OXZON.AI

AI/ML Interns Task Plan

Guide - Creating, Hosting, and Using Your Own API

User Story - 0054

Task 1: Design and Build Your API

1. Choose a Framework:
- Use Flask (lightweight) or FastAPI (fast and asynchronous-friendly) for Python.

2. Define Endpoints:
- Plan what your API will do and define the endpoints (e.g., GET, POST).
- Example: a simple API with a greeting endpoint.

3. Write the Code:
- Create a new Python file (e.g., app.py) and write your API code:

```
from flask import Flask, jsonify, request

app = Flask(__name__)

@app.route('/greet', methods=['GET'])
def greet():
    name = request.args.get('name', 'World')
    return jsonify({'message': f'Hello, {name}!'})

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

4. Test Locally:
- Run `python app.py` in your terminal to start the server.
- Open Postman:
  - Enter the URL `http://127.0.0.1:5000/greet`.
  - Set the method to `GET`.

- Add a query parameter: key=`name`, value=`YourName`.
- Send the request and check for a response like `{"message": "Hello, YourName!"}`.

Task 2: Prepare for Deployment

1. Add Dependencies:
- Create a `requirements.txt` file with dependencies by running:

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

2. Create a `Procfile` (for platforms like Heroku):
- Add this line to a file named `Procfile`:

```
web: python app.py
```

Task 3: Deploy to a Free Hosting Platform

Option A: Heroku
- Create a Heroku Account at heroku.com.
- Install Heroku CLI: Follow Heroku's installation guide.
- Deploy to Heroku:
  - Initialize Git: `git init`
  - Add files: `git add .`
  - Commit changes: `git commit -m "Initial commit"`
  - Create Heroku app: `heroku create`
  - Deploy: `git push heroku master`
- Test your API by visiting the provided Heroku URL.

Option B: Render (Another Free Platform)
- Create a Render Account at render.com.
- Create a New Web Service:
  - Link your GitHub repository or upload your code manually.
  - Set Python as the environment.
  - Specify `app.py` as the start command.
- Render will provide a public URL for your API once it's live.

Task 4: Test and Use Your Deployed API

1. Access the Deployed URL:
- Once deployed, visit your API URL (e.g., `https://your-api-url.com/greet?name=John`) to verify that it works.

2. Make API Calls from a Client:
- Example Python code to call your deployed API:

```
import requests

response = requests.get('https://your-api-url.com/greet', params={'name': 'John'})
print(response.json())  # Expected output: {'message': 'Hello, John!'}
```

3. Use Postman to Test the Deployed API:
- Enter the deployed URL in Postman and follow the same steps as in local testing to confirm it works as expected.

Task 5: Prepare a full Documentation

Add Screenshots and Proper Documentation

Optional: Add Authentication and Security

1. For securing your API, consider adding API keys or using OAuth.

2. Free platforms may have limitations on security features, but you can often manage this in-app.
