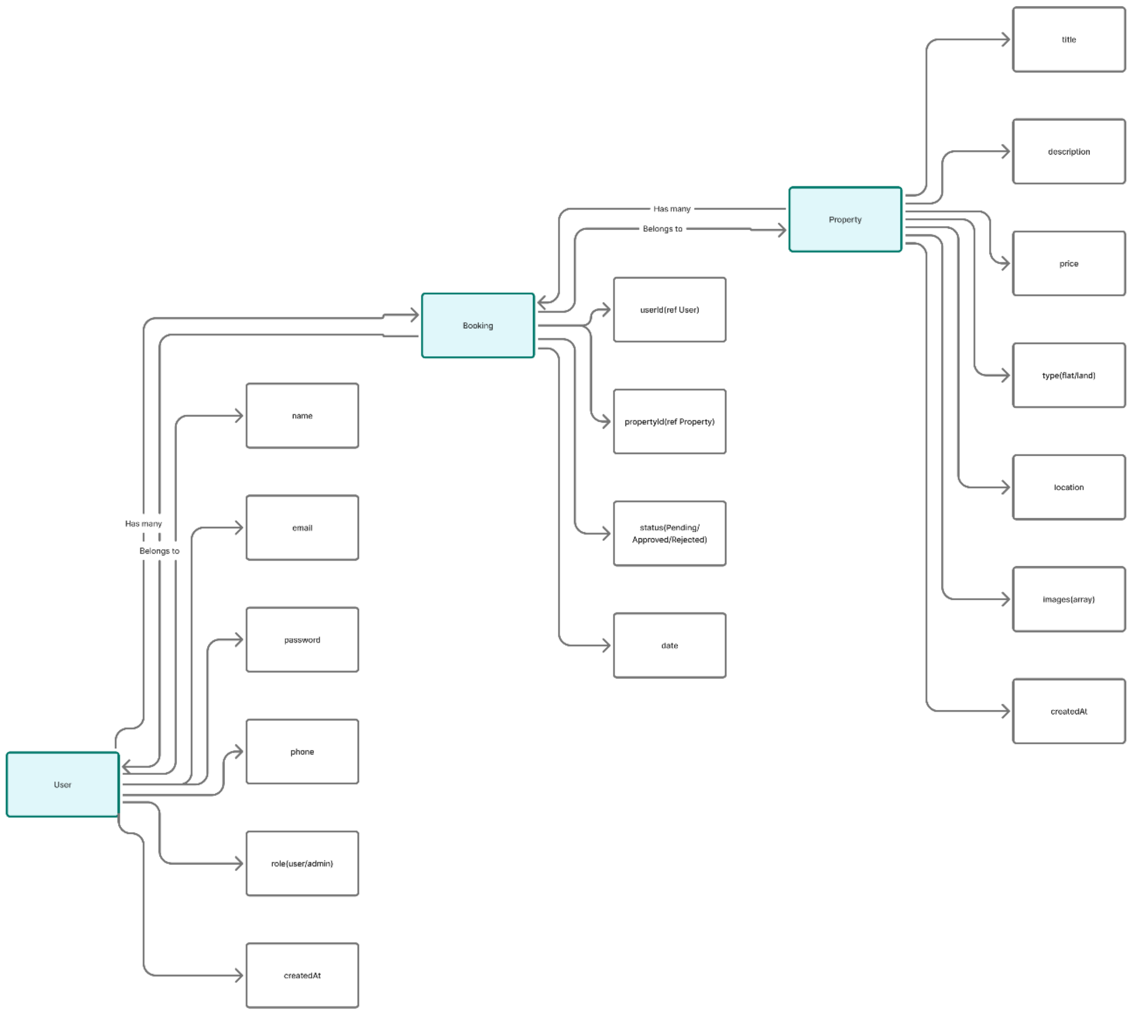


Diagram 5-Database Schema Diagram

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.

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.

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.

4.2 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)**.

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

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

4.3 System Overview Diagram

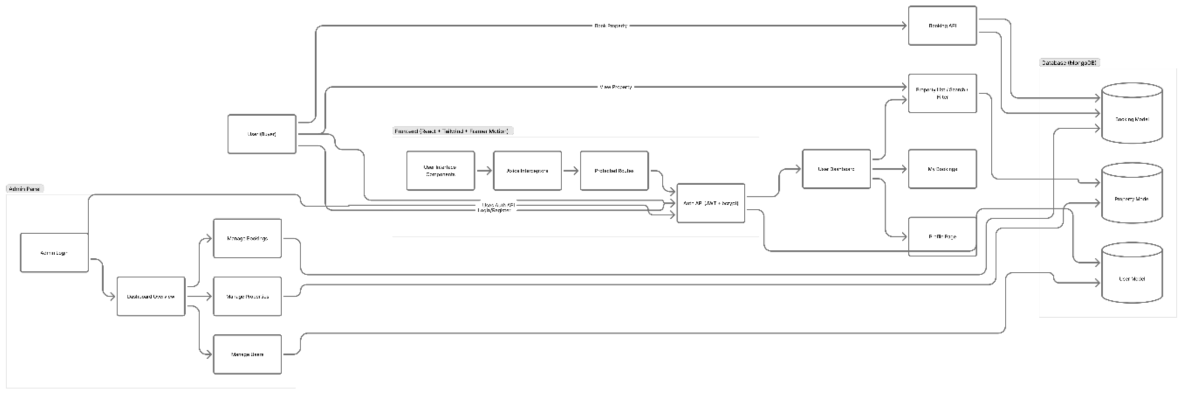


Diagram 6-System Overview Diagram

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.

4.4 Developed Dashboards

4.4.1 User Dashboard

The **User Dashboard** is designed with simplicity and usability in mind. After successful login, the user can:

- View all available properties (flat/land) with search and filter options.
- Check property details like **price, location, and images**.
- Book a property, generating a booking record with a **"Pending"** status.
- View their own booking history with status updates ("Approved", "Rejected", or "Pending").
- Update personal information such as name, phone number, and email.
- Switch between **light and dark themes** for better user experience.

The user interface follows **Bangladeshi cultural design inspiration**, maintaining a professional yet aesthetic look with Tailwind CSS.

4.4.2 Admin Dashboard

The **Admin Dashboard** is developed for the platform administrators to manage all system operations efficiently. The dashboard includes:

- **Overview Section:** Displays total properties, total users, total bookings, and recent bookings.
- **Property Management:** Admins can add, edit, or delete properties with detailed descriptions and image URLs.
- **User Management:** Admins can view all registered users and their activities.
- **Booking Management:** Admins can view all user bookings and change their statuses (Pending → Approved/Rejected).
- **Notification Management:** Admins receive notifications of new bookings and can send confirmation emails to users.

All admin routes are protected and accessible only to authorized admin users through **role-based authentication**.

4.5 Features Implementation

4.5.1 Authentication and Authorization

- Implemented using **JWT tokens** for secure authentication.
- **bcrypt** is used to hash passwords for enhanced security.
- Login tokens are stored in **HTTP-only cookies** to prevent XSS attacks.
- Role-based routing ensures that users and admins only access their designated areas.

4.5.2 Property Management

- Admins can create, read, update, and delete properties.
- Properties include **title, description, price, type (flat or land), location, and images**.
- Properties are displayed dynamically on the home page with a responsive grid layout.

4.5.3 Booking System

- Users can manually book properties (no online payment).
- Bookings are stored in the database with user and property references.
- Admins can change the booking status to Approved or Rejected.
- Email notifications are sent using **Nodemailer** upon booking confirmation.

4.5.4 User Profile Management

- Users can edit their personal details.
- Profile updates are reflected instantly in the MongoDB database.
- Secure endpoints ensure that users can only edit their own profiles.

4.5.5 Dashboard Analytics

- Admin overview dashboard uses **real-time data fetching** through Axios.
- Displays:
  - Total Users
  - Total Properties
  - Total Bookings
  - Recent Activities
- Dashboard charts can be integrated using **Recharts or Chart.js**.

4.5.6 UI and UX Design

- Fully responsive design using **Tailwind CSS**.
- 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.
- **TypeScript:** Ensures type safety and reduces runtime errors.
- **Tailwind CSS:** Provides a utility-first responsive design system.
- **Axios:** Handles API communication between frontend and backend.
- **React Router:** Manages navigation and route protection.
- **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.
- **MongoDB with Mongoose:** NoSQL database for flexible data storage.
- **JWT & bcrypt:** Authentication and password encryption tools.
- **Nodemailer:** Sends confirmation emails to users and notifications to admin.
- **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.

Relationships:

- Each booking references one user and one property via **ObjectId**.
- Data consistency is maintained using **Mongoose population** features.

4.9 Testing and Validation

Multiple testing phases were conducted to ensure performance and security:

- **Unit Testing:** Individual components and API endpoints tested using Jest.
- **Integration Testing:** Verified the seamless communication between frontend and backend.
- **User Acceptance Testing:** Conducted to ensure usability, navigation, and responsiveness.
- **Performance Testing:** Ensured smooth operation under concurrent user load.

4.10 Performance and Result

The final system successfully meets all design and functional requirements:

- Efficient property management and dynamic dashboards.

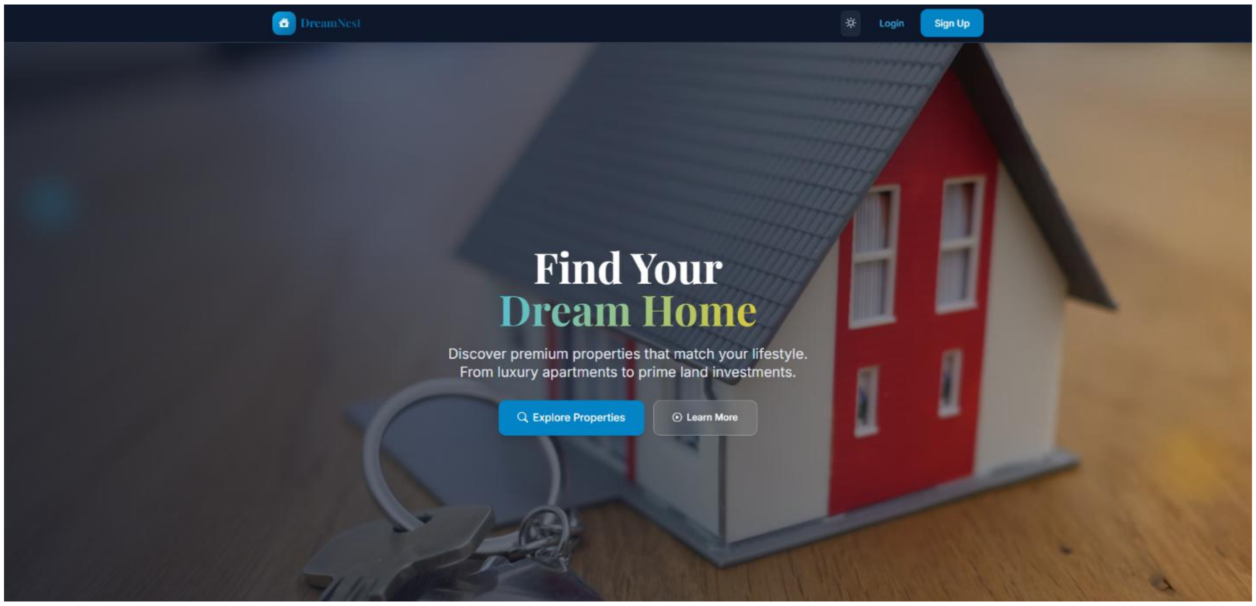
- Secure authentication and role-based routing.
- Seamless CRUD operations across all entities.
- Real-time updates and notifications enhance interactivity.

Performance benchmarks indicate:

- Average page load time: < 2.5 seconds
- Booking confirmation time: < 1 second
- Admin dashboard data fetch: < 800 ms

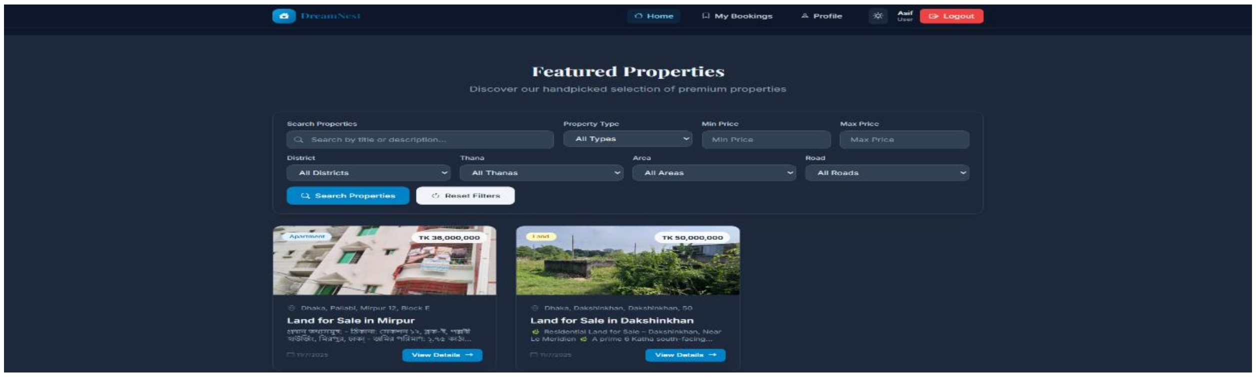
These results confirm that **DreamNest** is reliable, scalable, and production-ready.

4.11 Screenshot Placeholders



Discover premium properties that match your lifestyle. From luxury apartments to prime land investments.

Screenshot of Home Page



Featured Properties

Discover our handpicked selection of premium properties

Search by title or description...

All Types

Min Price

Max Price

All Districts

All Thanas

All Areas

All Roads

Apartment — TK 38,000,000

Dhaka, Pallabi, Mirpur 12, Block E

Land for Sale in Mirpur

[illegible]

11/7/2025

Land — TK 50,000,000

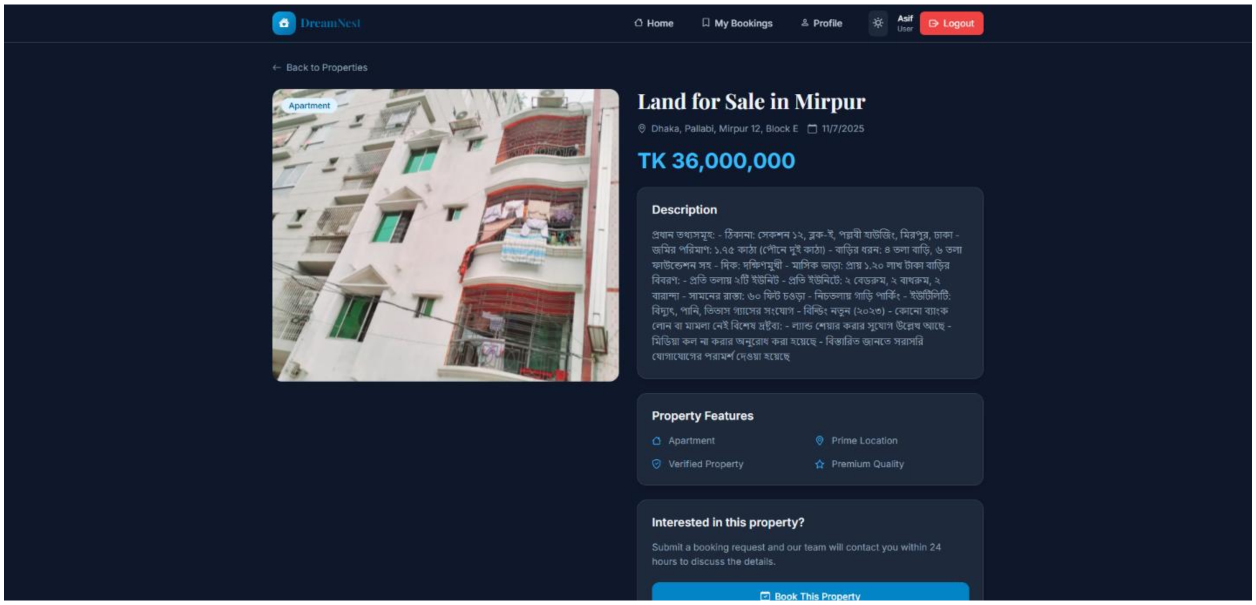
Dhaka, Dakshinkhan, Dakshinkhan, 50

Land for Sale in Dakshinkhan

Residential Land for Sale – Dakshinkhan, Near Le Meridien A prime 6 Katha south-facing...

11/7/2025

Screenshot of User Dashboard



Land for Sale in Mirpur

Dhaka, Pallabi, Mirpur 12, Block E 11/7/2025

TK 36,000,000

Description

প্রধান তথ্যসমূহ: - ঠিকানা: সেকশন ১২, ব্লক-ই, পল্লবী হাউজিং, মিরপুর, ঢাকা - জমির পরিমাণ: ১.৭৫ কাঠা (পৌনে দুই কাঠা) - বাড়ির ধরন: ৪ তলা বাড়ি, ৬ তলা ফাউন্ডেশন সহ - দিক: দক্ষিণমুখী - মাসিক ভাড়া: প্রায় ১.২০ লাখ টাকা বাড়ির বিবরণ: - প্রতি তলায় ২টি ইউনিট - প্রতি ইউনিটে: ২ বেডরুম, ২ বাথরুম, ২ বারান্দা - সামনের রাস্তা: ৬০ ফিট চওড়া - নিচতলায় গাড়ি পার্কিং - ইউটিলিটি: বিদ্যুৎ, পানি, তিতাস গ্যাসের সংযোগ - বিল্ডিং নতুন (২০২৩) - কোনো ব্যাংক লোন বা মামলা নেই বিশেষ দ্রষ্টব্য: - ল্যান্ড শেয়ার করার সুযোগ উল্লেখ আছে - মিডিয়া কল না করার অনুরোধ করা হয়েছে - বিস্তারিত জানতে সরাসরি যোগাযোগের পরামর্শ দেওয়া হয়েছে

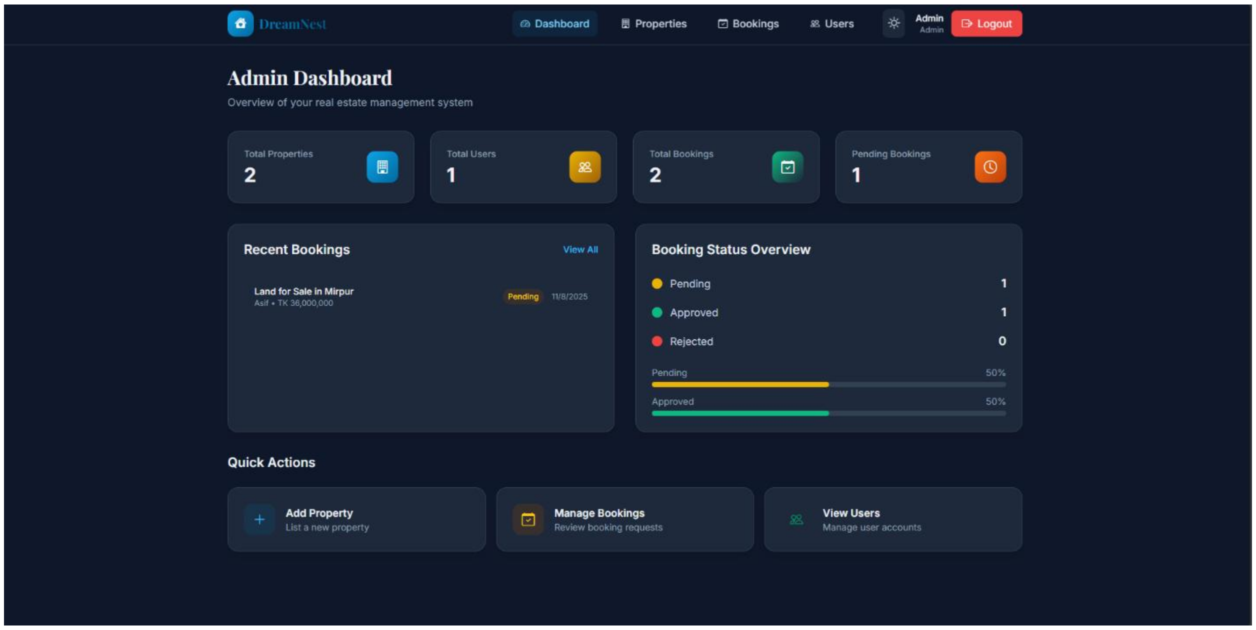
Property Features

Apartment	Prime Location
Verified Property	Premium Quality

Interested in this property?

Submit a booking request and our team will contact you within 24 hours to discuss the details.

Screenshot of Property Details Page



Admin Dashboard

Overview of your real estate management system

Total Properties	Total Users	Total Bookings	Pending Bookings
2	1	2	1

Recent Bookings

Land for Sale in Mirpur
Asif • TK 36,000,000 Pending 11/8/2025

Booking Status Overview

● Pending	1
● Approved	1
● Rejected	0

Pending 50%
Approved 50%

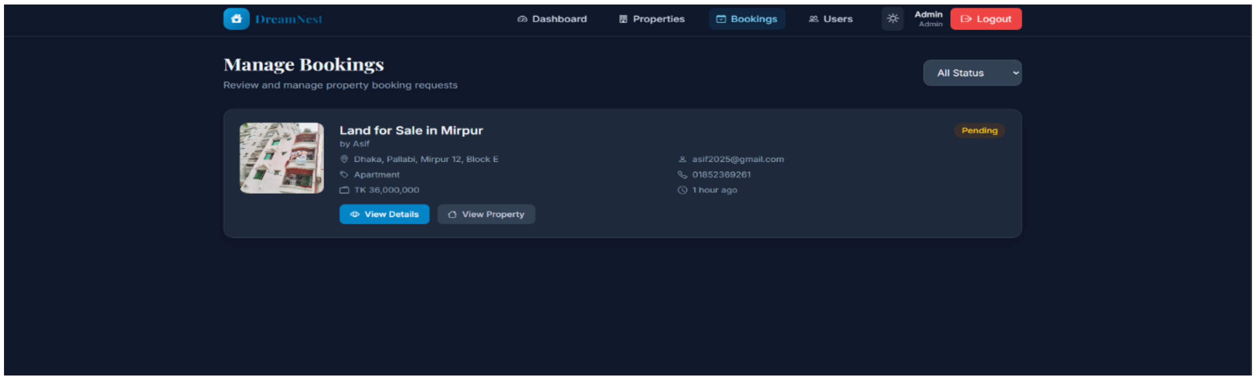
Quick Actions

Add Property
List a new property

Manage Bookings
Review booking requests

View Users
Manage user accounts

Screenshot of Admin Dashboard



Manage Bookings

Review and manage property booking requests

All Status

Land for Sale in Mirpur

by Asif

Dhaka, Pallabi, Mirpur 12, Block E

Apartment

TK 36,000,000

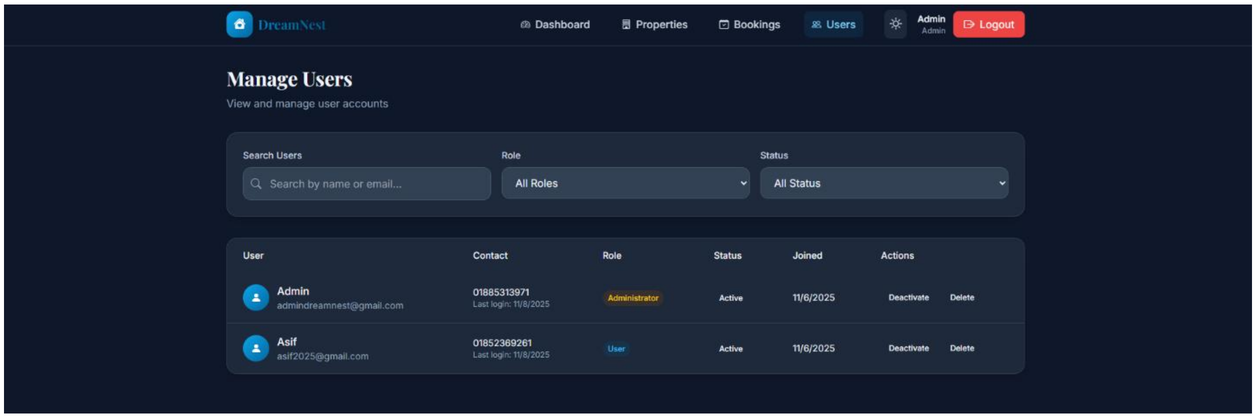
asif2025@gmail.com

01852369261

1 hour ago

Pending

Screenshot of Booking Management Page



Manage Users

View and manage user accounts

Search Users: Search by name or email...

Role: All Roles

Status: All Status

User	Contact	Role	Status	Joined	Actions
Admin admindreamnest@gmail.com	01885313971 Last login: 11/8/2025	Administrator	Active	11/6/2025	Deactivate Delete
Asif asif2025@gmail.com	01852369261 Last login: 11/8/2025	User	Active	11/6/2025	Deactivate Delete

Screenshot of 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.

Table 5.1: 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

1. The system is designed for **manual property booking only** — no online payment gateway is currently implemented.
2. The **Google Maps integration** uses a limited free API key, which may restrict location rendering in heavy traffic.
3. **Email notifications** are limited to admin alerts; users do not receive booking confirmation emails yet.
4. Image uploads are handled via **URL input** rather than drag-and-drop or local file storage due to server limitations.
5. The backend server currently runs on a **single-node environment**; no load balancing or clustering is configured.

5.2.2 Operational Limitations

1. The admin panel requires **manual status updates** for every booking.
2. The system currently supports **only one administrator account** for all backend operations.
3. Role management is **hard-coded**, meaning dynamic role creation (e.g., “Agent” role) is not yet supported.
4. No **mobile application version** has been developed yet; the system is web-based only.

5.2.3 Functional Limitations

1. The system does not include **property comparison** or **price trend analysis** features.
2. There is no **AI-based property recommendation** engine to suggest similar listings.
3. The **review/comment system** for users to provide feedback on properties is not implemented.
4. **Multilingual support** is not available; the system runs only in English.

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.

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.

Table 5.2: Estimated Annual Maintenance Cost

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 follows a **preventive and corrective** approach:

1. **Preventive Maintenance** – Regular monitoring of server logs, dependency updates, and API checks to prevent downtime.
2. **Corrective Maintenance** – Fixing any bugs reported by users or detected in testing.
3. **Adaptive Maintenance** – Updating the system as new technologies or security requirements emerge.
4. **Perfective Maintenance** – Adding new features or UI improvements based on user feedback.

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.

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.

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.

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.

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.

[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.