**Database Management system**

**Project Report**

**Real Estate Management System**



**Group members**

|  |  |
| --- | --- |
| Areeza Maryam | 01-135232-013 |
| Rubab Hanif | 01-135232-082 |
| Hamza Masood | 01-135232-023 |

**Class:** BS-IT(5-B)  
**Submitted to:** Mam Mehwish Pervaiz

**Project Overview**

The **Real Estate Management System** is a comprehensive web-based application designed to streamline the operations of a real estate agency. It facilitates the management of properties, agents, clients, and financial transactions through a centralized platform. The system employs Role-Based Access Control (RBAC) to ensure that Administrators (Agencies), Agents, and Clients have access to appropriate features and data.

**Technology Stack**

**Frontend**

* **HTML5 & CSS3**: Provides structural foundation and styling for the user interface.
* **Vanilla JavaScript**: Handles client-side logic, API interactions, and dynamic DOM manipulation without the overhead of heavy frameworks.
* **Bootstrap / Custom CSS**: Ensures a responsive design that works across devices.

**Backend**

* **Node.js**: The runtime environment executing JavaScript on the server side.
* **Express.js**: A minimal and flexible web application framework for handling API routes and middleware.
* **Authentication**:
  + **JSON Web Tokens (JWT)**: Securely manages user sessions and authorization.
  + **Bcrypt.js**: Hashes passwords to ensure sensitive data security.

**Database**

* **MySQL**: Relational database management system used for storing structured data.
* **MySQL2 Client**: A fast, reliable Node.js driver for communicating with the MySQL database.

**System Architecture & Logic**

**Role-Based Access Control (RBAC)**

The system logic is divided into three distinct user flows:

1. **Admin (Agency)**: The superuser with full control over the system. Can manage agents, properties, listings, transactions, and view global statistics.
2. **Agent**: Represents agency employees. They can manage their specific listings, schedule inspections, and track their sales/commissions.
3. **Client**: The end-users (Buyers/Sellers/Renters). They can browse listings, request inspections, and provide feedback.

**Database Schema Design**

The database is normalized to ensure data integrity and efficiency. Key entities include:

* **Core Entities**: Agency, Agent, Client.
* **Property Management**: Property (physical asset), Listing (sales instance), Property\_Type, Location.
* **Business Logic**: Transaction (sales/rent records), Payment (financials), Contract (legal agreements).
* **Operations**: Inspection (viewings), Feedback (quality assurance).

**API Design**

The backend exposes a RESTful API structure:

* /api/auth: Handles login and signup for all roles.
* /api/admin: Protected routes involving full CRUD operations for system administration.
* /api/agent: Routes specific to agent workflows (e.g., managing own listings).
* /api/client: Public and protected routes for property browsing and user user interactions.

**Key Features & Outcomes**

**Dashboard Analytics**

* **Admin Dashboard**: Provides high-level metrics such as Total Sales, Active Listings, Properties Sold/Rented, and Agent Performance.
* **Data Visualization**: Statistical summaries help the agency make data-driven decisions.

**Property Lifecycle Management**

The system tracks the entire lifecycle of a property:

1. **Acquisition**: Property details are entered into the system.
2. **Listing**: The property is published for Sale or Rent with a specific price.
3. **Inspection**: Potential buyers schedule visits.
4. **Transaction**: A deal is made, recording the final price and commission.
5. **Contract & Payment**: Legal terms are drafted, and payments are tracked until completion.

**Financial Management**

* **Transaction Tracking**: Records every sale and rental agreement.
* **Commission Calculation**: Automatically associates commissions with specific agents.
* **Payment Status**: Tracks individual payments (deposits, installments) against transactions.

**Operational Efficiency**

* **Automated Relations**: Linking Properties to Agents and Transactions to Clients reduces manual data entry errors.
* **Search & Filter**: Allows easy retrieval of properties based on location, type, price, and status.

**Deployment & Setup**

The project includes automation scripts for easy deployment:

* setup-database.bat: Initializes the MySQL database and schema.
* add-sample-data.bat: Populates the database with seed data for testing.
* start-project.bat: Launches the Node.js server and client simultaneously.

**Project Overview**

**Database Snapshots**

****

**Database Tables**

**Agent:**

**A screenshot of a computer

AI-generated content may be incorrect.**

**Agency:**

**A screenshot of a computer

AI-generated content may be incorrect.**

**Client:**

**A screenshot of a computer

AI-generated content may be incorrect.**

**Inspection:**

**A screenshot of a computer

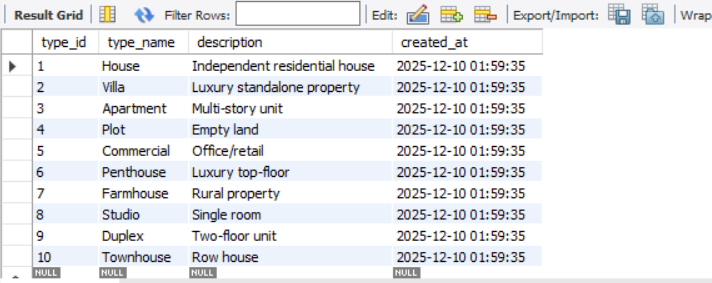
AI-generated content may be incorrect.**

**Property:**

**A screenshot of a computer

AI-generated content may be incorrect.**

**Property type:**

****

**Feedback:**

**A screenshot of a computer

AI-generated content may be incorrect.**

**Listing:**

**A screenshot of a computer

AI-generated content may be incorrect.**

**Location:**

**A screenshot of a computer

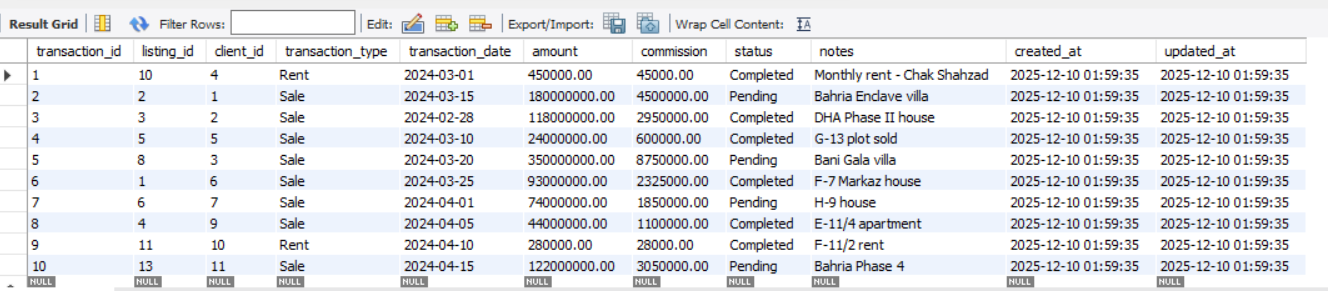
AI-generated content may be incorrect.**

**Payment:**

**A screenshot of a computer

AI-generated content may be incorrect.**

**Transaction:**

****

**Database Queries:**

We have two files for implementation of database:

* 1. Scheme.sql
  2. Seed.sql

**GUI:**

**Homepage**

****

**A green and black logo

AI-generated content may be incorrect.**

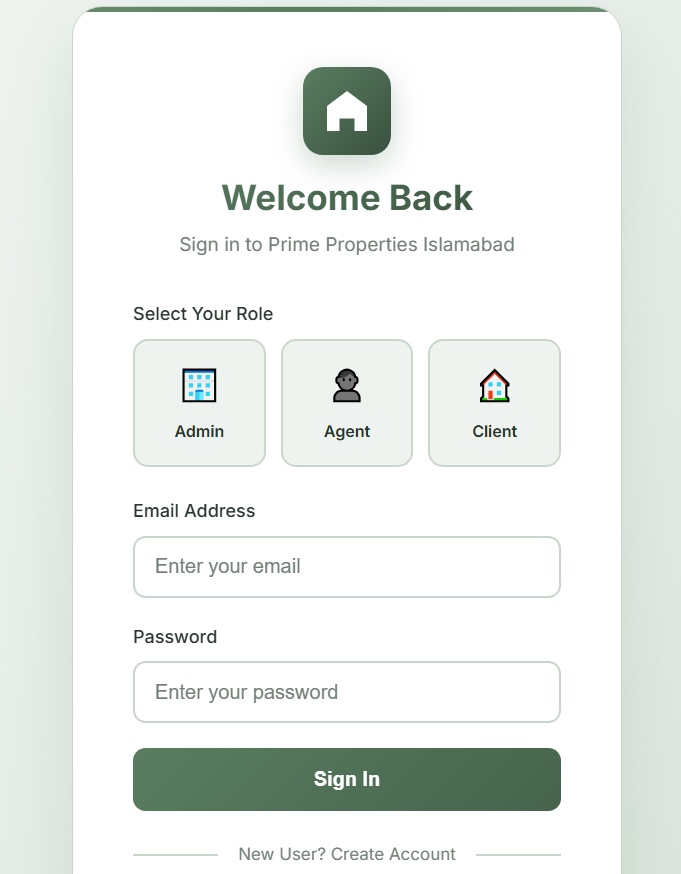
**A screenshot of a computer

AI-generated content may be incorrect.**

**A black screen with white text

AI-generated content may be incorrect.**

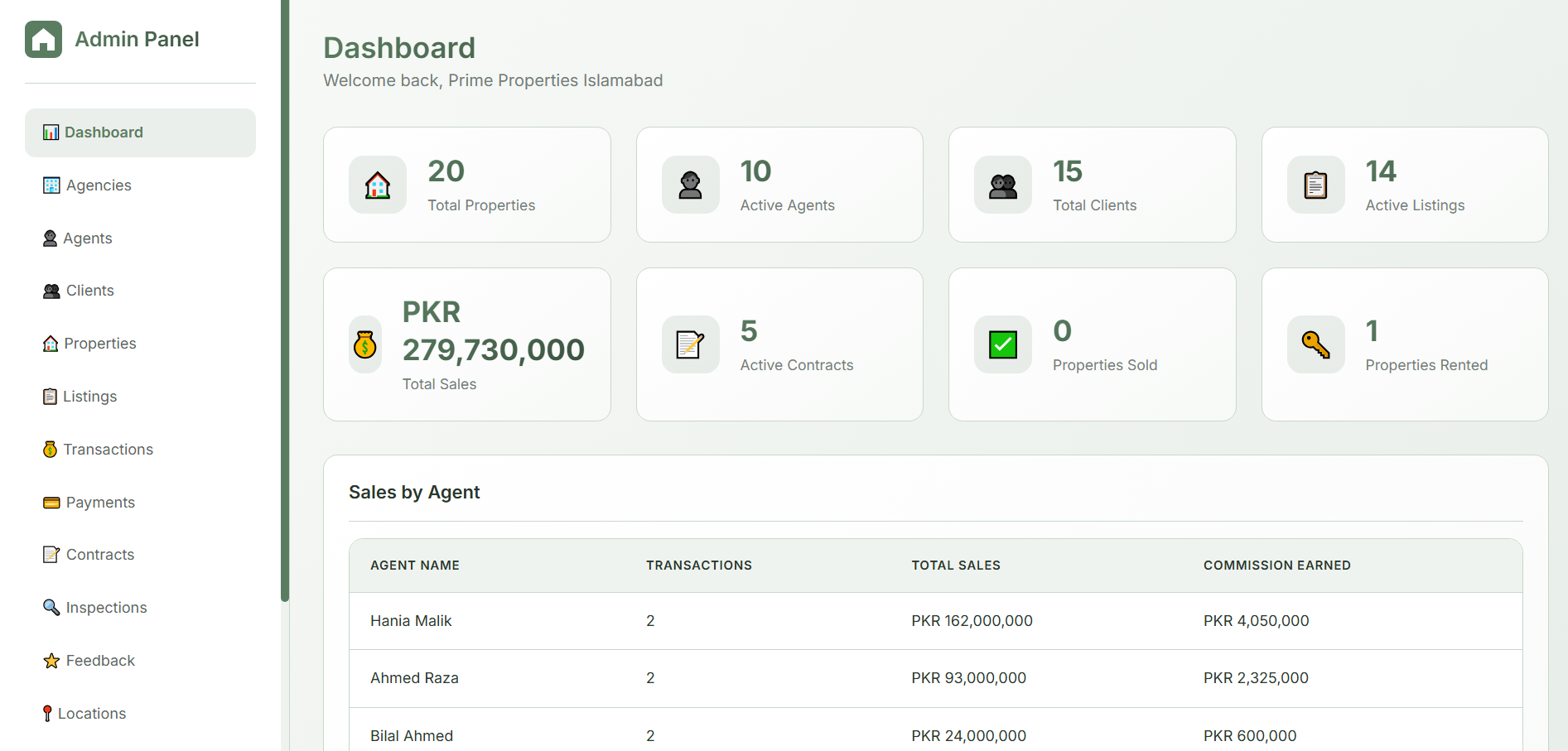
**Login Page**

****

**A screenshot of a login form

AI-generated content may be incorrect.**

**Admin (Agency) Panel:**

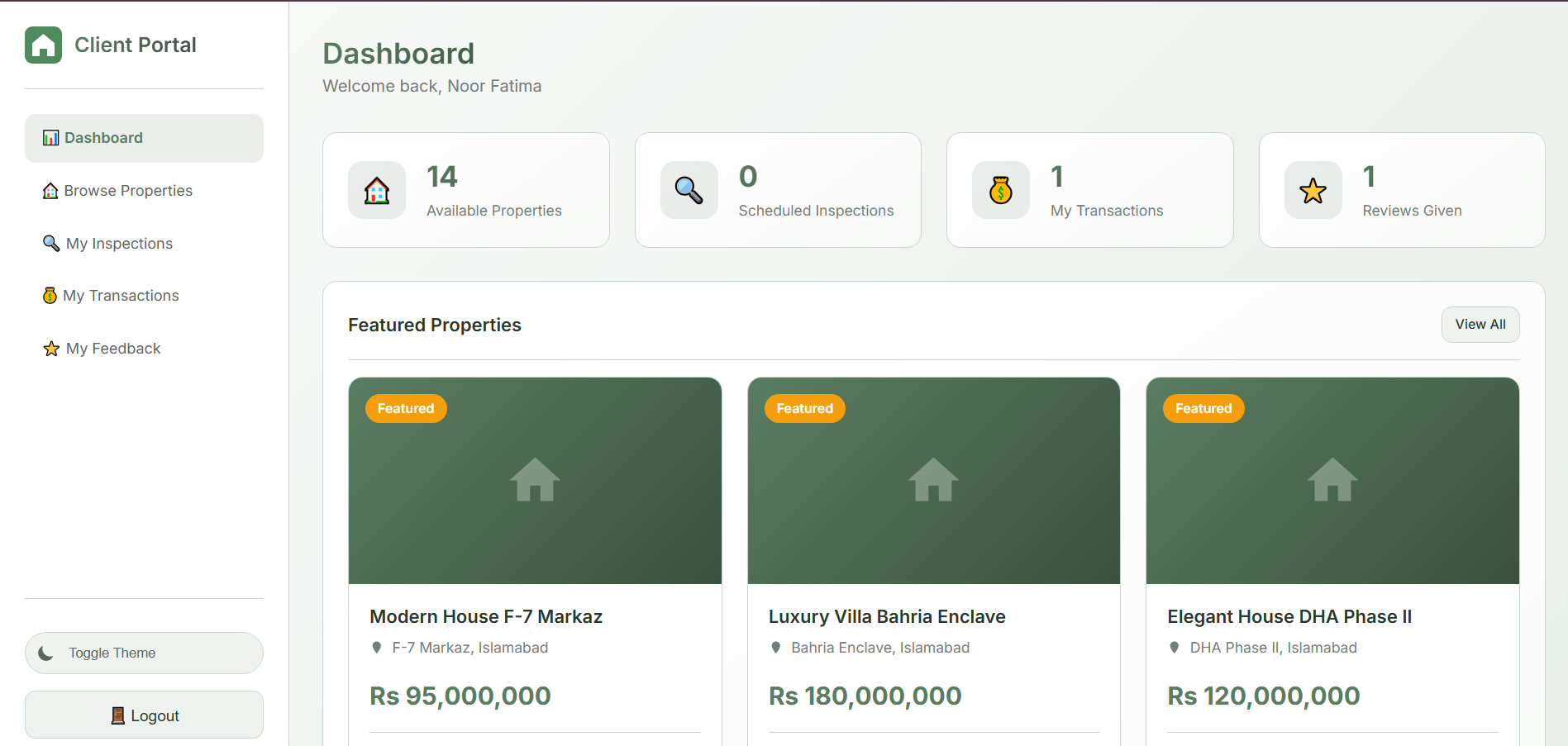
****

**Agent Panel:**

**A screenshot of a computer

AI-generated content may be incorrect.**

**Client Panel:**

****

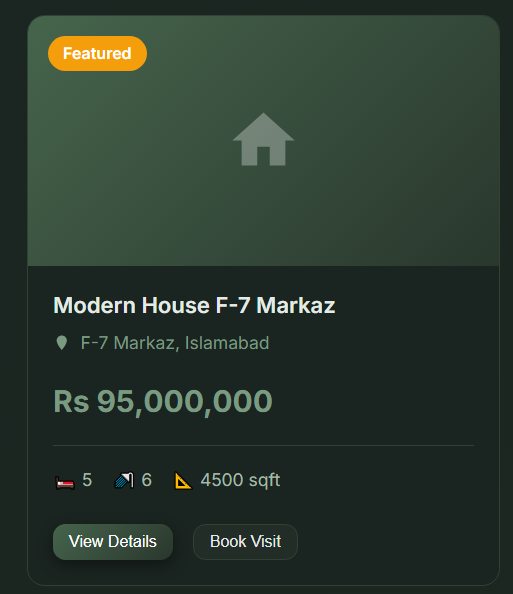
**Theme Can be changed to dark too:**

**A screenshot of a computer

AI-generated content may be incorrect.**

**System Functionality in Action**

* + 1. **Client books an inspection**



A screenshot of a book inspection

AI-generated content may be incorrect.

A screenshot of a computer

AI-generated content may be incorrect.

**A screenshot of a computer

AI-generated content may be incorrect.**

* + 1. **Inspection details are shown to Agent**

****

* + 1. **After inspection on the specific date agent updates it’s status as completed and it is shown to client**

**A screenshot of a computer

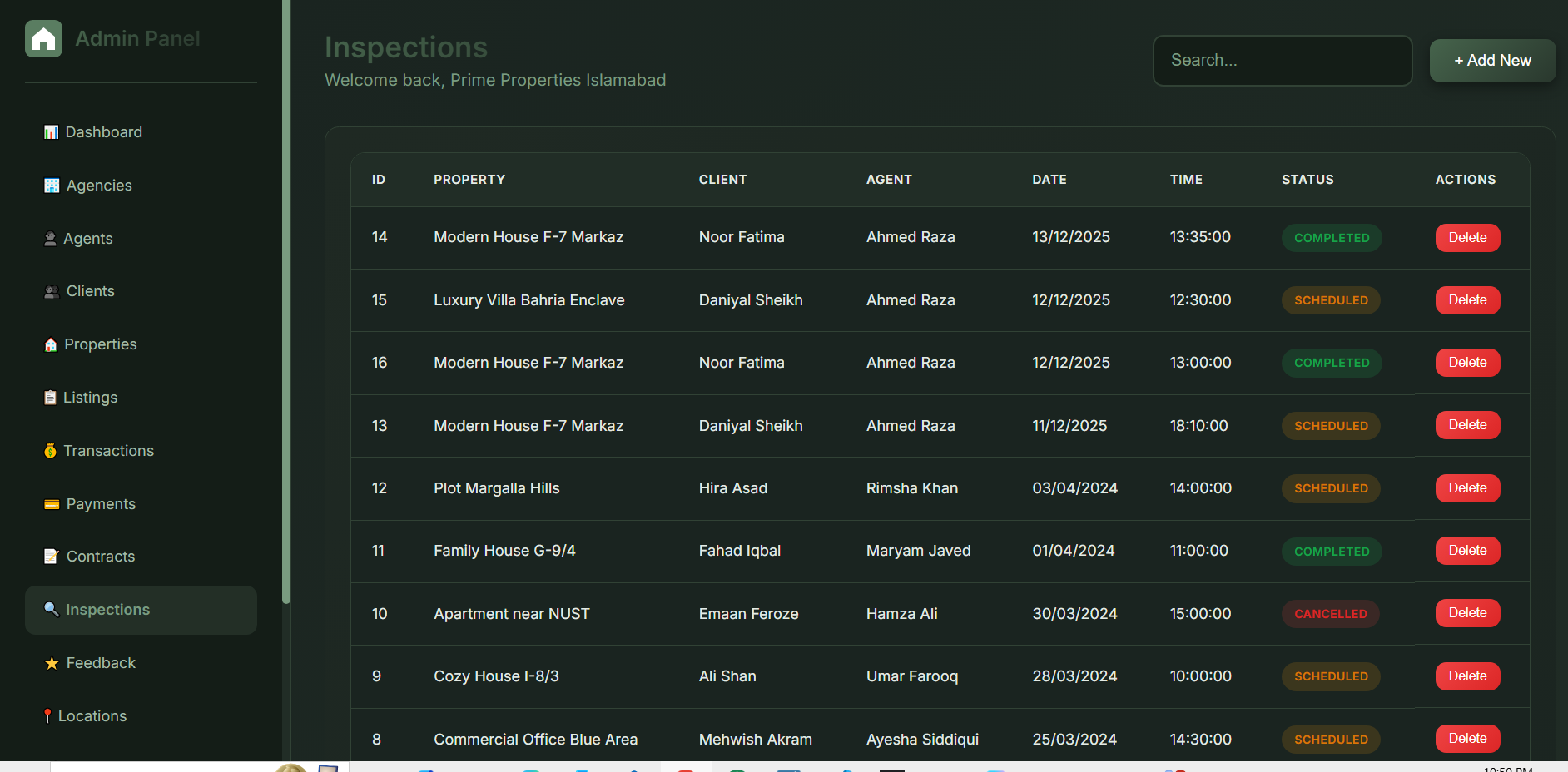
AI-generated content may be incorrect.**

* + 1. **Inspection updated successfully on client panel.**

**A screenshot of a computer

AI-generated content may be incorrect.**

* + 1. **Admin can see all the inspections.**

****

**Conclusion**

The Real Estate Management System successfully digitizes the core operations of a real estate business. By integrating property management, CRM (Customer Relationship Management), and financial tracking into a single solution, it reduces administrative overhead and improves transparency between the agency, its agents, and its clients.