Software Engineering Architectural Document

Real Estate Database Management System

Rahul Pandith - PES1UG22CS462 Rakshit Girish - PES1UG22CS465

INTRODUCTION

Purpose

To efficiently store, manage, and retrieve property-related data, such as listings, customer information, transactions, and agent details. It aims to streamline real estate operations by automating tasks like property search, lead management, and tracking sales, while ensuring data integrity, accessibility, and security for stakeholders, including agents, buyers, and sellers.

Scope

The real estate database management system project involves storing and managing property listings, user profiles (buyers, sellers, agents, administrators), and transaction records. It provides search and filtering options based on various property criteria, tracks sales and payments, and generates reports on market trends and performance.
The system implements role-based access control for different user types and ensures data security for sensitive information throughout the process.

INTRODUCTION

Definitions , Acronyms and Abbreviations

SQL: Structured Query Language for managing databases.
CRUD: Create, Read, Update, Delete - basic operations of persistent storage.
UX: User Experience - overall experience of users interacting with the system.
API: Application Programming Interface - rules for software communication.

References

Architectural Representation

The architectural representation of a Real Estate DBMS includes

1. **Presentation Layer :** User interface for property browsing, account management, and feedback submission.
2. **Application Layer :** Handles core business logic, user roles, and interactions between the front-end and database.
3. **Database Layer :** Stores all user, property, transaction, and location data with relationships between tables.
4. **Security Layer :** Ensures data protection through user authentication, role-based access, and encryption.
5. **Reporting & Analytics :** Monitors and analyses property views, transactions, and user activity for insights.

Architectural Goals and Constraints

Goals

The architectural goals of a Real Estate DBMS project are

- **Property Management :** Efficiently manage property listings, including details like type, price, and location.
- **User and Role Management** : Handle different user roles (admins, agents, users) with varying permissions and actions.
- **Transaction Tracking** : Record and monitor property transactions, including payment status and user details.
- **Feedback and Communication** : Collect user feedback and manage inquiries through contact forms for system improvement.
- **Wishlist and View Tracking** : Allows users to save properties to a wishlist and track property views for engagement insights.
- **Security and Access Control** : Ensure data integrity and security through authentication, password encryption, and role-based access.
- **Reporting and Analytics** : Provide insights into user behaviour , property popularity, and transaction trends for better decision-making.

Architectural Goals and Constraints

Constraints

1. **Scalability Limits** : Performance issues as data volume increases.
2. **Query Complexity** : Potential latency from complex searches with multiple filters.
3. **Role-Based Access** : Users (agents, regular users) and admins have different permissions, restricting actions based on roles.
4. **Limited Reporting Capabilities :** The current system lacks built-in advanced reporting and analytics, requiring external tools for in-depth insights.
5. **Resource Constraints** : Limitations on hardware, bandwidth, and memory affecting performance.
6. **Third-Party Integration**: Dependencies on external APIs impacting system design and reliability.

Use-Case View

Architecturally-Significant Use Cases

Key architecturally-significant use cases of a Real Estate DBMS project include:

1. **Property Listing Management :** Agents and admins manage property listings, requiring secure CRUD operations and data validation.
2. **User Authentication and Role Management :** Secure login and role-based access control for different user types (admin, agent, buyer).
3. **Transaction Processing :** Manages property purchases and payment tracking, ensuring data integrity and secure transactions.
4. **Property Search and View Tracking :** Efficient property searching and tracking of user views for performance and data insights.
5. **Reporting and Analytics :** Generates reports on transactions, property views, and user activity for insights and decision-making.

Architecture Overview

Frontend

- **User Interface :** Built with HTML, CSS, and JavaScript frameworks (e.g., React) for a responsive experience, enabling property browsing, user registration, and feedback submission.

Backend

- **Server-side Logic :**

  - Utilises frameworks like PHP or Node.js to handle business logic, process requests, and implement role-based access control for different user types.
- **Database Layer :**

  - MySQL for storing user, property, and transaction data, ensuring data integrity through relational tables and foreign keys.
- **Security :**

  - Implements role-based access control and data encryption.

Integration

- **API Integration :**

  - Exposes functionalities for property search, user interactions, and third-party service integrations (e.g., payment processing).

Subsystem Layering

1. **Presentation Layer :**
   - Interfaces for property listings, user registration, authentication, and feedback submission, designed for optimal user experience.
2. **Application Layer :**
   - Handles core functionalities such as user management, property transactions, and role-based access control, ensuring secure interactions and data processing.
3. **Database Layer :**
   - MySQL database for structured data storage, encompassing tables for users, properties, transactions, and feedback, with relationships enforced through foreign keys.
4. **Integration Layer :**
   - Exposes RESTful APIs for communication between the front end and back end, facilitating operations like property searches and payment processing.
5. **Security Layer :**
   - Implements user authentication via encrypted credentials and enforces role-based access to protect sensitive operations and data.

Process View

Processes

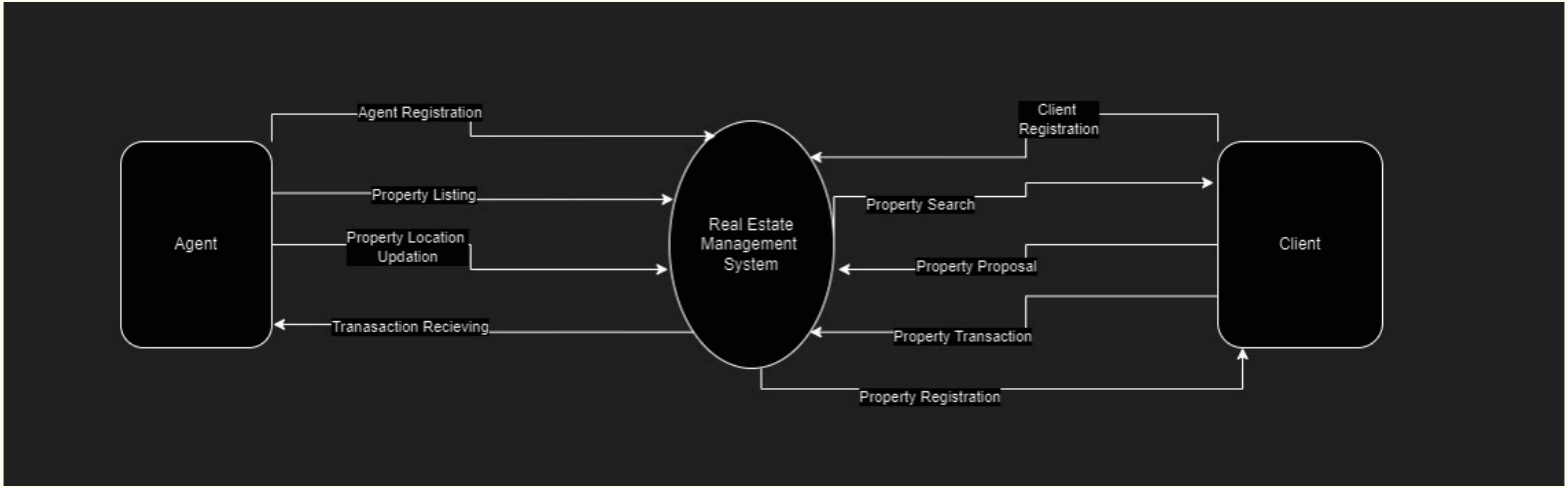
1. **User Registration and Authentication**
    - Secure user registration and login with data validation.
2. **Property Listing Management**
    - Create, update, and delete property listings by agents and admins.
3. **Property Search**
    - Enable users to search and filter properties based on criteria.
4. **Transaction Processing**
    - Handle offers and complete transactions, updating statuses accordingly.
5. **Reporting and Analytics**
    - Generate reports on market trends and performance metrics.

Deployment View
External Components

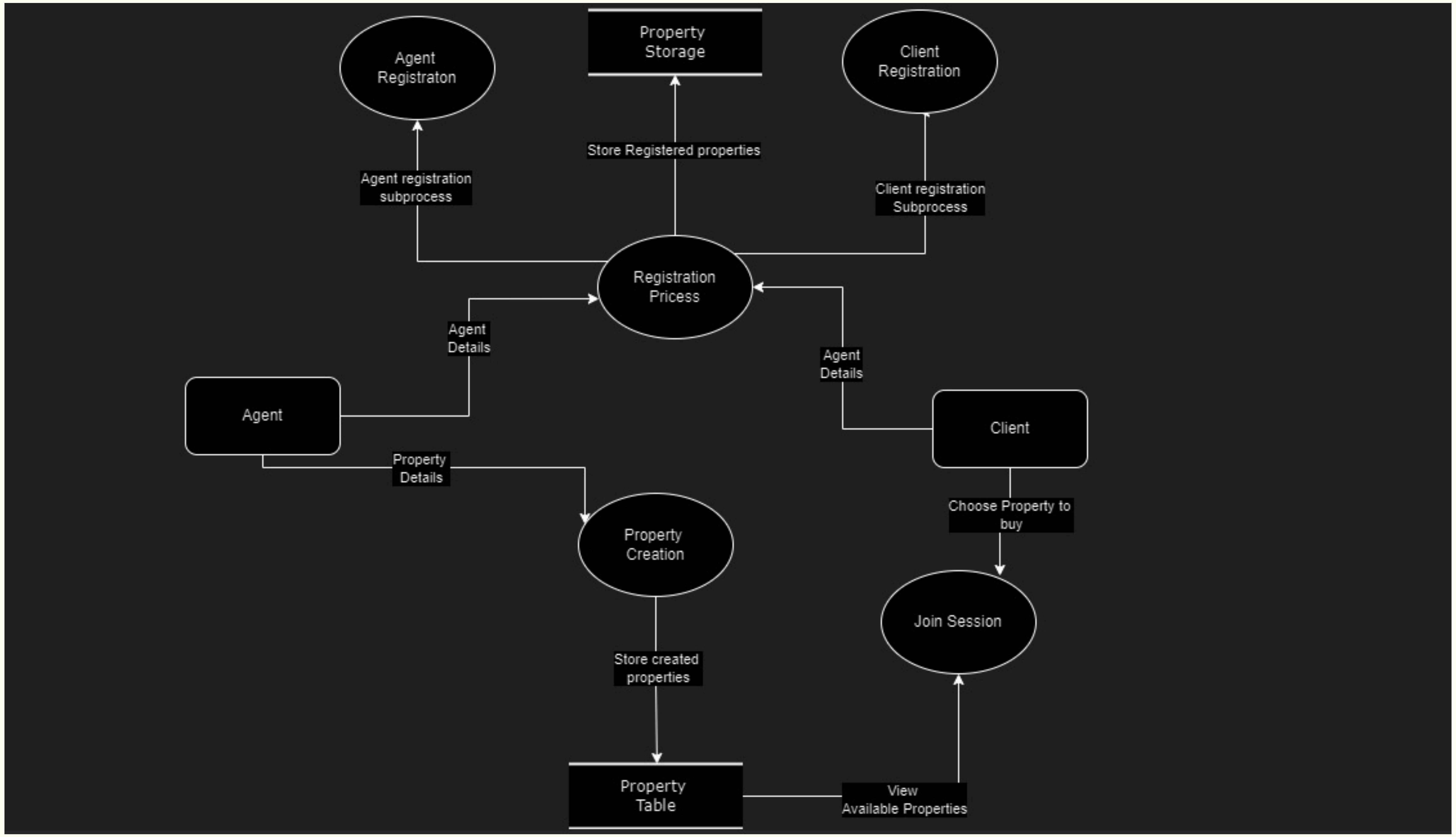
1. **User Devices**
    - Web Browsers : Access via desktops and laptops.
    - Mobile Devices : Interact through mobile apps.
2. **Web Server**
    - Application Hosting : Hosts frontend and backend components (e.g., Apache, Nginx).
3. **Application Server**
    - Backend Processing : Manages business logic and API requests (e.g., Node.js, Java).
4. **Database Server**
    - Data Storage : Hosts relational database (e.g., MySQL, PostgreSQL) with security measures.
5. **Third-Party Services**
    - Payment Gateways : Facilitates secure transaction processing.

Dataflow Diagram

Level 1

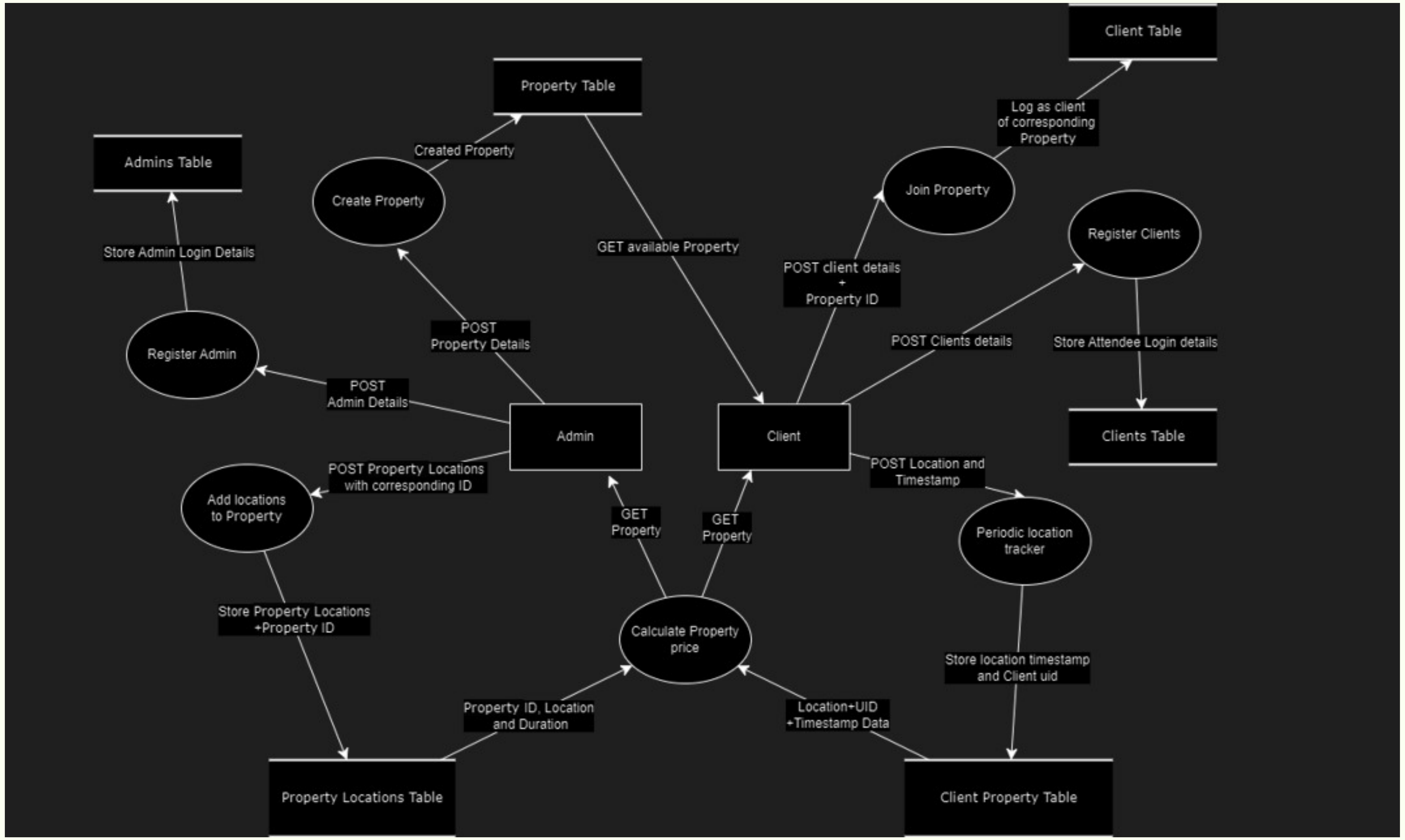


Dataflow Diagram
Level 2



Dataflow Diagram

Level 3


Client Table
Property Table
Log as client of corresponding Property
Admins Table
Created Property
Join Property
Create Property
Register Clients
Store Admin Login Details
GET available Property
POST client details + Property ID
POST Property Details
POST Clients details
Store Attendee Login details
Register Admin
POST Admin Details
Admin
Client
Clients Table
POST Property Locations with corresponding ID
POST Location and Timestamp
Add locations to Property
GET Property
GET Property
Periodic location tracker
Store Property Locations +Property ID
Calculate Property price
Store location timestamp and Client uid
Property ID, Location and Duration
Location+UID +Timestamp Data
Property Locations Table
Client Property Table

]

Performance and Quality

Performance Considerations

1. **Scalability**
   - Implement both horizontal and vertical scaling to accommodate user growth.
2. **Database Optimisation**
   - Use indexing and caching to enhance query performance.
3. **Frontend Performance**
   - Utilise a Content Delivery Network (CDN) and minify assets for faster load times.
4. **Asynchronous Processing**
   - Employ background tasks for non-critical operations to maintain responsiveness.

Performance and Quality

Quality Attributes

1. **Performance**
   - Fast loading times and scalability to handle increased user loads.
2. **Security**
   - Strong data protection measures, including encryption and secure authentication.
3. **Usability**
   - Intuitive interface design for easy navigation and accessibility.
4. **Reliability**
   - High availability and robust error handling to maintain system functionality.
5. **Maintainability**
   - Well-documented code and monitoring for quick issue resolution.

Results

The Real Estate DBMS project improved user experience with faster searches and streamlined property management, leading to higher satisfaction. It ensured secure transactions through robust security measures and demonstrated scalability to handle increased traffic. The system's reporting features provided valuable market insights, and successful integration with third-party services enhanced overall functionality.

Conclusion

In conclusion, the Real Estate DBMS project successfully delivered a user-friendly platform that enhances property management and ensures secure transactions. Its scalability allows it to handle increasing user demand, while integrated reporting features provide valuable insights for decision-making. Overall, the project meets user needs effectively and positions the business for future growth.