Propex – AI-Assisted Real Estate Platform

MUNIB KHAN
01-135221-111

MUHAMMAD FAIZAN
01-135221-106

Supervisor: Saira Hameed

Bachelor of Science in Information Technology

Department of Computer Science
Bahria University, Islamabad

December 2025

Certificate

We accept the work contained in the report titled "Propex – AI-Assisted Real Estate Platform", written by Mr. Munib Khan AND Mr. Muhammad Faizan as a confirmation to the required standard for the partial fulfillment of the degree of Bachelor of Science in Information Technology.

Approved by . . . :

Supervisor: Saira Hameed

_____

Internal Examiner:

_____

External Examiner:

_____

Project Coordinator: Dr. Kabeer Ahmed Bhatti

_____

Head of the Department: Dr. Khurram Ehsan

_____

Abstract

In the case of Pakistan the real estate market is very much dependent on manual valuation the agents experience and different levels of market knowledge which are not always consistent. Due to this lack of clarity buyers and sellers get perplexed about the price thereby making it difficult for them to take right decision. To counter this problem an AI driven web portal with a stylish easy to use interface is the project to which accurate property price predictions based on data are given as a solution. The system operates on a twofold backend infrastructure where computational work as well as I/O processes are separately taken care of for superior performance. A key factor of the platform is a Price Prediction Model that utilizes the XGBoost Regressor which is trained on local real estate data to produce accurate automated price estimates and thus helps users to have a clearer understanding of the market value and lowers the reliance on subjective manual evaluations. The application utilizes React.js for the mobile friendly frontend, a dedicated Flask service for quick ML inference and a Node.js/Express backend for administering user authentication API routing and database operations. PostgreSQL is the primary database to ensure that the data is managed in a secure and reliable manner. Tests performed indicate that the XGBoost model has a reliable performance since it has a R2 score of 0.87 on the test dataset. The system gives speedy predictions seamless interaction and a simplified property evaluation process which is why Propex is regarded as a viable digital transformation driver in the real estate sector of Pakistan.

Acknowledgment

I would like to express my deepest gratitude to my project supervisor, , for her continuous guidance, motivation, and constructive feedback throughout the development of this Final Year Project. Her support not only shaped the direction of this work but also strengthened my understanding of practical software development and research.

I am also thankful to the faculty members of the Information Technology Department at Bahria University for creating an encouraging academic environment and providing the essential resources required to complete this project successfully.

I sincerely appreciate the global open-source community whose contributions made this work possible. Tools and technologies such as React.js, Flask, Node.js/Express, PostgreSQL, scikit-learn, and XGBoost played a vital role in building and deploying the system.

Lastly, I extend my heartfelt thanks to my family and friends for their patience, encouragement, and unwavering support throughout this academic journey. Their belief in me remained a constant source of strength.

MUNIB KHAN and MUHAMMAD FAIZAN
Islamabad, Pakistan

December 2025

Contents

List of Figures

List of Tables

Acronyms and Abbreviations

Acronym	Description
AI	Artificial Intelligence
ML	Machine Learning
API	Application Programming Interface
UI	User Interface
GUI	Graphical User Interface
JWT	JSON Web Token
XGBoost	Extreme Gradient Boosting
ERD	Entity Relationship Diagram
CRUD	Create, Read, Update, and Delete
SQL	Structured Query Language
JSON	JavaScript Object Notation
MAE	Mean Absolute Error
RMSE	Root Mean Square Error
HTTP	Hypertext Transfer Protocol

Chapter 1

Introduction

This chapter provides a foundational overview of the Propex project, outlining the motivation behind its development, the specific challenges it seeks to address in the Pakistani real estate market, and the technical objectives established to achieve a modern, AI-assisted solution.

1.1 Project Overview

The real estate sector is one of the largest and most active industries in Pakistan, playing a pivotal role in economic development and urban expansion. Despite its significance, the industry relies heavily on traditional practices, including manual price estimation, agent-driven negotiations, and limited access to verified market data. Buyers and sellers often base decisions on personal experience or informal market insights, leading to uncertainty, inconsistent pricing, and a lack of trust in the process.

While Pakistan is undergoing a digital transformation across various sectors, the real estate industry remains only partially digitized. The majority of existing platforms serve merely as listing directories, presenting static information without analytical depth. Consequently, users continue to face challenges in accurately valuing properties or understanding price variations across different locations [12].

With the emergence of data-driven technologies and Machine Learning (ML), there is a significant opportunity to modernize this industry. Large volumes of property data can be leveraged to build predictive models that identify meaningful correlations between features such as location, area size, and market trends. This approach enables the generation of

objective and reliable price insights, often utilizing advanced similarity measures and content analysis to bridge the gap between user needs and available data [2, 11].

Propex is a novel online application that employs artificial intelligence to assist users in estimating property prices. The system aims to minimize uncertainty and facilitate informed decision-making by integrating a clean user interface with a secure, high-performance database structure.

1.2 Problem Description

Despite the existence of numerous property websites in Pakistan, most lack advanced functionality and serve primarily as digital noticeboards where users view listings. These platforms are often devoid of analytical tools, relying on manually entered prices that may not reflect actual market value. Many of these existing systems rely on basic information retrieval techniques designed for static data rather than dynamic analytical processing [3, 4].

The key issues currently facing the industry include:

- **Inaccurate or Subjective Price Estimation:** Most online portals display prices determined solely by sellers or agents without validation. These figures often diverge significantly from current sales data or neighborhood trends.
- **Limited Use of Data and Analytics:** There is a distinct lack of automated tools capable of analyzing property attributes to generate estimated values.
- **Lack of Technical Optimization:** Many existing systems rely on monolithic or outdated architectures, resulting in limited flexibility and an inability to handle efficient ML computations.
- **Manual Research Burden:** Buyers and sellers are compelled to spend excessive time manually checking listings. This process is time-consuming and prone to misinformation.

These challenges highlight the critical need for a platform that utilizes modern technology to deliver automated, trustworthy, and rapid price predictions.

1.3 Project Objectives

The primary objective of the Propex platform is to offer a transparent, data-driven property price estimation tool. The specific technical and functional objectives are as follows:

- **To develop an automated price prediction model** using the XGBoost regression algorithm, trained on local real estate data to ensure market relevance.
- **To build a responsive, user-friendly interface** using React.js [5], allowing users to intuitively input property details and receive real-time price estimates.
- **To implement a scalable dual-backend architecture**, where Flask [6] handles machine learning inference and Node.js [7] manages core API logic and authentication.
- **To ensure secure and structured data management** through PostgreSQL [8], enabling the efficient storage and retrieval of listings and user information.
- **To provide a streamlined user experience**, focusing on speed, reliability, and accessibility during the price prediction process.

1.4 Project Scope

The following sub-sections delineate the boundaries of the Propex platform, ensuring clarity regarding the system's capabilities and distinguishing between current functionalities and future development goals.

1.4.1 Included Features

The scope of this project encompasses the design, development, and deployment of a web-based application with the following core functionalities:

- **Responsive User Interface:** A dynamic architecture developed with React.js, ensuring cross-platform compatibility across desktop, tablet, and mobile browsers.
- **AI-Powered Price Estimation:** Integration of an ML inference engine capable of predicting prices based on inputs such as location and area.
- **Robust Security Architecture:** Implementation of secure authentication protocols including Bcrypt and JSON Web Tokens (JWT).

- **Administrative Control:** A dedicated Admin Dashboard allowing administrators to moderate listings and manage user accounts efficiently.

1.4.2 Excluded Features & Limitations

To maintain project feasibility within the allotted timeline, the following aspects are explicitly excluded from the current scope:

- **Native Mobile Development:** The project is strictly a web application; native Android (.apk) or iOS (.ipa) applications are not included.
- **Financial Transaction Integration:** The platform does not support payment gateway integration or online booking at this stage.
- **Synchronous Communication:** Real-time chat features are out of scope; communication is limited to static contact details.

1.4.3 Design and Implementation Constraints

The development of the system is subject to the following technical strictures:

- **Architectural Pattern:** The system must adhere to a component-based architecture for the frontend and a RESTful API architecture for backend communication.
- **Technology Stack:** The solution is constrained to the MERN stack with Python utilized exclusively for the Machine Learning microservice.

1.4.4 Assumptions and Dependencies

The successful implementation of the project relies on the continuous availability of third-party libraries and the assumption that the training dataset accurately reflects current market trends.

Chapter 2

Literature Review

This chapter explores the existing technological landscape of the real estate industry, comparing international standards with the current state of the Pakistani market. Furthermore, it examines the theoretical foundations of machine learning algorithms for price prediction and the architectural justifications for the chosen dual-backend system.

2.1 Existing Real Estate Technology

Real estate technology has undergone a significant transformation in developed countries, with platforms like Zillow, Redfin, and Realtor.com utilizing automated valuation models (AVMs) to ascertain property prices. These platforms rely heavily on massive datasets comprising past sales records, community statistics, and proficient machine learning techniques to deliver users real-time and data-backed insights [12].

In Pakistan, the majority of online property platforms, including Zameen.com and Graana, primarily function as listing services. These sites allow users to search for available properties but rarely offer sophisticated analytical tools such as automated price predictions or market insights. Frequently, the prices indicated on these platforms are entered manually by sellers or agents without verification, rendering them subjective and often inconsistent [12]. Consequently, both buyers and sellers experience a state of uncertainty and struggle to make decisions with confidence.

The absence of predictive analytics in the local market highlights the need for systems that not only display properties but also apply data-backed valuation methods. This gap represents the primary motivation for the development of Propex.

2.2 Machine Learning for Price Prediction

Machine Learning (ML) technology has transformed the landscape of real estate valuation by revealing complex relationships between property characteristics and market prices that are difficult to detect manually. Among various techniques, Linear Regression, Random Forest, and Gradient Boosting have been popular. However, tree boosting methods such as XGBoost have achieved particular recognition due to their excellent performance on structured tabular datasets [1].

XGBoost (Extreme Gradient Boosting) assembles a group of decision trees by optimizing the bias-variance trade-off [1]. It utilizes regularization, handles missing data efficiently, and provides outstanding performance even when features are of mixed types [13], such as:

- Location information
- Number of rooms
- Plot size and covered area
- Property type (house, flat, commercial)
- Amenities and structural attributes

For categorical features, One Hot Encoding and other preprocessing approaches are utilized to convert textual representations into numerical formats. This enables the model to process the data effectively and identify significant patterns during training [14]. The key advantages of XGBoost—such as managing non-linearities, preventing overfitting, and fast inference—make it a highly suitable choice for predicting prices in the real estate domain [1]. Its widespread global adoption suggests it is well-suited for modeling the real estate data of Pakistan as well.

2.3 Data Challenges in Real Estate (Pakistan Context)

Unlike countries with standardized real estate databases, Pakistan faces several data-related challenges that impact the accuracy of automated tools [12]. These challenges require rigorous data handling protocols:

- **Lack of Structured Data:** Data concerning listings is fragmented across different websites and agents, making the maintenance of consistent datasets difficult.
- **Inconsistent Feature Reporting:** Crucial details, such as the number of bedrooms, exact area measurements, and renovation conditions, are often omitted or incomplete in listings.
- **Unverified Prices:** Prices are usually seller-driven and may not reflect actual market value, complicating the model training process.
- **Location Complexity:** Discrepancies often arise due to inconsistent neighborhood borders, street naming conventions, and geographic coordinates.

Data must be rigorously preprocessed, cleaned, and engineered before any predictive model can be trained [14]. Literature indicates that the primary driver of reliable ML performance is high-quality data rather than the complexity of the algorithm itself.

2.4 Architectural Rationale

Contemporary web applications are increasingly adopting distributed architectures to enhance performance, scalability, and manageability. Separating resource-intensive processes from user interaction layers is a standard practice in AI-based platforms that integrate ML services with conventional web APIs.

Propex adopts a **dual backend architecture** consisting of:

- **Flask (Python) Backend:** Handles ML model loading, preprocessing, and inference. The Python ecosystem makes it ideal for implementing predictive analytics [6, 13].
- **Node.js/Express Backend:** Manages authentication, HTTP routing, CRUD operations, and communication with the PostgreSQL database [7, 8]. Its asynchronous nature allows it to efficiently handle a high volume of I/O requests.

This separation of workloads ensures that user interaction tasks are not delayed by heavy ML computations. Furthermore, it allows for independent scaling; for instance, the ML service can be deployed on servers optimized for CPU/GPU, while the API server manages concurrent user traffic [9]. This architectural approach is strongly supported by academic literature and is a proven strategy for scalable AI-driven web systems.

Chapter 3

Requirement Specifications

This chapter delineates the functional and non-functional requirements of the Propex platform. These requirements ensure the system meets user expectations, performs reliably through a secure infrastructure, and delivers accurate, data-driven price predictions based on local market insights [12].

3.1 Functional Requirements

Functional requirements define the essential features and capabilities the system must provide. To ensure clarity during development and testing, each requirement is assigned a unique identifier (FR-ID).

3.1.1 User Management and Authentication

The following requirements outline the security protocols and account management features necessary to protect user data and maintain individualized sessions.

FR-1 **User Registration:** The system shall allow new users to create an account by providing a valid name, email address, and secure password.

FR-2 **Secure Login:** Registered users shall be able to log in to the platform. The system must authenticate credentials against the stored records.

FR-3 **Password Encryption:** The system shall utilize bcrypt hashing to encrypt passwords before persisting them in the database.

FR-4 Session Management: The system shall issue JSON Web Tokens (JWT) upon successful login to maintain secure, stateless user sessions.

3.1.2 Property Discovery and Interaction

To facilitate an efficient search experience, the platform provides tools for browsing and filtering real estate listings. These functionalities are built upon established information retrieval principles to ensure relevant data is accessible to the user [3, 4].

FR-5 Property Listing Display: The system shall retrieve real estate listings from the database and display them in a responsive grid layout.

FR-6 Advanced Filtering: Users shall be able to filter property listings based on specific criteria, including location, property type, and area size.

FR-7 Property Details View: Users shall be able to select a listing to view comprehensive details, such as price, number of rooms, and available amenities.

3.1.3 Admin Dashboard & Property Management

Administrators require a specialized interface to maintain the accuracy of the platform and monitor the lifecycle of listings within the ecosystem.

FR-8 Dashboard Overview: The system shall provide an administrator dashboard displaying key metrics, such as the total number of registered users and properties listed.

FR-9 Property CRUD Operations: The system shall enable administrators to Create, Read, Update, and Delete property records to ensure data accuracy.

FR-10 User Moderation: The system shall allow administrators to perform moderation actions, such as account suspension, if necessary.

3.1.4 System Feedback & Validation

To ensure the integrity of the predictive model and the database, the system enforces strict validation on all user and administrator inputs.

FR-11 **Input Validation:** The system shall enforce data validation on the frontend to ensure numerical inputs are positive integers before submission.

FR-12 **Error Handling:** The system shall display descriptive error messages to the user in cases of failed prediction requests or network timeouts.

3.2 Use Case Table

The interaction between the primary actors and the functional requirements is summarized in **Table 3.1**. This mapping identifies how different users utilize the platform's features to achieve specific goals.

Table 3.1: System Use Cases and Associated Requirements

UC ID	Use Case Name	Actor	Ref. Functional Req.
UC-01	Register Account	User	FR-1, FR-3
UC-02	Authenticate/Login	User / Admin	FR-2, FR-4
UC-03	Browse Listings	User	FR-5
UC-04	Filter Properties	User	FR-6
UC-05	Predict Price	User	FR-8, FR-9, FR-10
UC-06	Add New Listing	Admin	FR-11
UC-07	Edit/Delete Listing	Admin	FR-11
UC-08	View Dashboard	Admin	FR-12

3.3 Non-Functional Requirements

Non-functional requirements define the quality attributes, performance standards, and operational constraints that determine the overall user experience and system reliability.

- **Performance:** The system shall process ML inference requests and return a price prediction in less than 100 ms under normal load.
- **Security:** All sensitive credentials (API keys, DB URIs) must be managed securely via environment variables (`.env`).
- **Usability:** The UI shall adhere to modern clean-design principles to ensure full responsiveness across devices.
- **Data Integrity:** PostgreSQL constraints must be enforced to prevent inconsistent or duplicate entries in the database.

3.4 Use Case Diagram

The visual representation of actor interactions is provided in **Figure 3.1**. This diagram illustrates the relationship between the User and Admin actors and the primary system functionalities, including searching properties and requesting AI-driven predictions.

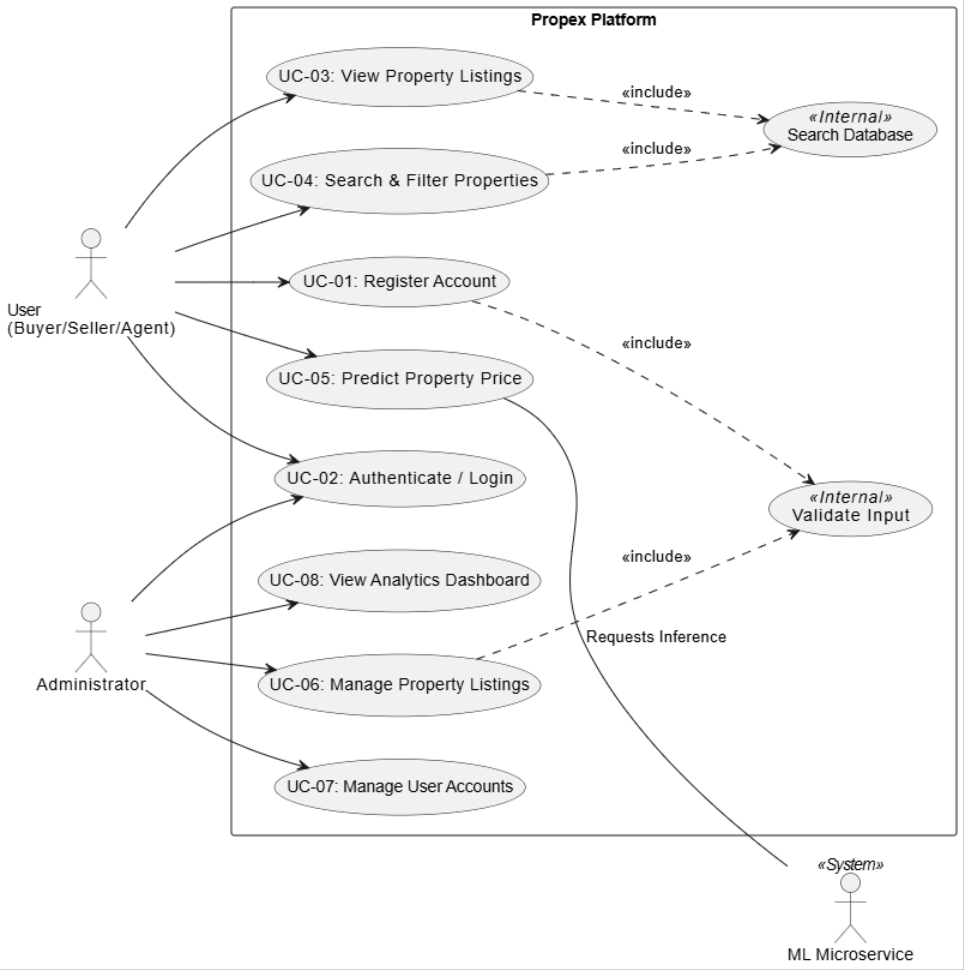

Figure 3.1: Use Case Diagram for Propex Platform

Chapter 4

Design

The current chapter presents the architectural choices, the structures of the components, UML diagrams, and database design which together form the core of the Propex platform. The main objective is to develop an elastic modular and efficient setting in which machine learning and web technologies interact flawlessly to solve property valuation challenges.

4.1 System Architecture

The system's architecture is a modular **dual backend architecture** allowing the separation of machine learning computation from traditional web API processing. This separation not only ensures but also boosts total performance, reduces server burden, and allows components to scale according to specific needs [7, 6].

4.1.1 Frontend (React.js)

The frontend layer is responsible for providing a seamless and responsive user experience. It utilizes a component-based structure to handle complex UI states and asynchronous data fetching from the backend services [5].

4.1.1.1 Authentication Interface

The Login Page serves as the secure entry point for the application. It incorporates strict input validation to ensure email formatting and password length requirements are met

before submission. Upon successful authentication, the system retrieves a JWT token to manage the user's session state. The visual layout for the secure authentication portal is illustrated in **Figure 4.1**.

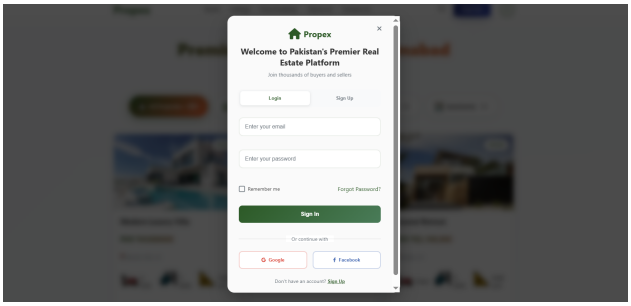

Figure 4.1: Login Page Interface

4.1.1.2 Landing Page

The Home Page acts as the central hub of the Propex platform. It features a responsive navigation bar that adapts to different screen sizes, providing users with quick access to property listings and the prediction tool. The interface design, shown in **Figure 4.2**, emphasizes a clean, user-friendly layout to guide new visitors toward key functionalities immediately.

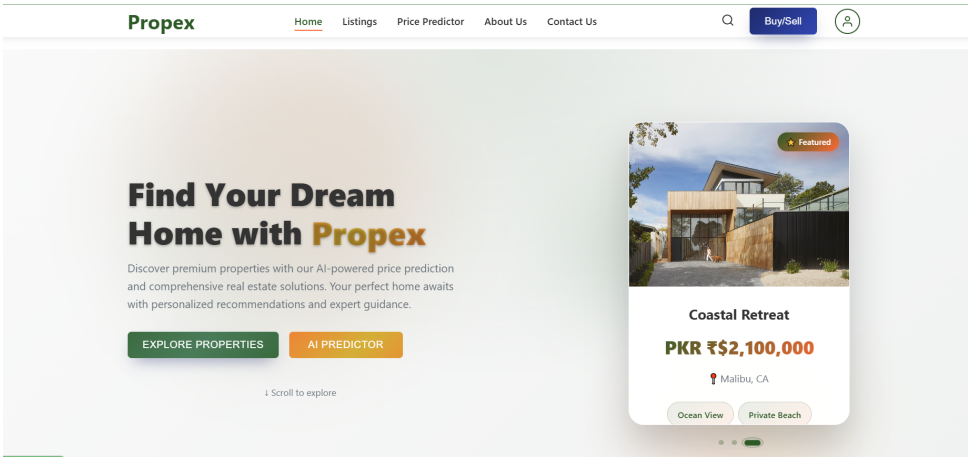

Figure 4.2: Home Page Interface of Propex

4.1.1.3 Prediction Module

The Price Prediction feature is implemented as a modal overlay, allowing users to interact with the AI tool without navigating away from their current page. This interface, depicted

in **Figure 4.3**, collects essential property attributes—such as location, area size (in Marla), and the number of bedrooms.

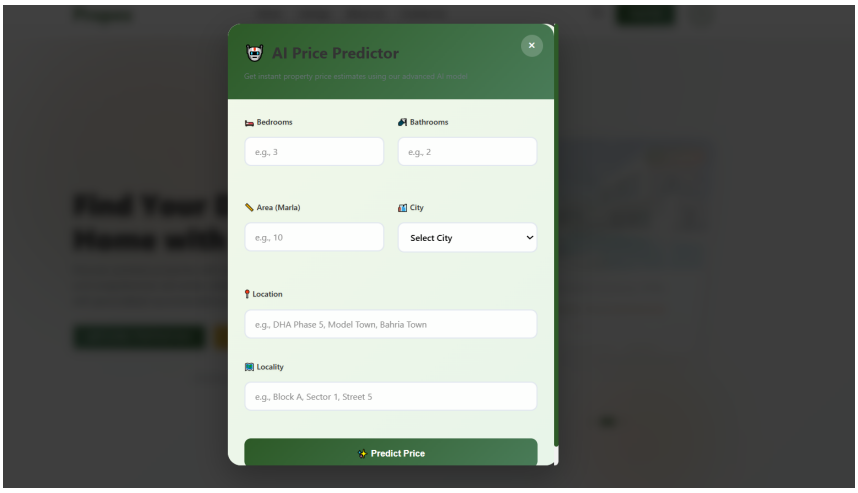

Figure 4.3: Price Prediction Modal Interface

Once the prediction is triggered, the data flow is managed through the following specialized channels:

- **Inter-Service Communication:** The Node.js server functions as the primary API Gateway. It forwards prediction-specific requests to the Flask service via internal RESTful API calls, ensuring that heavy ML computational tasks do not block the main application event loop.
- **Database Integration:** The persistence layer is managed exclusively by the Node.js service, which utilizes an ORM/ODM to enforce schema constraints and perform secure transactions for user and property data [7].

4.1.2 Backend Data Flow

The interaction between the client and the dual-backend architecture follows a synchronous pipeline designed for efficiency and data integrity. This flow ensures that user requests are processed by the AI model according to modern retrieval standards [3].

The process is defined by the following stages:

1. **Request Initiation:** The client (React) sends a JSON payload to the Node.js `/predict` endpoint.

2. **Validation & Forwarding:** The Node.js server validates the JWT and forwards data to the Flask ML service.
3. **Inference Processing:** The Flask service passes data through the XGBoost model to generate a price estimate.
4. **Response Aggregation:** The predicted value is returned to the client through the Node.js gateway.

4.2 Database Design

The database is implemented via **PostgreSQL** [8], ensuring data integrity and fast access. The design prioritizes relational consistency to support the advanced filtering requirements of the platform.

4.2.1 Entity Relationship Diagram (ERD)

Data integrity is maintained through foreign keys among the Users, Admins, Properties, and Prediction Logs entities. The logical structure of the database and the relationships between these entities are visualized in the Entity Relationship Diagram provided in **Figure 4.4**.

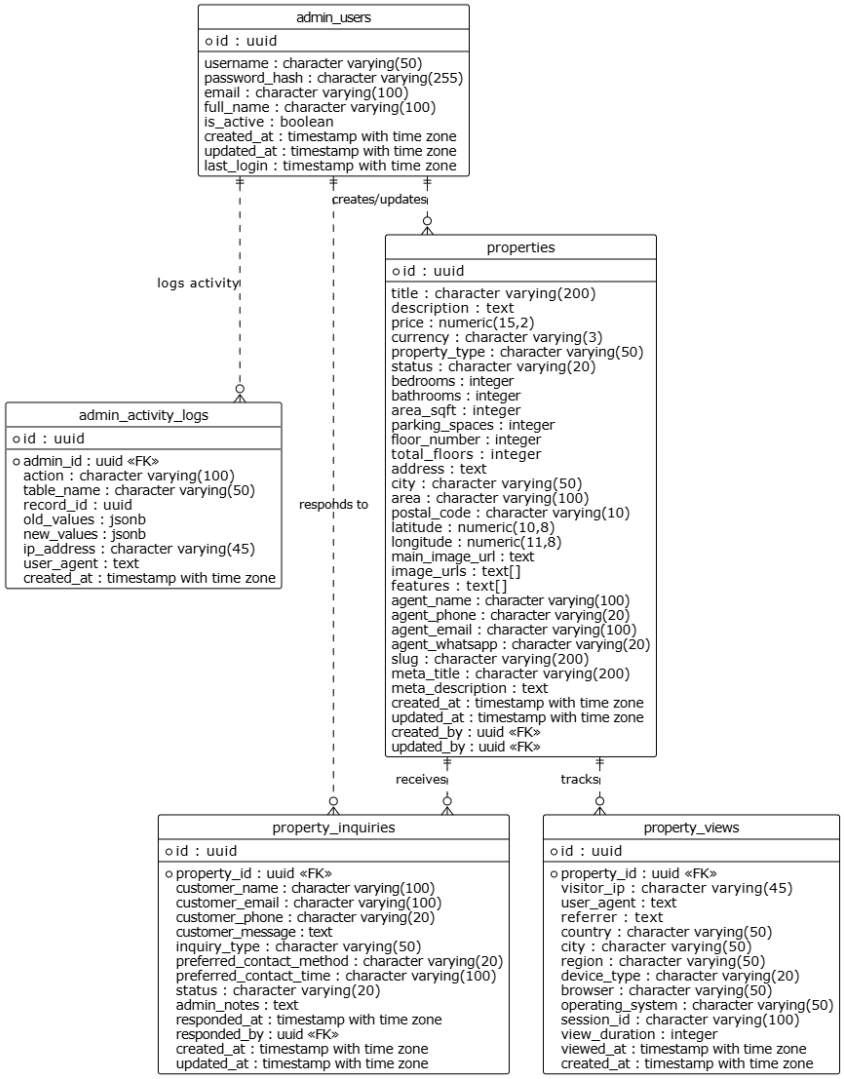

Figure 4.4: Entity–Relationship Diagram (ERD) for Propex Database

4.3 Security and Reliability Design

To ensure the platform remains robust against external threats, the architecture incorporates several layers of protection, following best practices for secure information systems [4]:

- **Authentication Security:** Utilizes JWT and Role-Based Access Control (RBAC).
- **Data Privacy:** Sensitive keys are managed via `.env` files and passwords are hashed using `bcrypt`.
- **Integrity Protection:** All inputs undergo frontend and backend validation to prevent SQL Injection and XSS attacks.

Chapter 5

System Implementation

The implementation phase involves translating the design specifications into a functional software product. This chapter narrates how the Propex platform was constructed, detailing the instruments utilized and the methodology for combining the machine learning model with the web-based ecosystem. The execution focuses on creating a seamless connection amongst the React.js frontend, Node.js backend, Flask ML service, and PostgreSQL database.

5.1 Tools and Technologies Used

Modern and well-established technologies were selected to ensure the system meets high standards of performance, scalability, and maintainability. These selections allow for a decoupled architecture where each component can be optimized independently.

The comprehensive technology stack used for the development of Propex is summarized in **Table 5.1** below.

Table 5.1: Technologies Used in Propex Implementation

Machine Learning Model	XGBoost Regressor [1]
ML Backend	Flask 2.x (Python) [6]
API Backend	Node.js / Express.js [7]
Frontend Framework	React.js 18 [5]
Database	PostgreSQL [8]
Data / ML Libraries	pandas, numpy, scikit-learn [13], joblib [10]
Other Tools	JWT, bcrypt, pgAdmin, VS Code, Postman

5.2 Frontend Implementation

The frontend layer provides the interface through which users interact with the AI-driven valuation tools and property listings. It is designed to be lightweight and responsive to ensure accessibility across different network conditions.

The frontend part of the app was constructed using React.js 18 and a component-based architecture, which made it very easy to change and reuse parts of the application. The interface can respond to all types of screen sizes, thus delivering a continuous and delightful user experience on mobile as well as desktop devices.

5.2.1 Major Frontend Implementation Features

The development of the user interface was guided by modularity, ensuring that functional elements such as search bars and prediction forms remain consistent throughout the application.

Major frontend implementation features include:

- **Reusable Components:** Navbar, Footer, Forms, Buttons, and Cards were developed as reusable UI components.
- **Property Listing Pages:** The process of data presentation happens in these components as the information of the property flows from the Node.js backend to the appropriate representation.
- **Price Prediction Form / Modal:** The future price of the property is predicted based on the main features provided by the user. The modal handles validation of user inputs, unit conversions (e.g., Marla $\rightarrow$ Square Feet), and JSON transmission.
- **State Management:** Using React Hooks (useState, useEffect), component states and APIs are handled efficiently.
- **Routing:** React Router is responsible for moving users between the Home, Listings, Login, Admin Panel, and Price Prediction pages.

The Login Page implementation procedure, illustrated in **Figure 5.1**, utilizes a secured state management system for the protection of user credentials. In the end, the front end sends a POST request to the auth API and keeps the JWT returned in local storage for session persistence.

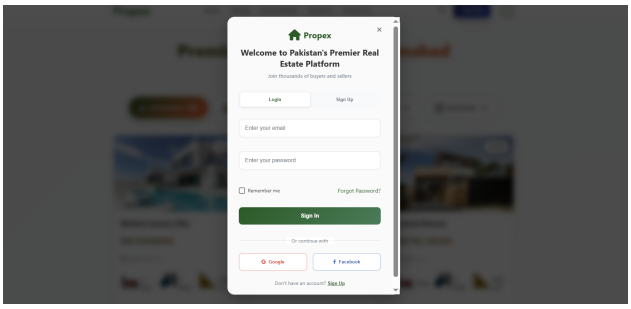

Figure 5.1: Login Page

The Signup Page, depicted in **Figure 5.2**, is an example of a live form validating function made in React. It assures that the users will not send any information to the backend registration point until they enter an email format and a password that meet the validity criteria.

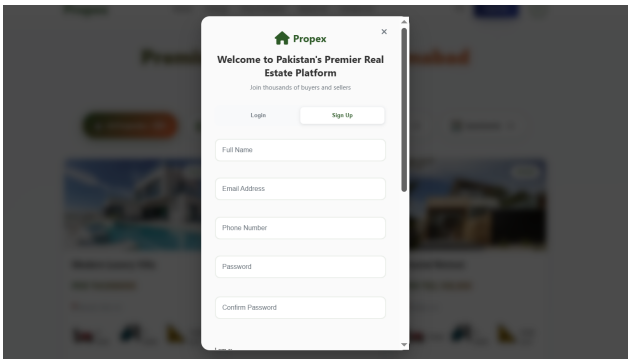

Figure 5.2: Signup Page

The Home Page acts as the primary gateway featuring a lively hero section and fast access connections. The layout shown in **Figure 5.3** utilizes responsive Flexbox layouts, ensuring a consistent display on both mobile and desktop devices.

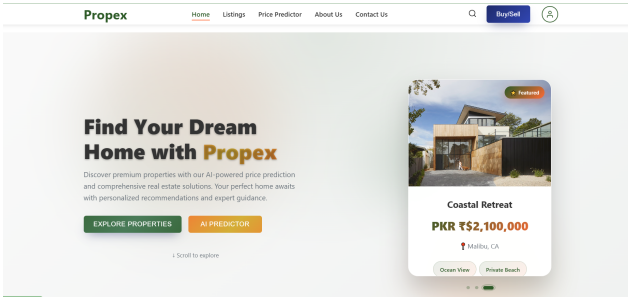

Figure 5.3: Home Page

The Property Listing Page, visualized in **Figure 5.4**, employs a dynamic rendering approach

where property cards are created by traversing the data retrieved from the backend API. Pagination and filtering techniques are implemented to handle larger datasets in an efficient manner.

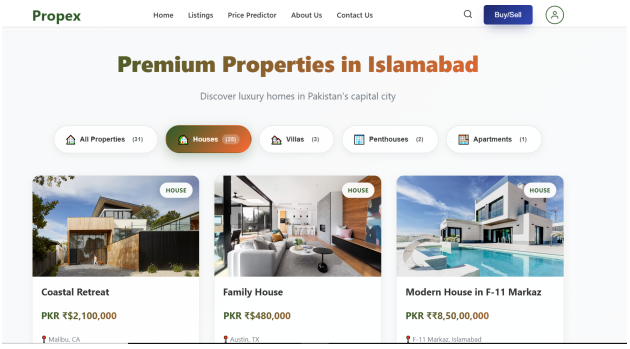

Figure 5.4: Property Listing Page

The Price Prediction Modal, which is a crucial feature shown in **Figure 5.5**, interacts with the user and takes the input regarding the property specifications. After that, it communicates with the Flask ML service in an asynchronous manner and shows the predicted price on the same page without refreshing it.

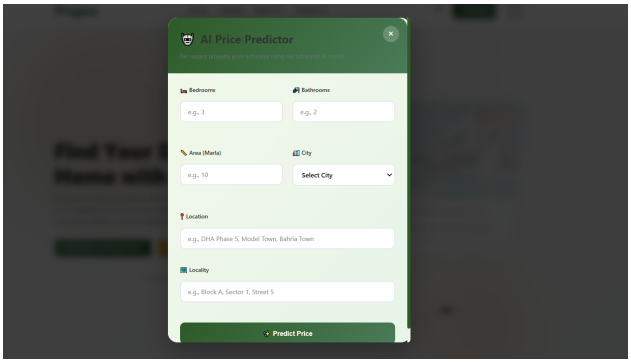

Figure 5.5: Price Prediction Model

5.3 Backend Implementation

The backend infrastructure is responsible for the core business logic and the execution of the machine learning model. To optimize performance, the backend is divided into two independent services to ensure scalability and maintain clear separation of responsibilities.

5.3.1 Node.js / Express Backend

The Node.js environment handles the standard application traffic, including user accounts and property data management. Node.js back-end manages the principal application logic and communicates with the PostgreSQL database [7].

The main features are:

- **User Authentication:** Passwords are hashed with bcrypt and session control is done through JWT tokens.
- **CRUD Operations:** Admins are allowed to create, edit, view and delete the records of the properties.
- **Input Validation:** All incoming requests undergo sanitization and validation to stop injection attacks.
- **API Routing:** Offers endpoints for login, registration, property listing, and connecting to the ML backend.

5.3.2 Flask Machine Learning Backend

Machine learning inference is the only task that the Flask backend is responsible for. It stays simple and fast and is thus able to quickly handle many prediction requests [6].

Main implementation steps are:

- **Model Loading:** The XGBoost model is loaded for the first time when the server starts up and is done so by `joblib` [10]. Storing the model in RAM helps in cutting down inference time.
- **Prediction Endpoint `/api/predict`:** It takes in JSON format data and performs the necessary preprocessing so that it is in the form expected by the model.
- **Inference Pipeline:** This involves checking inputs, applying encoding for categorical variables, and sending processed data to the model.

5.4 Machine Learning Deployment

The final stage of implementation involves deploying the trained model into the production environment. This requires careful consideration of data normalization to ensure consistency between the training and inference stages.

The XGBoost regression model [1] is the only machine learning element used in Propex. The whole process includes:

- **Data Preprocessing:** The dataset went through a thorough cleaning process where imputation was done for missing values and encoding was performed on categorical variables [14].
- **Feature Engineering:** Features like price per square foot and location mappings were added to improve the model's accuracy.
- **Model Training:** hyperparameters were altered so that the R^2 was increased while the error metrics were decreased.
- **Model Exporting:** The model was saved in a `.joblib` file [10] for rapid access in the Flask service.
- **Deployment:** The model file (`real_estate_price_predictor.joblib`) is loaded during Flask startup once and then used for all incoming predictions.

Chapter 6

System Testing and Evaluation

System testing ensures that the Propex platform operates seamlessly, maintains reliability, and performs according to specifications under real-world conditions. This chapter details the testing methodology, validation scenarios, performance metrics, and user feedback. The primary objective of the testing phase was not only to verify technical correctness but also to ensure the system is intuitive, responsive, and user-friendly.

6.1 Testing Approach

To ensure the integrity of the platform, a multi-faceted testing strategy was adopted, covering everything from individual code blocks to end-to-end user journeys.

Multiple complementary testing methodologies were employed to achieve robust quality assurance:

- **Unit Testing:** PyTest and Jest were utilized to test individual functions within the Flask and Node.js backends, ensuring that each isolated component operates correctly [13].
- **Integration Testing:** Validated the communication between distinct modules (Frontend $\rightarrow$ Node.js $\rightarrow$ Flask $\rightarrow$ Database) to ensure data flows without errors.
- **Performance Testing:** Focused on measuring API response times, particularly for the price prediction endpoint, to guarantee stable results under concurrent load.
- **User Interface Testing:** Manual testing was conducted on desktop and mobile devices to verify navigation, form validation, and responsiveness.

6.2 Key Test Scenarios

The following core scenarios were defined and tested to ensure the stability of the platform's primary features and the accuracy of the underlying AI model.

The specific scenarios examined during the testing process included:

- **User Login and Registration:** Verification of input validation, error messaging, JWT generation, and secure redirection.
- **Price Prediction Request:** Confirmation that the frontend correctly submits property details and receives a valid prediction from the Flask backend.
- **Admin Operations:** Validation of CRUD actions for property listings, including adding, updating, and deleting records.
- **Database Interactions:** Ensuring that PostgreSQL correctly stores, updates, and retrieves all data attributes without loss or corruption.

All scenarios were executed successfully over several iterative rounds of testing.

6.3 Detailed Test Cases

This section documents the specific validation scenarios executed to verify system performance. This documentation covers critical test cases ranging from frontend usability checks to backend security protocols and machine learning inference accuracy.

6.3.1 GUI Testing – Price Prediction Module

This test case validates the core functionality of the AI Price Predictor. It verifies the successful transmission of property attributes from the React frontend to the Flask backend and confirms the low-latency rendering of the predicted price. The details of this test case are summarized in **Table 6.1**.

Table 6.1: GUI Testing - Price Prediction Module

Test Case ID	TC006
Module Name	Price Prediction Tool - ML Inference
Description	Validate that the user can input property features and receive a mathematically accurate price estimate from the Flask ML backend.
Pre-Requisites	User logged in. Flask ML service and Node.js API running. XGBoost model loaded.
Environment	Windows 11, Chrome/Firefox, Node.js, Flask, PostgreSQL
Test Steps	1. Open Price Prediction page. 2. Enter valid inputs (City, Area, Bedrooms, Property Type). 3. Click "Predict Price". 4. Check Flask returns a prediction.
Test Inputs	City: Islamabad, Area: 10 Marla, Bedrooms: 3, Type: House
Expected Results	Valid PKR price returned; Response time < 100ms.
Actual Results	Predicted Price: PKR 4,850,000; Response Time: 92ms
Status	Pass

6.3.2 Security Testing – Role-Based Access Control

This test verifies system security by enforcing Role-Based Access Control (RBAC). It ensures that standard users are strictly prohibited from accessing administrative dashboards. The results of the unauthorized access attempts are documented in **Table 6.2**.

Table 6.2: Security Testing - Role-Based Access Control

Test Case ID	STC003
Module Name	Authentication and Authorization Security
Description	Verify that a regular user cannot access admin dashboard or perform admin CRUD operations.
Pre-Requisites	Regular user account and admin account must exist.
Environment	Windows 11, Chrome/Firefox, Node.js/Express (JWT Auth)
Test Steps	1. Log in as regular user. 2. Attempt to open "/admin_dashboard". 3. Check API response codes.
Expected Results	403 Forbidden for admin dashboard; API returns 401/403.
Actual Results	Dashboard redirected to user panel; POST returned 403 Forbidden.
Status	Pass

6.3.3 Authentication Test Case – User Registration

This test validates the user registration workflow. It confirms the successful creation of accounts, secure hashing of passwords, and the prevention of duplicate email registrations. **Table 6.3** outlines the validation steps for the user registration process.

Table 6.3: Authentication Test Case - User Registration

Test Case ID	TC001
Module Name	User Authentication - Registration
Description	Verify that a new user can successfully register and that duplicate emails are handled correctly.
Test Steps	1. Open Sign-Up page. 2. Enter valid Name, Email and Password. 3. Click “Register”. 4. Attempt registration again with same email.
Test Inputs	Name: Ali Khan, Email: ali.khan@example.com, Password: SecurePass123!
Expected Results	User redirected to login page; Duplicate email shows “User already exists”.
Actual Results	User created; Hash verified; Duplicate prevented.
Status	Pass

6.3.4 Admin Dashboard Test Case – Add Property Listing

This validation focuses on administrative CRUD operations. It ensures that administrators can successfully add new property listings via the dashboard. Details of this test case are provided in **Table 6.4**.

Table 6.4: Admin Dashboard Test Case - Add Property Listing

Test Case ID	TC010
Module Name	Admin Dashboard - Add Property Listing
Description	Verify that Admin can add new property listings and they appear correctly on frontend.
Test Steps	1. Open Admin Dashboard. 2. Go to “Add Property”. 3. Enter Title, Price, Area, City. 4. Submit listing.
Test Inputs	Title: Modern Villa F-7, Price: 55,000,000, Area: 1 Kanal, City: Islamabad
Expected Results	Success message displayed; Entry added to Properties table.
Actual Results	Listing added; Property ID #105 created.
Status	Pass

6.3.5 Performance Test Case – System Load

This test examines system stability under high-load conditions, requiring the API to respond consistently during concurrent requests. The metrics for system stability under load are presented in **Table 6.5**.

Table 6.5: Performance Test Case - System Load

Test Case ID	PTC001
Module Name	System Performance - Load Testing
Description	Verify the system handles 50 concurrent prediction requests.
Environment	AWS EC2 / Local server
Test Steps	1. Configure 50 concurrent users. 2. Send POST requests to "/api/predict". 3. Record response times and error rates.
Expected Results	Avg response < 200ms; 0% error rate.
Actual Results	Avg response: 145ms; Error rate: 0%; Throughput: 45 req/sec.
Status	Pass

6.3.6 Usability Test Case – Navigation

This test evaluates User Experience (UX) and navigational efficiency by measuring how intuitively a user can access the Price Prediction feature. The outcome of the usability assessment is shown in **Table 6.6**.

Table 6.6: Usability Test Case - Interface Navigation

Test Case ID	UTC001
Module Name	Usability - Interface Navigation
Description	Evaluate ease of finding Price Prediction tool from homepage.
Success Metrics	Time < 10s; Clicks < 3.
Actual Results	Time: 4s; Clicks: 1.
Status	Pass

6.3.7 Compatibility Test Case – Cross-Device

This test ensures design responsiveness and cross-device compatibility, verifying that the layout adjusts correctly for mobile users. **Table 6.7** summarizes the findings from the compatibility testing.

Table 6.7: Compatibility Test Case - Cross-Device

Test Case ID	CTC001
Module Name	Compatibility - Mobile Responsiveness
Description	Verify interface works correctly on mobile devices.
Environment	iPhone 14 (Safari), Samsung S22 (Chrome)
Test Steps	1. Open website on mobile. 2. Navigate to Listings. 3. Open Predict Price modal.
Expected Results	No horizontal scroll; Modal fits screen; Buttons are tappable.
Actual Results	Responsive layout worked; Modal accessible.
Status	Pass

6.4 Model Performance Evaluation

The XGBoost algorithm [1] was evaluated using standard regression metrics to assess its accuracy in predicting property prices. These results, discussed in conjunction with local market trends [12], demonstrate the model's reliability for the local context.

The core performance metrics recorded are as follows:

- **R^2 Score:** 0.87 (Indicates strong correlation between predicted and actual values).
- **Mean Absolute Error (MAE):** 125,000 PKR.
- **Root Mean Squared Error (RMSE):** 185,000 PKR.

The evaluation results and the comparison of these metrics are visually represented in the chart provided in **Figure 6.1**.

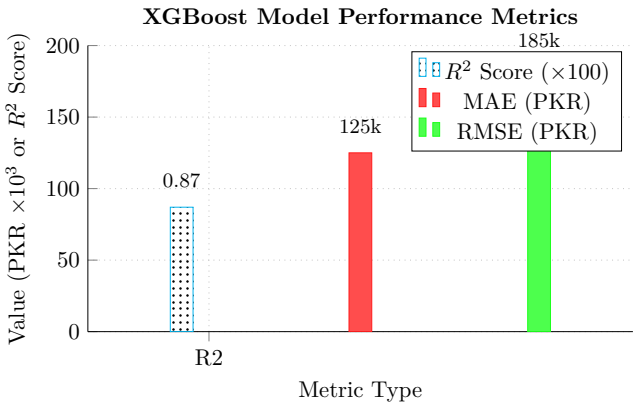

Figure 6.1: Model Performance Evaluation Chart

These results demonstrate that the model generalizes effectively, providing reliable approximations of real market prices.

6.5 User Feedback

To assess real-world usability, a cohort of property seekers and students tested the system. Their feedback was collected to identify both strengths and areas for future iterative development.

The key findings from the feedback sessions include:

- **Visual Design:** The interface was described as clean, modern, and intuitive.

- **Utility:** Users found the price prediction feature highly valuable for gaining quick insights.
- **Performance:** The simplicity of input forms and rapid response times were widely appreciated.
- **Suggestions:** Users recommended adding visual cues, such as icons and input examples, for future iterations.

Chapter 7

Conclusion and Future Enhancements

This final chapter summarizes the significant milestones achieved during the development of the Propex platform. It reflects on how the integrated technologies address the core problems of the real estate sector and outlines a strategic roadmap for future feature expansions to further enhance the system's utility.

7.1 Conclusion

The Propex platform was developed with the goal of introducing modern, data-driven tools into Pakistan's real estate market. Throughout this project, significant progress was made in solving long-standing challenges such as unclear property valuation and the lack of transparent digital tools for buyers and sellers [12].

The system successfully integrates a reliable XGBoost-based price prediction model [1], a clean and responsive React.js interface, and a dual-backend architecture built with Node.js and Flask. This mixture guarantees rapidity and precision in result delivery. The users experienced a high degree of interactivity with the platform, from the submission of property data to the immediate reception of market-based predicted values.

In the end, Propex illustrates that artificial intelligence-based solutions can offer significant advantages to the real estate industry by increasing the accessibility of data, cutting down on uncertainties, and providing better information for decision-making. Apart from that, the project lays down a strong base for further growth, sophisticated functionalities, and real-world application in a developing digital economy.

7.2 Future Enhancements

While the current platform meets all defined functional requirements, there are several avenues to raise and fortify the system's potential as the dataset grows. The following activities are proposed for future iterations:

- **Improved Prediction Model:** The best way to move forward in the direction of higher prediction accuracy would be to augment the dataset with more features like schools, roads, amenities, and market trends. Applying advanced data mining techniques can reveal deeper correlations [14].
- **Automated Retraining Pipeline:** The introduction of an automatic retraining procedure would mean the ML model could constantly refresh itself whenever fresh property data comes in, keeping it relevant to fluctuating market prices.
- **Interactive Visual Dashboards:** The creation of analytics dashboards for administrators (showing price trends and city-wise statistics) will help make the decision-making process smoother and more transparent.
- **Recommendation Engine:** Future versions could implement content-based recommendation systems to suggest properties to users based on their search history and price preferences [2].
- **Mobile Application:** Launching native mobile applications for both Android and iOS will bring the platform closer and more convenient for users while they are on the move.
- **Enhanced Security Layers:** The incorporation of two-factor authentication (2FA) and more sophisticated monitoring systems will safeguard user information more strongly than ever before.

The Propex prediction tool would be turned into an entirely intelligent real estate ecosystem through these improvements, which would then be able to offer value to everyone including buyers, sellers, and property professionals. The platform has a strong potential for widespread use and practical impact in the real world with constant refining and data updates.

References

[1] Tianqi Chen and Carlos Guestrin, *XGBoost: A Scalable Tree Boosting System*, Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pp. 785–794, 2016, doi:10.1145/2939672.2939785. Cited on pp. 6, 17, 22, 28, and 30.

[2] Pasquale Lops, Marco De Gemmis, and Giovanni Semeraro, *Content-based Recommender Systems: State of the Art and Trends*, Recommender Systems Handbook, Springer, pp. 73–105, 2011. Cited on pp. 2 and 31.

[3] Amit Singhal, *Modern Information Retrieval: A Brief Overview*, IEEE Data Engineering Bulletin, vol. 24, no. 4, pp. 35–43, 2001. Cited on pp. 2, 9, and 14.

[4] Gerard Salton and Michael J. McGill, *Introduction to Modern Information Retrieval*, McGraw-Hill, 1983. Cited on pp. 2, 9, and 16.

[5] Meta Platforms Inc., *React Documentation*, 2024, https://react.dev Cited on pp. 3, 12, and 17.

[6] Pallets Projects, *Flask Documentation*, 2024, https://flask.palletsprojects.com Cited on pp. 3, 7, 12, 17, and 21.

[7] Node.js Foundation, *Node.js Documentation*, 2024, https://nodejs.org/en/docs Cited on pp. 3, 7, 12, 14, 17, and 21.

[8] PostgreSQL Global Development Group, *PostgreSQL 16.3 Documentation*, 2024, https://www.postgresql.org/docs/ Cited on pp. 3, 7, 15, and 17.

[9] Docker Inc., *Docker Documentation*, 2024, https://docs.docker.com Cited on p. 7.

[10] Python Software Foundation, *Joblib Documentation*, 2024, https://joblib.readthedocs.io Cited on pp. 17, 21, and 22.

[11] Ankit Kumar and T. Rajan, *An Efficient Approach to Compute Cosine Similarity for Real-Time Recommendation Systems*, International Journal of Computer Applications, vol. 176, no. 10, pp. 1–6, 2020, doi:10.5120/ijca2020919732. Cited on p. 2.

[12] Debapratim Ghose and Ravi Shankar, *Artificial Intelligence in Real Estate: Opportunities and Challenges*, Real Estate Technology Journal, vol. 5, no. 2, pp. 45–58, 2021. Cited on pp. 1, 5, 6, 8, 28, and 30.

[13] Scikit-learn Developers, *Scikit-learn Documentation*, 2024, https://scikit-learn.org/stable/ Cited on pp. 6, 7, 17, and 23.

[14] Pang-Ning Tan, Michael Steinbach, Anuj Karpatne, and Vipin Kumar, *Introduction to Data Mining*, Pearson Education, 2018. Cited on pp. 6, 7, 22, and 31.