Lab1 AirBnB

Vidushi Verma, Shristi Kumar *(San Jose State University)*

1 Introduction

The Airbnb Prototype is a full-stack web application built using ReactJS for the frontend and Node.js (Express.js) for the backend, integrated with a Python FastAPI-based AI concierge agent. The platform replicates core Airbnb functionalities for two primary user roles:

- Traveler (can search, book, and manage trips)
- Owner/Host (can list properties and manage bookings)

The goal of this project is to develop a scalable, responsive, and intelligent booking platform that provides personalized recommendations using AI.

2 System Design

2.1 Architecture overview

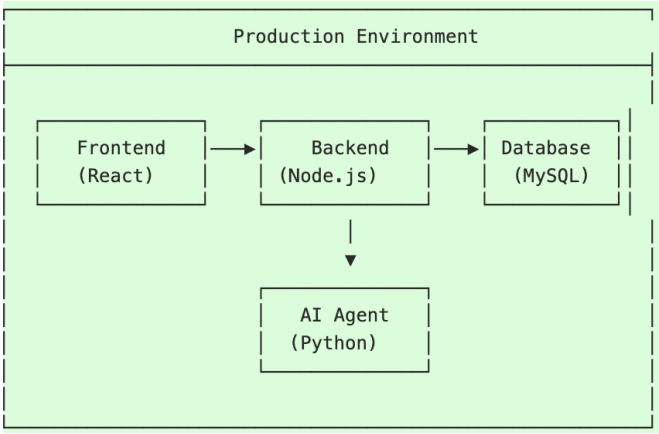

Figure 1: *AirBnB – Production Environment*

The Airbnb Prototype follows a modular, layered architecture to ensure scalability and clear separation of concerns. It consists of four main layers: **Frontend**, **Backend**, **AI Service**, and **Database**.

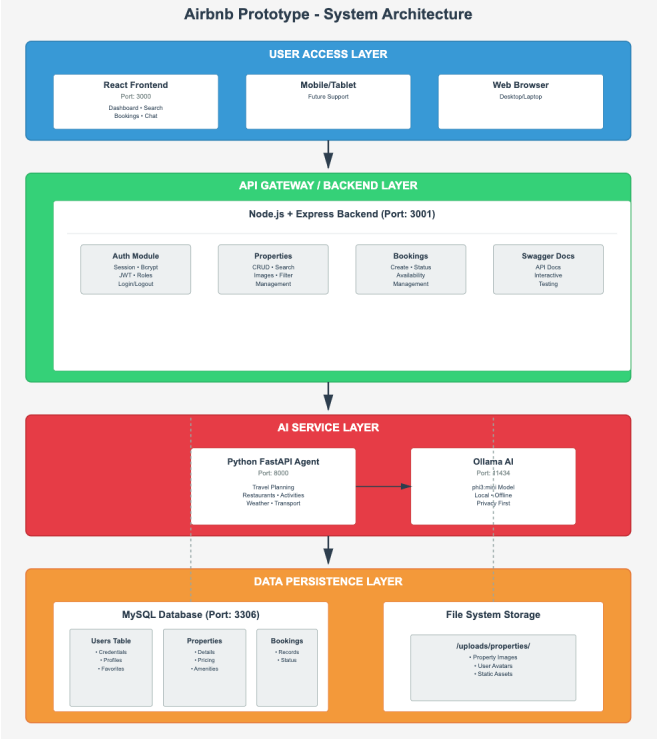

Figure 2: *AirBnB – System Architecture*

1. User Access Layer (Frontend) — Built using ReactJS, this layer provides a responsive interface for both Travelers and Owners. It handles authentication, property search, booking, and user dashboards. The frontend communicates with backend APIs via Axios and runs on port 3000.

2. API Gateway / Backend Layer — Implemented using Node.js with Express.js, this layer manages user authentication, property listings, and book-

ing operations. It exposes REST APIs consumed by both the frontend and AI agent, supports Swagger documentation, and operates on port 3001.

3. AI Service Layer — Powered by Python (FastAPI + LangChain), this component generates travel itineraries, activity suggestions, and restaurant recommendations based on user preferences and booking data. It integrates with Ollama AI (port 11434) for local reasoning and operates on port 8000.

4. Data Persistence Layer — Consists of a MySQL database (port 3306) and a file storage system. MySQL stores user, property, and booking information, while the file system manages uploaded assets such as property images and profile pictures.

Data Flow Summary:

- The Traveler initiates actions via the React frontend.
- The Node.js backend processes requests and interacts with the MySQL database.
- For personalized recommendations, backend data is passed to the FastAPI AI service.
- The AI agent returns structured JSON responses displayed on the frontend.

This architecture ensures scalability, modularity, and responsiveness while maintaining security through bcrypt-based authentication and session management.

2.2 Technology Stack

- **Frontend:**
  - React.js – Modern UI framework
  - Tailwind CSS – Utility-first CSS framework
  - Axios – HTTP client for API calls
  - React Router – Client-side routing
  - React Datepicker – Date selection component
- **Backend:**
  - Node.js – JavaScript runtime
  - Express.js – Web application framework
  - MySQL – Relational database
  - bcryptjs – Password hashing
  - express-session – Session management
  - multer – File upload handling
  - swagger-jsdoc – API documentation
- **AI Service:**
  - Python – Programming language
  - Fast API – Modern web framework
  - Ollama – Local AI model
  - MySql Connector – Database connectivity
- **Database layer:**
  - *Users Table*: id, name, email, passwordhash, role, phone, aboutme, city, country, languages, gender, favoritepropertyids, createdat
  - *Properties Table*: id, ownerid, propertyname, propertytype, description, location, city, country, pricepernight, bedrooms, bathrooms, maxguests, amenities, images, createdat
  - *Bookings Table*: id, propertyid, travelerid, startdate, enddate, numguests, totalprice, status, createdat

3 API documentation

3.1 Authentication

POST/api/auth/signup: User registration

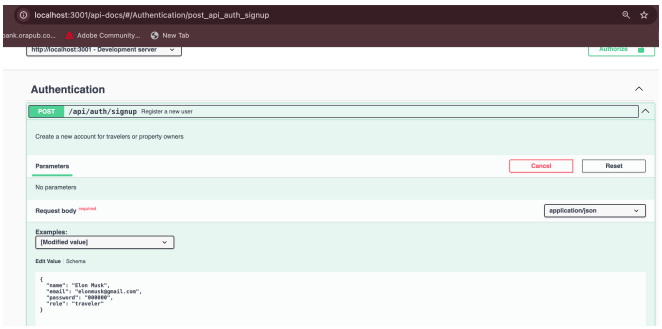

Figure 3: *Input Parameters*

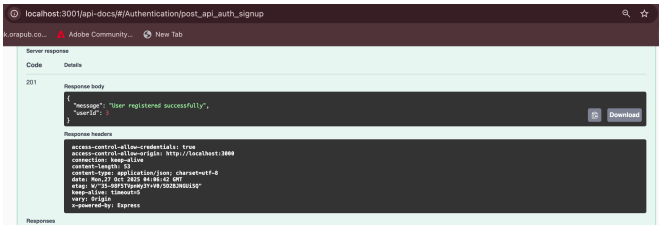

Figure 4: *Output Result*

POST /api/auth/login: User authentication

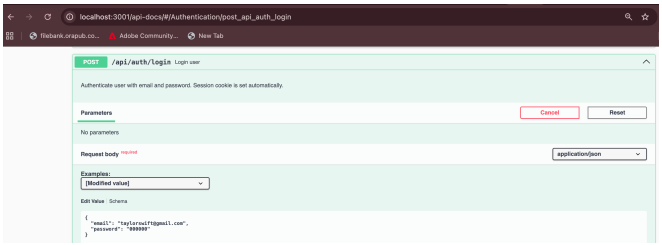

Figure 5: *Input Parameters*

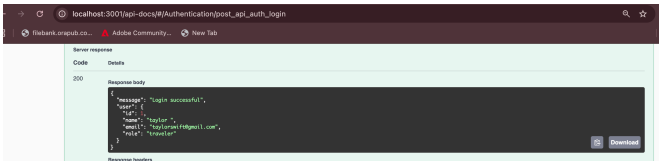

Figure 6: *Output Result*

3.2 Bookings

GET /api/bookings: Create new booking

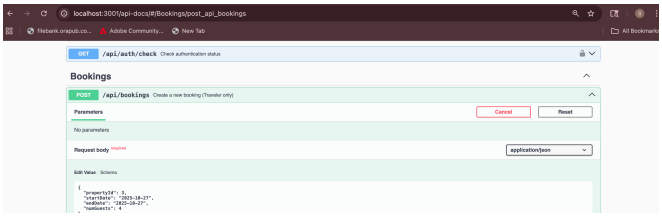

Figure 7: *Input Parameters*

Figure 8: *Output Result*

GET /api/bookings/traveller: Booking are retrieved travellers only

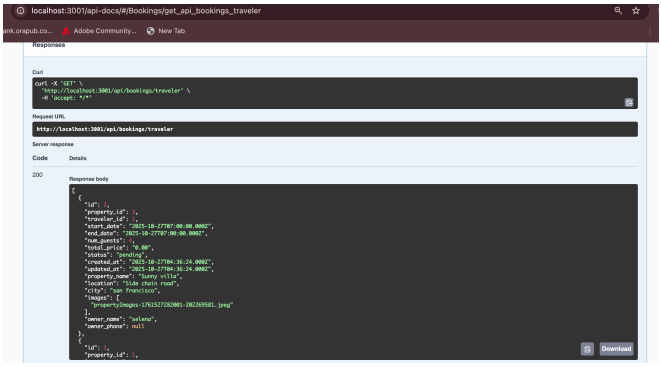

Figure 9: *Input/Output Parameters*

GET /api/bookings/owner: Bookings are retrieved (Owner only)

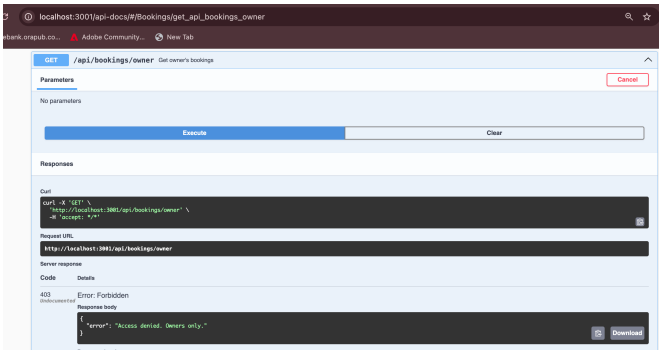

Figure 10: *Input/Output Parameters*

3.3 Properties

GET /api/properties/search‘: Search properties with filters

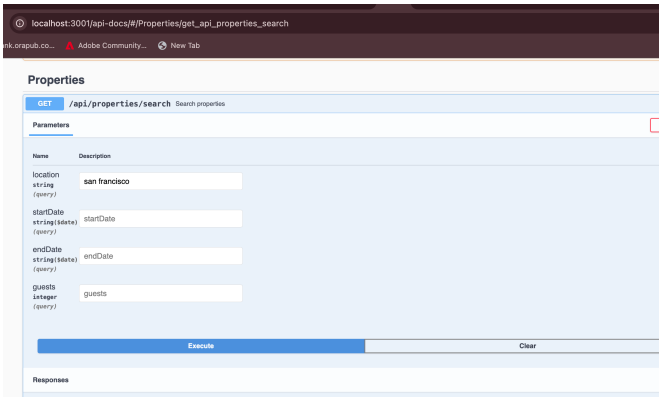

Figure 11: *Input Parameters*

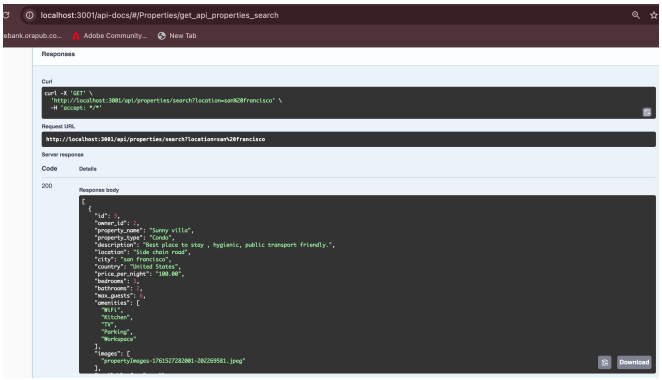

Figure 12: *Output Result*

GET /api/properties/:id‘: Get property details

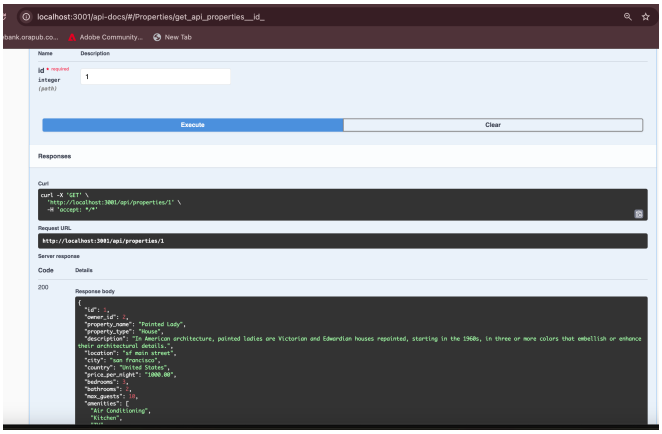

Figure 13: *Input Parameters*

PUT /api/properties/:id‘: Update property (Owner only)

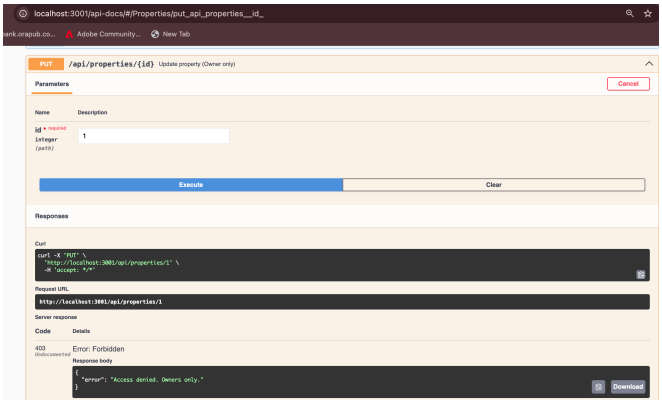

Figure 14: *Input Parameters*

3.4 Users

GET /api/users/me‘: Get current user profile

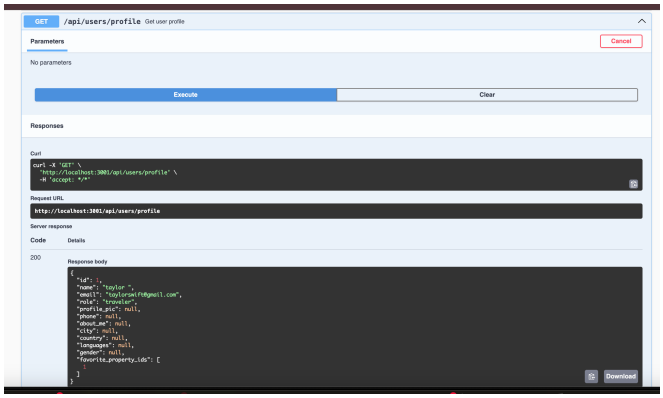

Figure 15: *Input/Output Parameters*

PUT /api/users/me‘: Update user profile

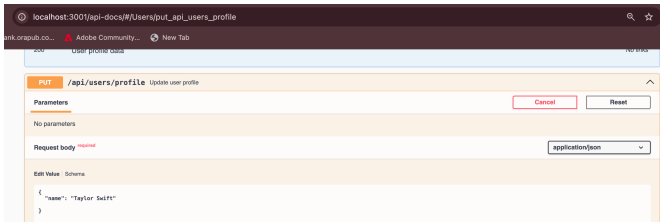

Figure 16: *Input Parameters*

Figure 17: *Output Result*

There are 20+ API points and the same were tested successfully via Swagger API documentation.

```
Airbnb API - Complete Collection
├── Authentication
│   ├── Sign Up
│   ├── Login
│   ├── Check Auth Status
│   └── Logout
├── Properties
│   ├── Search Properties
│   ├── Get Property by ID
│   ├── Upload Images
│   ├── Create Property
│   ├── Update Property
│   ├── Delete Property
│   └── Get Owner's Properties
├── Bookings
│   ├── Create Booking
│   ├── Get My Bookings
│   └── Update Booking Status
├── Users
│   ├── Get User Profile
│   ├── Update User Profile
│   └── Toggle Favorite
└── Health Check
```

Figure 18: *API Authentication Structure*

4 Non-Functional Requirements

4.1 Responsiveness

The application is fully responsive across desktop, tablet, and mobile devices. Tailwind CSS breakpoints and a flexible grid system ensure adaptive layouts for all screen sizes. Features like collapsible navigation, single-column mobile views, and touch-friendly controls maintain usability and consistency across platforms.

4.2 Accessibility

Accessibility is implemented using semantic HTML, ARIA roles, and descriptive alt text for images. The interface supports complete keyboard navigation, visible focus indicators, and skip links. Color contrast follows WCAG AA standards, ensuring clarity and inclusivity for all users.

4.3 Scalability

The system ensures scalability through optimized database queries, connection pooling, and pagination. APIs use asynchronous operations, caching, and minimal payloads for high performance. Modular code, lazy loading, and cloud-ready file management support future growth without performance degradation.

5 Airbnb App Screenshot

5.1 Traverler

Traveller Login Page

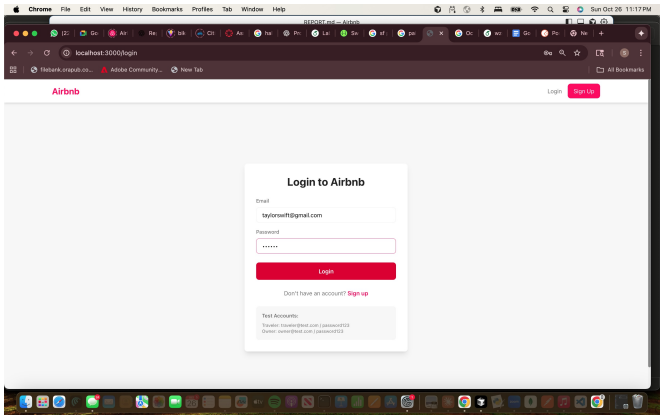

Figure 19: *Traveller:Taylor's Login*

Traveller Dashboard

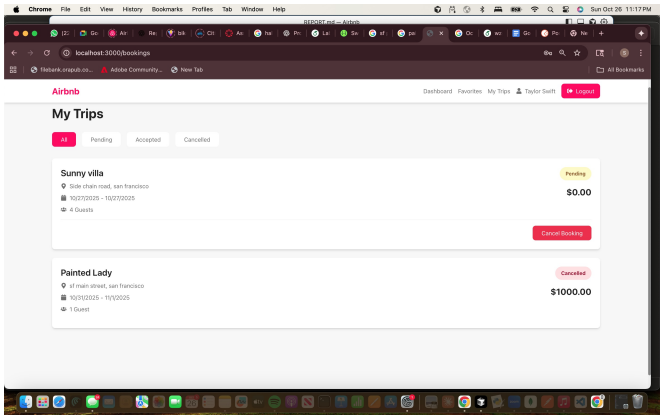

Figure 20: *Traveller:Taylor's Dashboard*

Traveller Favourites

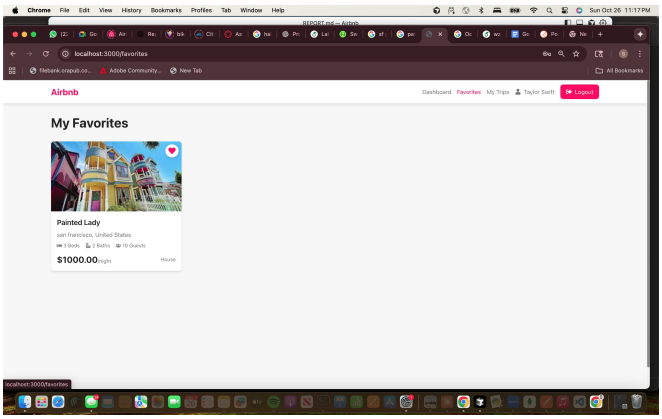

Figure 21: *Traveller:Taylor's Favourites*

Traveller Trips

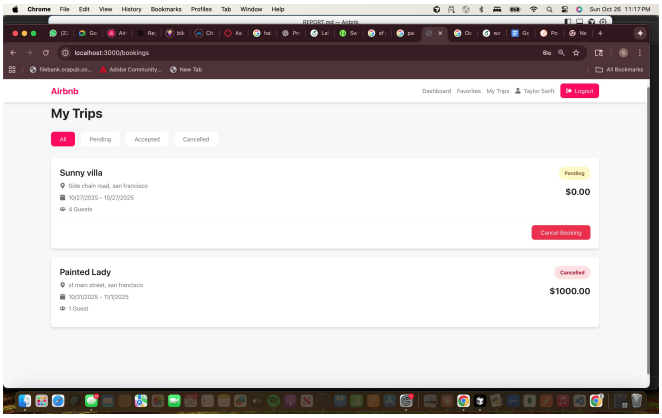

Figure 22: *Traveller:Taylor's Trips*

5.2 Owner

Owner's list of properties

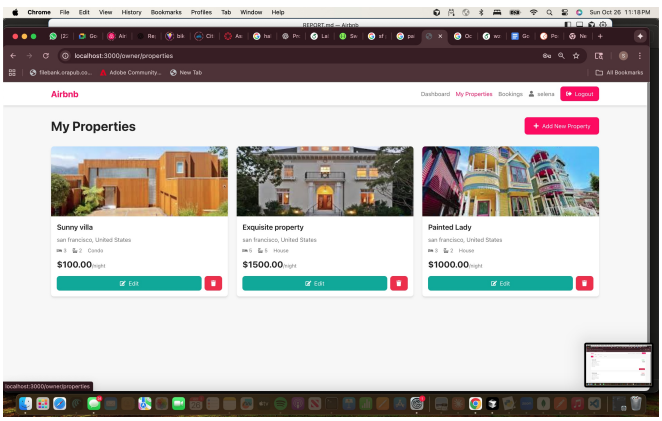

Figure 23: *Owner:Selena's properties page*

Owner's dashboard

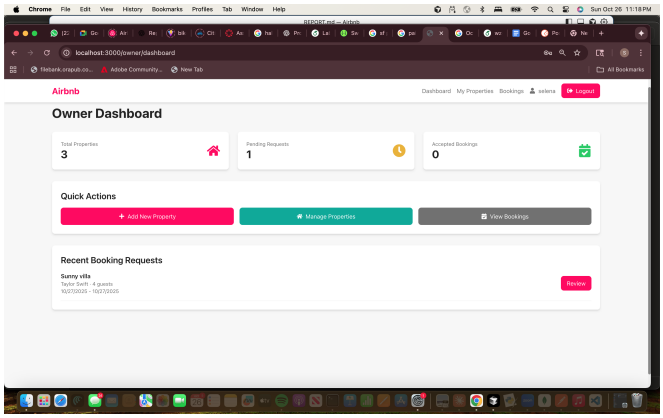

Figure 24: *Owner:Selena's dashboard*

Agent Chatbox

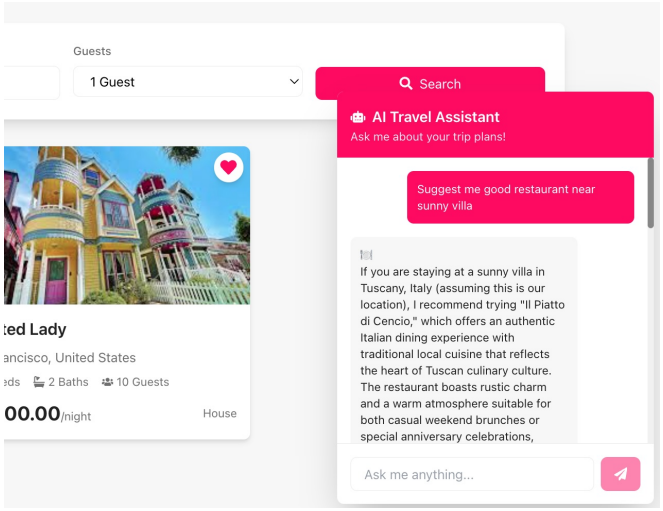

Figure 25: *AI Assistant*