Guide to Creating and Deploying a Django Project with AI and ML Integration

1 Introduction

This guide provides a step-by-step approach to creating and deploying a Django project integrated with AI and ML functionalities, using only free technologies and services.

2 Setup and Development

2.1 Install Required Tools

- **Python:** Download and install from `https://www.python.org/downloads/`.
- **Django:** Install Django using pip.

```
pip install django
```

- **Git:** Install Git from `https://git-scm.com/downloads`.
- **Text Editor/IDE:** Install [Visual Studio Code](https://code.visualstudio.com/) or [PyCharm Community Edition](https://www.jetbrains.com/pycharm/download).

2.2 Create a Django Project

```
django-admin startproject myportfolio
cd myportfolio
python manage.py startapp chatbot
```

2.3 Develop the Chatbot Integration

2.3.1 Choose a Chatbot Framework

- **Dialogflow:** Create a Dialogflow agent at `https://dialogflow.cloud.google.com/`.

2.3.2 Install Dialogflow Client Library

```
pip install google-cloud-dialogflow
```

2.3.3 Setup Dialogflow Authentication

```
export GOOGLE_APPLICATION_CREDENTIALS="path/to/your/service-account-file.json"
```

2.3.4 Create Django Views for Chatbot

```
# chatbot/views.py
from django.shortcuts import render
from django.http import JsonResponse
from google.cloud import dialogflow_v2 as dialogflow
from google.oauth2 import service_account

def chatbot_view(request):
    return render(request, 'chatbot/chat.html')

def chat(request):
    text = request.GET.get('text')
    project_id = 'your-project-id'
    credentials = service_account.Credentials.from_service_account_file('path/to/your/keyfil
    client = dialogflow.SessionsClient(credentials=credentials)
    session_id = 'unique-session-id'
    session = client.session_path(project_id, session_id)

    text_input = dialogflow.TextInput(text=text, language_code='en')
    query_input = dialogflow.QueryInput(text=text_input)
    response = client.detect_intent(session=session, query_input=query_input)

    return JsonResponse({'fulfillmentText': response.query_result.fulfillment_text})
```

2.3.5 Add URL Routing

```
# chatbot/urls.py
from django.urls import path
from . import views

urlpatterns = [
    path('', views.chatbot_view, name='chatbot'),
    path('chat/', views.chat, name='chat'),
]
```

2.3.6 Create Frontend for Chatbot

```
<!-- chatbot/templates/chatbot/chat.html -->
```

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>Chatbot</title>
    <script>
        async function sendMessage() {
            const message = document.getElementById('message').value;
            const response = await fetch(`/chatbot/chat/?text=${message}`);
            const data = await response.json();
            document.getElementById('chat').innerHTML += `<div>User: ${message}</div><div>Bo
            document.getElementById('message').value = '';
        }
    </script>
</head>
<body>
    <h1>Chatbot</h1>
    <div id="chat"></div>
    <input type="text" id="message" />
    <button onclick="sendMessage()">Send</button>
</body>
</html>
```

2.4 Implement AI/ML Features

2.4.1 Build and Integrate ML Models

```
# chatbot/ml_model.py
import tensorflow as tf

def load_model():
    model = tf.keras.models.load_model('path/to/your/model')
    return model

def predict(input_data):
    model = load_model()
    prediction = model.predict(input_data)
    return prediction
```

2.4.2 Expose ML Functionality through Django Views

```
# chatbot/views.py
from .ml_model import predict

def ml_predict(request):
    input_data = request.GET.get('input')
```

```
    prediction = predict(input_data)
    return JsonResponse({'prediction': prediction.tolist()})
```

2.4.3 Update URLs

```
# chatbot/urls.py
urlpatterns = [
    path('ml_predict/', views.ml_predict, name='ml_predict'),
]
```

3 Deployment

3.1 Prepare for Deployment

3.1.1 Create `requirements.txt`

```
pip freeze > requirements.txt
```

3.1.2 Create `Procfile`

```
web: gunicorn myportfolio.wsgi
```

3.1.3 Add Configuration for Static and Media Files

```
# myportfolio/settings.py
STATIC_URL = '/static/'
MEDIA_URL = '/media/'
STATIC_ROOT = os.path.join(BASE_DIR, 'staticfiles')
MEDIA_ROOT = os.path.join(BASE_DIR, 'media')
```

Collect static files:

```
python manage.py collectstatic
```

3.2 Deploy to Heroku

3.2.1 Sign Up for Heroku

- Go to `https://www.heroku.com/` and create a free account.

3.2.2 Install Heroku CLI

- Follow installation instructions at `https://devcenter.heroku.com/articles/heroku-cli`.

3.2.3 Login to Heroku

```
heroku login
```

3.2.4 Create a New Heroku App

```
heroku create your-app-name
```

3.2.5 Deploy Your Code

```
git add .
git commit -m "Deploying to Heroku"
git push heroku main
```

3.2.6 Set Environment Variables

```
heroku config:set DJANGO_SECRET_KEY='your-secret-key'
```

3.2.7 Run Migrations

```
heroku run python manage.py migrate
```

3.2.8 Open Your App

```
heroku open
```

3.3 Alternative Deployment with Railway

3.3.1 Sign Up for Railway

- Go to `https://railway.app/` and create a free account.

3.3.2 Deploy from GitHub

- Connect your GitHub repository to Railway and follow the deployment steps.

3.3.3 Set Environment Variables

- Add environment variables through the Railway dashboard.

3.3.4 Deploy and Monitor

- Railway will automatically build and deploy your project.

4 Post-Deployment

- **Monitor Logs:** Check for errors and performance issues.
- **Update and Maintain:** Regularly update your project and models as needed.

- **Collect Feedback:** Gather user feedback to improve the chatbot and AI features.