Virtual Tour API: Ennejma Ezzahra Palace

Dhia Eddine Zouari
Senior BA-IT Student
Tunis Business School

January 20, 2025

Abstract

This document presents the development of a **Virtual Tour API** for **Ennejma Ezzahra Palace**, a historical site in Tunisia. The system is designed to allow users to explore a **360° virtual tour**, submit feedback, and authenticate securely using JWT-based authentication. The backend is implemented with **Django REST Framework**, integrated with **Google Maps API**, and deployed using **Docker**. This report thoroughly explains the architecture, database structure, security measures, API endpoints, and deployment strategy. Additionally, it discusses the testing approach using **Insomnia** and presents potential future enhancements.

Contents

1 Introduction

1.1 Context and Importance

With the rise of digital transformation, virtual tours have become an essential tool for museums, historical sites, and tourism agencies. Ennejma Ezzahra Palace, a historical landmark in Tunisia, provides an opportunity to showcase its beauty and historical significance through an interactive, digital experience.

1.2 Problem Statement

Many people are unable to visit historical sites due to geographic, financial, or accessibility constraints. A virtual tour provides a way for users worldwide to experience these locations remotely. However, a tour system must also allow user engagement, such as providing feedback and interacting with the content dynamically.

1.3 Proposed Solution

To address this, a web service has been developed that:

- Provides a **360° virtual experience** using Google Maps API.
- Implements **secure authentication** for access control.
- Allows users to **submit feedback** about their experience.
- Ensures a **scalable and maintainable architecture** using Docker.

2 System Architecture

2.1 Overview

The Virtual Tour API follows a **client-server architecture**, with:

- A **frontend** for users to interact with the virtual tour.
- A **backend API** for handling authentication, tour hotspots, and feedback.
- A **database** for storing user credentials, hotspots, and feedback data.

2.2 Technology Stack

The system is built using:

- **Django REST Framework**: Efficient API development.
- **Google Maps API**: Fetching and displaying 360° virtual tours.
- **JWT Authentication**: Secure user sessions.
- **Docker**: Containerized deployment for consistency.

3 UML Diagram for API Project

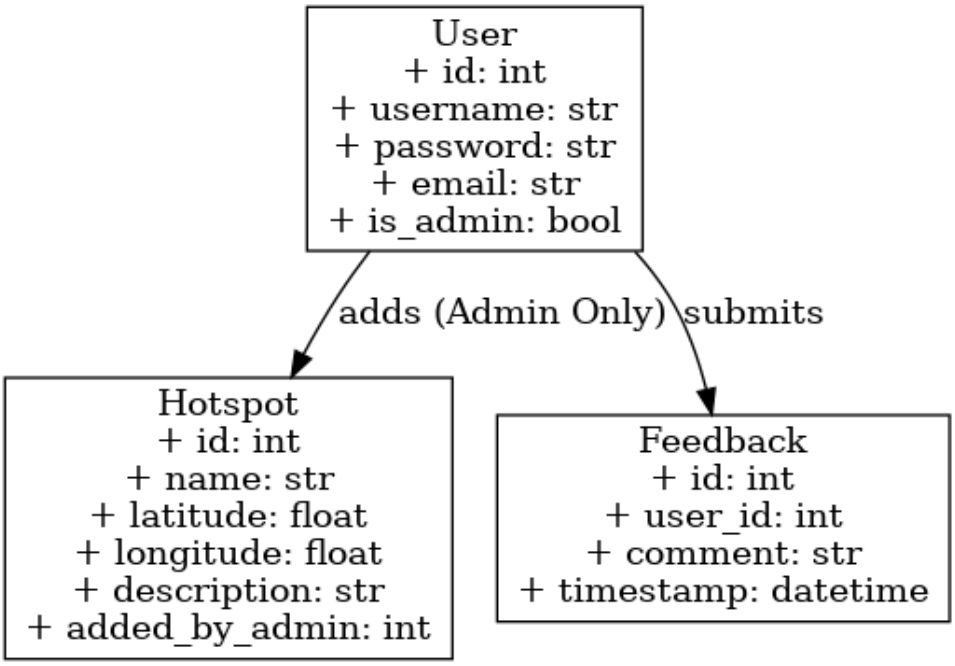

Figure 1: UML Diagram for API Design

3.1 Component Breakdown

- **User**: Represents authenticated users and superusers.
- **Hotspot**: Represents a virtual tour location added by an admin.
- **Feedback**: Represents user-submitted reviews.

4 API Design

4.1 Authentication API

4.1.1 User Registration

```
POST /auth/register/
Content-Type: application/json
{
    "username": "user1",
    "password": "securepassword",
    "email": "user1@example.com"
}
```

4.1.2 User Login

```
POST /auth/login/
Content-Type: application/json
{
    "username": "user1",
    "password": "securepassword"
}
```

Response

```
{
    "token": "your_jwt_token_here"
}
```

4.2 Tour API

4.2.1 Fetch Available Hotspots

```
GET /tour/hotspots/
Authorization: Bearer <JWT_TOKEN>
```

4.3 Feedback API

4.3.1 Submit Feedback

```
POST /tour/feedback/
Authorization: Bearer <JWT_TOKEN>
Content-Type: application/json
{
    "comment": "Amazing virtual tour experience!"
}
```

5 Security Implementation

5.1 JWT Authentication Flow

1. User submits credentials via login API.
2. System validates credentials and issues a JWT token.
3. User includes this token in subsequent requests.
4. Backend verifies token before granting access.

5.2 Best Security Practices

- **Password Hashing**: Storing hashed passwords to prevent leaks.
- **JWT Expiry and Refresh Tokens**: Enhancing session security.
- **CORS Configuration**: Restricting unauthorized external API calls.

6 Deployment Strategy

6.1 Docker-Based Deployment

Why Docker?

- Ensures consistent environment across development and production.
- Simplifies deployment and scaling.

Running the Project

```
docker-compose up --build
```

7 Testing

7.1 Using Insomnia

1. Register a new user.
2. Authenticate using login credentials.
3. Fetch available hotspots.
4. Submit user feedback.

8 Conclusion and Future Enhancements

8.1 Key Takeaways

- Provides a **secure 360° virtual tour**.
- Implements **JWT authentication**.
- Allows **user feedback collection**.

8.2 Future Plans

- **Multilingual Support**: Arabic and French versions.
- **Mobile App Integration**: Dedicated iOS and Android applications.
- **AI Chatbot**: Virtual tour guide assistant.