Complete setup

Complete Setup

Microsoft Standard Directory Structure

Here is a refined directory structure for Dwellingly.ai. This format aligns with Microsoft's standards for creating scalable and secure web applications on Azure. The structure incorporates best practices for deployment, security, scalability, and maintainability.

```
bashdwellingly_ai/|├── src/|   ├── app/|   |   ├── __init__.py|   |   ├── routes.py|   |
├── models.py|   |   ├── forms.py|   |   ├── templates/|   |   |   ├── base.html|   |   |
├── index.html|   |   |   ├── signup.html|   |   |   ├── login.html|   |   |   ├──
dashboard.html|   |   |   ├── admin-settings.html|   |   |   ├── create-cma.html|   |   |
├── create-offers.html|   |   └── static/|   |       ├── css/|   |       ├── js/|   |
├── images/|   ||   ├── tests/|   |   ├── __init__.py|   |   ├── test_routes.py|   |   ├──
test_models.py|   |   ├── test_forms.py|   ||   ├── .env|   ├── .gitignore|   ├── config.py|
├── manage.py|   ├── requirements.txt|   ├── run.py|├── deployment/|   ├── azure-
pipelines.yml|   ├── Dockerfile|   ├── docker-compose.yml|   ├── infrastructure/|   |   ├──
main.bicep|   |   ├── network.bicep|   |   ├── storage.bicep|   |   ├── appservice.bicep|├──
docs/|   ├── architecture.md|   ├── setup_guide.md|   ├── user_guide.md|   ├──
api_reference.md|└── scripts/    ├── setup_env.sh    ├── install_dependencies.sh    ├──
run_tests.sh
```

Detailed Overview

1. **src/app**: This directory houses the core application code including routes, models, forms, templates, and static files.
2. **src/tests**: This directory contains unit tests and integration tests to ensure the code functions as expected.
3. **deployment**: This directory stores CI/CD pipeline configurations, Docker files, and Infrastructure as Code (IaC) scripts.
4. **docs**: This directory includes all the documentation related to the project, such as architecture, setup guides, user guides, and API references.

5. **scripts**: This directory contains shell scripts for setting up the environment, installing dependencies, and running tests.

Updating the Code to Follow the New Structure

1. Transfer Existing Code into `src/app`

```
bashmv routes.py models.py forms.py templates static src/app/
```

2. Modify Import Paths

Adjust the import paths in your Python files to match the new structure.

3. Add Deployment Configuration

azure-pipelines.yml (for Azure DevOps)

```
pool:  vmImage: 'ubuntu-latest'
variables:  PIP_CACHE_DIR: $(Pipeline.Workspace)/.pip_cache
steps:- task: UsePythonVersion@0  inputs:    versionSpec: '3.
x'    addToPath: true
- script: |    python -m venv venv    source venv/bin/activat
e    python -m pip install --upgrade pip    pip install -r sr
c/app/requirements.txt  displayName: 'Install dependencies'
- script: |    source venv/bin/activate    python -m unittest
discover -s src/tests  displayName: 'Run Tests'
- task: CopyFiles@2  inputs:    Contents: '**'    TargetFolde
r: '$(Build.ArtifactStagingDirectory)'
- task: PublishBuildArtifacts@1  inputs:    PathtoPublish:
'$(Build.ArtifactStagingDirectory)'    ArtifactName: 'drop'``
`
**Dockerfile**
```dockerfile# Use an official Python runtime as a parent ima
geFROM python:3.9-slim
# Set the working directoryWORKDIR /app
# Copy the current directory contents into the container at /
appCOPY src/app /app
# Install any needed packages specified in requirements.txtRU
```
```

```
N pip install --no-cache-dir -r requirements.txt
# Make port 80 available to the world outside this containerE
XPOSE 80
# Define environment variableENV NAME Dwellingly.ai
# Run app.py when the container launchesCMD ["python", "run.p
y"]```
**docker-compose.yml**
```yamlversion: '3.8'
services:  app:    build: .    ports:      - "5000:80"    env
ironment:      - FLASK_ENV=development      - FLASK_APP=run.p
y    volumes:      - .:/app    command: flask run --host=0.0.
0.0 --port=80```
**infrastructure/main.bicep**
```bicepresource group 'dwellingly-ai-rg' existingmodule netw
ork './network.bicep' = {  name: 'network'  params: {    vnet
Name: 'dwellingly-ai-vnet'    subnetName: 'default'  }}
module storage './storage.bicep' = {  name: 'storage'  param
s: {    storageAccountName: 'dwellinglyaistorage'    locatio
n: resourceGroup().location  }}
module appservice './appservice.bicep' = {  name: 'appservic
e'  params: {    appServicePlanName: 'dwellingly-ai-plan'
webAppName: 'dwellingly-ai-app'    location: resourceGroup().
location    storageAccountName: storage.outputs.storageAccoun
tName    vnetName: network.outputs.vnetName    subnetName: ne
twork.outputs.subnetName  }}```
**scripts/setup_env.sh**
```bash#!/bin/bash# Create a virtual environmentpython3 -m ve
nv venv
# Activate the virtual environmentsource venv/bin/activate
# Install dependenciespip install --upgrade pippip install -r
src/app/requirements.txt```
**scripts/install_dependencies.sh**
```bash#!/bin/bashsource venv/bin/activatepip install -r src/
app/requirements.txt```
**scripts/run_tests.sh**
```bash#!/bin/bashsource venv/bin/activatepython -m unittest
```
```

```
discover -s src/tests```
### Update Forms, Templates, and Static Files
Make sure that forms, templates, and static files are correct
ly referenced in the updated structure.
### Documentation and Best Practices
Generate comprehensive documentation for architecture, setup,
and user guides in the `docs` directory.
**docs/architecture.md**
```markdown# Architecture Overview
## Components- Flask for the web application- Azure for deplo
yment and hosting- Docker for containerization- PostgreSQL fo
r the database- Azure Storage for static files
## Deployment- CI/CD pipeline using Azure DevOps- Infrastruct
ure as Code using Bicep
```

docs/setup_guide.md

```
## Prerequisites- Python 3.9- Docker- Azure CLI- Azure DevOps
account
## Steps1. Clone the repository2. Navigate to the project dir
ectory3. Run `./scripts/setup_env.sh`4. Deploy using Azure De
vOps pipeline
```

docs/user_guide.md

```
## Features- User signup and login- Property valuation- CMA c
reation- Offer submission
## Usage1. Navigate to the landing page2. Sign up for a new a
ccount3. Log in to access the dashboard4. Use the dashboard t
o create CMAs and submit offers
```

docs/api_reference.md

```
## Endpoints- `/signup`: User signup- `/login`: User login-
`/dashboard`: User dashboard- `/admin-settings`: Admin settin
gs- `/create-cma`: Create CMA- `/create-offers`: Create offer
```
```

```
s
## Models- User- Property- Offer- CMA
```

Completing the Setup

After updating the structure and content, follow the steps below to ensure everything is set up correctly:

1. **Initialize Git Repository** `bash git init git add . git commit -m "Initial commit with updated structure"`
2. **Setup Virtual Environment** `bash ./scripts/setup_env.sh`
3. **Run Tests** `bash ./scripts/run_tests.sh`
4. **Deploy to Azure** Utilize the Azure DevOps pipeline and Bicep files to deploy the infrastructure and application.
   By adhering to these steps and using this directory structure, you will ensure that Dwellingly.ai adheres to industry standards for scalability, security, and best practices. This setup will contribute to maintaining a robust, scalable, and secure web application ready for deployment and use.

Microsoft Standard Directory Structure

Here's a revised directory structure for Dwellingly.ai, which aligns with Microsoft's standards for building scalable and secure web applications on Azure. This structure includes best practices for deployment, security, scalability, and maintainability.

```bash
dwellingly_ai/
├── src/
│   ├── app/
│   │   ├── __init__.py
│   │   ├── routes.py
│   │   ├── models.py
│   │   ├── forms.py
│   │   ├── templates/
│   │   │   ├── base.html
│   │   │   ├── index.html
│   │   │   ├── signup.html
│   │   │   ├── login.html
│   │   │   ├── dashboard.html
│   │   │   ├── admin-settings.html
│   │   │   ├── create-cma.html
│   │   │   ├── create-offers.html
│   │   └── static/
│   │       ├── css/
│   │       ├── js/
│   │       ├── images/
│   │
│   ├── tests/
│   │   ├── __init__.py
│   │   ├── test_routes.py
│   │   ├── test_models.py
│   │   ├── test_forms.py
│   │
│   ├── .env
│   ├── .gitignore
│   ├── config.py
│   ├── manage.py
│   ├── requirements.txt
│   ├── run.py
├── deployment/
│   ├── azure-pipelines.yml
│   ├── Dockerfile
│   ├── docker-compose.yml
│   ├── infrastructure/
│   │   ├── main.bicep
│   │   ├── network.bicep
│   │   ├── storage.bicep
│   │   ├── appservice.bicep
├── docs/
│   ├── architecture.md
│   ├── setup_guide.md
│   ├── user_guide.md
│   ├── api_reference.md
└── scripts/
    ├── setup_env.sh
    ├── install_dependencies.sh
    ├── run_tests.sh
```

Detailed Explanation
1. **src/app**: This directory contains the core application code including routes, models, forms, templates, and static files.2. **src/tests**: Contains unit tests and integration tests to ensure the code works as expected.3. **deployment**: Contains CI/CD pipeline configurations, Docker files, and infrastructure as code (IaC) scripts.4. **docs**: Contains all the documentation related to the project, including architecture, setup guides, user guides, and API references.5. **scripts**: Contains shell scripts for setting up the environment, installing dependencies, and running tests.
Update the Code to Follow the New Structure
1. Move Existing Code into `src/app`
```bashmv routes.py models.py forms.py templates static src/app/```
2. Update Import Paths
Adjust the import paths in your Python files to reflect the new structure.
3. Add Deployment Configuration
azure-pipelines.yml (for Azure DevOps)
```yamltrigger:- main
pool:  vmImage: 'ubuntu-latest'
variables:  PIP_CACHE_DIR: $(Pipeline.Workspace)/.pip_cache
steps:- task: UsePythonVersion@0  inputs:    versionSpec: '3.x'    addToPath: true
- script: |    python -m venv venv    source venv/bin/activate    python -m pip install --upgrade pip pip install -r src/app/requirements.txt  displayName: 'Install dependencies'
- script: |    source venv/bin/activate    python -m unittest discover -s src/tests  displayName: 'Run Tests'
- task: CopyFiles@2  inputs:    Contents: '**'    TargetFolder: '$(Build.ArtifactStagingDirectory)'
- task: PublishBuildArtifacts@1  inputs:    PathtoPublish: '$(Build.ArtifactStagingDirectory)' ArtifactName: 'drop'```
**Dockerfile**
```dockerfile# Use an official Python runtime as a parent imageFROM python:3.9-slim
# Set the working directoryWORKDIR /app
# Copy the current directory contents into the container at /appCOPY src/app /app
# Install any needed packages specified in requirements.txtRUN pip install --no-cache-dir -r requirements.txt
# Make port 80 available to the world outside this containerEXPOSE 80
# Define environment variableENV NAME Dwellingly.ai
# Run app.py when the container launchesCMD ["python", "run.py"]```
**docker-compose.yml**
```

```yamlversion: '3.8'
services: app: build: . ports: - "5000:80" environment: - FLASK_ENV=development
- FLASK_APP=run.py volumes: - .:/app command: flask run --host=0.0.0.0 --port=80```
**infrastructure/main.bicep**
```bicepresource group 'dwellingly-ai-rg' existingmodule network './network.bicep' = { name:
'network' params: { vnetName: 'dwellingly-ai-vnet' subnetName: 'default' }}
module storage './storage.bicep' = { name: 'storage' params: { storageAccountName:
'dwellinglyaistorage' location: resourceGroup().location }}
module appservice './appservice.bicep' = { name: 'appservice' params: {
appServicePlanName: 'dwellingly-ai-plan' webAppName: 'dwellingly-ai-app' location:
resourceGroup().location storageAccountName: storage.outputs.storageAccountName
vnetName: network.outputs.vnetName subnetName: network.outputs.subnetName }}```
**scripts/setup_env.sh**
```bash#!/bin/bash# Create a virtual environmentpython3 -m venv venv
# Activate the virtual environmentsource venv/bin/activate
# Install dependenciespip install --upgrade pippip install -r src/app/requirements.txt```
**scripts/install_dependencies.sh**
```bash#!/bin/bashsource venv/bin/activatepip install -r src/app/requirements.txt```
**scripts/run_tests.sh**
```bash#!/bin/bashsource venv/bin/activatepython -m unittest discover -s src/tests```
### Update Forms, Templates, and Static Files
Ensure that forms, templates, and static files are correctly referenced in the updated structure.
### Documentation and Best Practices
Create comprehensive documentation for architecture, setup, and user guides in the `docs` directory.
**docs/architecture.md**
```markdown# Architecture Overview
## Components- Flask for the web application- Azure for deployment and hosting- Docker for containerization- PostgreSQL for the database- Azure Storage for static files
## Deployment- CI/CD pipeline using Azure DevOps- Infrastructure as Code using Bicep
```
docs/setup_guide.md
```markdown# Setup Guide
## Prerequisites- Python 3.9- Docker- Azure CLI- Azure DevOps account
## Steps1. Clone the repository2. Navigate to the project directory3. Run
`./scripts/setup_env.sh`4. Deploy using Azure DevOps pipeline

```

docs/user_guide.md

```markdown# User Guide

## Features- User signup and login- Property valuation- CMA creation- Offer submission

## Usage1. Navigate to the landing page2. Sign up for a new account3. Log in to access the dashboard4. Use the dashboard to create CMAs and submit offers

```

docs/api_reference.md

```markdown# API Reference

## Endpoints- `/signup`: User signup- `/login`: User login- `/dashboard`: User dashboard- `/admin-settings`: Admin settings- `/create-cma`: Create CMA- `/create-offers`: Create offers

## Models- User- Property- Offer- CMA

```

Finalizing the Setup

After updating the structure and content, follow the below steps to ensure everything is set up correctly:

1. **Initialize Git Repository** ```bash git init git add . git commit -m "Initial commit with updated structure" ```
2. **Setup Virtual Environment** ```bash ./scripts/setup_env.sh ```
3. **Run Tests** ```bash ./scripts/run_tests.sh ```
4. **Deploy to Azure** Use the Azure DevOps pipeline and Bicep files to deploy the infrastructure and application.

By following these steps and using this directory structure, you will ensure that Dwellingly.ai adheres to industry standards for scalability, security, and best practices. This setup will help in maintaining a robust, scalable, and secure web application ready for deployment and use.
```