Trip Planner GPT Project Overview

1. Project Goal

The Trip Planner GPT project aims to create an intelligent travel planning assistant that combines the power of GPT models with real-world travel data. The system helps users plan their trips by:

- Finding suitable accommodations through Airbnb integration
- Providing real-time pricing and availability information
- Offering a clean, modern interface for travel planning
- Supporting natural language interactions for travel queries
- Maintaining deep links to booking platforms for seamless user experience

2. Architectural Overview

FastAPI Backend Server

- Core API server built with FastAPI for high performance and modern Python features
- RESTful endpoints for accommodation searches and travel planning
- Comprehensive error handling and logging
- CORS support for frontend integration
- Environment-based configuration management

RapidAPI Integration

- Integration with RapidAPI's Airbnb API for real accommodation data
- Fallback system to mock data when API is unavailable
- Standardized URL construction for deep linking to Airbnb listings
- Rate limiting and error handling for API calls

Data Models

- Pydantic models for request/response validation
- Structured data models for accommodations
- Type hints and validation throughout the codebase
- Consistent error response formats

Mock Data System

- Realistic mock data for development and testing
- Automatic fallback when external APIs are unavailable
- Consistent data structure matching real API responses
- Easy switching between mock and real data

Deployment Infrastructure

- Render.com for API hosting
- Environment variable management for sensitive data
- Production-ready configuration
- Static site hosting for documentation

Security Features

- API key management through environment variables
- CORS configuration for security
- Input validation and sanitization
- Secure handling of sensitive data

3. Implementation and Deployment Steps

Initial Setup

1. Created FastAPI application structure
2. Implemented basic endpoints and data models
3. Set up development environment with virtual env
4. Added dependency management with requirements.txt

API Integration

1. Integrated RapidAPI's Airbnb endpoint
2. Implemented error handling and logging
3. Created mock data system for development
4. Added URL construction for deep linking

Deployment Process

1. Set up Render.com account and configuration
2. Created render.yaml for deployment settings
3. Configured environment variables:
    - RapidAPI key
    - Other service credentials
4. Set up continuous deployment from GitHub

Testing and Validation

1. Implemented endpoint testing
2. Validated API responses
3. Tested error handling
4. Verified mock data system

Documentation

1. Created comprehensive README
2. Added API documentation
3. Included setup instructions
4. Documented environment variables

Security Implementation

1. Set up secure key management
2. Implemented input validation
3. Added error handling
4. Configured CORS policies

Current Status

The project is fully functional with:

- Working Airbnb integration
- Deployed API endpoint
- Comprehensive documentation
- Security measures in place
- Error handling and logging
- Mock data system for development

Next Steps

1. Add more travel-related integrations
2. Enhance the user interface
3. Implement additional trip planning features
4. Expand test coverage
5. Add more sophisticated GPT integrations