Phase 1 Dependencies Install in VS

Table of Contents

1. Phase 1 Dependencies Install in VS
    - PowerShell Script for Windows (setup_phase_1.ps1)
    - Usage Instructions
    - Example Project Structure
    - Script Contents for Phase 1
        - analyze_model.py
        - data_preparation.py
        - model_development.py
        - update_api.py
    - Summary
    - Setup Script (setup_phase1.sh)
    - Additional Environment Configuration (Optional)

- Save the Updated Script

2. Steps to Install Dependencies for Phase 1 in Visual Studio
3. Script to Automate the Setup (setup_phase1.bat for Windows)
4. Script to Automate the Setup (setup_phase1.sh for macOS/Linux)
5. How to Use the Script in Visual Studio

PowerShell Script for Windows (`setup_phase_1.ps1`)

```
# Check if the current environment is running as an administr
ator
$currentPrincipal = New-Object Security.Principal.WindowsPrin
cipal([Security.Principal.WindowsIdentity]::GetCurrent())
if (-not $currentPrincipal.IsInRole([Security.Principal.Windo
wsBuiltInRole]::Administrator)) {
    Write-Host "Please run this script as an administrator."
    exit
}

# Function to install pip if not installed
function Install-Pip {
    $pipUrl = "<https://bootstrap.pypa.io/get-pip.py>"
    $pipPath = "get-pip.py"
    Invoke-WebRequest -Uri $pipUrl -OutFile $pipPath
    python $pipPath
    Remove-Item $pipPath
}

# Check if Python is installed
try {
    python --version
} catch {
    Write-Host "Python is not installed. Please install Pytho
n and run the script again."
    exit
```

```
}

# Check if pip is installed
try {
    pip --version
} catch {
    Write-Host "pip is not installed. Installing pip..."
    Install-Pip
}

# Create virtual environment
Write-Host "Creating virtual environment..."
python -m venv venv

# Activate virtual environment
Write-Host "Activating virtual environment..."
& .\\venv\\Scripts\\Activate.ps1

# Install dependencies
Write-Host "Installing dependencies..."
pip install pandas scikit-learn xgboost Flask joblib pytest p
ylint

Write-Host "Virtual environment and dependencies setup comple
te."
Write-Host "To activate the virtual environment, run: .\\venv
\\Scripts\\Activate.ps1"
```

Usage Instructions

1. **Save the Script:**
    - Save the script as `setup_phase_1.ps1` in the root directory of your Visual Studio project.
2. **Open PowerShell:**

- Open PowerShell as an administrator.

3. **Navigate to the Script Directory:**
   - Use `cd` to navigate to the directory where `setup_phase_1.ps1` is located.
4. **Run the Script:**
   - Execute the script by typing `.\\setup_phase_1.ps1` and pressing Enter.

Example Project Structure

Ensure your project structure looks something like this:

```
MyProject
|-- venv/
|-- scripts/
|   |-- analyze_model.py
|   |-- data_preparation.py
|   |-- model_development.py
|   |-- update_api.py
|-- setup_phase_1.ps1
|-- run_pipeline.py
|-- test/
|   |-- test_analyze_model.py
|   |-- test_data_preparation.py
|   |-- test_model_development.py
|   |-- test_pipeline.py
|-- README.md
```

Script Contents for Phase 1

analyze_model.py

```
from sklearn.metrics import accuracy_score, precision_score,
recall_score, f1_score, mean_absolute_error
import joblib
import pandas as pd
```

```
import logging

logging.basicConfig(level=logging.INFO)

def evaluate_model(model, X_test, y_test):
    try:
        predictions = model.predict(X_test)
        metrics = {
            'accuracy': accuracy_score(y_test, predictions),
            'precision': precision_score(y_test, predictions,
average='macro'),
            'recall': recall_score(y_test, predictions, avera
ge='macro'),
            'f1_score': f1_score(y_test, predictions, average
='macro'),
            'mean_absolute_error': mean_absolute_error(y_tes
t, predictions)
        }
        logging.info("Model evaluation successful.")
        return metrics
    except Exception as e:
        logging.error(f"Error evaluating model: {e}")
        raise

if __name__ == "__main__":
    try:
        model = joblib.load('property_valuation_model.pkl')
        X_test = pd.read_csv('X_test.csv')
        y_test = pd.read_csv('y_test.csv')
        metrics = evaluate_model(model, X_test, y_test)
        print(metrics)
    except Exception as e:
        logging.error(f"Error running analysis: {e}")
```

data_preparation.py

```
import pandas as pd
import logging

logging.basicConfig(level=logging.INFO)

def load_data():
    try:
        demographics = pd.read_csv('neighborhood_demographic
s.csv')
        school_info = pd.read_csv('school_district_info.csv')
        market_trends = pd.read_csv('real_estate_market_trend
s.csv')
        property_records = pd.read_csv('property_records.cs
v')
        public_records = pd.read_csv('public_records.csv')
        economic_indicators = pd.read_csv('economic_indicator
s.csv')
        social_data = pd.read_csv('social_data.csv')
        geospatial_data = pd.read_csv('geospatial_data.csv')
        logging.info("Data loaded successfully.")
        return demographics, school_info, market_trends, prop
erty_records, public_records, economic_indicators, social_dat
a, geospatial_data
    except Exception as e:
        logging.error(f"Error loading data: {e}")
        raise

def feature_engineering(df):
    try:
        df['price_per_sqft'] = df['price'] / df['sqft']
        df['age_of_property'] = 2024 - df['year_built']
        logging.info("Feature engineering successful.")
        return df
    except Exception as e:
        logging.error(f"Error in feature engineering: {e}")
```

```
        raise

if __name__ == "__main__":
    try:
        demographics, school_info, market_trends, property_re
cords, public_records, economic_indicators, social_data, geos
patial_data = load_data()
        all_data = pd.concat([demographics, school_info, mark
et_trends, property_records, public_records, economic_indicat
ors, social_data, geospatial_data], axis=1)
        all_data = feature_engineering(all_data)
        all_data.to_csv('prepared_data.csv', index=False)
        logging.info("Data preparation complete.")
    except Exception as e:
        logging.error(f"Error in data preparation: {e}")
```

model_development.py

```
import xgboost as xgb
from sklearn.model_selection import train_test_split
import pandas as pd
import logging

logging.basicConfig(level=logging.INFO)

def train_model(X, y):
    try:
        model = xgb.XGBRegressor()
        model.fit(X, y)
        logging.info("Model training successful.")
        return model
    except Exception as e:
        logging.error(f"Error training model: {e}")
        raise
```

```
if __name__ == "__main__":
    try:
        data = pd.read_csv('prepared_data.csv')
        X = data.drop(columns=['price'])
        y = data['price']
        X_train, X_test, y_train, y_test = train_test_split
(X, y, test_size=0.2, random_state=42)
        model = train_model(X_train, y_train)
        model.save_model('new_property_valuation_model.json')
        metrics = evaluate_model(model, X_test, y_test)
        print(metrics)
    except Exception as e:
        logging.error(f"Error in model development: {e}")
```

update_api.py

```
from flask import Flask, request, jsonify
import xgboost as xgb
import pandas as pd
import logging

logging.basicConfig(level=logging.INFO)

app = Flask(__name__)

try:
    model = xgb.XGBRegressor()
    model.load_model('new_property_valuation_model.json')
    logging.info("Model loaded successfully.")
except Exception as e:
    logging.error(f"Error loading model: {e}")
    raise

@app.route('/predict', methods=['POST'])
def predict():
```

```
    try:
        data = request.get_json()
        df = pd.DataFrame(data)
        prediction = model.predict(df)
        return jsonify({'prediction': prediction.tolist()})
    except Exception as e:
        logging.error(f"Error in prediction: {e}")
        return jsonify({'error': str(e)}), 500

if __name__ == '__main__':
    app.run(debug=True)
```

Summary

This PowerShell script ensures that your environment for Phase 1 is properly configured and ready for development in Visual Studio. It sets up a virtual environment, installs necessary dependencies, and provides instructions for activation. Remember to run this script in an elevated PowerShell session (Run as Administrator) to ensure proper execution.

Setup Script (setup_phase1.sh)

This script will:

1. Create a virtual environment.
2. Activate the virtual environment.
3. Install the required dependencies.
4. Set up the necessary environment variables.

```
bashCopy code
#!/bin/bash

# Create a virtual environment
echo "Creating virtual environment..."
python -m venv venv
```

```
# Activate the virtual environment
echo "Activating virtual environment..."
source venv/bin/activate

# Install dependencies
echo "Installing dependencies..."
pip install pandas scikit-learn xgboost Flask joblib

# Verify installation
echo "Verifying installation..."
pip freeze

echo "Setup complete. To activate the virtual environment, us
e 'source venv/bin/activate' on macOS/Linux or 'venv\\Scripts
\\activate' on Windows."
```

Usage Instructions:

1. **Create the Script:**
   Save the above script as
   `setup_phase1.sh` in your project directory.
2. **Run the Script:**
   Open a terminal in your project directory and run the following command to make the script executable and then execute it:

   ```
   bashCopy code
   chmod +x setup_phase1.sh
   ./setup_phase1.sh
   ```

Additional Environment Configuration (Optional)

If there are any additional environment variables or configurations needed for your project, you can add them to the script.

Example:

```
bashCopy code
#!/bin/bash

# Create a virtual environment
echo "Creating virtual environment..."
python -m venv venv

# Activate the virtual environment
echo "Activating virtual environment..."
source venv/bin/activate

# Install dependencies
echo "Installing dependencies..."
pip install pandas scikit-learn xgboost Flask joblib

# Set environment variables (add any necessary variables her
e)
echo "Setting environment variables..."
export FLASK_APP=update_api.py
export FLASK_ENV=development

# Verify installation
echo "Verifying installation..."
pip freeze

echo "Setup complete. To activate the virtual environment, us
e 'source venv/bin/activate' on macOS/Linux or 'venv\\Scripts
\\activate' on Windows."
```

Save the Updated Script

1. **Save the Script:**
   Save the updated script as

`setup_phase1.sh` in your project directory.

2. **Run the Script:**
   Open a terminal in your project directory and run the following command to make the script executable and then execute it:

```
bashCopy code
chmod +x setup_phase1.sh
./setup_phase1.sh
```

This script will create a virtual environment, install all necessary dependencies, and set up any required environment variables for Phase 1 of your project.

now to install the depandcey for phase 1 in VS

To install the dependencies for Phase 1 in Visual Studio (VS), you can follow these steps:

Steps to Install Dependencies for Phase 1 in Visual Studio

1. **Open Your Project:**
   Open your project in Visual Studio.
2. **Open Terminal in Visual Studio:**
   - Go to `View` > `Terminal` or use the shortcut `Ctrl +` (backtick) to open the integrated terminal.
3. **Create and Activate a Virtual Environment:**
   - In the terminal, navigate to your project directory if you are not already there.
   - Run the following commands to create and activate a virtual environment:

```
bashCopy code
# Create a virtual environment
python -m venv venv

# Activate the virtual environment
# On Windows
venv\Scripts\activate
# On macOS/Linux
source venv/bin/activate
```

4. **Install Dependencies:**
   - With the virtual environment activated, run the following command to install the required dependencies:

```
bashCopy code
pip install pandas scikit-learn xgboost Flask joblib
```

5. **Verify Installation:**
   - Run the following command to verify that all dependencies have been installed correctly:

```
bashCopy code
pip freeze
```

6. **Optional - Set Environment Variables:**
   - If you need to set any environment variables, you can do so in the terminal:

```
bashCopy code
# Example of setting environment variables
```

```
export FLASK_APP=update_api.py
export FLASK_ENV=development
```

Script to Automate the Setup (setup_phase1.bat for Windows)

To automate the setup process in Visual Studio, you can create a batch script for Windows:

```
batCopy code
@echo off

:: Create a virtual environment
echo Creating virtual environment...
python -m venv venv

:: Activate the virtual environment
echo Activating virtual environment...
call venv\Scripts\activate

:: Install dependencies
echo Installing dependencies...
pip install pandas scikit-learn xgboost Flask joblib

:: Verify installation
echo Verifying installation...
pip freeze

echo Setup complete. To activate the virtual environment in t
he future, use 'venv\Scripts\activate'.
pause
```

Script to Automate the Setup (setup_phase1.sh for macOS/Linux)

For macOS/Linux users, you can use the previously provided `setup_phase1.sh` script:

```
bashCopy code
#!/bin/bash

# Create a virtual environment
echo "Creating virtual environment..."
python -m venv venv

# Activate the virtual environment
echo "Activating virtual environment..."
source venv/bin/activate

# Install dependencies
echo "Installing dependencies..."
pip install pandas scikit-learn xgboost Flask joblib

# Verify installation
echo "Verifying installation..."
pip freeze

echo "Setup complete. To activate the virtual environment, us
e 'source venv/bin/activate' on macOS/Linux or 'venv\Scripts
\activate' on Windows."
```

How to Use the Script in Visual Studio

1. **Create the Script File:**
   - Save the script as `setup_phase1.bat` for Windows or `setup_phase1.sh` for macOS/Linux in your project directory.
2. **Run the Script:**
   - Open the terminal in Visual Studio (`View` > `Terminal`).
   - Navigate to your project directory.
   - Run the script:

- For Windows:

```
bashCopy code
setup_phase1.bat
```

- For macOS/Linux:

```
bashCopy code
chmod +x setup_phase1.sh
./setup_phase1.sh
```

This will automate the process of setting up your environment and installing the necessary dependencies for Phase 1 in Visual Studio.

design and implementation plan for the frontend user interface for property search, viewing property details, and managing favorites using Blazor:

Embedded Ai agents