Room Booking Assistant – Project Documentation

Overview

This project implements an AI-powered room booking assistant using:

- **FastAPI** for backend API
- **React (JSX)** for chat UI
- **LLM (e.g., DeepSeek)** for prompt-based intent and entity extraction
- **spaCy** and `dateutil` for entity extraction fallback
- **SQLite/MySQL** for storage (via SQLAlchemy)
- **Multi-turn session handling** using in-memory or token-based fallback
- **Feature Suggestions** when booking is not possible

System Architecture

```
User Input → Frontend Chat Interface (React)
                ↓
            FastAPI Backend
                ↓
            ┌──────────────┐
            │ LLM (DeepSeek) ──> Prompt & JSON Action
            └──────────────┘
                ↓
       Action Routing & Handling
         ↓                ↓
  Availability Logic   Entity Extraction (spaCy)
         ↓                ↓
     DB CRUD          Fallback + Prompt Tuning
                ↓
          Structured Response
```

Supported Actions

```
{
  "action": "check_availability" | "add_booking" | "cancel_booking" |
"alternatives",
  "parameters": {
    "room_name": "...",
    "date": "YYYY-MM-DD",
    "start_time": "HH:MM",
    "end_time": "HH:MM",
    "booking_id": "..."  // only for cancel_booking
  }
}
```

Prompt Example

System Prompt to LLM:

```
You are an intelligent assistant that helps manage room bookings.

From the following user request:
"I want to book LT1 on 2025-06-25 from 3PM to 4PM"

Extract the action and parameters in strict JSON format:
```

Expected Output:

```
{
  "action": "add_booking",
  "parameters": {
    "room_name": "LT1",
    "date": "2025-06-25",
    "start_time": "15:00",
    "end_time": "16:00"
  }
}
```

Fallback Handling

If the LLM returns incomplete data, we check for missing parameters and respond:

Backend Response:

```
{
  "status": "missing_parameters",
  "message": "Missing required fields: room_name, date, start_time, end_time",
  "next_step": "Please provide the missing information."
}
```

Entity Extraction Layer (Step 3)

- Uses `spaCy` + `dateutil.parser` to extract parameters from user message if LLM misses them.
- Example:

User: "Book on June 20th from 3 to 4PM"

Extracted Entities:

```
{
  "date": "2025-06-20",
  "start_time": "15:00",
  "end_time": "16:00"
}
```

💬 React ChatInterface Logic

```
const botMessage = response.data.message;
addMessage({ role: "assistant", text: botMessage });

if (response.data.next_step) {
  const botNextMessage = response.data.next_step;
  addMessage({ role: "assistant", text: botNextMessage });
}
```

🗓️ Example Conversations

➕ Add Booking (Complete Info)

> User: I want to book LT1 on 2025-06-25 from 2 to 3PM
> Bot: ☑ Booking confirmed for LT1 on 2025-06-25 from 14:00 to 15:00

➕ Add Booking (Missing Info)

> User: I want to book a room
> Bot: Missing required fields: room_name, date, start_time, end_time
> Bot: Please provide the missing information.

❌ Room Not Available

> User: Book LT1 on 2025-06-19 from 5PM to 6PM
> Bot: LT1 is NOT available at that time.
> Bot: Suggested slot: LT1 on 2025-06-19 from 6PM to 7PM

📦 Technologies Used

Layer	Tech Stack
Frontend	React, JSX, Zustand, MUI
Backend	FastAPI

Layer	Tech Stack
LLM Engine	DeepSeek / OpenAI
NLP Layer	spaCy, dateutil
DB	SQLite/MySQL
Entity Fallback	spaCy, regex, prompt tuning

🚧 Future Improvements

- [ ] Add session persistence
- [ ] Save user profile for better booking preferences
- [ ] Support calendar integration
- [ ] More advanced intent disambiguation

📂 How to Run

```
# Backend
cd backend
uvicorn main:app --reload

# Frontend
cd frontend
npm install
npm start
```