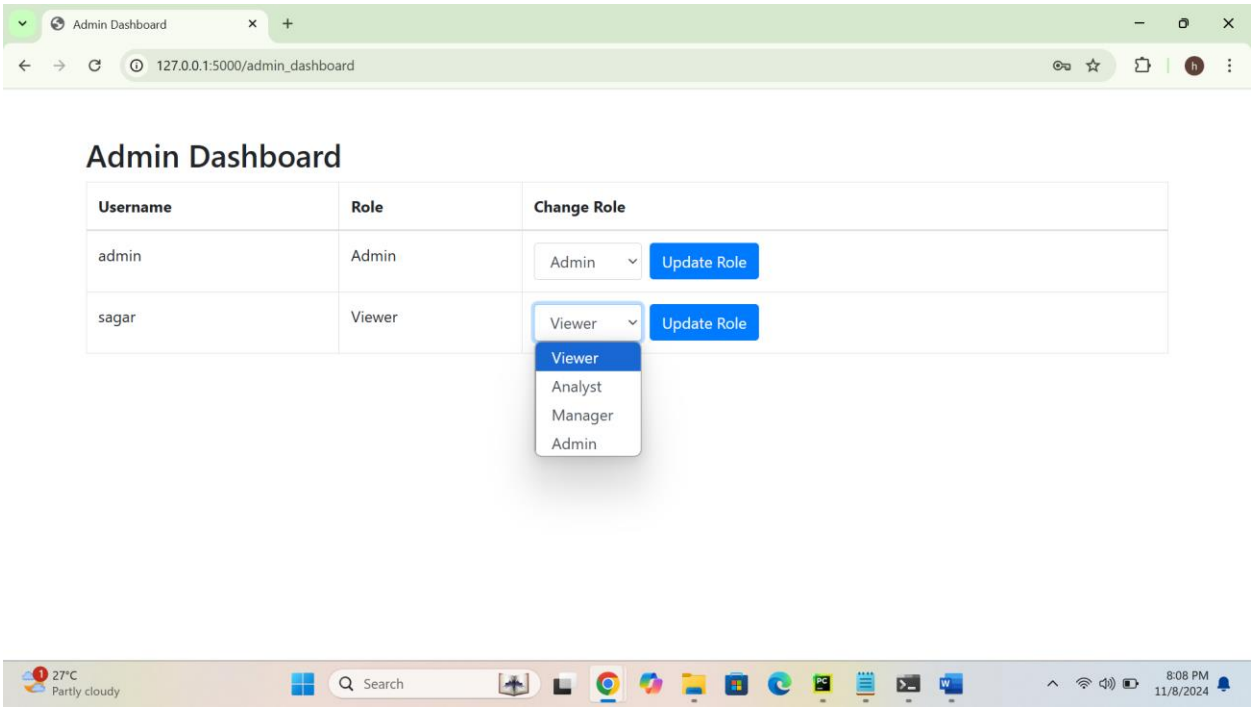
Real Estate Market Trend Dashboard App Documentation

Overview

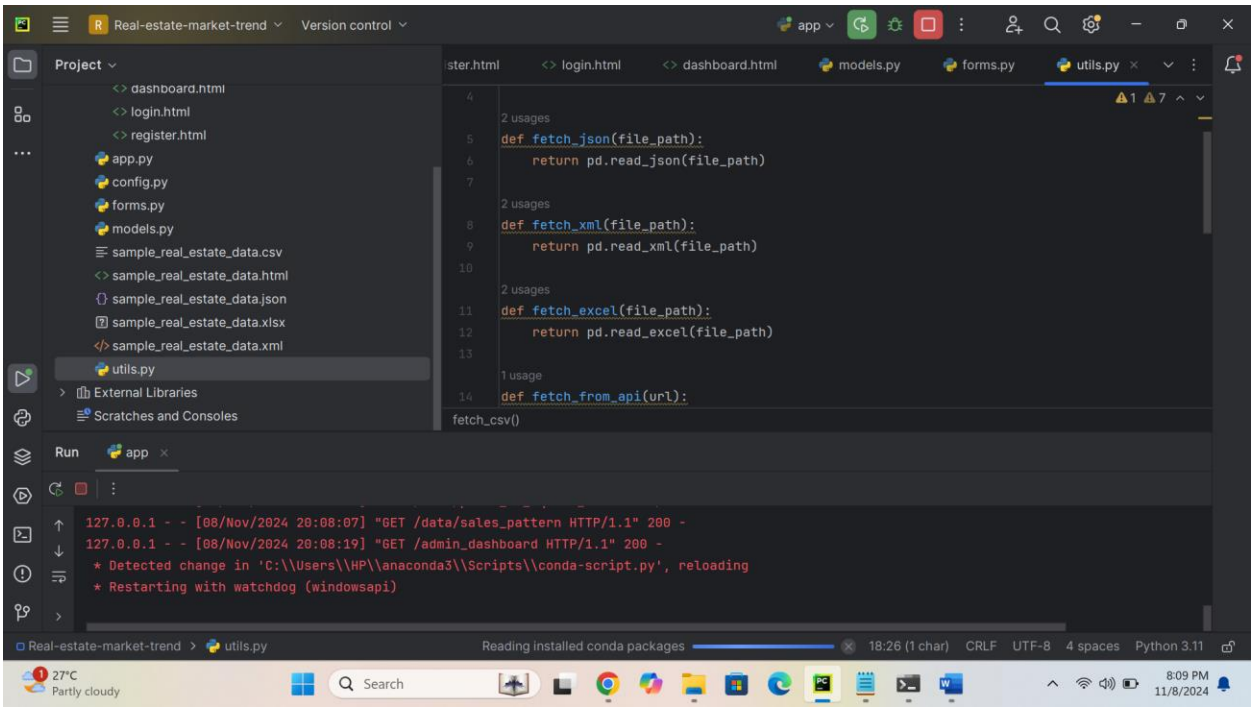
The Real Estate Market Trend Dashboard app is a Flask-based application designed for real estate data visualization with user authentication, role-based access, dynamic filtering, and database migration capabilities.

Key Features:

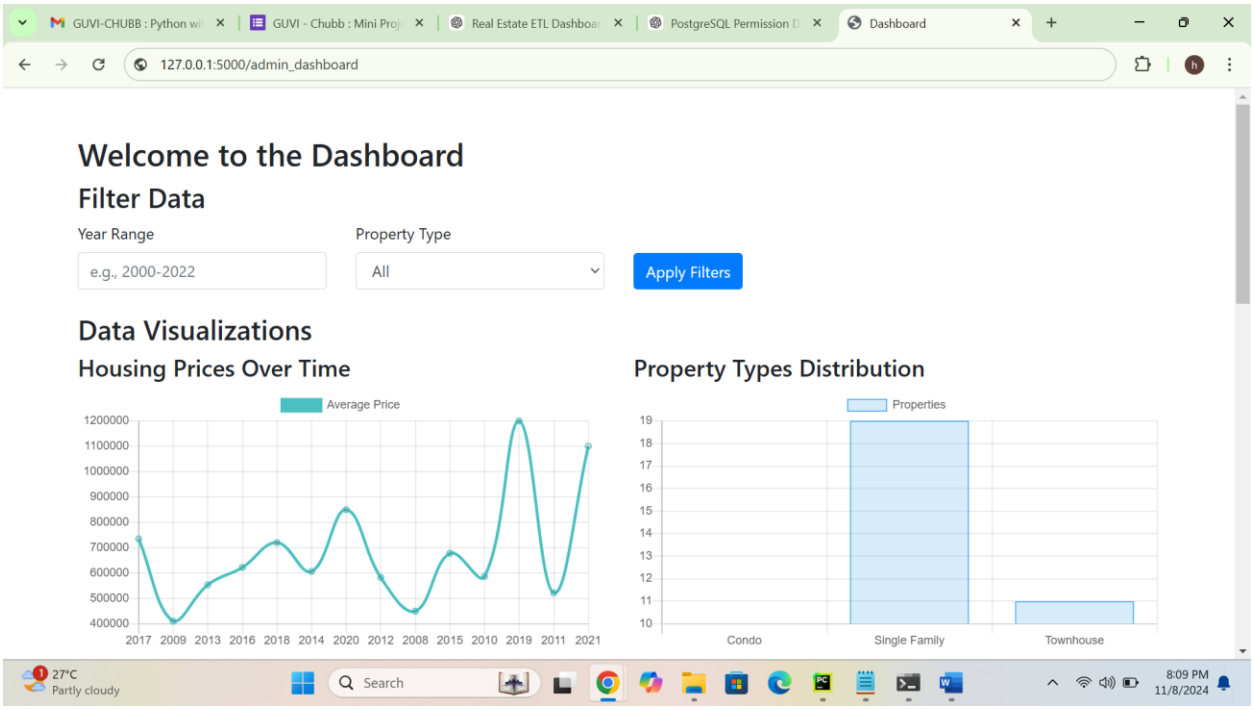
- **Role-Based Access Control: Admins can manage user roles, with each role seeing different dashboard views and having different levels of access.**

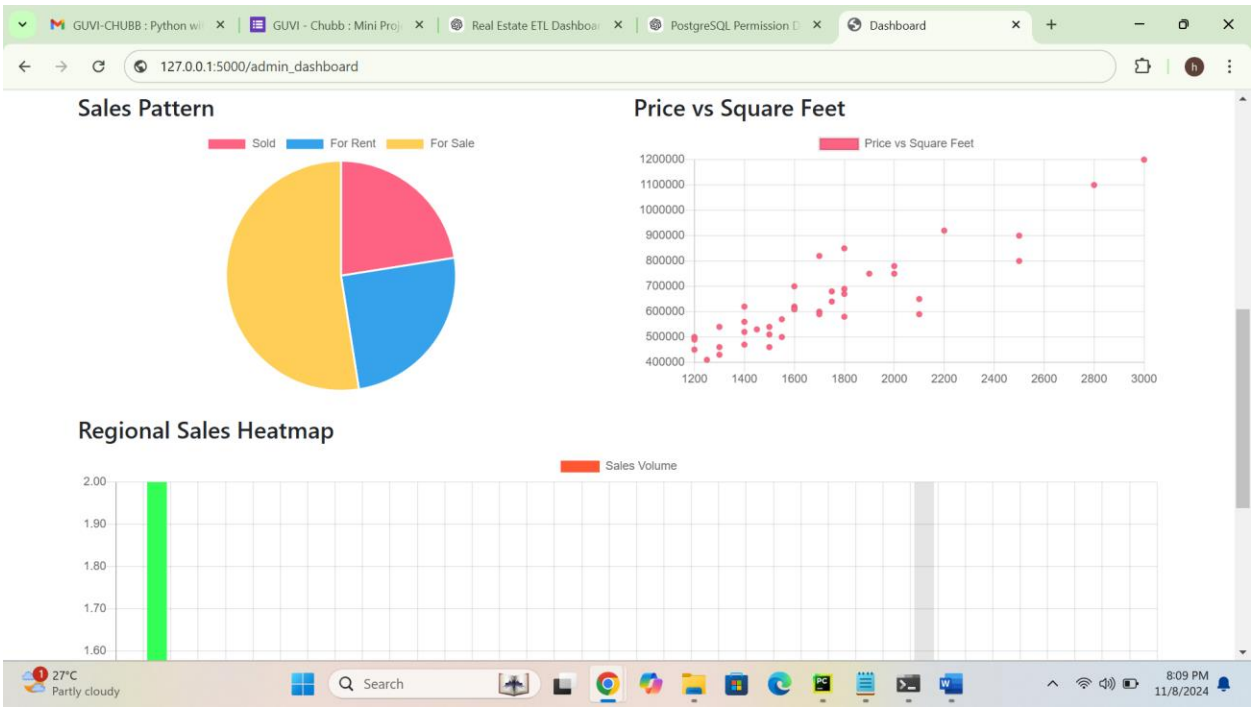


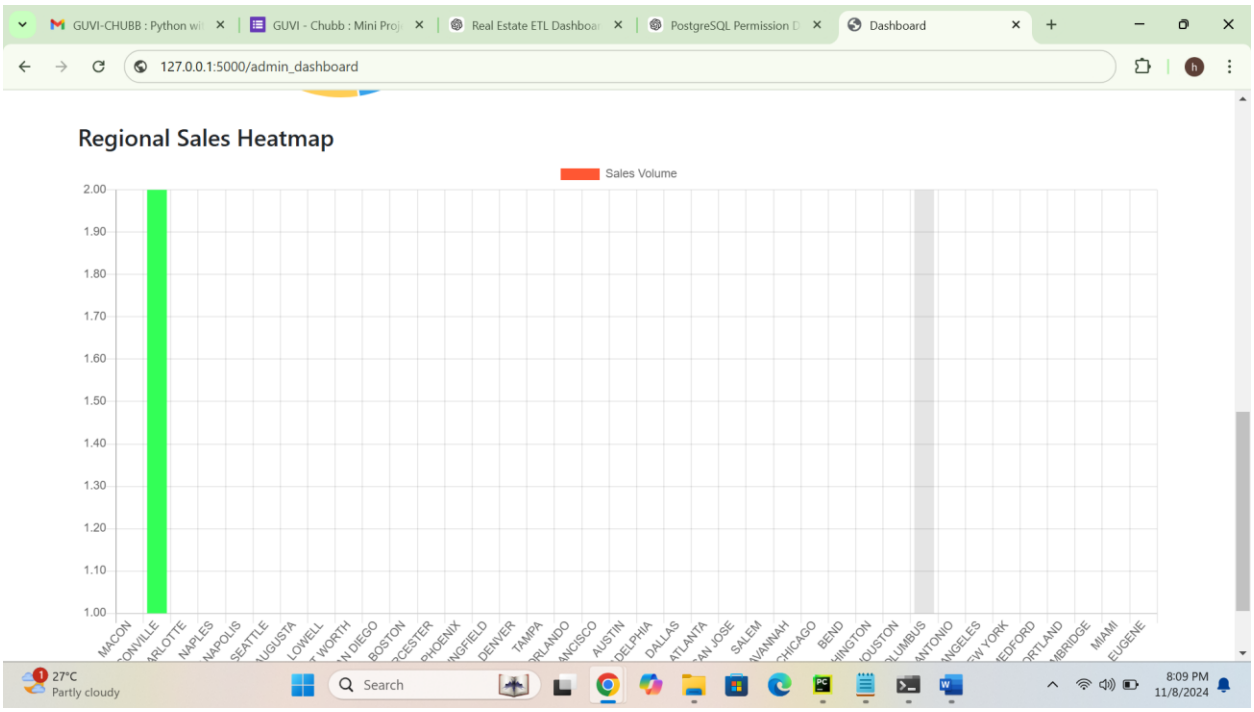
- **Data Fetching and Validation: Integrates data from multiple sources (JSON, XML, CSV, Excel, HTML) with validation/transformation before storing it in the database.**



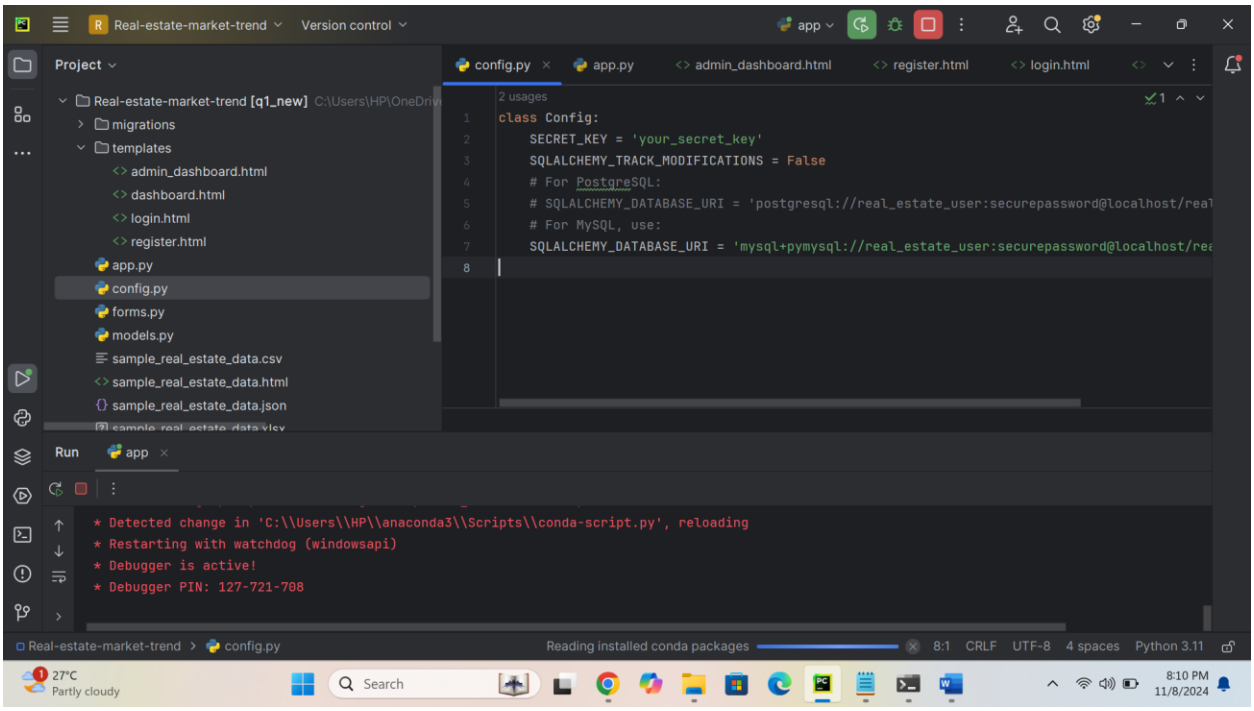
- **Data Visualization: Interactive charts for housing prices, property types, sales patterns, and more.**

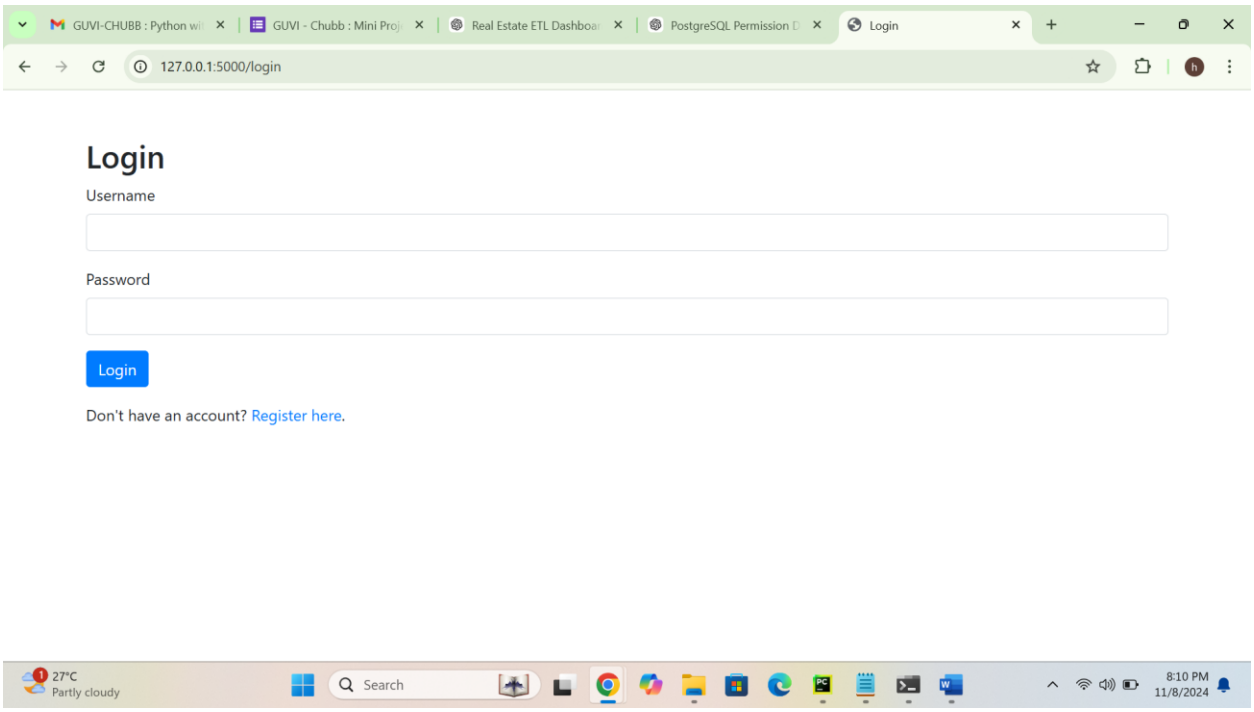


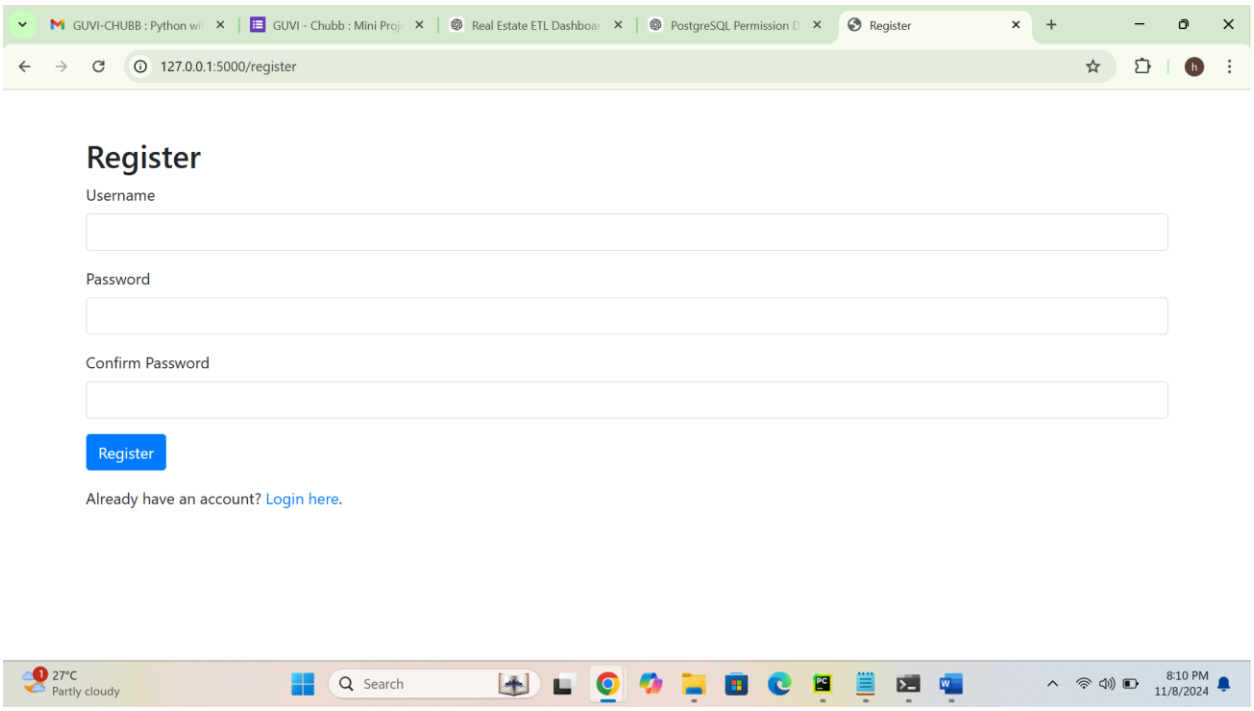




- **Dynamic Filtering: Filter data by year range and property type.**
- **Database Migration: Database schema can be easily updated to support new features or changes.**


Real-estate-market-trend
Version control
app
Project
Real-estate-market-trend [q1_new] C:\Users\HP\OneDriv
migrations
templates
admin_dashboard.html
dashboard.html
login.html
register.html
app.py
config.py
forms.py
models.py
sample_real_estate_data.csv
sample_real_estate_data.html
sample_real_estate_data.json
config.py
app.py
admin_dashboard.html
register.html
login.html
2 usages
class Config:
SECRET_KEY = 'your_secret_key'
SQLALCHEMY_TRACK_MODIFICATIONS = False
# For PostgreSQL:
# SQLALCHEMY_DATABASE_URI = 'postgresql://real_estate_user:securepassword@localhost/real
# For MySQL, use:
SQLALCHEMY_DATABASE_URI = 'mysql+pymysql://real_estate_user:securepassword@localhost/rea
Run
app
* Detected change in 'C:\\Users\\HP\\anaconda3\\Scripts\\conda-script.py', reloading
* Restarting with watchdog (windowsapi)
* Debugger is active!
* Debugger PIN: 127-721-708
Real-estate-market-trend
config.py
Reading installed conda packages
8:1
CRLF
UTF-8
4 spaces
Python 3.11
27°C
Partly cloudy
Search
8:10 PM
11/8/2024


GUVI-CHUBB : Python wi
GUVI - Chubb : Mini Proj
Real Estate ETL Dashboar
PostgreSQL Permission D
Login
127.0.0.1:5000/login
Login
Username
Password
Login
Don't have an account? Register here.
27°C
Partly cloudy
Search
8:10 PM
11/8/2024



Database Migrations

The app uses Flask-Migrate (powered by Alembic) for managing database schema changes. This allows you to add, modify, or remove columns and tables without disrupting existing data.

Typical Migration Workflow

1. **Make Changes to Models: Update the models in models.py to reflect new columns, tables, or relationships.**
2. **Create a New Migration: After changing the models, generate a new migration:**

flask db migrate -m

3. **Apply the Migration: Apply the changes to the database:**

flask db upgrade

4. **Downgrade (If Necessary): To roll back the last migration:**

flask db downgrade

Switching Databases (PostgreSQL and MySQL)

To switch between MySQL and PostgreSQL:

1. **Update SQLALCHEMY_DATABASE_URI in config.py to the new database URI.**
2. **Run flask db upgrade again to apply all migrations to the new database.**

Key Components

User Roles

The app includes the following roles:

1. **Admin: Full access to all features and user management.**
2. **Manager: Access to all visualizations.**
3. **Analyst: Limited access to charts and filters.**
4. **Viewer: Read-only access to basic visualizations.**

Data Fetching, Validation, and Transformation

The app fetches data from various formats (JSON, XML, CSV, HTML, Excel) and validates it. Key validations include checking for required fields and filtering data within specified ranges. This process is handled in utils.py.

Data Visualization

Visualizations are rendered with Chart.js in dashboard.html, based on data fetched via API routes from the Flask backend. Available charts include:

- **Housing Prices Over Time (Line Chart)**
- **Property Types Distribution (Bar Chart)**
- **Sales Pattern (Pie Chart)**
- **Price vs Square Feet (Scatter Plot)**
- **Regional Sales Heatmap (Bar Chart)**

Routes and Functionality

- **Authentication Routes: /register, /login, /logout.**
- **Dashboard Routes:**
    - **/dashboard: Displays charts with role-based access.**

    - **/admin_dashboard: Allows admin users to manage roles.**
- **Data API Routes:**
    - **/data/price_over_time: Returns price data by year.**
    - **/data/property_type_distribution: Returns distribution of property types.**
    - **/data/sales_pattern: Returns sales statuses.**
    - **/data/price_vs_square_feet: Returns property size vs. price.**
    - **/data/region_sales_heatmap: Returns sales volume by region.**

Sample Workflows

Admin Workflow

1. **Log in as admin (username: admin, password: admin).**
2. **Access /admin_dashboard to view and manage user roles.**
3. **Access /dashboard for full data filtering and visualization.**

New User Workflow

1. **Register through /register.**
2. **Log in and access /dashboard to view basic visualizations based on default role Viewer.**

Additional Information

Error Handling

1. **Database Errors: SQLAlchemy's IntegrityError catches errors during data insertion (e.g., duplicate property IDs).**
2. **Role-Based Access: Unauthorized access redirects users to the login page.**
3. **Data Validation: Data fetched from sources is validated and transformed before storage.**

Adding New Features

1. **Additional Roles: Update models.py, admin_dashboard.html, and role-based conditions in routes.**
2. **New Visualizations: Define new data API routes and add corresponding <canvas> elements and JavaScript logic in dashboard.html.**
3. **Extended Filtering: Add fields in the filter form in dashboard.html and update the backend routes.**