dev10-Capstone-Project

Title: Real Estate Price Prediction

Author: David McCain

This project is a web-based application designed to predict real estate prices based on features such as location, square meters, number of bedrooms, and additional property features. Users can also view existing properties and save their predictions for future updates or reference.

User Permissions

Key Functionalities:

1. **Input Property Details**
    - Users can input property details (location, square meters, number of bedrooms, and features) to receive price predictions.
2. **View Predicted Price**
    - Users can view the predicted price along with a breakdown of contributing factors.
3. **Search and Filter Properties**
    - Users can search through existing properties and apply filters based on location, price range, square meters, and additional features.
4. **Save and Edit Predictions**
    - Authenticated users can save property predictions. Saved predictions can be edited to reflect changes in property details, which will automatically update the predicted price.

Technologies Used

Frontend:

- **React** Used for dynamic user interfaces.
- **Bootstrap** Used for responsive design and styling.

Backend:

- **FastAPI**: FastAPI is a modern, fast (high-performance), web framework for building APIs with Python 3.7+ based on standard Python type hints. It is designed to be easy to use and to provide automatic interactive API documentation with Swagger and ReDoc. FastAPI is built on top of Starlette for the web parts and Pydantic for the data parts, ensuring high performance and ease of development.

- **SQLAlchemy**: SQLAlchemy is a powerful and flexible SQL toolkit and Object-Relational Mapping (ORM) library for Python. It provides a full suite of well-known enterprise-level persistence patterns, designed for efficient and high-performing database access. SQLAlchemy allows developers to work with databases in a more Pythonic way, using classes and objects to represent tables and rows, and provides a high-level abstraction for database operations.
- **Pydantic**: Pydantic is a data validation and settings management library for Python, using Python type annotations. It provides a way to define data models with type hints and validate data against these models, ensuring that the data conforms to the specified types and constraints.
- **Uvicorn**: Uvicorn is a lightning-fast ASGI server implementation, using `uvloop` and `httptools`. It is designed to be lightweight and efficient, making it ideal for serving high-performance web applications built with frameworks like FastAPI and Starlette.
- **Starlette**: Starlette is a lightweight ASGI framework/toolkit, which is ideal for building high-performance asyncio services. It provides a simple and flexible foundation for building web applications and includes features such as routing, middleware, and background tasks.
- **PyMySql**: PyMySql is a pure-Python MySQL client library. It allows you to connect to a MySQL database server from Python code, execute SQL queries, and retrieve results. It is a drop-in replacement for MySQLdb and provides a simple and consistent interface for interacting with MySQL databases.

Machine Learning: (Linear regression model using supervised learning)

- **Numpy**: Used for numerical computations, handling arrays, and performing mathematical operations efficiently.
- **Pandas**: Used for data manipulation, analysis, and providing data structures to handle and process data.
- **Scikit-learn**: Used for data preprocessing, model selection, training, and evaluation of machine learning algorithms.

Database:

- **MySQL** Used for structured data storage.
- Schema includes tables for Users, Locations, Features, Properties, Property Features, and Predicted Price History

Authentication:

- **OAuth2** with JWT tokens for secure user authentication.
- **FastAPI Security** utilities for handling authentication and authorization.
- **JOSE** for JWT token encoding and decoding.
- **Passlib** for password hashing and verification.
- **Bcrypt** used alongside Passlib for password hashing.
- **Dependencies** for managing authentication dependencies in routes.

Deployment: (stretch goal)

- Docker for containerization.
- Hosted on platforms like AWS, Heroku, or Render.

Database Schema

Tables:

1. **Users**
   - Stores user details and authentication information.
2. **Locations**
   - Stores detailed location data, including country, state/province, city, and address.
3. **Features**
   - Lists predefined features (e.g., Pool, Gym).
4. **Properties**
   - Stores property details, linked to Locations and Users.
5. **Property Features**
   - Many-to-many relationship between Properties and Features.
6. **Predicted Price History**
   - Stores logs of property prices (predicted from the model) with timestamps. Predictions are updated if details are edited.

Installation and Setup

Steps:

1. **Clone the repository**:

```
git clone https://github.com/dgmccain/dev10-Capstone-Project.git
cd dev10-Capstone-Project
```

2. **Set up the backend**:
   - Create a virtual environment:

```
python -m venv venv
source venv/bin/activate  # On Windows: venv\Scripts\activate
```

   - Install dependencies:

```
pip install -r requirements.txt
```

- Run the server:

```
uvicorn main:app --reload
ctrl + c # Quits running backend
```

3. **Set up the frontend**:
   - Navigate to the frontend directory:

```
cd frontend # Should be immediately within the main project
folder, or a src || app folder
```

   - Install dependencies:

```
npm install
```

   - Start the development server:

```
npm run dev # After deployment, instead run "npm start"
o # Opens webpage
q # Quits running frontend
```

4. **Access the application**:
   - Backend: `http://localhost:8000`
   - Frontend: `http://localhost:5173`

Contributing

Contributions are welcome! Currently the project is privated, but when it goes public here is how to contribute:

1. Fork the repository.
2. Create a new branch (`git checkout -b feature-branch`).
3. Commit your changes (`git commit -m 'Add new feature'`).
4. Push to the branch (`git push origin feature-branch`).
5. Open a pull request.

Future Enhancements

- Differentiate roles from users, owners, etc., and allow owners to post into the database "properties" table.
- Integrate more advanced AI/ML models for better predictions.
- Allow realtors to list properties directly for sale or rent.
- Enable export of predictions and data to PDF/Excel formats.
- Consider using NGINX and Heroku for deployment.

Important Notes

For Testing

run the following command in the terminal after navigating to the backend folder: "pytest tests" testing database credentials are hard-coded to a restricted test_user, but production database is stored in a .env file.

For requirements.txt (dependencies)

"brew install pkg-config" may need to be run as well

- it is a system package, not a Python package
