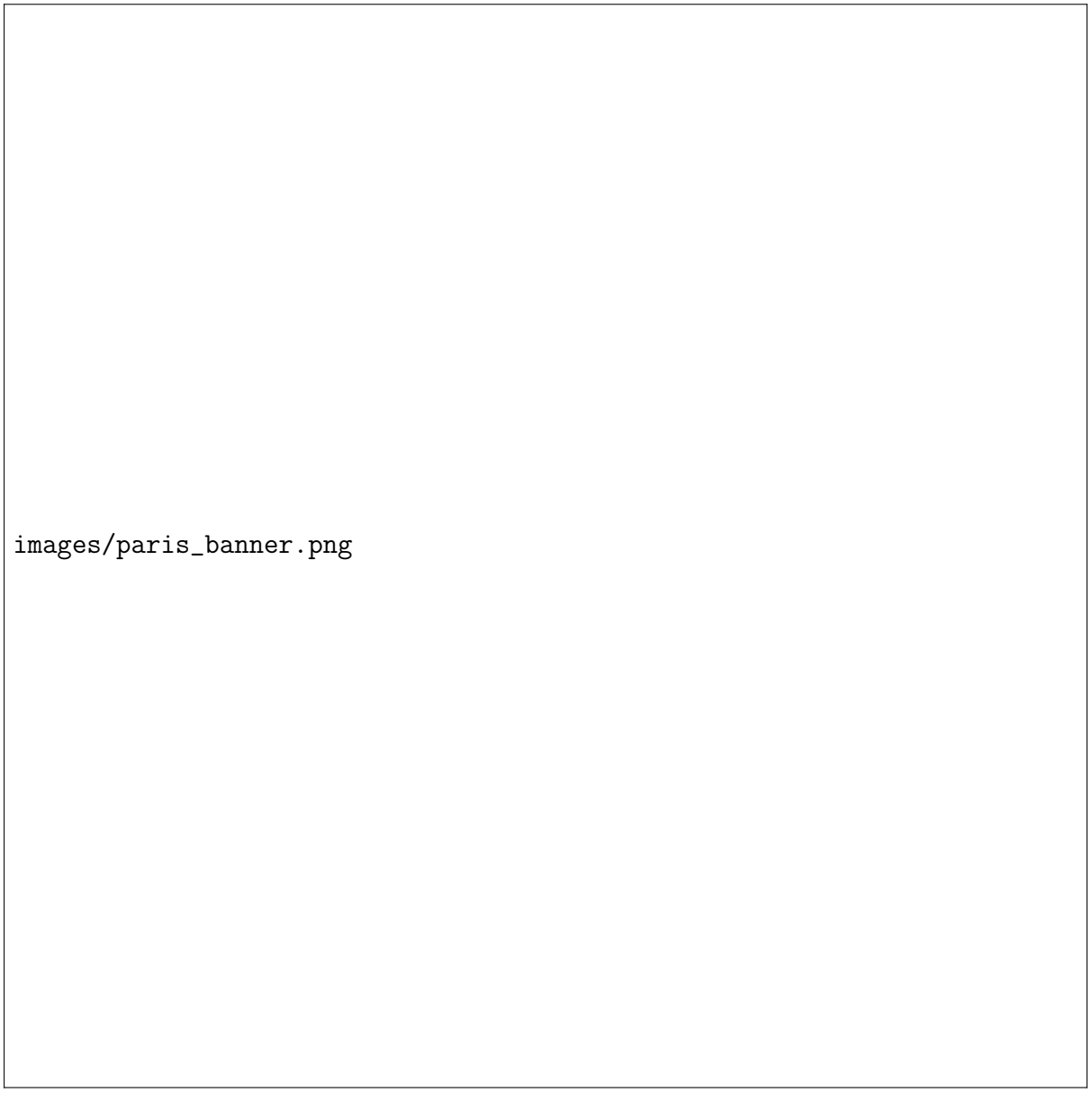

AI Paris Travel Chatbot

Comprehensive Stakeholder Project Report

Prepared by:

Olalemi John Oluwatosin

AI Developer, Peterman Reality Tours

November 12, 2025

Contents

Chapter 1

Introduction

This report presents a detailed account of the **AI-powered Paris Travel Chatbot**, an intelligent virtual assistant developed for Peterman Reality Tours. Initially designed as a CLI tool, the project has now evolved into a visually appealing **Streamlit web application**, offering a modern user interface for tourists and stakeholders.

The chatbot delivers real-time, context-aware, and factually accurate responses about Paris landmarks, travel routes, museums, and cultural attractions, leveraging OpenAI's API.

Chapter 2

Project Objectives

- Provide reliable and instant travel guidance about Paris.
- Enhance user engagement through an intuitive web interface (Streamlit).
- Showcase the integration of OpenAI's GPT-4o-mini model for natural language interaction.
- Ensure maintainable, modular, and portable architecture for scalability.
- Demonstrate best practices for AI deployment, including security and transparency.

Chapter 3

System Overview

The Paris Travel Chatbot now supports both:

1. A **CLI version** — for testing, logging, and debugging.
2. A **Streamlit Web App** — for an interactive, end-user experience.

3.1 Key Technologies

- Python 3.10+ for backend logic.
- OpenAI GPT-4o-mini model for AI-driven responses.
- Streamlit for the graphical user interface.
- Rich for enhanced CLI formatting.
- Docker & Docker Compose for containerization.
- dotenv for secure API key management.

Chapter 4

Project Architecture

4.1 Folder Structure

Below is the well-structured, hierarchical layout of the Paris Travel Chatbot project, showing both the CLI and Streamlit application components:

```
Paris-Travel-Chatbot/
|
+-- src/
|   +-- paris_chatbot.py            # Command-Line Interface (CLI) chatbot script
|   +-- streamlit_app.py            # Streamlit web interface for interactive chat
|   +-- utils/                      # Utility modules (helpers, logging, etc.)
|       +-- __init__.py
|       +-- helpers.py
|
+-- data/
|   +-- conversation_log.json       # Chat logs automatically generated
|
+-- images/
|   +-- paris_banner.png            # Project banner (used in README and Streamlit)
|   +-- streamlit_home.png          # Streamlit homepage screenshot
|   +-- streamlit_chat.png          # Conversation interface screenshot
|   +-- streamlit_results.png       # Example of chatbot output
|
```

```
+-- requirements.txt          # Python dependencies
+-- Dockerfile                # Docker container configuration
+-- docker-compose.yml        # Orchestration for multi-container setup
+-- .env.example              # Example environment variable file
+-- README.md                 # Project documentation with banner
+-- .gitignore                # Git ignored files and directories
+-- LICENSE                   # Open-source license information
```

4.2 User Interface Screenshots

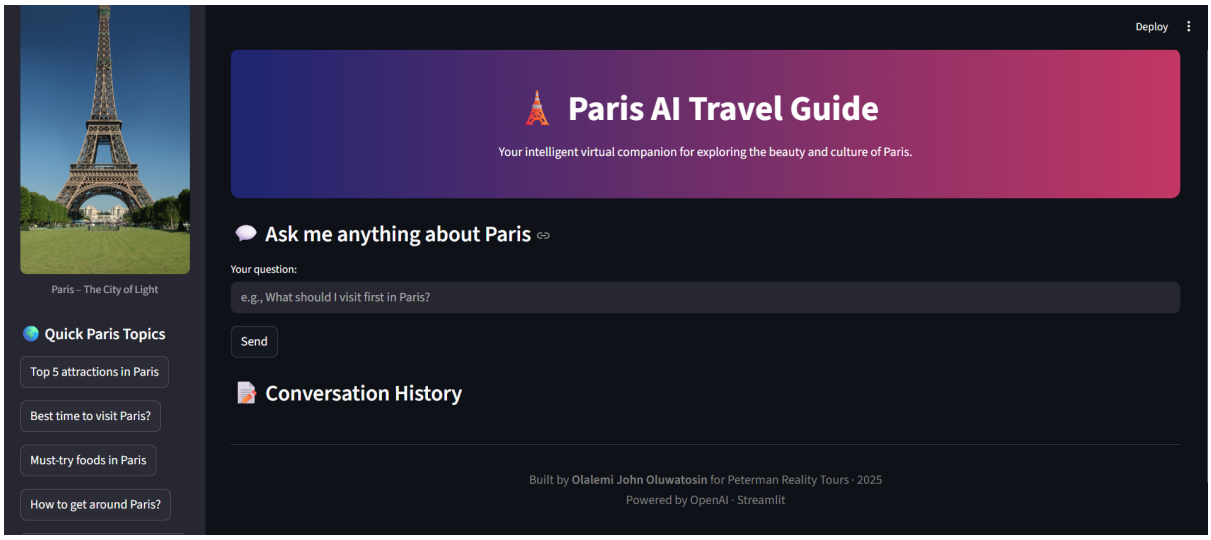

Figure 4.1: Streamlit App – Homepage interface

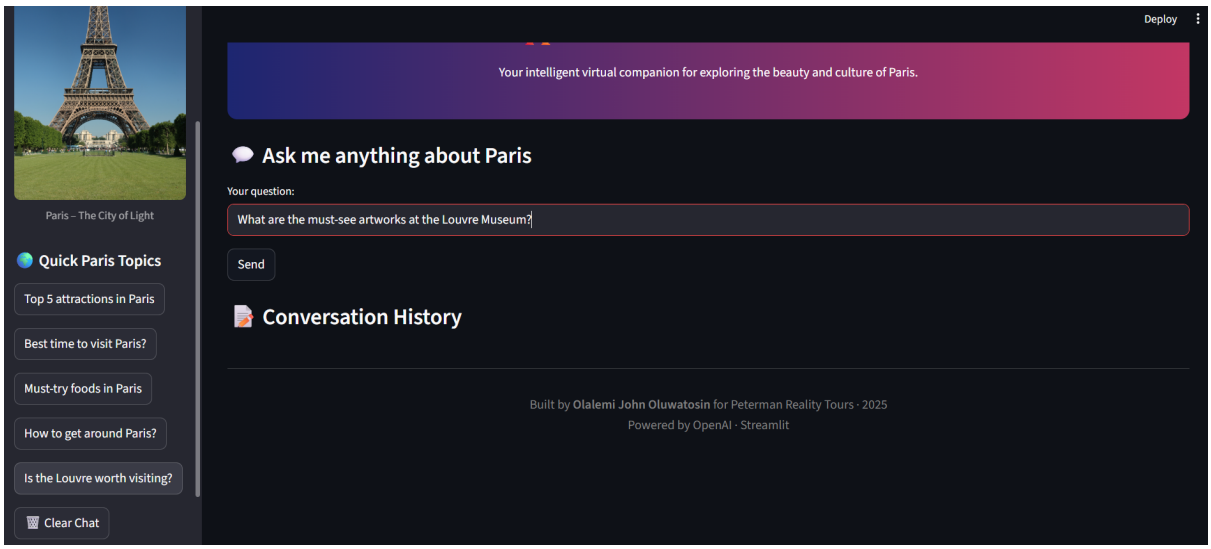

Figure 4.2: Streamlit App – Chat conversation interface

Chapter 5

How the Chatbot Works

The chatbot follows a structured pipeline:

1. **Initialization:** Loads environment variables and API client.
2. **Prompt Setup:** Defines system behavior for Paris tourism context.
3. **User Query:** Accepts input via CLI or Streamlit.
4. **Response Generation:** OpenAI API processes and returns responses.
5. **Logging:** All messages are saved in JSON format for analysis.

Chapter 6

Docker Deployment

Containerization simplifies running the chatbot locally or on servers.

- **Build image:** `docker compose build`
- **Run container:** `docker compose up`
- **Stop container:** `docker compose down`

Docker environment includes Streamlit, OpenAI, and dotenv configurations, ensuring reproducibility.

Chapter 7

Stakeholder Benefits

- **Tourists:** Real-time, friendly AI assistance for exploring Paris.
- **Executives:** Tangible example of AI applied to travel technology.
- **Developers:** Modular architecture for further innovation.
- **Investors:** Demonstrates potential scalability into multilingual or mobile versions.

Chapter 8

Conclusion

The AI Paris Travel Chatbot demonstrates the integration of advanced AI models with user-friendly interfaces. The migration from a CLI-based system to a full Streamlit web app marks a significant improvement in user experience, scalability, and visual engagement.

Future improvements include:

- Multilingual capabilities for diverse users.
- Integration with real-time weather and route APIs.
- Deployment on cloud services for global accessibility.

Appendix: Contact Information

- Developer: Olalemi John Oluwatosin
- Email: `johnnysnipes90@gmail.com`
- LinkedIn: `https://www.linkedin.com/in/john-olalemi`
- GitHub: `https://github.com/Johnnysnipes90`