Palestine Polytechnic University

College of Information Technology and Computer Engineering

Project Title:

Machine-Learning-Based Land-Price Prediction System

Team Members:

Mohammad Alqadi | Mohammad Alamlah

Supervisor: Dr. Hashem Altamimi

إهداء

إلى والدينا

بفصاحة القلب وبكل احترام وتقدير، نتوجه إليكم برسالة ممتلئة بعمق المشاعر وارتفاع الجلال. إن ما نحمله في قلوبنا من امتنان ومودة لا يمكن وصفه بكلمات بسيطة، فأنتما الركيزة الثابتة التي بنينا عليها حياتنا، والشمعة الساطعة التي أضاءت دربنا في ظلمة الليالي..

...

إلى أصدقائنا

بكل احترام وتقدير، نرفع لكم تحية الود والاعتزاز، فأنتم أصدقاؤنا الأوفياء والرفاق المخلصون. لقد كنتم دائماً العون والسند في السراء والضراء، والصخرة الصلبة التي نستند إليها في عبور مياه الحياة العميقة. فشكرًا لكم على كل لحظة قضيناها معًا، وعلى كل دعمكم اللامحدود وتضحياتكم الجليلة.

...

شكر وتقدير

إلى أساتذتنا الكرام، نتقدم بأخلص الشكر والتقدير على الجهود الجبارة التي بذلتموها خلال سنواتنا في الدراسة. لقد كنتم قدوةً ومصدرَ إلهامٍ لنا، وساهمتم بشكل كبير في تشكيل مستقبلنا الأكاديمي والمهني.

نود أن نخصّ بالشكر **الدكتور هاشم هشام التميمي** على تفانيه وإرشاده القيم، وعلى كل العلم والمعرفة التي شاركنا بها. لقد كنتم داعمًا لنا في كل خطوة نخطوها في طريقنا التعليمي.

نشكركم على صبركم الذي لا يُضاهى واحتوائكم لنا في كل الظروف، وعلى توفير بيئة تعليمية محفِّزة ومليئة بالتشجيع. إن مساهماتكم **لن تُنسى**، وستظل خالدة في ذاكرتنا.

ونخصّ بالشكر **المخمّن العقاري قيس ادعيس** على تزويدنا ببيانات ميدانية واقعية وإرشادات مهنية أسهمت مباشرةً في بناء قاعدة البيانات واختبار النموذج.

ونتوجه بالشكر أيضًا إلى **جامعة بوليتكنك فلسطين** على توفير المرافق التعليمية المتميزة والخدمات التي ساعدتنا على تحقيق أهدافنا الأكاديمية بنجاح.

ندعو الله أن يجزيكم خير الجزاء وأن يوفقكم في كل ما تسعون إليه من خير وتطور في خدمة العلم والتعليم.

Abstract

In the era of artificial intelligence and technological advancements, the process of predicting (or estimating) land prices is still implemented using traditional methods that rely on human estimation, which makes it prone to bias and inconsistency in results. In response to these challenges, this project aims to develop an intelligent system that depends on machine learning and real-world data to estimate land prices more objectively and more accurately, and in less time compared to traditional methods. The town of Bani Na'im, located in the Hebron Governorate in Palestine, was chosen as an experimental area to apply the system because there is enough available data about its lands, and local land appraisers cooperated by providing us with this data.

The system is designed with an interactive user interface that provides a form with fields to enter land features such as area, location, political classification, and other influencing factors, to provide an immediate estimated price for the user. The regression tree algorithm was chosen for the project in its early stage due to its simplicity and efficiency in dealing with a limited amount of data, which is the case with the data currently available. The used data included both numerical and categorical features, the model was trained on this data to estimate the price based on the entered factors. The data was collected from various sources, the most important being the land appraisers from Bani Na'im as well as referring to official maps and structural plans to extract important information about the lands, such as their location, classification, shape, and price, so they can be manually entered into the system. The diverse sources helped build a realistic database, and reinforced the authenticity of the model and its relevance to the practical field. Although the available data is limited, the model was optimized to achieve a balance between precision and speed, which makes it an effective helping tool for the decisions of real estate appraisers, since they are the main users who benefit from it. This project represents the first step in automating the process of real estate valuation, and it is planned to develop it in the future using more advanced algorithms such as CatBoost and Random Forest, to keep up with the increasing volume and variety of available data.

الخلاصة

في عصر نهضة الذكاء الاصطناعي والتطور الملحوظ لا تزال عملية تخمين (أو تثمين) أسعار الأراضي تُنفّذ بأساليب تقليدية تعتمد على التقدير البشري؛ فهذا يجعلها عرضة للتحيّز والتفاوت في النتائج. استجابةً لهذه التحديات، يهدف هذا المشروع إلى تطوير نظام ذكي يعتمد على تعلم الآلة لتقدير أسعار الأراضي بموضوعية ودقة أعلى، وفي زمن أقل مقارنةً بالأساليب التقليدية، وذلك بالاعتماد على بيانات واقعية تم جمعها. وقد تم اختيار بلدة بني نعيم الواقعة في محافظة الخليل، فلسطين، كنموذج أولي لتطبيق النظام. نظراً لتوفر بيانات كافية حول أراضيها، وتعاون مخمني الأراضي من خلال تزويدنا بها. تم تصميم النظام بواجهة مستخدم تفاعلية تتيح إدخال خصائص الأرض مثل المساحة، والموقع، والتصنيف السياسي، وغيرها من العوامل المؤثرة، ليحصل المستخدم على سعر تقديري فوري. اعتمد المشروع في مرحلته الأولى خوارزمية شجرة الانحدار (Regression Tree)، نظرًا لبساطتها وقدرتها على التعامل بكفاءة مع أحجام بيانات محدودة، كما هو الحال مع البيانات المتوفرة حاليًا. تضمنت البيانات المستخدمة خصائص عددية وأخرى تصنيفية وتم تدريب النموذج عليها لتقدير السعر بناءً على العوامل المدخلة. وقد تم تحصيل هذه البيانات من مصادر متنوعة أهمها مخمنو الأراضي في بلدة بني نعيم، بجانب الرجوع إلى خرائط رسمية ومخططات هيكلية لاستخلاص معلومات تنظيمية عن الأراضي مثل موقعها وتصنيفها وشكلها، وسعرها، وذلك لإدخالها يدويًا إلى النظام. ساعد هذا التنوع في بناء قاعدة بيانات واقعية، وعزّز من موثوقية النموذج وارتباطه بالميدان العملي. ورغم محدودية البيانات المتوفرة، تم ضبط النموذج لتحقيق توازن فعّال بين الدقة وسرعة التنفيذ، مما يجعله أداة مساعدة فعالة لقرارات المخمّنين العقاريين، كونهم الفئة المستفيدة منه بشكل رئيسي. يمثّل هذا المشروع الخطوة الأولى في أتمتة عملية التثمين العقاري، ويُخطط لتطويره لاحقًا باستخدام خوارزميات أكثر تقدمًا مثل CatBoost وRandom Forest، بما يتماشى مع الازدياد في حجم وتنوع البيانات المتوفرة.

Contents

List of Tables

List of Figures

Chapter 1: Introduction

1.1 Overview

This chapter introduces the main elements of the project. It begins with the idea of the project, then goes to its importance, followed by the goals of the project, the scope and limitations, the theoretical background, and finally the chosen algorithm and alternatives.

1.2 Idea of the Project

The main idea of the project is building a web application for an intelligent system that is capable of accurately predicting the land prices in the town of Bani Na'im. In order for the system to predict accurately, it will depend on the techniques of artificial intelligence (AI) and machine learning (ML). The machine learning model will be trained by providing for it all the major factors affecting land prices, these factors include: area, distance to main roads and markets, availability of water and electricity supplies, among others. By feeding the machine learning algorithm with this data the project aims to provide a faster, more accurate, and more transparent alternative to traditional land valuation methods. The project intends to benefit the appraisers and help them make objective and data driven decisions.

1.3 Importance

Due to the frequent transactions in the area, the need for a faster and more efficient pricing method is growing, and one of the main advantages of this project's AI-powered approach is speed, while traditional/manual methods can take several hours to evaluate the price, the trained machine learning model can do it in seconds. Another need is reducing subjectivity, it is crucial to avoid the human bias in the field of land price evaluation because human bias in land valuation can shift prices by thousands of shekels. The project eliminates such bias by relying on data and algorithms alone, ensuring objective, data-driven, and transparent predictions.

1.4 Goals of the Project

The main goal of the project is to develop a machine learning model capable of accurately estimating land prices in Bani Na'im and to achieve this goal, the project has the following objectives:

1- Data collection: Gather all relevant land data like the area, location, suitability for agriculture and more.

2- Data cleaning: After collecting the raw data, data cleaning is performed, where the data's quality will be enhanced by removing duplicate and irrelevant data entries, and correcting inconsistencies.

3- Model development: Train machine learning algorithms with the cleaned data and compare them to select the most suitable algorithm in predicting the prices.

4- Model evaluation: Test the model and evaluate it by comparing the results of the model with the actual results of traditional pricing methods.

5- Tool implementation: Design a user-friendly website as a tool for the land appraisers.

The project aims to increase the efficiency and transparency of the land price estimations as well as making the process of price estimation easier for the appraisers.

1.5 Scope and Limitations

This project aims to predict land prices in the town of Bani Na'im using machine learning techniques depending on available real Bani Na'im land data. The scope of the project includes developing a predictive machine learning model that predicts land prices depending on features like:

- Location
- Political classification of the land (area A, B, or C)
- Intended land use (e.g., residential, commercial)
- Land area
- Availability of infrastructure
- Proximity to essential public services (e.g., hospitals)

The project also involves designing a user-friendly interface that allows authorized users such as land appraisers, system admins, and data scientists to use the system — everyone as allowed to.

On the other hand, the project faces many challenges that may affect the accuracy of the prediction. The accuracy depends directly on data quality and completeness in addition to the used model. The most important limitations are:

- **Limited data availability:** The collected data may be outdated or incomplete, and some land prices may not be documented.

- **Assumption of data representativeness:** This project assumes that the available data reflects typical land characteristics in Bani Na'im.
- **Geographic limitation:** The model is specifically designed for lands just in Bani Na'im.
- **Not considering all external factors:** The model does not account for sudden market shifts, and in Palestine, Palestinians are vulnerable to forced displacement at any moment, which could cause a sudden gap in land prices.
- **Limited time:** Because of the limited time that we have, our team was not able to try many machine learning algorithms to choose the best one that validated our project.

1.6 Background

1.6.1 Artificial Intelligence (AI)

Artificial Intelligence (AI) is one of the most significant fields in modern computer science. It aims to develop intelligent systems capable of performing tasks that traditionally require human intelligence, such as reasoning, decision-making, pattern recognition, and prediction. AI systems rely on processing large datasets, extracting meaningful relationships, and generating insights that enable faster, more accurate, and more objective decisions compared to traditional manual approaches. In recent years, AI applications have expanded across numerous domains, including the real estate sector, where AI contributes to producing reliable, data-driven land and property price estimations.

1.6.2 Machine Learning (ML)

Machine Learning (ML) is a core discipline within AI that focuses on constructing models capable of learning automatically from data rather than being explicitly programmed for every possible case. ML models analyze historical data, discover patterns and relationships between input features and target variables, and utilize this knowledge to make predictions on new, unseen data.

Machine learning approaches are commonly categorized into three main types:

- **Supervised Learning:** The model is trained using labeled data that includes both input features and their corresponding correct outputs. This enables the model to learn the mapping between inputs and outputs.
- **Unsupervised Learning:** The model learns from unlabeled data, aiming to uncover hidden structures, clusters, or patterns within the dataset.

- **Reinforcement Learning:** The model learns by interacting with an environment, receiving feedback in the form of rewards or penalties, and improving its performance over time.

Since the objective of this project is to predict a continuous numerical value representing land price, the most suitable approach is supervised regression learning.

1.6.3 Regression in Machine Learning

Regression techniques are used when the target variable is continuous rather than categorical. In this project, regression is employed to estimate land prices based on multiple influential features, including:

- Geographic location
- Land area
- Administrative and political classification
- Availability of services and infrastructure
- Proximity to main roads and essential facilities

Using regression enables objective, consistent, and data-driven valuation while reducing reliance on subjective human estimation, which may vary among assessors.

1.6.4 Regression Tree

A Regression Tree is one of the widely used supervised learning algorithms for predicting continuous values such as real estate and land prices. A regression tree consists of internal nodes, branches, and terminal leaf nodes.

The dataset is recursively divided into increasingly homogeneous subsets by selecting the most informative feature and an appropriate splitting threshold at each node. This process continues until specific stopping criteria are satisfied, such as:

- Reaching a maximum tree depth
- Reaching a minimum number of samples in a node
- Achieving an acceptable prediction error

Ultimately, each leaf node represents a group of lands sharing similar characteristics, and a corresponding price estimation is assigned to that group.

1.6.5 Overfitting and Pruning

Regression trees may suffer from overfitting when the tree becomes too complex, learns noise from the training data, and performs poorly on new data. To address this limitation, pruning techniques are applied.

Pre-pruning restricts tree growth through constraints such as limiting maximum depth or requiring a minimum number of samples per node. Post-pruning builds a full tree and then removes branches that do not contribute significantly to prediction performance. These methods enhance the model's generalization capability and improve the stability of predictions.

1.6.6 Reason for Choosing Regression Tree

The Regression Tree algorithm was selected for this project because it provides interpretable decision-making, supports both numerical and categorical data, performs effectively with small to medium-sized datasets, captures non-linear relationships, and matches the multi-factor nature of land price estimation. Therefore, it represents a scientifically justified and practically efficient choice for predicting land prices in Bani Na'im.

1.7 Mathematical Background

The Regression Tree model implemented in this project is mathematically supported by several fundamental principles governing node splitting, prediction generation, and model complexity control.

1.7.1 Sum of Squared Residuals (SSR)

At each node, the quality of the grouping is evaluated using the Sum of Squared Residuals (SSR), which measures how close the values are to their mean:

$$SSR = \sum_{i=1}^{n} (y_i - \bar{y})^2$$

where y_i is the actual price of sample i, $\bar{y}$ is the mean price of samples in the node, and n is the number of samples in the node. A lower SSR indicates better homogeneity and therefore a better-quality node.

1.7.2 Best Split Criterion

For each potential split, SSR is computed for the left and right subsets. The total resulting error is:

$$SSR_{total} = SSR_{left} + SSR_{right}$$

The optimal split is the one that minimizes SSR_{total}, ensuring that the resulting subsets are more homogeneous and stable.

1.7.3 Leaf Node Prediction

After the splitting process terminates, each leaf node represents a set of similar samples. The predicted value assigned to a leaf node is the mean value of all samples within it:

$$\hat{y} = \frac{1}{n} \sum_{i=1}^{n} y_i$$

Any new instance that reaches this leaf will be assigned this value as its predicted land price.

1.7.4 Model Complexity Control (Cost Complexity Pruning)

To avoid overfitting, a penalty term is introduced to balance accuracy and structural complexity:

$$R_\alpha(T) = R(T) + \alpha|T|$$

where $R(T)$ is the prediction error of the tree, $|T|$ is the number of terminal nodes, and α is a regularization parameter controlling complexity. Increasing α reduces tree size and enhances generalization capability.

1.7.5 Summary

These mathematical foundations ensure objective and optimal splitting decisions, accurate prediction generation, and an appropriate balance between accuracy and complexity. As a result, the Regression Tree model used in this project provides reliable, stable, and data-driven land price estimation.

1.8 Alternatives

There are several alternative methods for evaluating the land prices, but the most common method used in Bani Na'im is **comparative market analysis (CMA)**, which is

comparing the land to be evaluated to similar lands that were recently sold. These lands have shared attributes to be compared.

Although this method is commonly used, it is less accurate and less effective than the machine learning method, and also more complicated to justify the result because the CMA method relies heavily on the subjective judgment of experts rather than objective land statistics, which can introduce bias and inconsistency.

In contrast, this project leverages machine learning models, which can automatically learn from data and adapt to changing market and political conditions to provide faster, more accurate predictions and justifiable, transparent results.

1.9 Chosen Algorithm for the Model

We adopt the decision tree regression algorithm as the base machine learning model in this project. This choice is consistent with the theoretical and mathematical background presented in the previous sections and is motivated by the following reasons:

1. It can handle both numerical and categorical data.
2. It works well with non-linear relationships between features and the target.
3. It provides a clear, interpretable visualization of the decision-making process.
4. It performs well with small to medium-sized datasets, which matches the available data for Bani Na'im.
5. It is relatively simple to implement and easy to understand, which facilitates further development and maintenance.

1.9.1 Practical Behavior of Decision Tree Regression in This Project

In the context of this project, the decision tree regression model operates as follows:

1. The input features (such as area, location, political classification, and infrastructure availability) are used to recursively split the data into smaller regions.
2. At each split, the model selects the feature and threshold that minimize the sum of squared residuals, as described in the mathematical background.
3. Splitting continues until one of the stopping criteria is met, such as maximum depth or minimum number of samples per leaf.
4. Each terminal node (leaf) stores the average land price of the training samples that fall into that region.

5. When a new land instance is entered into the system, it is routed through the tree according to its features until it reaches a leaf, where the stored average price is returned as the predicted land value.

This behavior allows the model to capture complex relationships between land characteristics and price, while maintaining a structure that is transparent and explainable to domain experts such as land appraisers.

Chapter 2: Requirement Specifications

2.1 Overview

This chapter identifies the main users of the land pricing system and describes the role of each. It also outlines the functional and non-functional requirements that define how the system should behave, and shows how the components of the system interact with each other. It also presents visual representations such as a use-case diagram and a context diagram as well as functional requirements tables.

2.2 Actors

The system has three main actors, each with distinct responsibilities:

1- **Land Appraiser** — Enters target-land characteristics and receives an automated price prediction. Uses the result to validate their own estimate or as a data-backed estimation.

2- **Admin** — Manages user accounts and system configuration (view roles/emails, activate/deactivate, update or remove users). Maintains a safe, secure, and smooth operation of the platform.

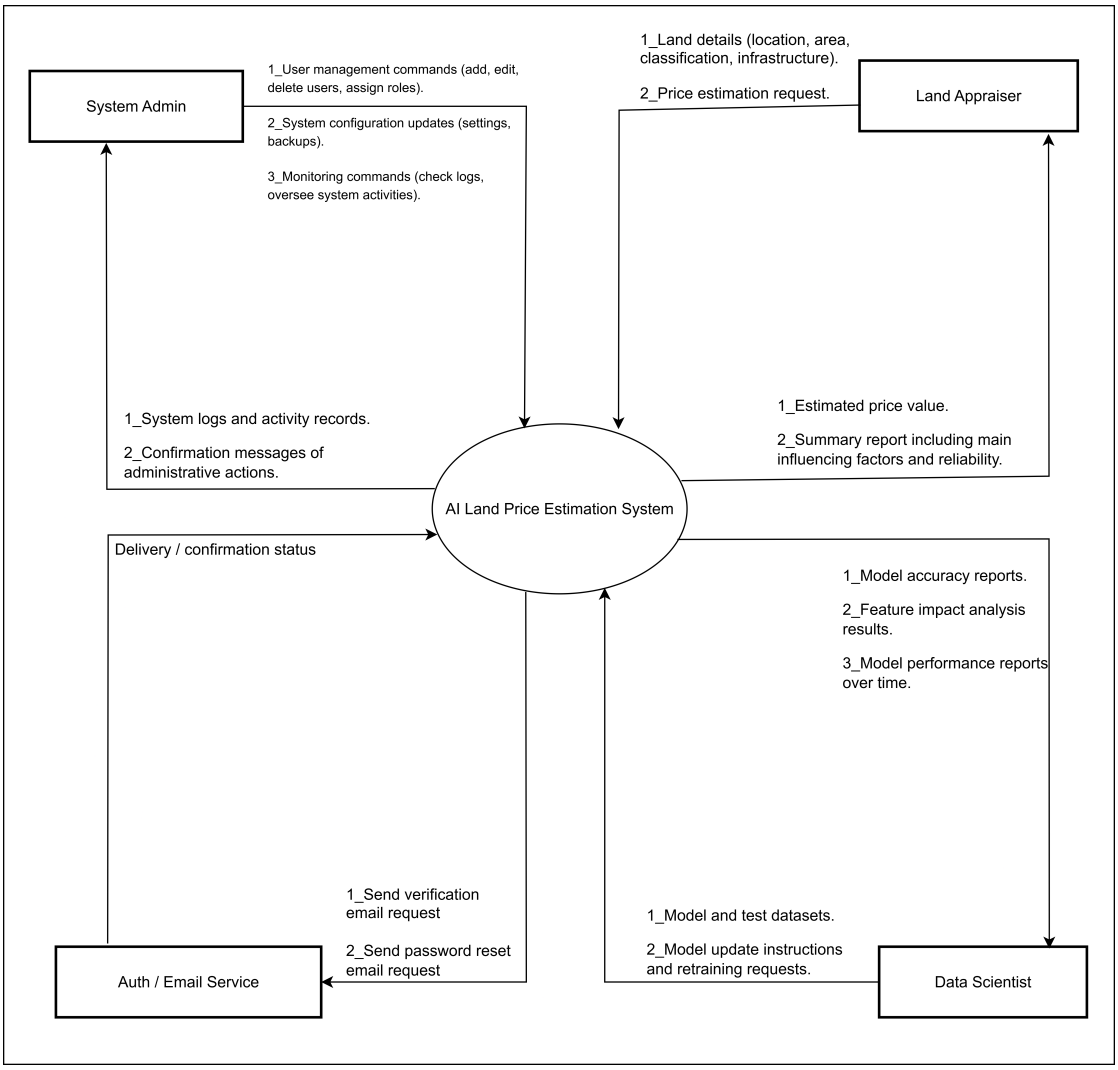
3- **Data Scientist** — Ensures model and platform quality. Prepares/curates datasets, tests and validates the ML model with real or synthetic data, monitors accuracy and performance, and suggests improvements.

Together, these actors keep the system reliable and continuously improving.

2.3 Context Diagram

Figure 2.3.1 illustrates the context diagram of the AI Land Price Estimation System, showing the main external entities and their interactions with the system. The key entities are the System Admin, Land Appraiser, Data Scientist, and Authentication/Email Service. Each entity communicates with the system through specific commands, data inputs, or reports, ensuring the overall functionality of user management, model development, account security, and land price estimation.

Figure 2.3.1 Context Diagram



2.4 Functional Requirements

2.4.1 Land Appraiser's Side

1. **User Registration and Login**
    - Appraisers must be able to register using a valid email address and password.
    - An activation code provided by the administrator is required to complete registration.
    - Once registered, appraisers can log in securely using their email and password.
    - A password reset option must be available in case appraisers forget their password.
2. **Profile Management**
    - View and edit personal information (e.g., name, email).
    - Change password from profile settings.
3. **Add a New Project**
    - Create a new project.
    - Input land details for estimation.
4. **Selecting an Old Project**
    - Select a previously created project.
    - Edit the input data and re-estimate the price.
5. **Price Estimation**
    - The system processes the entered data and displays the estimated land price.
    - The appraiser receives a summary of the estimation and the influencing factors.
6. **Project History**
    - The system saves each submitted land estimation as a separate project.
    - The appraiser can view a list of all past projects.
    - Each project shows input details, results, and the date of submission.
7. **Edit or Delete Land Inputs (Before Submission)**
    - Edit or clear the form data before submitting for estimation.

8. **Input Validation**
    - The system checks for missing or invalid entries and shows helpful error messages.
9. **Rating the Estimation Result**
    - The appraiser can rate the estimation result after it is displayed.

2.4.2 Admin's Side

1. **Login** — The admin can securely log in to the system using their credentials.
2. **Manage Users**
    - View all registered users.
    - Remove user accounts.
    - Edit user roles.
    - Activate / Deactivate accounts.
3. **Creating Admin Accounts** — Create new admin accounts when necessary to expand system management.
4. **Manage Form Data** — Manage selectable regions and update system data fields relevant to land evaluation to keep the platform consistent with current geographic and regulatory information.
5. **View System Logs** — See records of user activity and system events to monitor and diagnose issues.
6. **Manage Backups** — Save backups of system data and restore them in case of data loss or system problems.

2.4.3 Data Scientist's Side

1. **User Registration and Login**
    - Register using a valid email address and password.
    - Provide an activation code issued by the administrator to complete registration.
    - Log in securely using credentials once registered.
    - Reset password when needed.
2. **Test Model Accuracy** — Run tests using known or sample land data to evaluate model accuracy.

3. **Review Feature Impact** — View which land features (e.g., area, location) most influence the predicted price based on the model's analysis.
4. **Monitor Model Performance Over Time** — Track model performance across time and compare older versions with newer ones.
5. **Select Any Project to Analyze** — Select any existing project in the system (including those created by any land appraiser) for analysis.

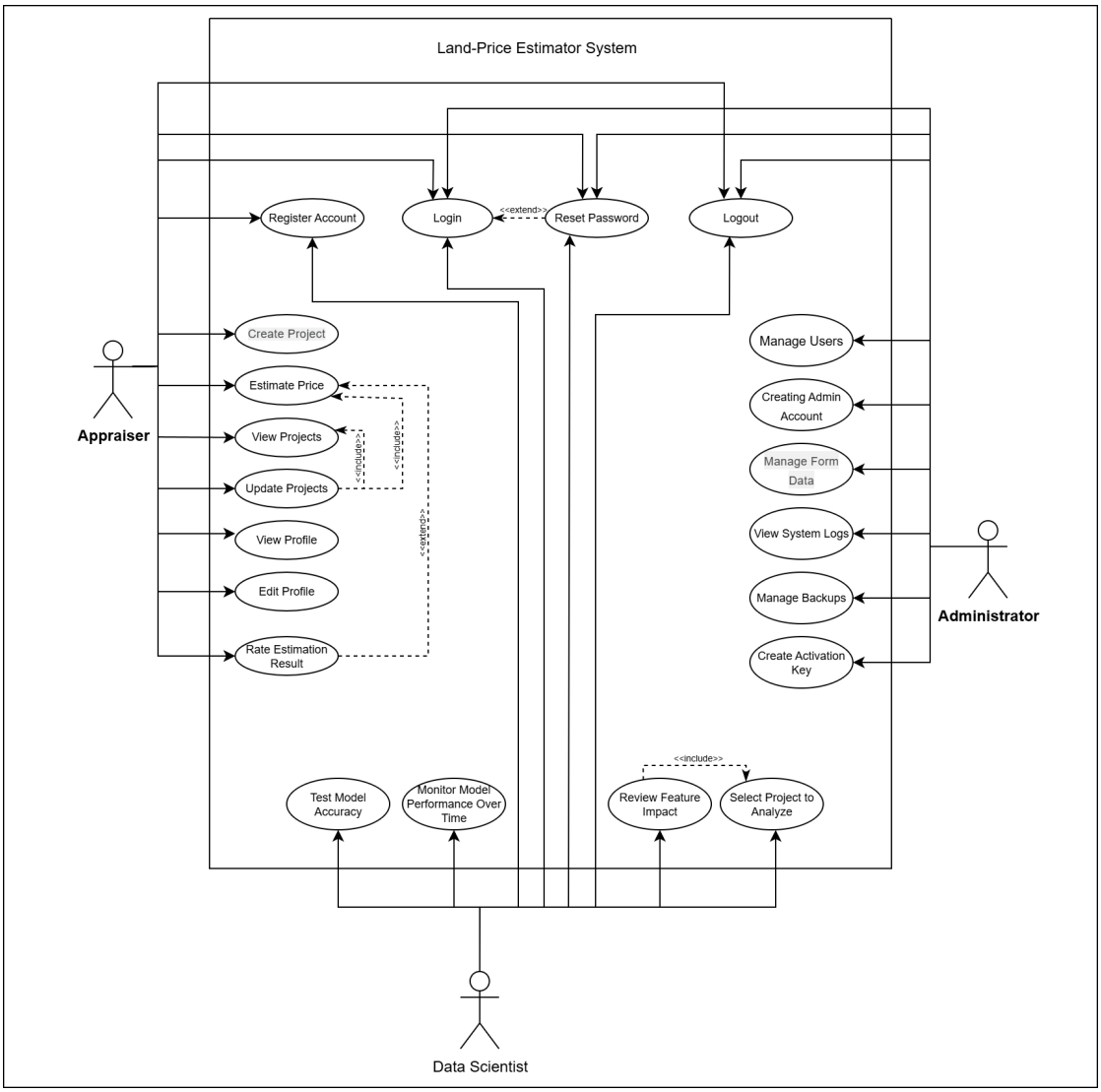
2.5 Nonfunctional Requirements

The nonfunctional requirements describe how the system should behave to provide the best user experience.

1. **Usability**
   - The system should provide a simple and user-friendly interface.
   - The interface should support both desktop and mobile browsers.
2. **Performance**
   - The system should return land price estimation results in less than 5 seconds after submission.
   - Login and registration should complete in less than 3 seconds under normal load.
3. **Availability**
   - The system should be available at least 99% of the time.
4. **Security**
   - The system must protect user information by applying strong encryption methods.
   - Passwords should be securely hashed.
   - Only authorized users can access their personal projects and information.
5. **Data Backup and Recovery**
   - All user accounts and project details should be backed up regularly.
   - When a system failure occurs, users should be able to recover their information without data loss.

2.6 Use-Case Diagram

Figure 2.5.1 Use Case Diagram



2.7 Appraiser's Functional Requirements Tables

Table 2.7.1 Login

Field	**Content**
Requirement	Login
Actor	Land Appraiser
Objective	Access the appraiser's account
Precondition	The appraiser must be registered.
Scenario	1. The appraiser enters email and password. 2. The appraiser clicks 'Submit'. 3. The system verifies credentials and grants access.
Exceptions	1. Incorrect credentials — the system displays an error message. 2. Account locked due to failed attempts. 3. Account not activated. 4. No internet connection. 5. Server or network error — system prompts the user to try again later.

Table 2.7.2 Register Account

Field	**Content**
Requirement	Register Account
Actor	Land Appraiser
Objective	Create a new appraiser account.
Precondition	The appraiser must have the activation key from the administrator.
Scenario	1. The appraiser selects ‘Register‘. 2. The appraiser enters the activation code. 3. The appraiser fills in required details (name, email, phone, password). 4. Verification code is sent to the entered email. 5. The appraiser clicks the link in the email to confirm the email. 6. The appraiser submits the form. 7. The system creates the account and confirms registration.
Exceptions	1. Email already in use. 2. Weak or invalid password. 3. Required fields missing. 4. Activation code expired or incorrect. 5. Server or network error.

Table 2.7.3 Logout

Field	**Content**
Requirement	Logout
Actor	Land Appraiser
Objective	Securely end the current session and prevent unauthorized access to the account.
Precondition	The appraiser is logged into the system.
Scenario	1. The appraiser clicks the Logout button from the system interface. 2. The system ends the current session. 3. The appraiser is sent to the login page.
Exceptions	1. Server or network error.

Table 2.7.4 Reset Password

Field	**Content**
Requirement	Reset Password
Actor	Land Appraiser
Objective	Change the password.
Precondition	Appraiser has a valid registered email.
Scenario	1. Appraiser clicks ”Forgot Password” on the login page or selects ”Change Password” from their profile. 2. If ”Forgot Password”: 2.1. System prompts for the registered email. 2.2. Appraiser enters email and submits. 2.3. System sends a password reset link or code to the email. 2.4. Appraiser clicks the link or enters the code, then sets a new password. 3. If ”Change Password” from profile: 3.1. Appraiser enters current password and new password. 3.2. System verifies the current password and updates it. 4. System confirms that the password has been successfully updated.
Exceptions	1. Email not found in the system. 2. Invalid or expired reset link/code. 3. Incorrect current password (when changing from profile). 4. Server or database error during update.

Table 2.7.5 Create Project

Field	**Content**
Requirement	Add Project
Actor	Land Appraiser
Objective	Create a new project and enter information needed to estimate the land price.
Precondition	Appraiser must be logged in.
Scenario	1. Appraiser selects ‘New Project’. 2. Names the Project. 3. Fills in land details. 4. The system checks and validates the input data.
Exceptions	1. Missing or invalid fields — display helpful error messages. 2. Network failure.

Table 2.7.6 Estimate Price

Field	**Content**
Requirement	Estimate Price
Actor	Land Appraiser
Objective	Predict and view the price of the land.
Precondition	Land data has been successfully submitted.
Scenario	1. Appraiser clicks “estimate price”. 2. System runs the model on the input. 3. Displays the estimated price and summary.
Exceptions	1. System error in model execution. 2. Timeout or delay in result. 3. Server or network error.

Table 2.7.7 View Projects

Field	**Content**
Requirement	View Projects
Actor	Land Appraiser
Objective	Access previously estimated land projects to view them.
Precondition	Appraiser must be logged in.
Scenario	1. Appraiser selects ‘My Projects’. 2. System displays a list of past projects. 3. Appraiser can select any project to view.
Exceptions	1. No saved projects. 2. Database access failure. 3. Network error.

Table 2.7.8 Update Projects

Field	**Content**
Requirement	Update Projects
Actor	Land Appraiser
Objective	Edit previously estimated land projects, perform new estimations, and keep a record of past estimations for the same project.
Precondition	Appraiser must be logged in.
Scenario	1. Appraiser selects ‘My Projects’. 2. System displays a list of past projects. 3. Appraiser selects a project to update. 4. Appraiser edits the project details. 5. System generates a new estimation for the updated data. 6. Previous estimations for the same project are saved and can be viewed.
Exceptions	1. No saved projects. 2. Database access failure. 3. Network error.

Table 2.7.9 View Profile

Field	**Content**
Requirement	View Profile
Actor	Land Appraiser
Objective	View the appraiser's account information.
Precondition	Appraiser must be logged in.
Scenario	1. Appraiser selects 'Profile'. 2. System displays current account information, including name, email, phone, and other profile details.
Exceptions	1. Profile data not found. 2. Database access failure. 3. Network error.

Table 2.7.10 Edit Profile

Field	**Content**
Requirement	Edit Profile
Actor	Land Appraiser
Objective	Update appraiser's account information.
Precondition	Appraiser must be logged in and in the profile.
Scenario	1. Appraiser selects 'Edit Profile'. 2. Updates name, email, phone number, or password. 3. Clicks 'Save Changes'.
Exceptions	1. Invalid email or password. 2. Server error during update. 3. Network or server error.

Table 2.7.11 Rate Estimation Result

Field	Content
Requirement	Rate Estimation Result
Actor	Land Appraiser
Objective	Provide feedback on the quality of the estimated result to help evaluate model performance.
Precondition	An estimation result must be displayed.
Scenario	1. After viewing the estimation result, the appraiser is prompted to provide feedback. 2. The appraiser selects a rating option (e.g., Thumbs Up or Thumbs Down). 3. The appraiser can provide a more logical/accurate estimation in case of negative rating. 4. The system saves the rating along with the estimation details and corrected estimation for the current project.
Exceptions	1. Rating submission fails due to network error. 2. Database save error.

2.8 Admin's Functional Requirements Tables

Table 2.8.1 Login

Field	Content
Requirement	Login
Actor	Admin
Objective	Allow admin to securely log in to the system.
Precondition	Admin must be registered and approved as an administrator by an existing admin.
Scenario	1. Admin visits login page. 2. Enters email and password. 3. Clicks ”Login”. 4. The system verifies credentials and grants access.
Exceptions	1. Invalid email or password — system displays an error message. 2. Server or network error — system prompts the user to try again later.

Table 2.8.2 Logout

Field	**Content**
Requirement	Logout
Actor	Admin
Objective	Securely end the current session and prevent unauthorized access to the account.
Precondition	Admin is logged into the system.
Scenario	1. Admin clicks the Logout button from the system interface. 2. The system ends the current session. 3. Admin is sent to the login page.
Exceptions	1. Server or network error.

Table 2.8.3 Creating Admin Account

Field	**Content**
Requirement	Creating Admin Accounts
Actor	Existing Admin
Objective	Create new admin accounts.
Precondition	Existing Admin is authorized to create admins.
Scenario	1. Authorized admin accesses the admin management interface. 2. Creates an admin and enters admin details. 3. Account is activated.
Exceptions	1. Invitation link expires. 2. Unauthorized requester attempts to create an admin. 3. Failure in account setup due to system error.

Table 2.8.4 Manage Users

Field	Content
Requirement	View Users
Actor	Admin
Objective	View a list of all registered users with their details, and the ability to select any user to edit their account.
Precondition	Admin is logged in.
Scenario	1. Admin opens the user management panel. 2. System displays a list of all registered users with basic details (e.g., name, email, registration date, role). 3. Admin can sort or filter the list. 4. Admin can select any user to make actions. 5. The actions are: Delete User, Deactivate Account (if activated), Activate Account (if deactivated), and Change Role.
Exceptions	1. No users found in the system. 2. Server error when retrieving user data. 3. Database connection failure. 4. Failure of action.

Table 2.8.5 Manage Form Data

Field	Content
Requirement	Manage Form Data
Actor	Admin
Objective	Edit or add options (e.g., locations or classifications) available during project creation.
Precondition	Admin has access rights.
Scenario	1. Admin selects ‘Manage Data’. 2. Chooses data category (e.g., regions). 3. Edits, adds, or deletes entries. 4. Saves changes.
Exceptions	1. Input is invalid. 2. Changes not saved due to a database error.

Table 2.8.6 View System Logs

Field	**Content**
Requirement	View System Logs
Actor	Admin
Objective	Monitor system events and user activity.
Precondition	System logging is enabled.
Scenario	1. Admin navigates to the ‘Logs’ section. 2. Filters by date or activity type. 3. Views login, registration, or error logs.
Exceptions	1. Logs not available. 2. Permission denied.

Table 2.8.7 Manage Backups

Field	**Content**
Requirement	Manage Backups
Actor	Admin
Objective	Ensure system and user data is regularly backed up.
Precondition	Backup system is active.
Scenario	1. Admin opens ‘Backup Settings’. 2. Triggers manual backup or sets automatic schedule. 3. Confirms successful completion.
Exceptions	1. Backup failed due to storage limit. 2. Scheduled backup skipped due to server downtime.

Table 2.8.8 Create Activation Key

Field	**Content**
Requirement	Create Activation Key
Actor	Admin
Objective	Generate a unique activation key for a data scientist or appraiser to use when registering their account.
Precondition	Admin is logged in.
Scenario	1. Admin opens the ”Activation Keys” panel. 2. Selects the account type (Data Scientist or Appraiser). 3. Clicks ”Generate Key”. 4. System generates a unique activation key.Clicks 5. Admin copies or sends the key Clicksto the intended recipient.
Exceptions	1. Server error during key generation. 2. Database access failure when saving the new key.

Table 2.8.9 Reset Password

Field	**Content**
Requirement	Reset Password
Actor	Admin
Objective	Change the password.
Precondition	Admin has a registered account.
Scenario	1. Admin clicks ”Forgot Password” on the login page or selects ”Change Password” from their profile. 2. If ”Forgot Password”: 2.1. System prompts for the registered email. 2.2. Admin enters email and submits. 2.3. System sends a password reset link or code to the email. 2.4. Admin clicks the link or enters the code, then sets a new password. 3. If ”Change Password” from profile: 3.1. Admin enters current password and new password. 3.2. System verifies the current password and updates it. 4. System confirms that the password has been successfully updated.
Exceptions	1. Email not found in the system. 2. Invalid or expired reset link/code. 3. Incorrect current password (when changing from profile). 4. Server or database error during update.

Table 2.8.10 View Profile

Field	**Content**
Requirement	View Profile
Actor	Admin
Objective	View the admin's account information.
Precondition	Admin must be logged in.
Scenario	1. Admin selects 'Profile'. 2. System displays current account information, including name, email, phone, and other profile details.
Exceptions	1. Profile data not found. 2. Database access failure. 3. Network error.

Table 2.8.11 Edit Profile

Field	**Content**
Requirement	Edit Profile
Actor	Admin
Objective	Update Admin's account information.
Precondition	Admin must be logged in and in the profile.
Scenario	1. Admin selects 'Edit Profile'. 2. Updates name, email, phone number, or password. 3. Clicks 'Save Changes'.
Exceptions	1. Invalid email or password. 2. Server error during update. 3. Network or server error.

2.9 Data Scientist's Functional Requirements Tables

Table 2.9.1 Login

Field	Content
Requirement	Login
Actor	Data Scientist
Objective	Allow the data scientist to securely log in to the system.
Precondition	The data scientist must be registered and approved as a data scientist by an admin.
Scenario	1. Data scientist visits login page. 2. Enters a valid email and password. 3. Clicks the Login button. 4. The system verifies credentials and grants access.
Exceptions	1. Invalid email or password — system displays an error message. 2. Server or network error — system prompts the user to try again later.

Table 2.9.2 Logout

Field	Content
Requirement	Logout
Actor	Data Scientist
Objective	Securely end the current session and prevent unauthorized access to the account.
Precondition	The Data Scientist is logged into the system.
Scenario	1. The data scientist clicks the Logout button from the system. 2. The system ends the current session. 3. The data scientist is sent to the login page.
Exceptions	1. Server or network error.

Table 2.9.3 Register Account

Field	**Content**
Requirement	Register Account
Actor	Data Scientist
Objective	Create a new Data Scientist account.
Precondition	The Data Scientist must have the activation key from the administrator.
Scenario	1. The Data Scientist selects ‘Register’. 2. Data Scientist enters the activation code. 3. The Data Scientist fills in required details (name, email, password). 4. Verification code is sent to the entered email. 5. Data Scientist clicks the link in the email to confirm the email. 6. The Data Scientist submits the form. 7. The system creates the account and confirms registration.
Exceptions	1. Email already in use. 2. Weak or invalid password. 3. Required fields missing. 4. Activation code expired or incorrect. 5. Server or network error.

Table 2.9.4 Test Model Accuracy

Field	**Content**
Requirement	Test Model Accuracy
Actor	Data Scientist
Objective	Evaluate the prediction accuracy of the machine learning model.
Precondition	The system must have a trained model and a dataset available for testing.
Scenario	1. The data scientist selects the ”Model Testing” section. 2. Uploads or selects a dataset for testing. 3. Runs the model to predict prices. 4. Compares predicted results with actual prices.
Exceptions	1. Incomplete or invalid test dataset. 2. Model not available or not trained.

Table 2.9.5 Review Feature Impact

Field	**Content**
Requirement	Review Feature Impact
Actor	Data Scientist
Objective	Analyze which features had the most influence on the land price prediction for a specific project.
Precondition	1. The model must be trained and support feature importance analysis. 2. The project must have completed the estimation.
Scenario	1. Access the “Projects” section. 2. Select a specific project from the list. 3. Open the “Feature Importance” view for that project. 4. View ranked list of features by their impact on the project’s prediction. 5. Export or download the report if needed.
Exceptions	1. Feature analysis tool is unavailable or unsupported for that project. 2. Insufficient project data to generate meaningful insights. 3. Selected project not found or inaccessible.

Table 2.9.6 Monitor Model Performance Over Time

Field	**Content**
Requirement	Monitor Model Performance Over Time
Actor	Data Scientist
Objective	Track how the model performs across different versions and datasets, and test any selected version on demand.
Precondition	1. System must store model versions, related datasets, and performance logs. 2. At least one model version must exist.
Scenario	1. Go to the “Model History” tab. 2. System displays a list of stored model versions with their details. 3. Select a version to see its past results. 4. Optionally, choose a dataset to re-test the selected model version. 5. System runs the test and shows the new results.
Exceptions	1. No stored model versions available. 2. Past performance data is missing or incomplete. 3. Testing fails due to corrupted data or unsupported dataset format. 4. Network or server error during testing.

Table 2.9.7 Select Project to Analyze

Field	**Content**
Requirement	Select Project to Analyze
Actor	Data Scientist
Objective	Choose a specific project from the list of available projects to perform analysis on its estimations and related data.
Precondition	1. Data Scientist must be logged in. 2. There must be at least one project available to analyze.
Scenario	1. Data Scientist opens the “Projects” section. 2. System displays a list of projects with basic details (e.g., project name, date, owner). 3. Data Scientist searches, filters, or sorts projects if needed. 4. Selects a project to open for detailed analysis.
Exceptions	1. No available projects to analyze. 2. Project data is incomplete or inaccessible. 3. Network or server error while loading projects.

Table 2.9.8 Reset Password

Field	**Content**
Requirement	Reset Password
Actor	Data Scientist
Objective	Change the password.
Precondition	The data scientist has a valid registered email.
Scenario	1. The data scientist clicks “Forgot Password” on the login page or selects “Change Password” from their profile. 2. If “Forgot Password”: 2.1. System prompts for the registered email. 2.2. The data scientist enters the email and submits. 2.3. System sends a password reset link or code to the email. 2.4. The data scientist clicks the link or enters the code, then sets a new password. 3. If “Change Password” from profile: 3.1. The data scientist enters current password and new password. 3.2. System verifies the current password and updates it. 4. System confirms that the password has been successfully updated.
Exceptions	1. Email not found in the system. 2. Invalid or expired reset link/code. 3. Incorrect current password (when changing from profile). 4. Server or database error during update.

Table 2.9.9 View Profile

Field	**Content**
Requirement	View Profile
Actor	Data Scientist
Objective	View the Data Scientist's account information.
Precondition	Data Scientist must be logged in.
Scenario	1. Data Scientist selects 'Profile'. 2. System displays current account information, including name, email, phone, and other profile details.
Exceptions	1. Profile data not found. 2. Database access failure. 3. Network error.

Table 2.9.10 Edit Profile

Field	**Content**
Requirement	Edit Profile
Actor	Data Scientist
Objective	Update Data Scientist's account information.
Precondition	Data Scientist must be logged in and in the profile.
Scenario	1. Data Scientist selects 'Edit Profile'. 2. Updates name, email, phone number, or password. 3. Clicks 'Save Changes'.
Exceptions	1. Invalid email or password. 2. Server error during update. 3. Network or server error.

Chapter 3: Architecture and Design

3.1 Overview

This chapter explains how the Land Price Estimator system is organized and how its parts work together. It covers the system's design, the chosen architecture and its possible alternatives, the database structure, and the main interfaces for the user, administrator, and data scientist.

3.2 Chosen Architecture Design

We studied multiple architecture design options, and concluded that the MVT (Model–View–Template) architecture is the best fit for our project.

Table 3.2.1 MVT Components

Component	**Role**
Model	Manages data, database structure, and rules.
View	Handles user actions; retrieves data from the model and selects the appropriate template to display results.
Template	Presentation layer controlling how data is rendered to the user (HTML).

The separation makes it easy for developers to work on different components of the application at the same time without affecting each other's work, and makes future scalability as well as maintaining, debugging, and testing the application easier.
Why MVT? The MVT architecture is provided by Django, which is the framework we are using to develop the web application for the Land-Price Prediction system.

3.3 Architecture Implementation

In our Land-Price Prediction system, the MVT architecture is implemented as follows:

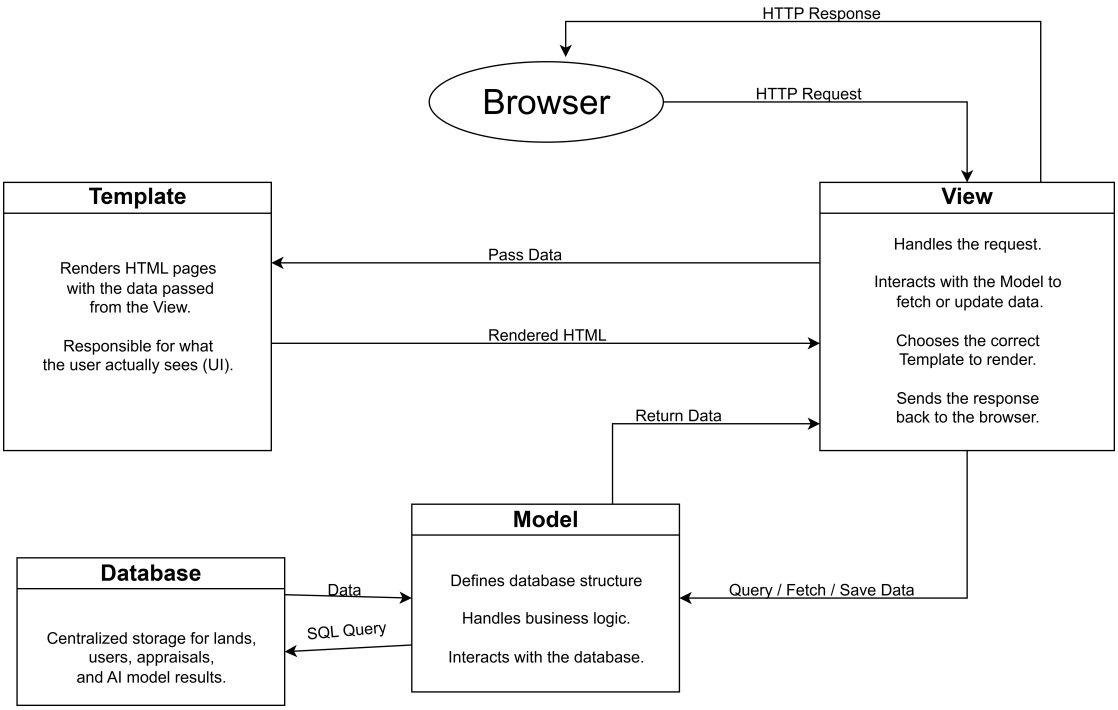
Model: Stores all the data related to each entity of the system, such as the lands and users attributes, and how they are stored in the database and how to retrieve them.

View: The view stores the business logic and connects the models with the templates; it processes user requests, retrieves data from the model, and gives it to the template.

Template: It's the interface that the user sees and interacts with. Through it, the user sees the results of the predictions and other information. The template has no business logic to ensure a clean separation from backend processing.

This structured method ensures that each layer is independent but still connected.

Figure 3.3.1 MVT Architecture



3.3.1 Example Models in the System

The project will have multiple models to manage the data effectively within the MVT architecture. Each model represents an entity of the Land-Price Prediction system and has its own attributes. Example models include:

1. **User:** Represents a system user. Attributes include `id`, `full_name`, `email`, `password`, `role`, and `created_at`.

2. **Project:** Represents a land estimation project. Attributes include `id`, `name`, `description`, `created_by`, `created_at`, and `status`.

3. **Plot (Land Parcel):** Represents a land parcel. Attributes include `id`, `plot_code`, `governorate_id`, `town_id`, `neighborhood_id`, `area_m2`, `slope`, `soil_type_id`, `rock_type_id`, `current_land_use`, `planned_land_use`, `far`, `coverage_ratio`, `is_current`, `version_no`, `created_by`, and `created_at`.

4. **Plot Document:** Represents documents related to a plot. Attributes include `id`,

plot_id, doc_type_id, issuing_authority_id, doc_number, issue_date, share_type, and share_ratio.

5. **Model:** Represents a machine learning model. Attributes include id, name, version, description, created_by, created_at, and is_active.

6. **Valuation:** Represents the output of a model prediction for a plot in a project. Attributes include id, project_id, plot_id, model_id, predicted_price, created_at, and created_by.

7. **Project Plot:** Represents the association between a project and its plots. Attributes include id, project_id, plot_id, valuation_id, and note.

8. **Plot Feedback:** Stores user feedback on a plot's estimated price. Attributes include id, plot_id, user_id, is_price_accepted, suggested_price, and created_at.

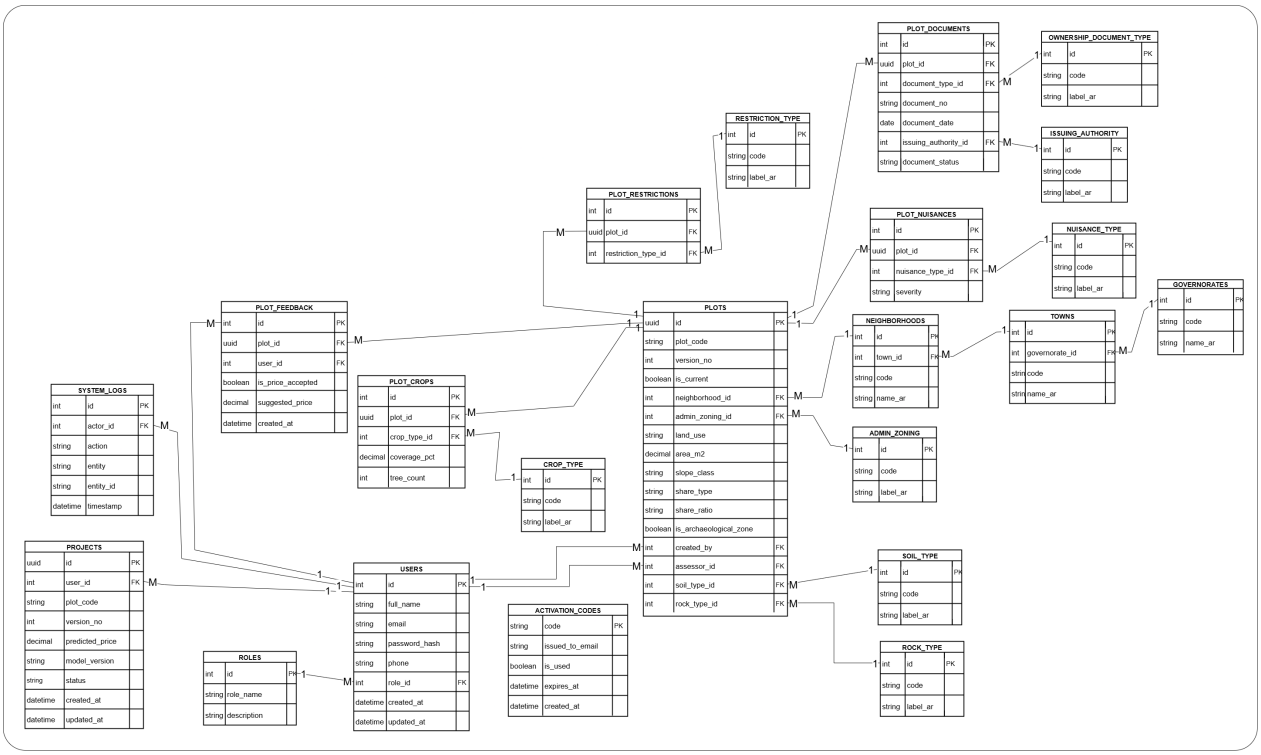
9. **Governorate, Town, Neighborhood:** Represent geographical hierarchy. Attributes include id, code, name_ar, and relevant foreign keys.

10. **Supporting Lookup Tables:** Includes Admin Zoning, Ownership Document Type, Issuing Authority, Soil Type, Rock Type, Crop Type, Nuisance Type, and Restriction Type with their respective codes and labels.

11. **Plot Crops, Plot Nuisances, Plot Restrictions:** Represent details of crops, nuisances, and restrictions on a plot. Attributes include plot foreign key, type foreign key, and specific measurements such as coverage_pct, tree_count, or severity.

3.4 ER Diagram

Figure 3.4.1 ER Diagram



3.5 Database Description

3.5.1 Users

- `id`: integer; PK; auto-increment.
- `full_name`: string; not-null.
- `email`: string; unique; not-null.
- `password`: string; not-null; ≥ 8 chars.
- `role`: enum {ADMIN, ASSESSOR, DATA_SCIENTIST}; not-null.
- `created_at`: datetime; not-null.

3.5.2 Projects

- `id`: UUID; PK.
- `name`: string; not-null.
- `description`: string; not-null.
- `created_by`: integer; FK $\rightarrow$ `users.id`; not-null.
- `created_at`: datetime; not-null.
- `status`: enum {ACTIVE, ARCHIVED}; not-null.

3.5.3 Plots (Land Parcels)

- `id`: UUID; PK.
- `plot_code`: string; unique; not-null.
- `governorate_id`: integer; FK $\rightarrow$ `governorates.id`; not-null.
- `town_id`: integer; FK $\rightarrow$ `towns.id`; not-null.
- `neighborhood_id`: integer; FK $\rightarrow$ `neighborhoods.id`; not-null.
- `area_m2`: `decimal(12,2)`; ≥ 0; not-null.
- `slope`: enum {FLAT, SLIGHT, MODERATE, STEEP}; not-null.
- `soil_type_id`: integer; FK $\rightarrow$ `soil_type.id`; nullable.
- `rock_type_id`: integer; FK $\rightarrow$ `rock_type.id`; nullable.
- `current_land_use`: enum {RES, COM, AGR, IND, MIX, VACANT}; not-null.

- planned_land_use: enum {RES, COM, AGR, IND, MIX, VACANT}; nullable.
- far: decimal(6,3); ≥ 0; nullable.
- coverage_ratio: decimal(4,2); 0–1; nullable.
- is_current: boolean; not-null.
- version_no: integer; not-null.
- created_by: integer; FK → users.id; not-null.
- created_at: datetime; not-null.

3.5.4 Plot_Documents

- id: integer; PK; auto-increment.
- plot_id: UUID; FK → plots.id; not-null.
- doc_type_id: integer; FK → ownership_document_type.id; not-null.
- issuing_authority_id: integer; FK → issuing_authority.id; not-null.
- doc_number: string; not-null.
- issue_date: date; not-null.
- share_type: enum {INDIVIDUAL, MUSHA}; not-null.
- share_ratio: string (e.g., "3/8"); nullable if INDIVIDUAL.

3.5.5 Models

- id: UUID; PK.
- name: string; not-null.
- version: string; not-null.
- description: string; not-null.
- created_by: integer; FK → users.id; not-null.
- created_at: datetime; not-null.
- is_active: boolean; not-null.

3.5.6 Valuations

- `id`: UUID; PK.
- `project_id`: UUID; FK → `projects.id`; not-null.
- `plot_id`: UUID; FK → `plots.id`; not-null.
- `model_id`: UUID; FK → `models.id`; not-null.
- `predicted_price`: `decimal(12,2)`; not-null.
- `created_at`: datetime; not-null.
- `created_by`: integer; FK → `users.id`; not-null.

3.5.7 Project_Plots

- `id`: integer; PK; auto-increment.
- `project_id`: UUID; FK → `projects.id`; not-null.
- `plot_id`: UUID; FK → `plots.id`; not-null.
- `valuation_id`: UUID; FK → `valuations.id`; nullable.
- `note`: string; optional short label.

3.5.8 Plot_Feedback

- `id`: integer; PK; auto-increment.
- `plot_id`: UUID; FK → `plots.id`; not-null.
- `user_id`: integer; FK → `users.id`; not-null.
- `is_price_accepted`: boolean; not-null.
- `suggested_price`: `decimal(12,2)`; required if not accepted.
- `created_at`: datetime; not-null.

3.5.9 Governorates

- `id`: integer; PK; auto-increment.
- `code`: string; unique; not-null.
- `name_ar`: string; not-null.

3.5.10 Towns

- `id`: integer; PK; auto-increment.
- `governorate_id`: integer; FK → `governorates.id`; not-null.
- `code`: string; unique; not-null.
- `name_ar`: string; not-null.

3.5.11 Neighborhoods

- `id`: integer; PK; auto-increment.
- `town_id`: integer; FK → `towns.id`; not-null.
- `code`: string; unique; not-null.
- `name_ar`: string; not-null.

3.5.12 Admin_Zoning

- `id`: integer; PK; auto-increment.
- `code`: string; unique; not-null.
- `label_ar`: string; not-null.

3.5.13 Ownership_Document_Type

- `id`: integer; PK; auto-increment.
- `code`: string; unique; not-null.
- `label_ar`: string; not-null.

3.5.14 Issuing_Authority

- `id`: integer; PK; auto-increment.
- `code`: string; unique; not-null.
- `label_ar`: string; not-null.

3.5.15 Soil_Type

- `id`: integer; PK; auto-increment.
- `code`: string; unique; not-null.
- `label_ar`: string; not-null.

3.5.16 Rock_Type

- `id`: integer; PK; auto-increment.
- `code`: string; unique; not-null.
- `label_ar`: string; not-null.

3.5.17 Crop_Type

- `id`: integer; PK; auto-increment.
- `code`: string; unique; not-null.
- `label_ar`: string; not-null.

3.5.18 Nuisance_Type

- `id`: integer; PK; auto-increment.
- `code`: string; unique; not-null.
- `label_ar`: string; not-null.

3.5.19 Restriction_Type

- `id`: integer; PK; auto-increment.
- `code`: string; unique; not-null.
- `label_ar`: string; not-null.

3.5.20 Plot_Crops

- `id`: integer; PK; auto-increment.
- `plot_id`: UUID; FK $\rightarrow$ `plots.id`; not-null.
- `crop_type_id`: integer; FK $\rightarrow$ `crop_type.id`; not-null.
- `coverage_pct`: `decimal(5,2)`; 0–100; nullable.
- `tree_count`: integer; ≥ 0; nullable.

3.5.21 Plot_Nuisances

- `id`: integer; PK; auto-increment.
- `plot_id`: UUID; FK $\rightarrow$ `plots.id`; not-null.

- `nuisance_type_id`: integer; FK → `nuisance_type.id`; not-null.
- `severity`: enum {LOW, MEDIUM, HIGH}; not-null.

3.5.22 Plot_Restrictions

- `id`: integer; PK; auto-increment.
- `plot_id`: UUID; FK → `plots.id`; not-null.
- `restriction_type_id`: integer; FK → `restriction_type.id`; not-null.

3.6 Interfaces

Figure 3.6.1 Account Registration

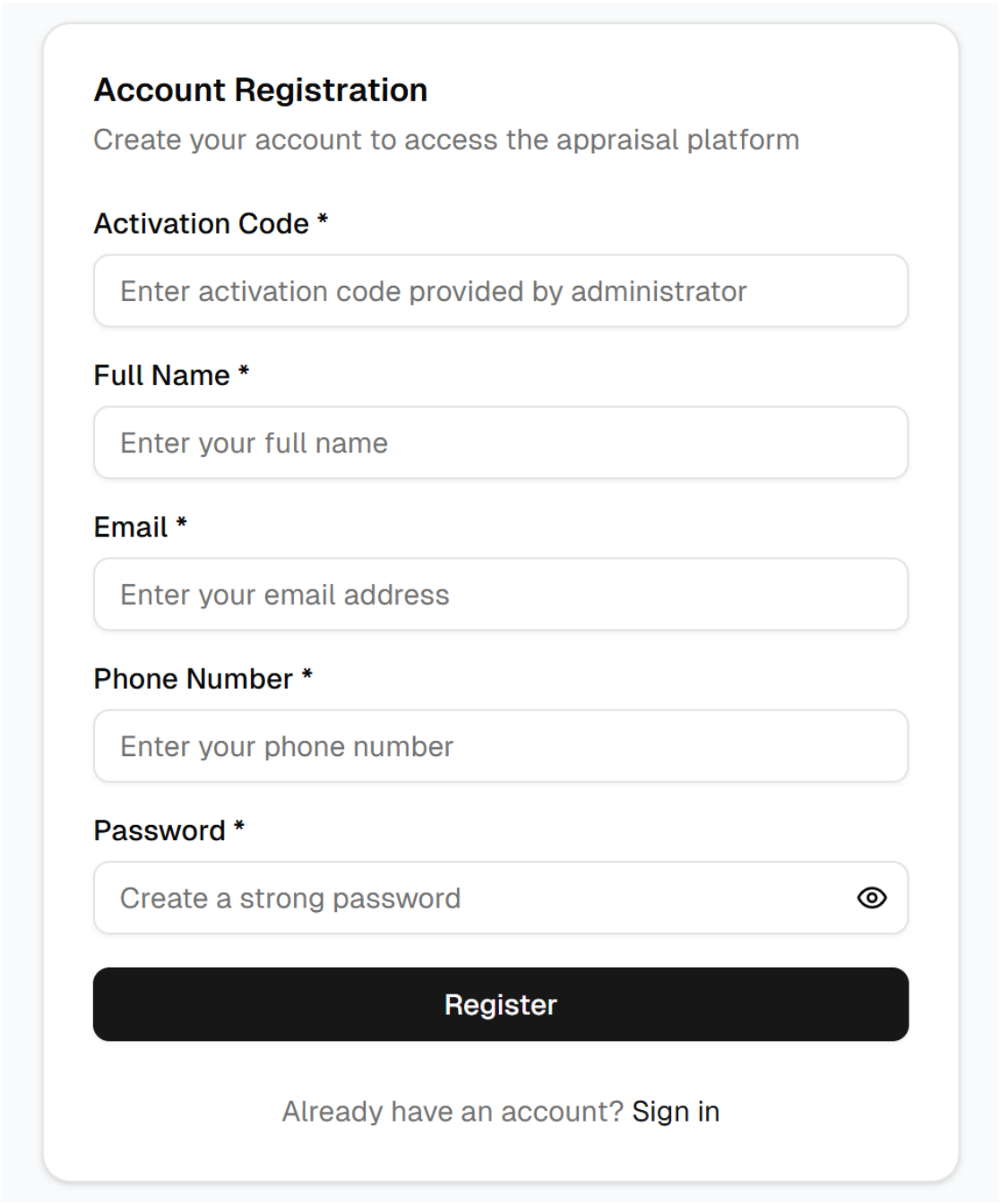


Figure 3.6.2 Login Page

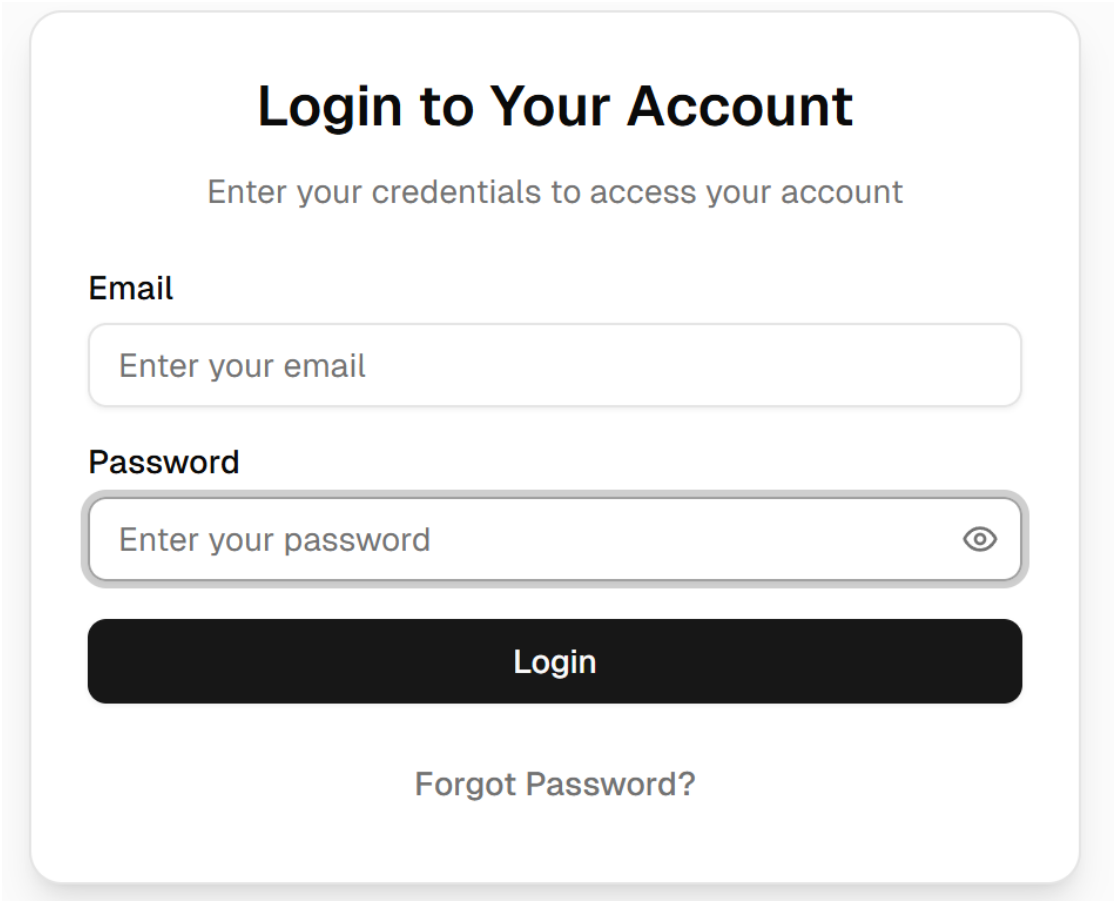


Figure 3.6.3 Forgot Password

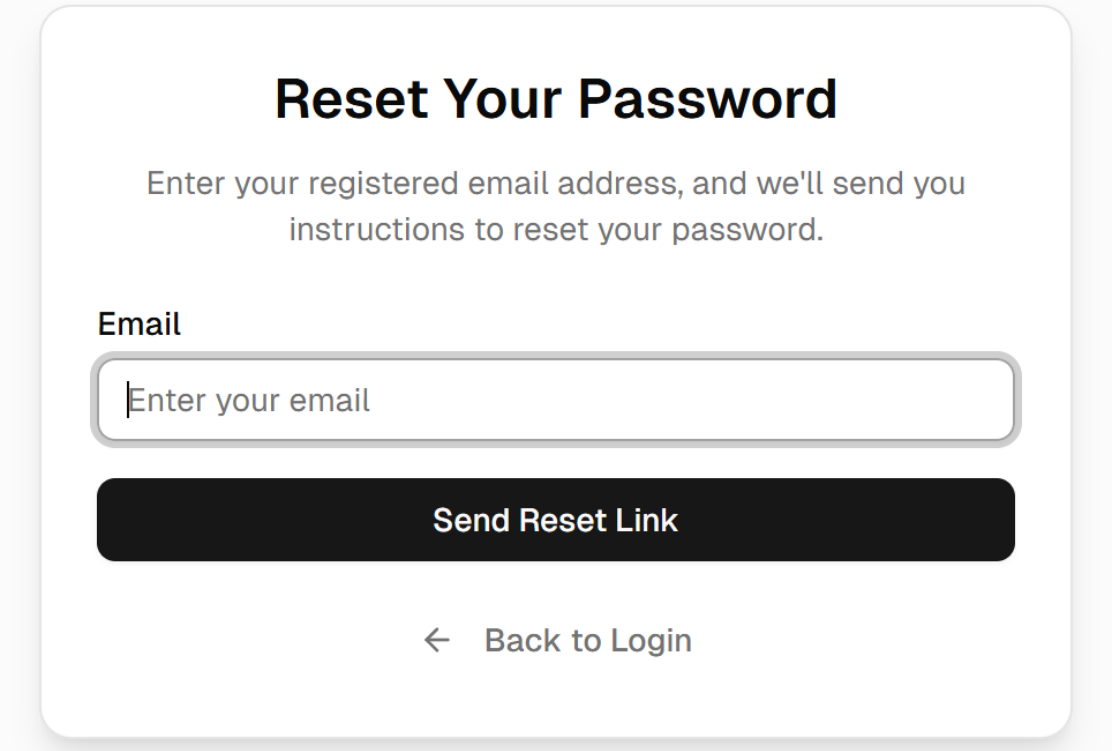


Figure 3.6.4 Home Page

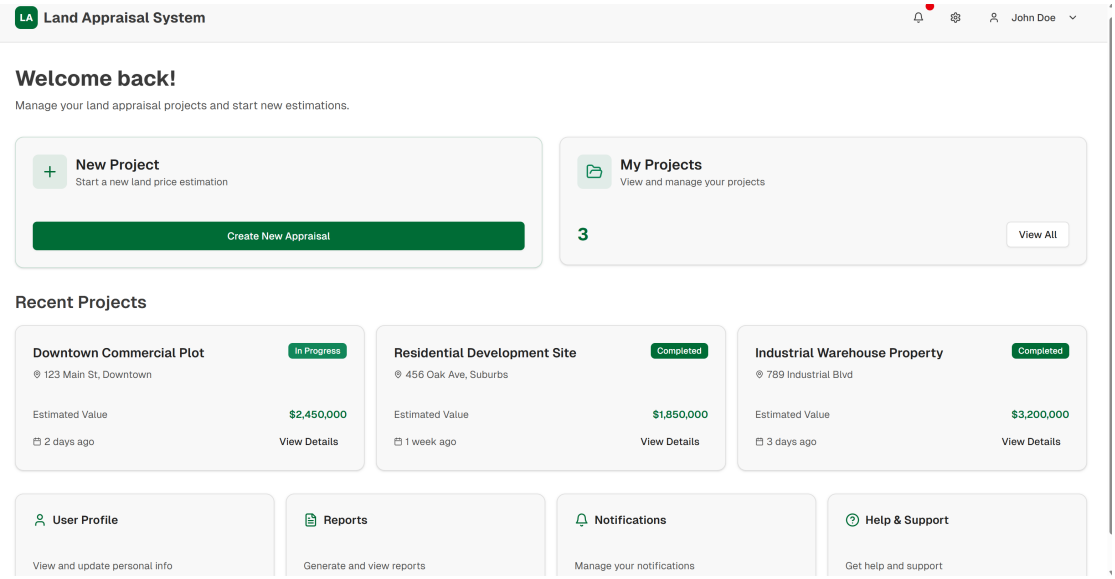


Figure 3.6.5 Create New Project

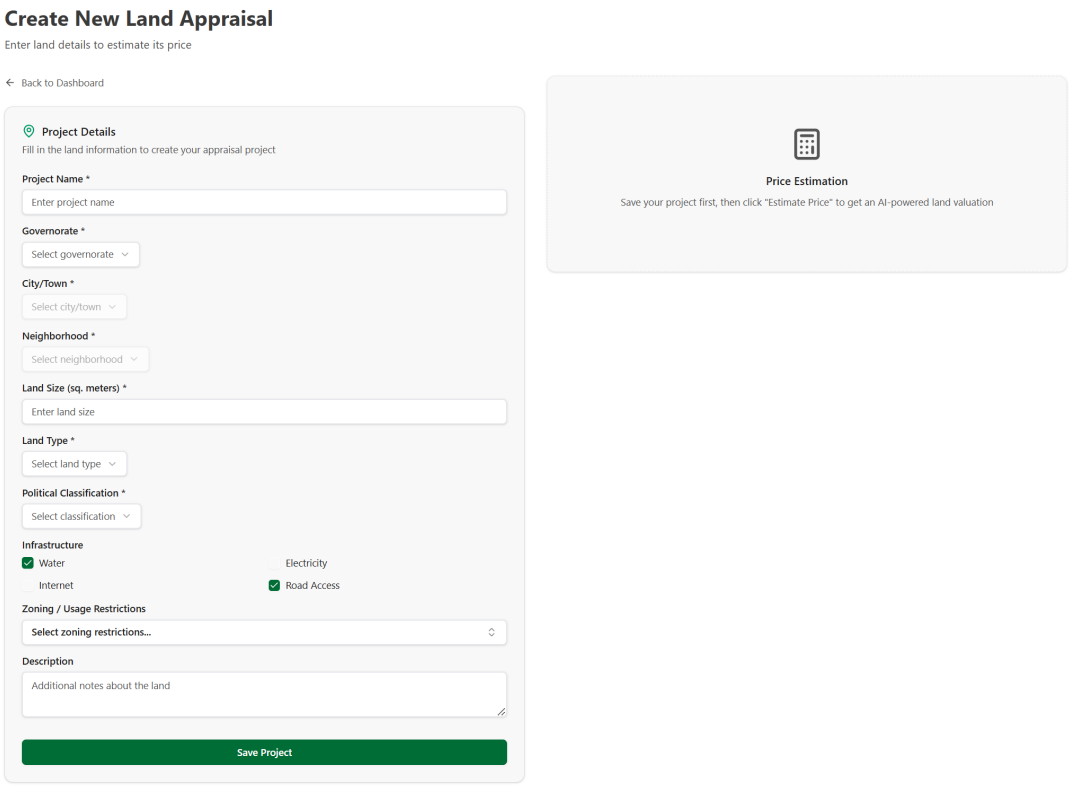


Figure 3.6.6 View Projects

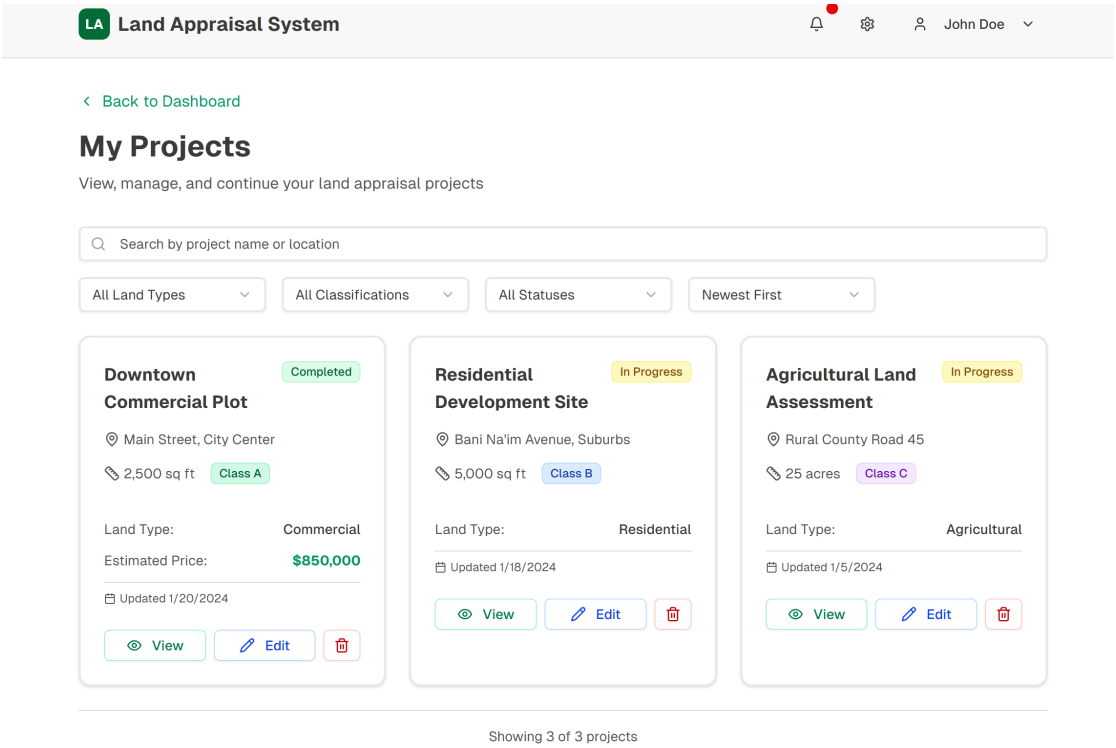


Figure 3.6.7 View Profile

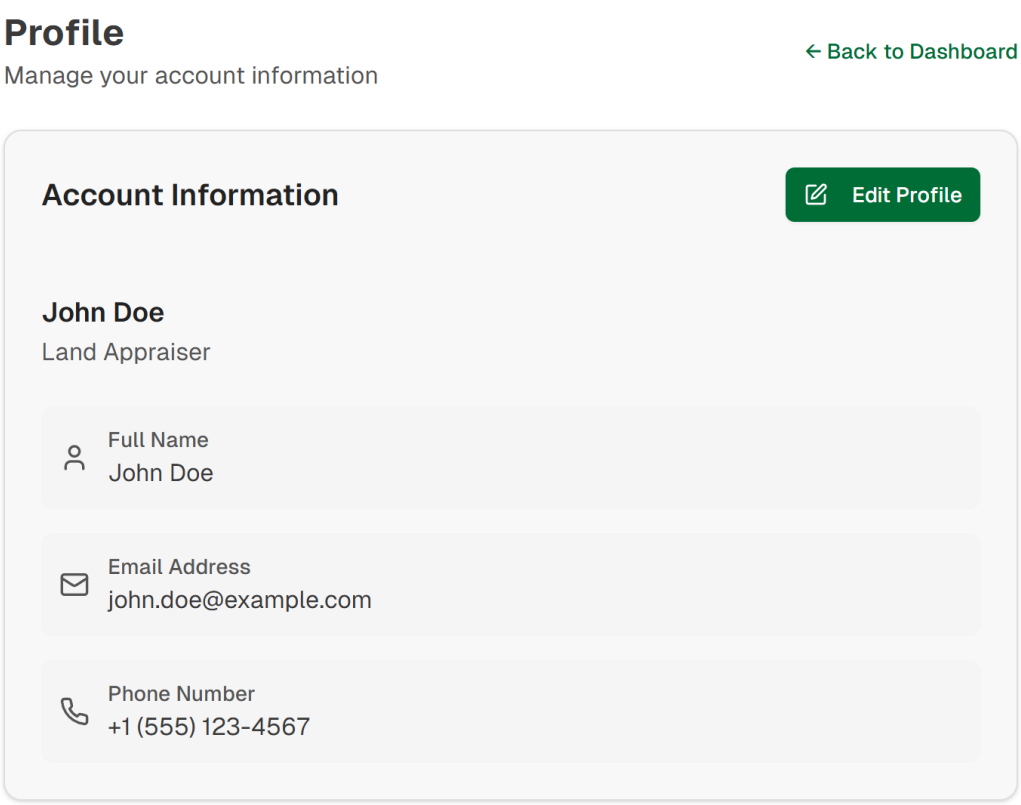


Figure 3.6.8 Edit Profile

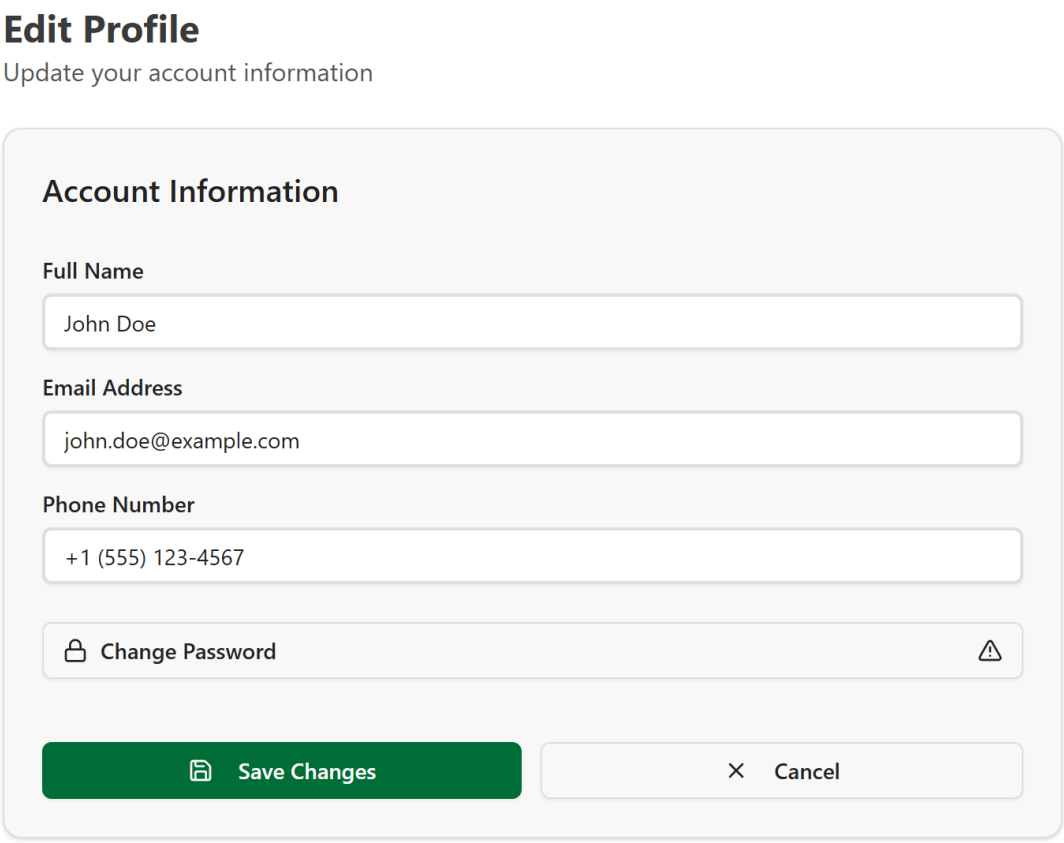


Figure 3.6.9 Select Project To Analyze

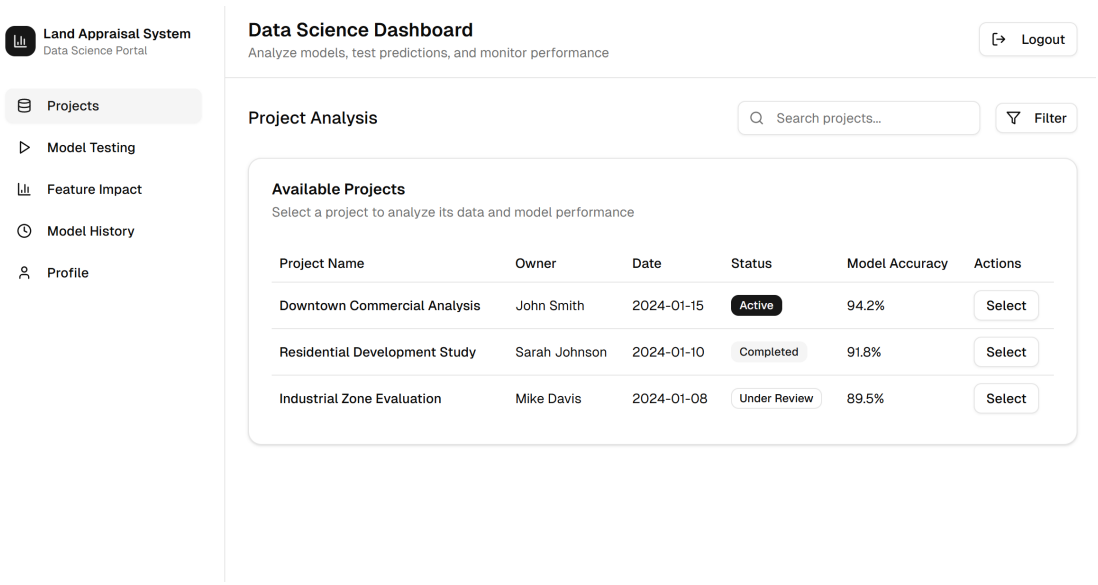


Figure 3.6.10 Test Model Accuracy

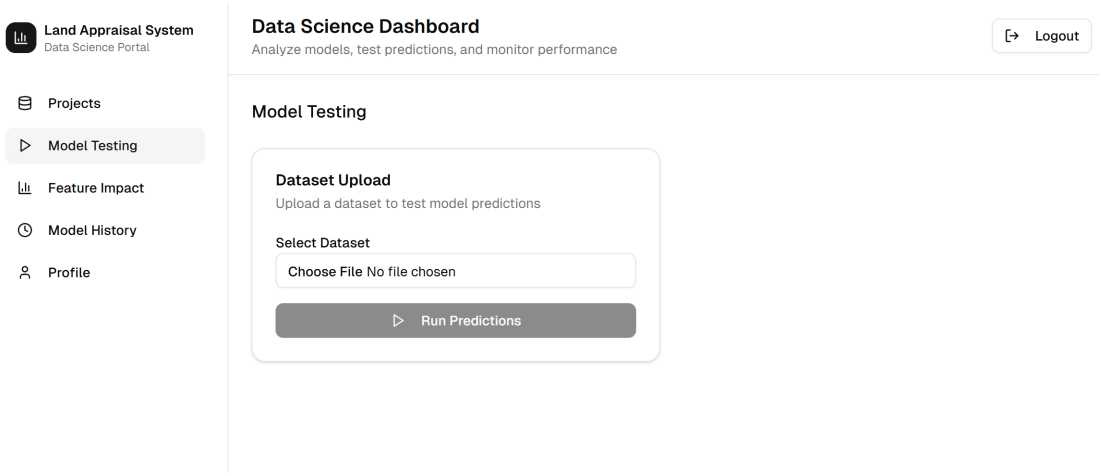


Figure 3.6.11 Review Feature Impact

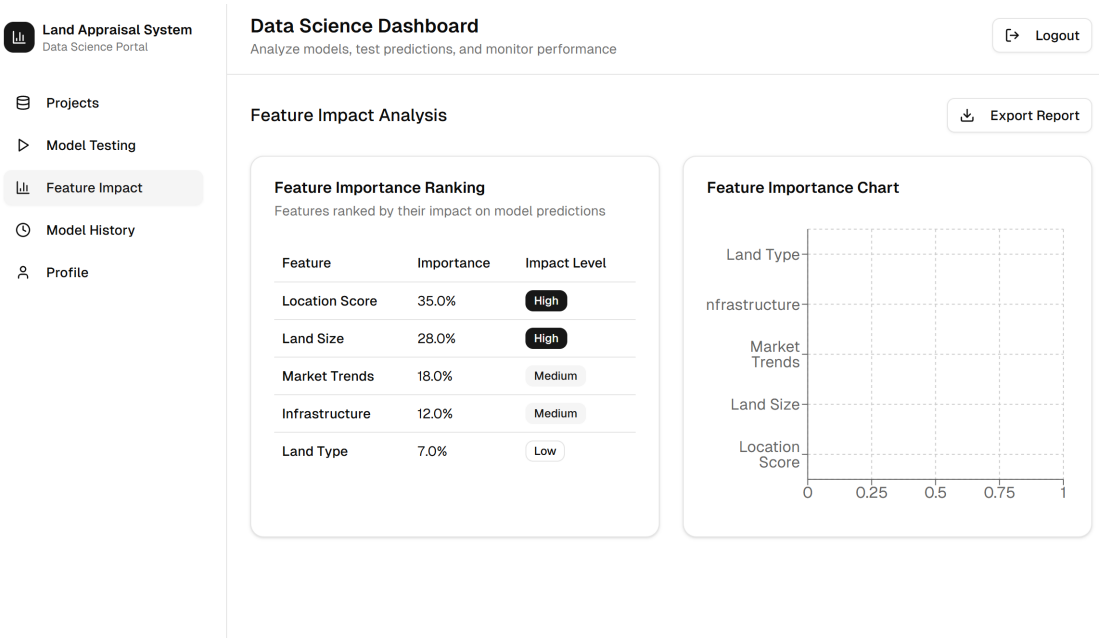


Figure 3.6.12 Monitor Model Performance Over Time

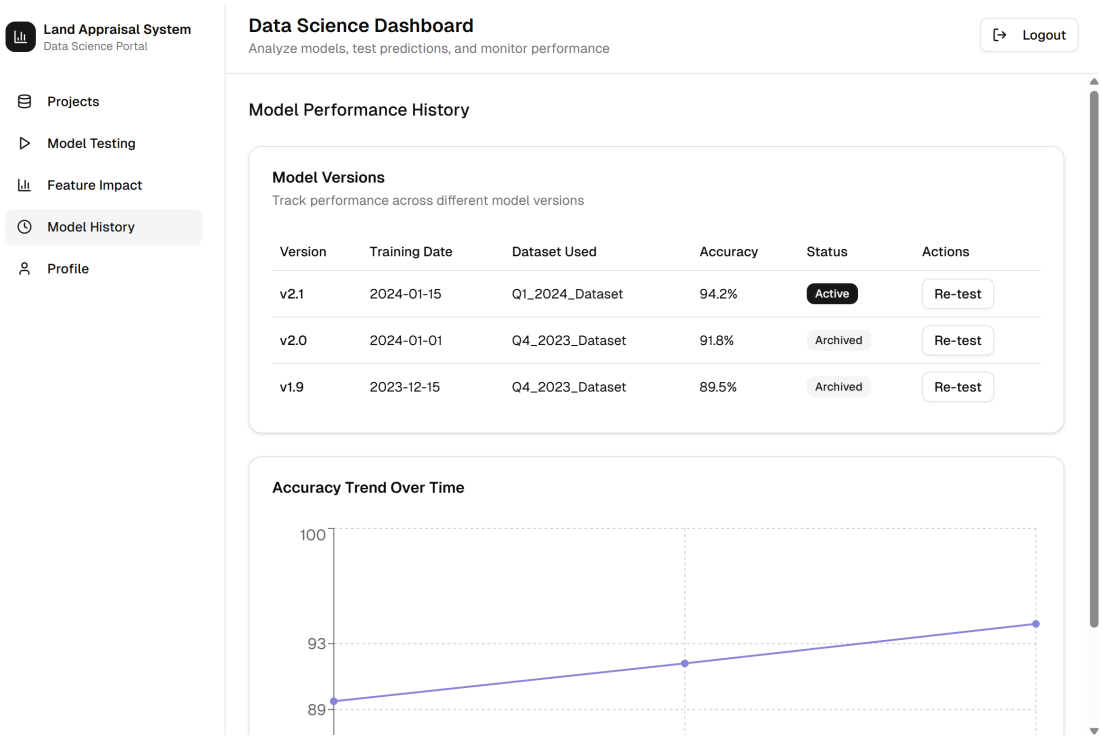


Figure 3.6.13 Admin Dashboard

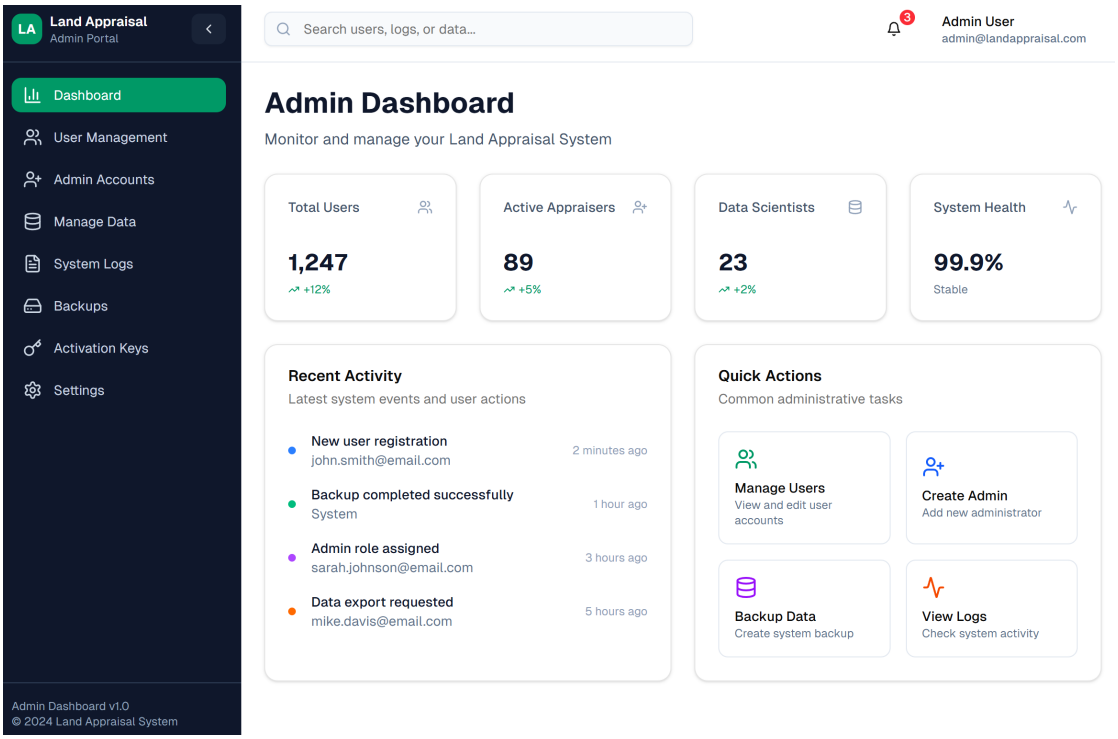


Figure 3.6.14 View And Manage Users

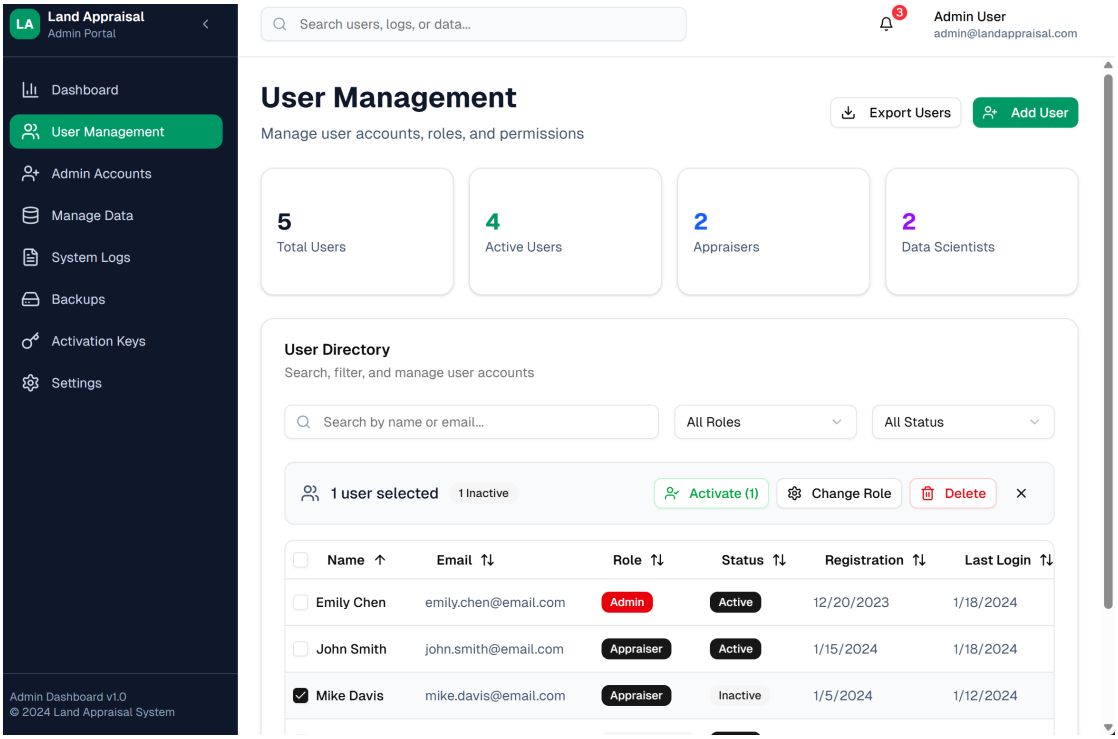


Figure 3.6.15 View Admin Accounts

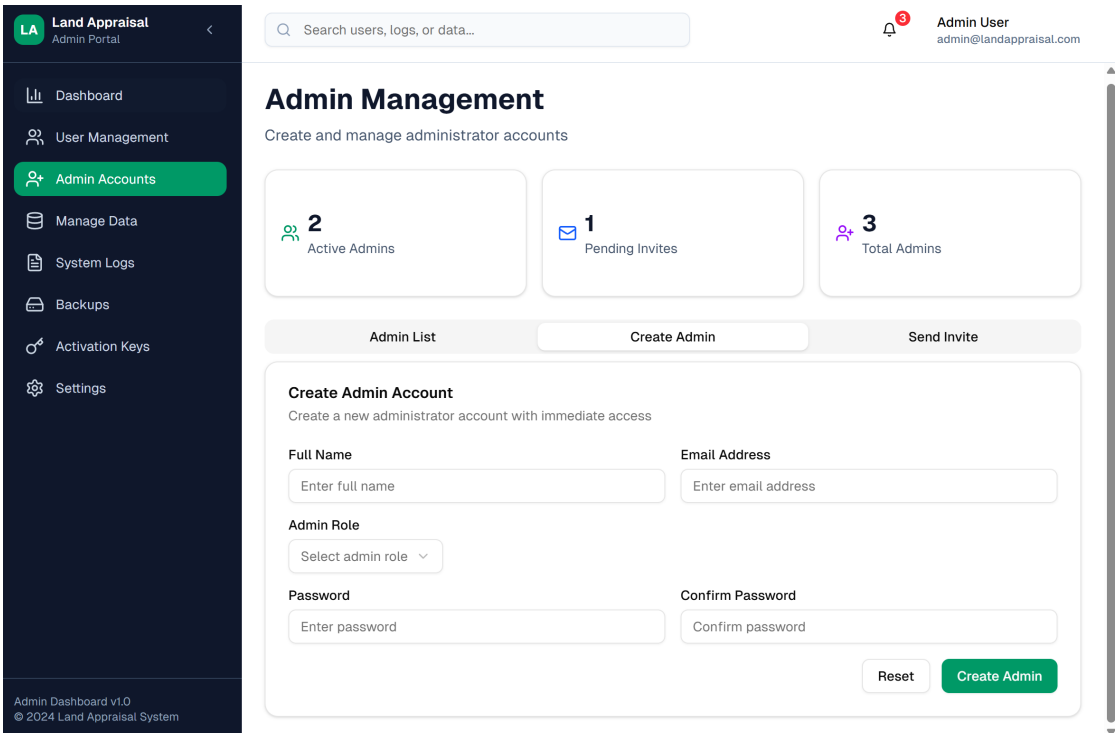


Figure 3.6.16 Create Admin Account

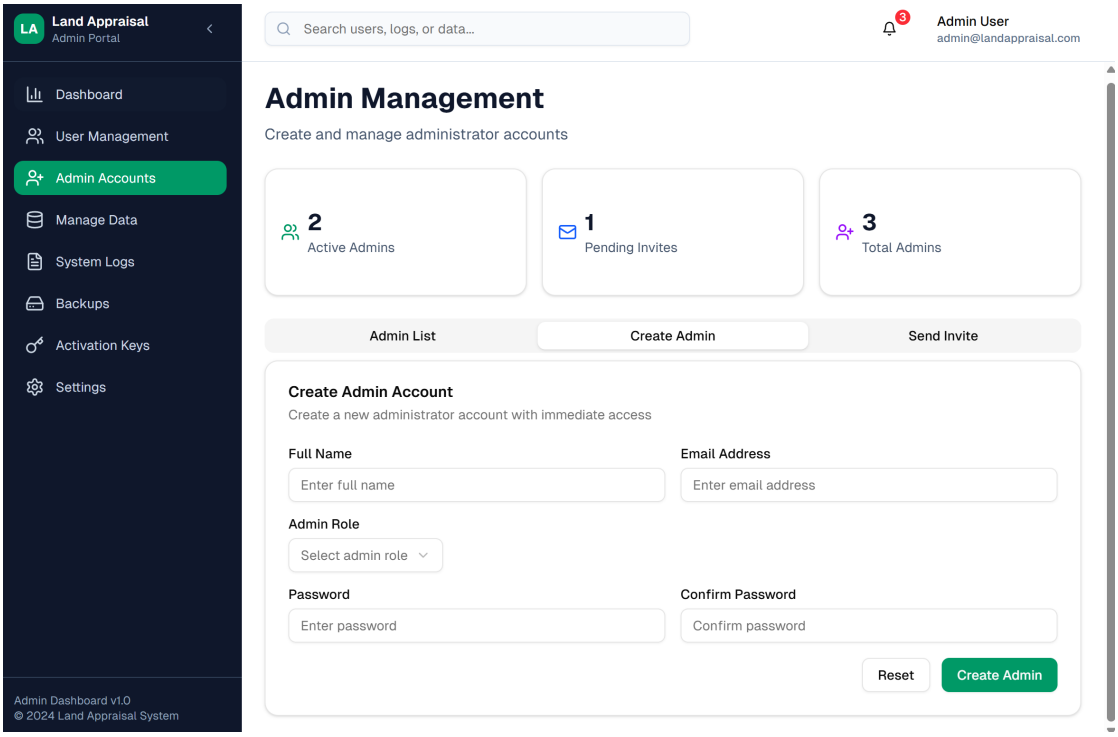


Figure 3.6.17 Manage Form Data

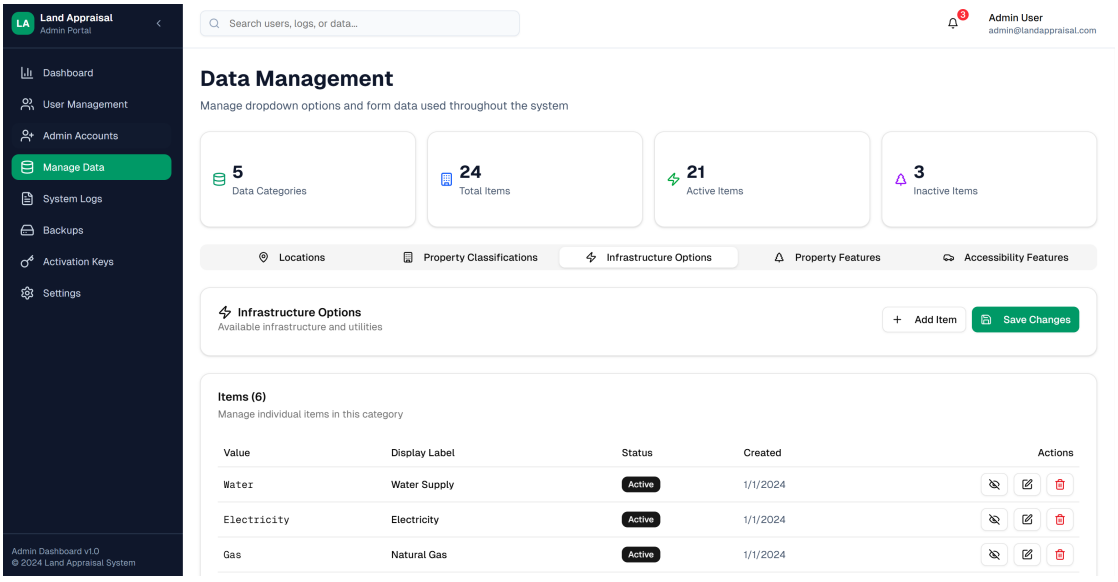


Figure 3.6.18 View System Logs

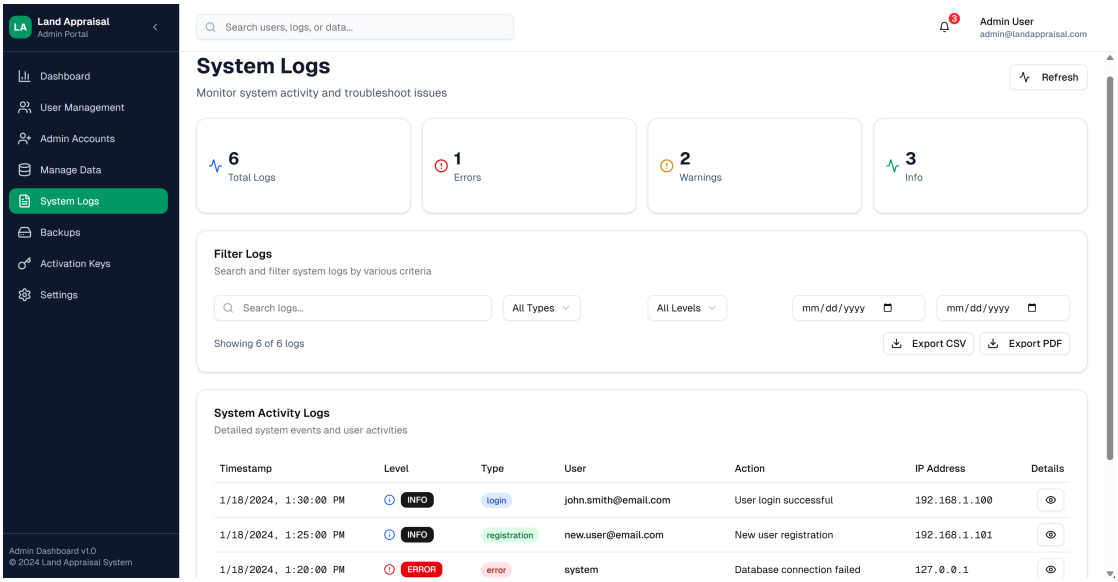


Figure 3.6.19 Manage Backups

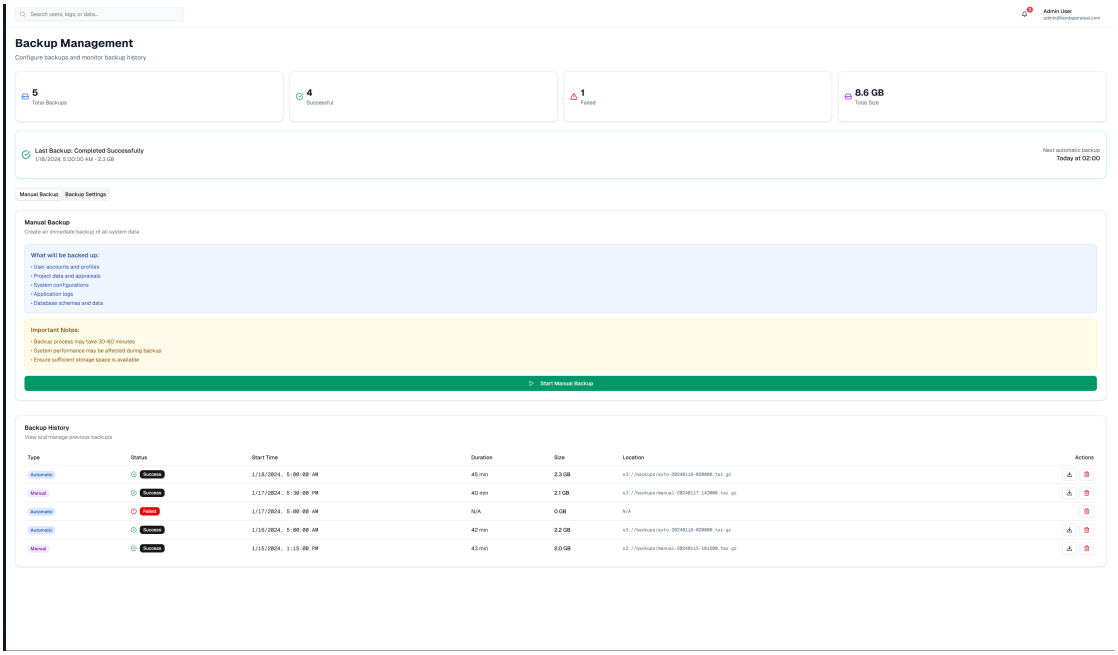
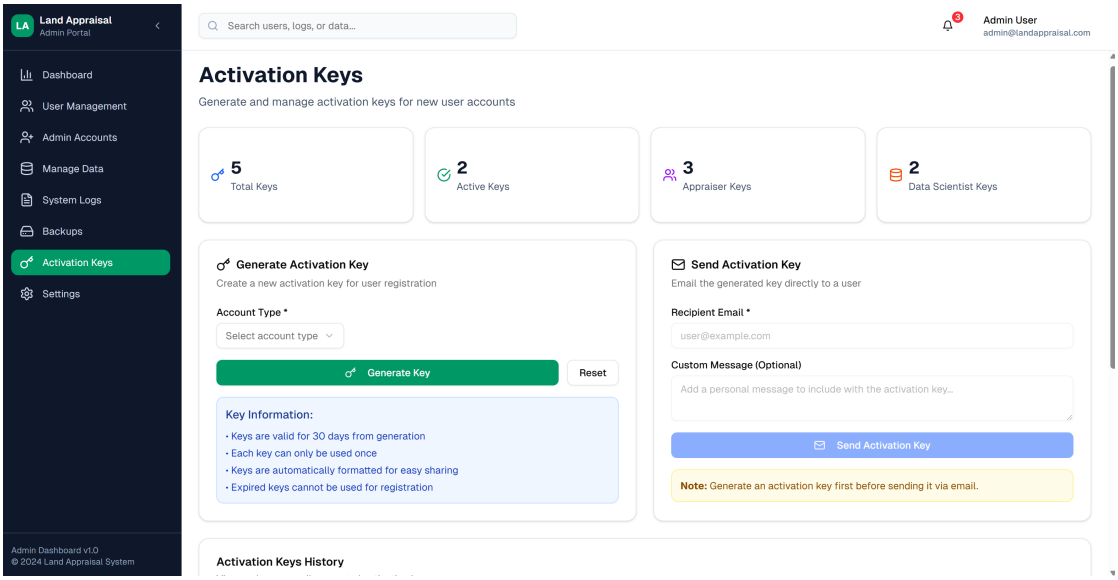


Figure 3.6.20 Create Activation Key



References

[1] J. Starmer, "Regression Trees, Clearly Explained!!!," *StatQuest with Josh Starmer*, YouTube, 2020. [Online]. Available: https://youtu.be/g9c66TUylZ4?si=0V35RgYavNRFB-yx. Accessed: Jul. 3, 2025.

[2] J. Starmer, "CatBoost Part 1: Ordered Target Encoding," *StatQuest with Josh Starmer*, YouTube, 2023. [Online]. Available: https://youtu.be/KXOTSkPL2X4?si=Iq890z0lxjhSlImH. Accessed: Jul. 3, 2025.

[3] J. Starmer, "CatBoost Part 2: Building and Using Trees," *StatQuest with Josh Starmer*, YouTube, 2023. [Online]. Available: https://youtu.be/3Bg2XRFOTzg?si=SUU2vVzNxbofFelA. Accessed: Jul. 3, 2025.