SE MENTOR MEETING ROOM BOOKING PORTAL

INDEX

Introduction

The **Meeting Room Booking Portal** developed for **SEMentor** is a focused, proof-of-concept implementation aimed at demonstrating how natural language interfaces can streamline internal scheduling and collaboration tasks. Built over the course of a rapid, three-day development cycle, this project showcases how an AI-assisted conversational interface can be effectively integrated into a traditional web-based infrastructure.

This portal enables employees to:

- Log in securely
- Book meeting rooms using natural language prompts
- Detect scheduling conflicts and purpose mismatches
- Send and respond to meeting invitations
- View and manage existing bookings through a simple dashboard

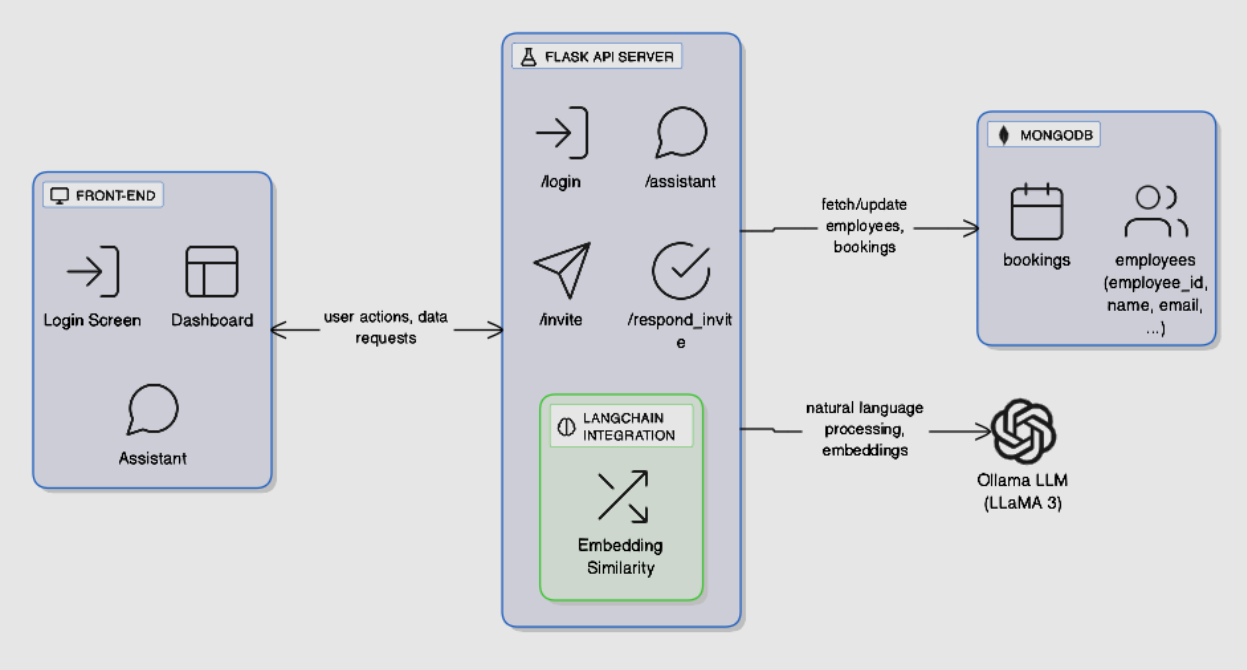
Rather than presenting a polished final product, the portal serves as a **functional prototype**—a practical foundation designed to validate integration workflows, database interactions, and the viability of LLM-driven intent recognition in enterprise use cases.

The focus of the build was to **test the end-to-end flow**: from user prompt interpretation by the language model (powered by LangChain and Ollama) to backend logic enforcement via Flask, and persistent storage through MongoDB. While the frontend was kept minimal, efforts were made to maintain consistency, responsiveness, and usability.

Problem Definition

In dynamic and fast-paced organizational environments, managing meeting room bookings efficiently remains a persistent challenge. Traditional systems often rely on rigid forms, multiple-step workflows, or fragmented interfaces that hinder quick scheduling and real-time coordination among employees. These limitations frequently result in overlapping bookings, underutilized spaces, and an overall lack of transparency in room availability. Additionally, most existing solutions require users to remember specific formats or interact with systems in a way that feels unintuitive and disconnected from natural workflow patterns. There is a noticeable absence of intelligent assistance that can understand user intent, verify details against existing data, and provide context-aware feedback. Employees should not have to manually check for availability or worry about exceeding room capacity constraints, nor should they need to understand the backend structure of the system. The need of the hour is a portal that simplifies this process where a user can express their intent in natural language, receive meaningful assistance, and complete room-related tasks swiftly and accurately. This project seeks to explore that very solution space through a functional, prototype-grade implementation of such a booking system, which integrates **natural language interaction, backend validation**, and minimalistic design into one coherent experience.

Architecture Diagram



Front End Specification

The front-end of the **SEMentor Meeting Room Booking Portal** serves as the primary interface through which employees interact with the system. Designed with simplicity and usability in mind, the portal is structured as a single-page web application, utilizing a combination of HTML, CSS, and vanilla JavaScript, with Bootstrap employed for basic responsiveness and aesthetic structure.

1. User Interface Pages

The portal contains three major interface views:

a) Login Page

- Purpose: Securely authenticates users based on employee ID and password.
- Features:
    - Form fields for employee ID and password
    - Client-side validation
    - LocalStorage integration for storing the logged-in user's employee ID
    - Error alerts for failed login attempts
- Upon successful login, users are redirected to the dashboard.

b) Dashboard Page

- Purpose: Display and manage the user's meeting bookings and invitations.
- Features:
    - Lists all current and upcoming bookings for the logged-in user
    - Dynamically fetches bookings from the backend using the employee ID
    - For each booking, displays:
        - Room name, Date, Time, Purpose, Attendees

        - Invitee list (with name and status)
        - “Invite” button to invite other employees using a dropdown
    - Under “Your Invites” section, the user sees:
        - Bookings they’ve been invited to
        - Who invited them
        - A status (sent, accepted, or rejected)
        - Action buttons to update their response

c) Assistant Chat Bubble

- Purpose: Provides an intelligent assistant to interpret natural language commands.
- Features:
    - Floating red circular chat icon marked with “SE”
    - Expands into a chat window when clicked
    - Users can issue free-form commands such as:
        - “Book Pinnacle for 2 people tomorrow at 2 PM”
        - “Cancel my booking on July 20”
        - “Show my bookings”
    - Chat window displays user queries and assistant responses in threaded format

2. Styling and Themes

- Theme Colors: Red primary palette (for brand alignment), white background, black and dark gray text
- Framework: Bootstrap 5
- Custom styling added for buttons, cards, and chat bubbles to match the SEMentor visual language
- All elements are responsive and render well on standard desktop browsers (Chrome, Firefox)

3. JavaScript Interactions

- Uses native fetch API for asynchronous communication with Flask backend
- Employs localStorage to persist user identity between page reloads
- Dynamically updates DOM based on API responses (bookings, invites, assistant outputs)
- Event listeners are used for buttons (e.g., login, invite, respond to invites)

Back End Specification

The backend of the **SEMentor Meeting Room Booking Portal** is a functional proof-of-concept powered by three core technologies: **Flask**, **MongoDB**, and **Ollama** (used in conjunction with **LangChain and LLaMA 3.2).** Together, these tools enable a lightweight, AI-driven system capable of understanding natural language commands and managing bookings dynamically.

1. Flask (Python Web Framework)

- **Role**: Acts as the primary backend server handling HTTP requests and routing.
- **Routing Structure**: RESTful routes like /assistant, /login, /invite, /my_invites provide isolated access to backend logic.
- **CORS Handling**: flask_cors.CORS is used to enable cross-origin requests between the frontend and backend.
- **API Endpoints**:
    - /login – Validates employee credentials
    - /assistant – Handles booking/view/cancellation based on natural language input
    - /invite, /respond_invite – Manages invitations
    - /my_invites, /employees – Returns data to populate frontend

The simplicity of Flask ensures that the backend logic remains clear, maintainable, and fast to prototype, which is ideal for early-stage or internal tooling.

---

2. MongoDB (NoSQL Document Store)

- **Role**: Provides flexible storage for both employee records and booking data.
- **Collections Used**:
    - employees: Static dataset containing user profiles, credentials, and access flags.
    - bookings: Dynamic dataset capturing room bookings, purpose embeddings, and invitations.


```
{
 "room": "Brainstorm Hub",
 "date": "2025-07-22",
 "time": "3:00 PM to 4:00 PM",
 "attendees": 4,
 "purpose": "strategy discussion",
 "booked_by": "EMP1001",
 "invites": [
  {
   "employee_id": "EMP1002",
   "status": "accepted"
  }
 ]
}
```

- **Querying**:
    - Time overlap logic uses Python's datetime for comparison
    - Invite lookups use $elemMatch on embedded arrays
    - Soft matching is used for purpose validation via embeddings

3. Ollama + LangChain + LLaMA 3.2

Ollama is used locally to serve LLMs like **LLaMA 3.2**, enabling high-quality offline natural language understanding.

Integration Highlights:

- **LangChain**: Provides a structured pipeline for prompt formatting, parsing, and output validation.
- **PydanticOutputParser**: Ensures that the language model output is converted into a predictable Python object.
- **Prompt Extracts the following:**

Extract:

- Room

- Attendees

- Date

- Time

- Purpose

- Employee ID if explicitly mentioned otherwise based on login

- Intent (book/view/cancel)

Workflow:

- Users send a free-form booking or inquiry prompt (e.g., *"Book Data Dome for 2 people on July 25th from 2:00 PM to 3:00 PM. My ID is EMP1002."*)
- The prompt is sent to the **LLaMA 3.2** model running locally via **Ollama**
- LangChain formats and parses the response into usable fields
- Flask evaluates the parsed output to perform the correct operation (booking, cancelling, viewing)
- MongoDB is queried or updated accordingly

User Story

User: Nitin Sai

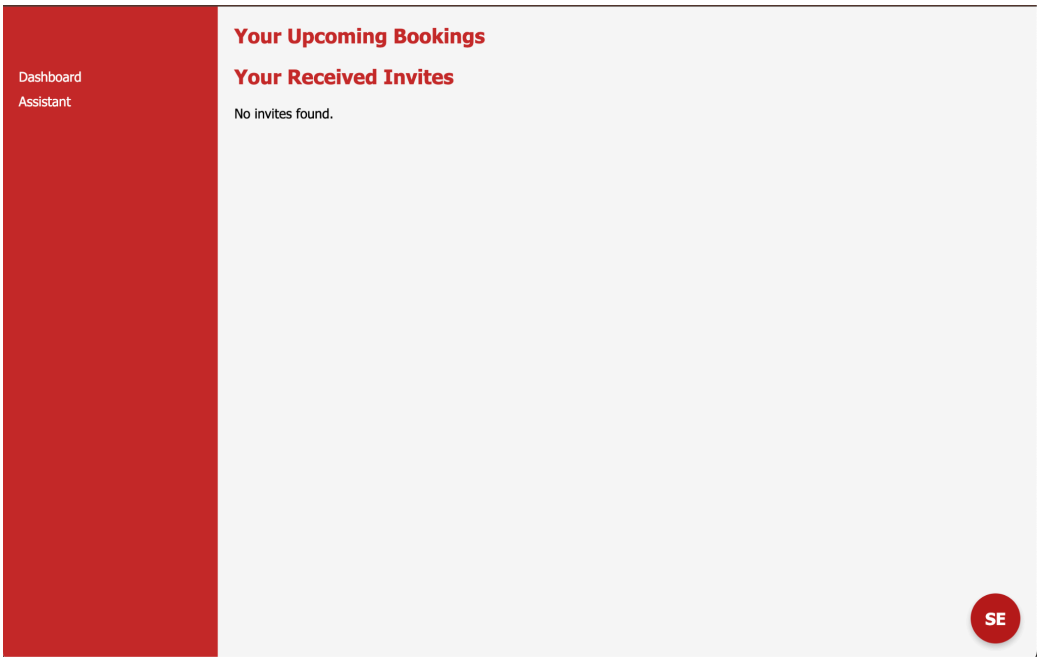
Role: Project Intern

Employee ID: EMP1001

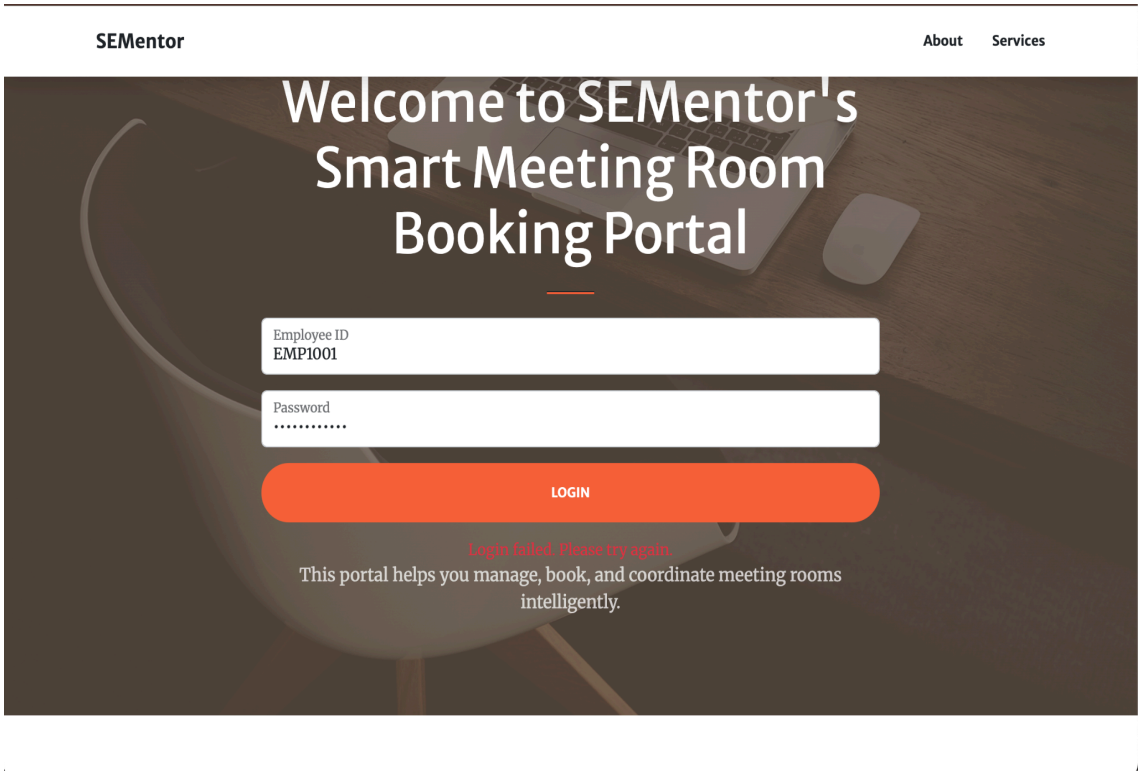
1. **Login to SEMentor Portal:**

   Nitin accesses the SEMentor platform through the login interface.
   He enters his **employee ID** and **password.**

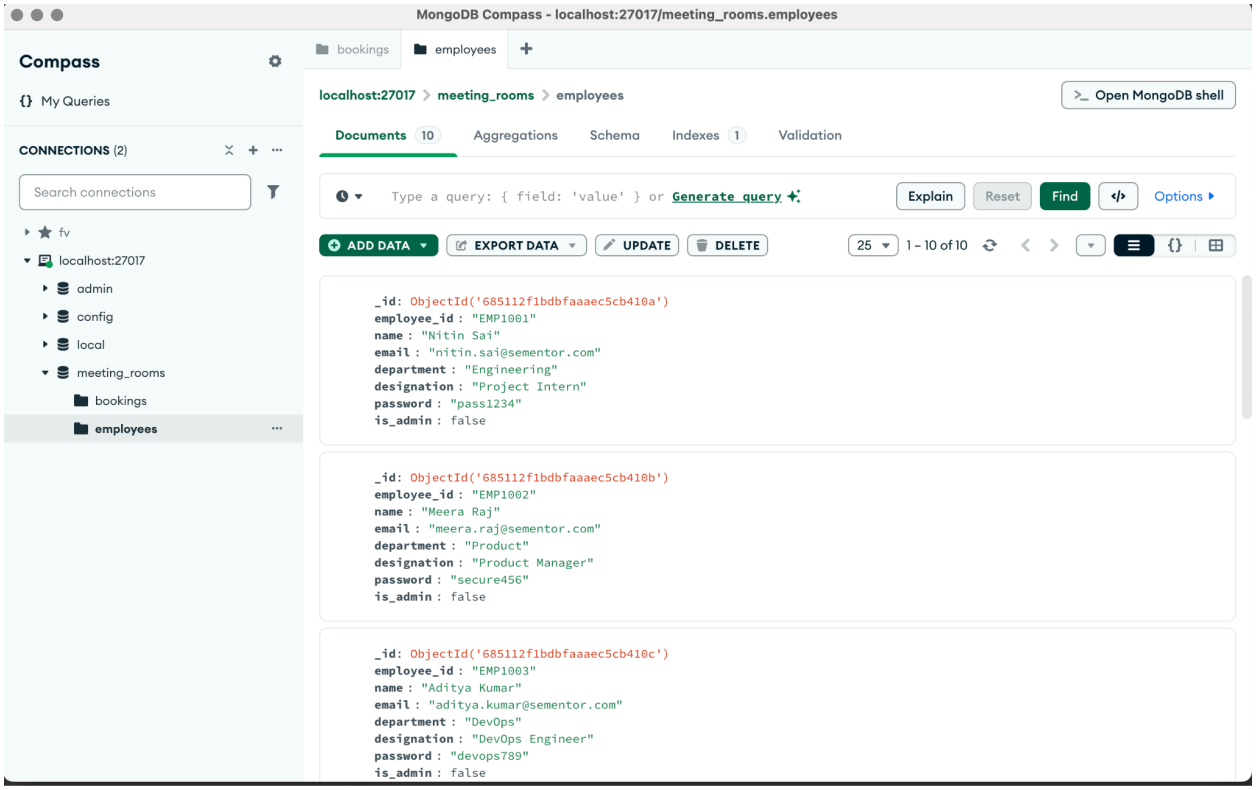
   - Successful login redirects to dashboard.
   - Incorrect password shows **“Incorrect password”** error.
   - Unknown **employee ID** is rejected with an error.



Correct Login



Incorrect Login



MongoDB Structure for Employee Details

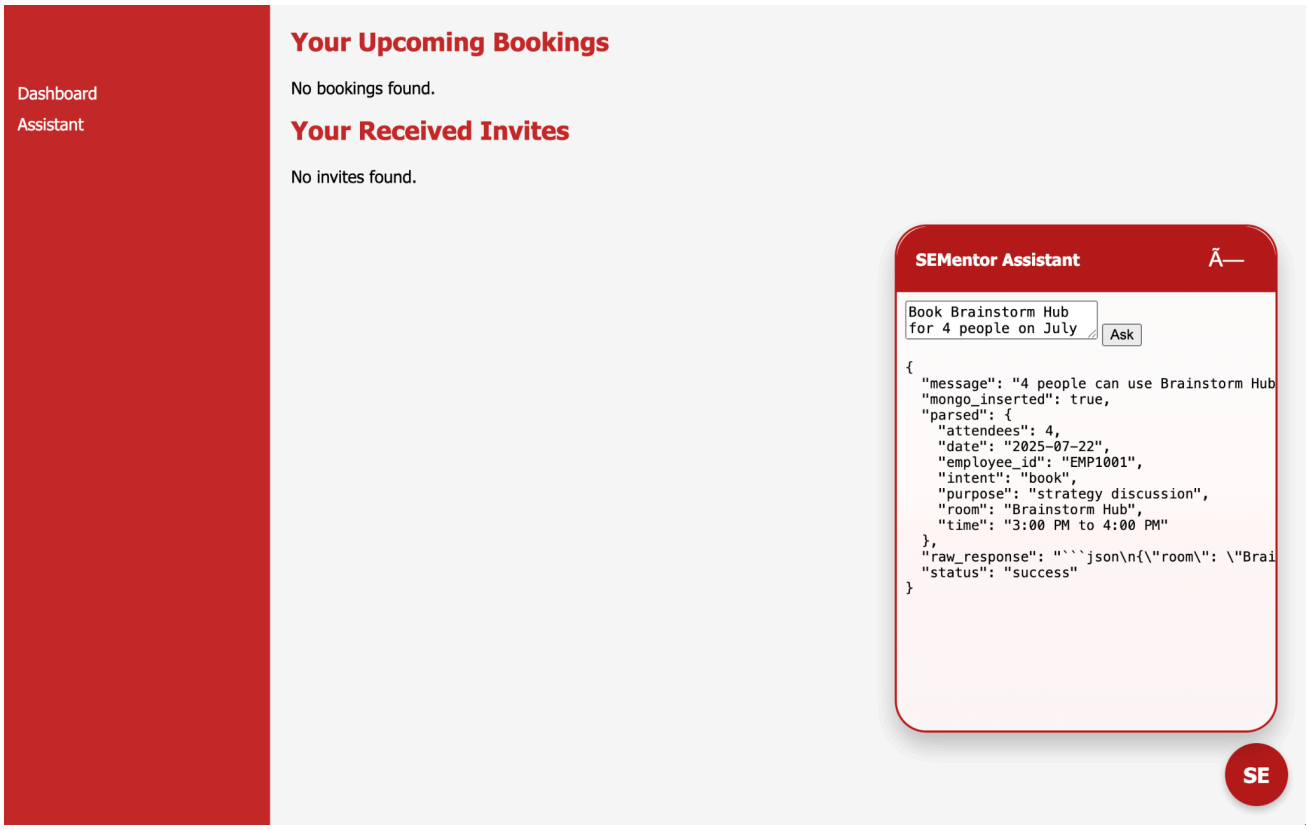
2. Making a Booking via the Assistant

Nitin clicks the floating assistant bubble and enters the command:

Prompt:

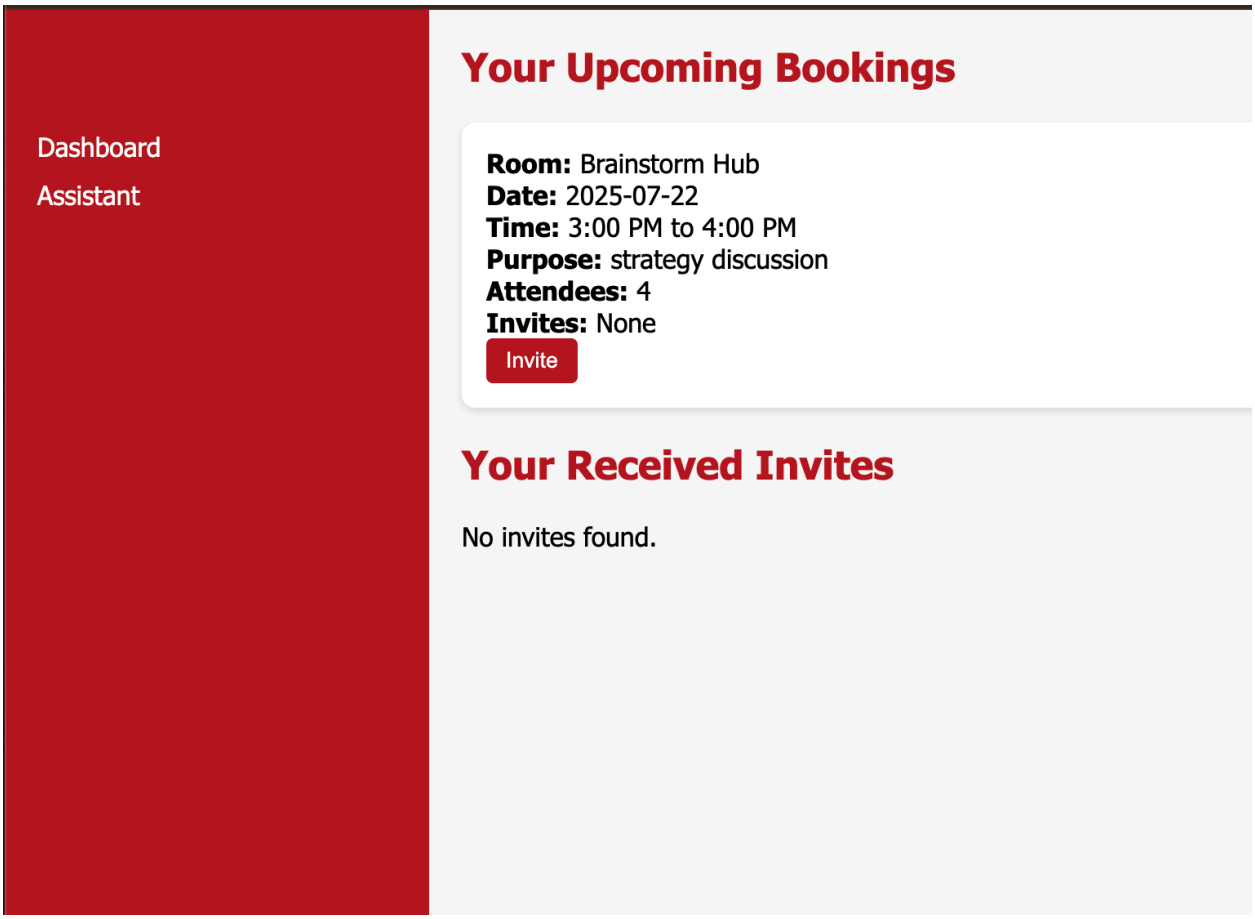
"Book Brainstorm Hub for 4 people on July 22 2025 from 3:00 PM to 4:00 PM for strategy discussion."

- The assistant processes the request.
- If successful and no overlapping bookings are found, the booking is confirmed and reflected on the dashboard.
- MongoDB bookings collection is updated with a new document.

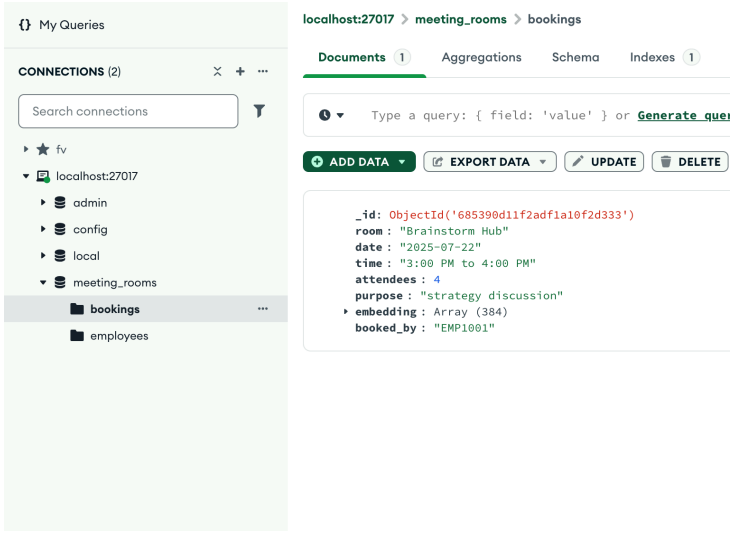



```
TERMINAL                                                    python3.12
 * Restarting with watchdog (fsevents)
 * Debugger is active!
 * Debugger PIN: 135-255-603
127.0.0.1 - - [19/Jun/2025 09:53:25] "OPTIONS /login HTTP/1.1" 200 -
127.0.0.1 - - [19/Jun/2025 09:53:25] "POST /login HTTP/1.1" 401 -
127.0.0.1 - - [19/Jun/2025 09:53:35] "OPTIONS /login HTTP/1.1" 200 -
127.0.0.1 - - [19/Jun/2025 09:53:35] "POST /login HTTP/1.1" 200 -
127.0.0.1 - - [19/Jun/2025 09:53:35] "OPTIONS /assistant HTTP/1.1" 200 -
127.0.0.1 - - [19/Jun/2025 09:53:35] "OPTIONS /my_invites HTTP/1.1" 200 -
Invites fetched for EMP1001: []
127.0.0.1 - - [19/Jun/2025 09:53:35] "POST /my_invites HTTP/1.1" 200 -
127.0.0.1 - - [19/Jun/2025 09:53:35] "GET /employees HTTP/1.1" 200 -
LLM Response:
 ```json
{
  "room": "Conference Room",
  "attendees": 5,
  "date": "2024-03-18",
  "time": "02:00 PM - 04:00 PM",
  "purpose": "Meeting for Team",
  "employee_id": "EMP1001",
  "intent": "view"
}
```
127.0.0.1 - - [19/Jun/2025 09:53:39] "POST /assistant HTTP/1.1" 200 -
127.0.0.1 - - [19/Jun/2025 09:53:39] "GET /employees HTTP/1.1" 200 -
127.0.0.1 - - [19/Jun/2025 09:53:41] "OPTIONS /assistant HTTP/1.1" 200 -
LLM Response:
 ```json
{"room": "Brainstorm Hub", "attendees": 4, "date": "2025-07-22", "time": "3:00 PM to 4:00 PM", "purpose": "s
trategy discussion", "employee_id": "EMP1001", "intent": "book"}
```
/Users/nitinsai/Documents/SEMentor/app2.py:336: PydanticDeprecatedSince20: The `dict` method is deprecated;
use `model_dump` instead. Deprecated in Pydantic V2.0 to be removed in V3.0. See Pydantic V2 Migration Guide
 at https://errors.pydantic.dev/2.11/migration/
  "parsed": parsed_output.dict(),
127.0.0.1 - - [19/Jun/2025 09:53:45] "POST /assistant HTTP/1.1" 200 -
```

Terminal logs showing parsed JSON and success



Nitin's booking's are reflected in the Dashboard



MongoDB Storing the booking with relevant details

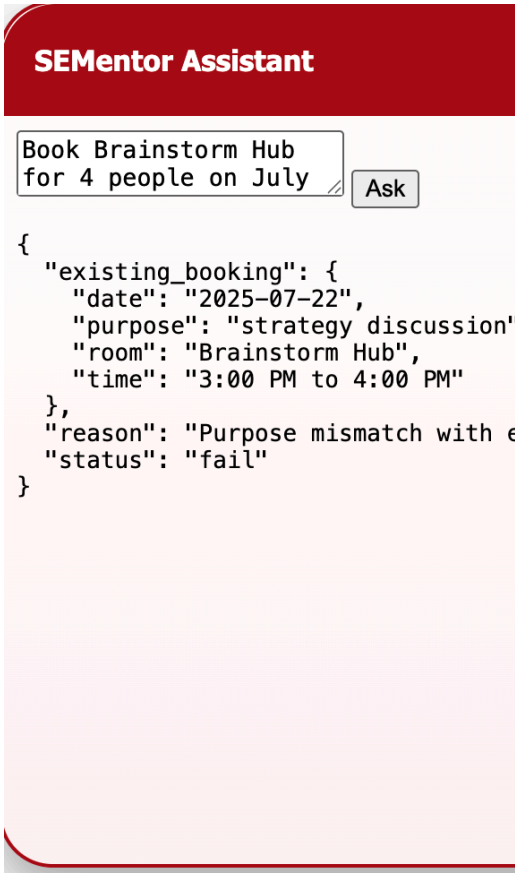
3. **Meeting Room Unavailable:**

   Prompt:

   "Book Brainstorm Hub for 4 people on July 22 2025 from 3:00 PM to 4:00 PM for client meetings."

   - Booking fails as the room is already booked.
   - If purpose mismatch, non-admin users receive purpose conflict error.

   This test case will be simulated by logging in as a different user:



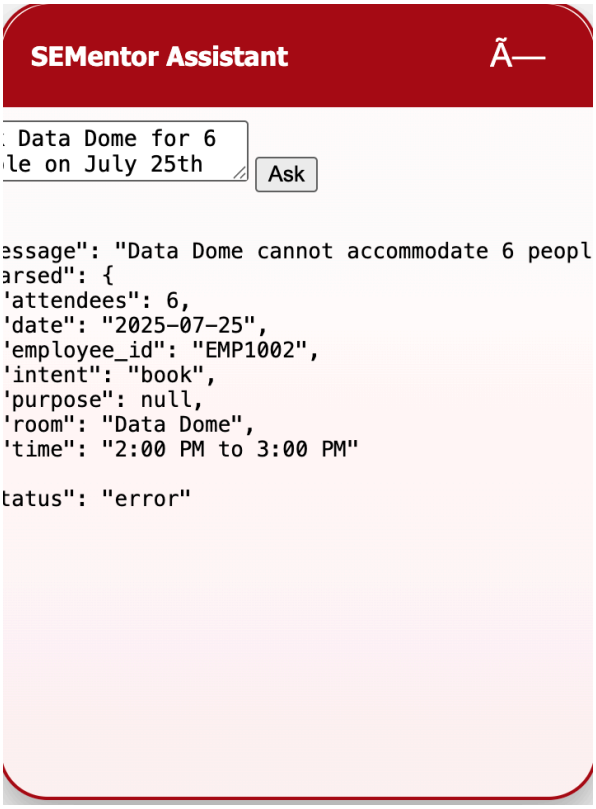
Existing Booking

4. **Edge Case – Capacity Violation**

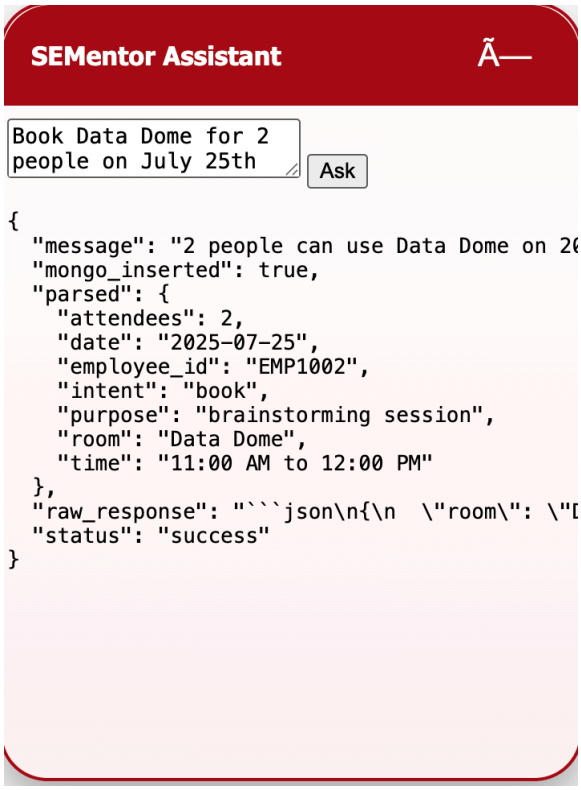
Prompt:

- Fails because Data Dome cannot accommodate more than 4 people.Book Data Dome for 6 people on July 25th 2025 from 2:00 PM to 3:00 PM.

- Checks handled by is_valid_room() function.



Meeting Room Capacity Violation

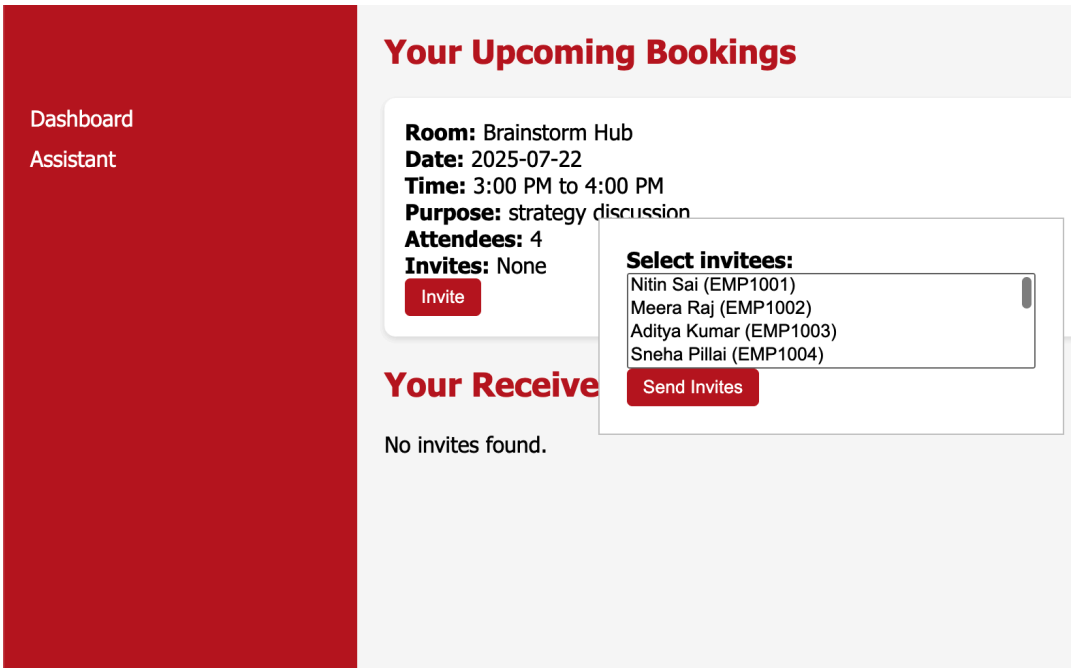


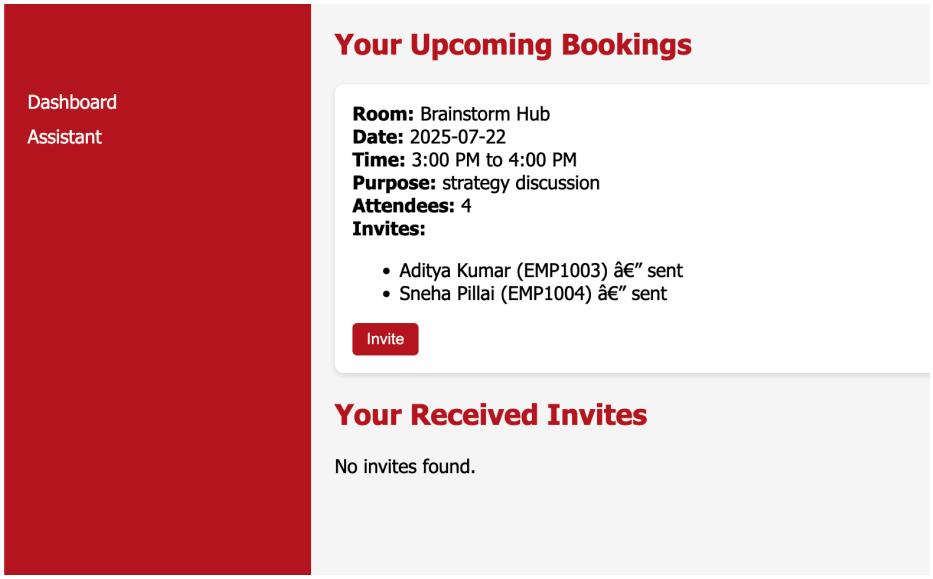
Valid Input

5. **Invitations**

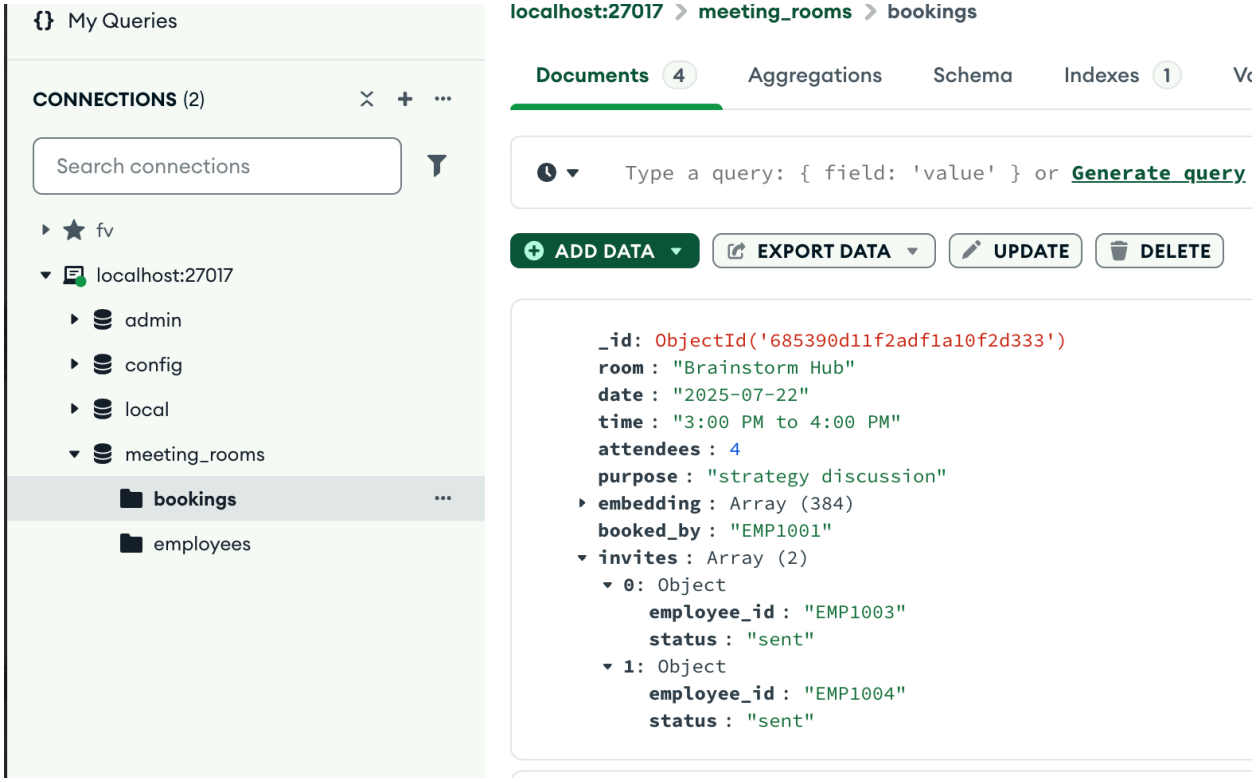
Nitin clicks “Invite” under one of his bookings.

- Dropdown of all employees is shown.
- Multiple invitees can be selected.
- Invitees are added to the booking under the invites field with status “sent”.



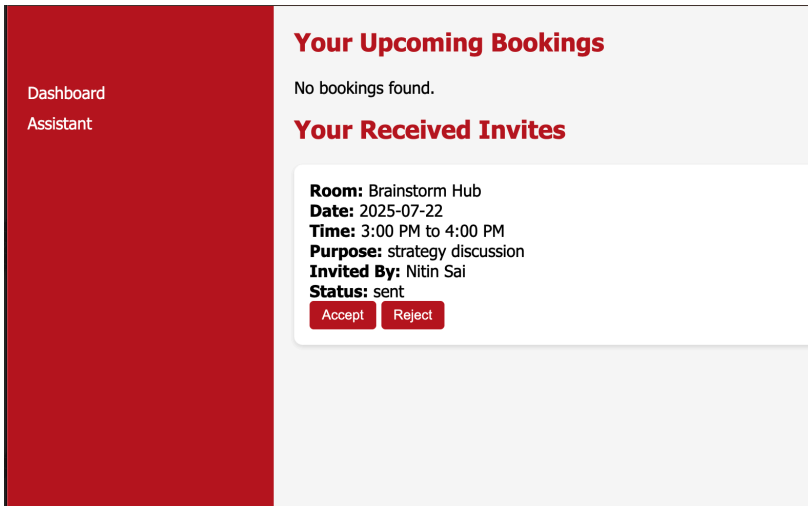


Invites Sent



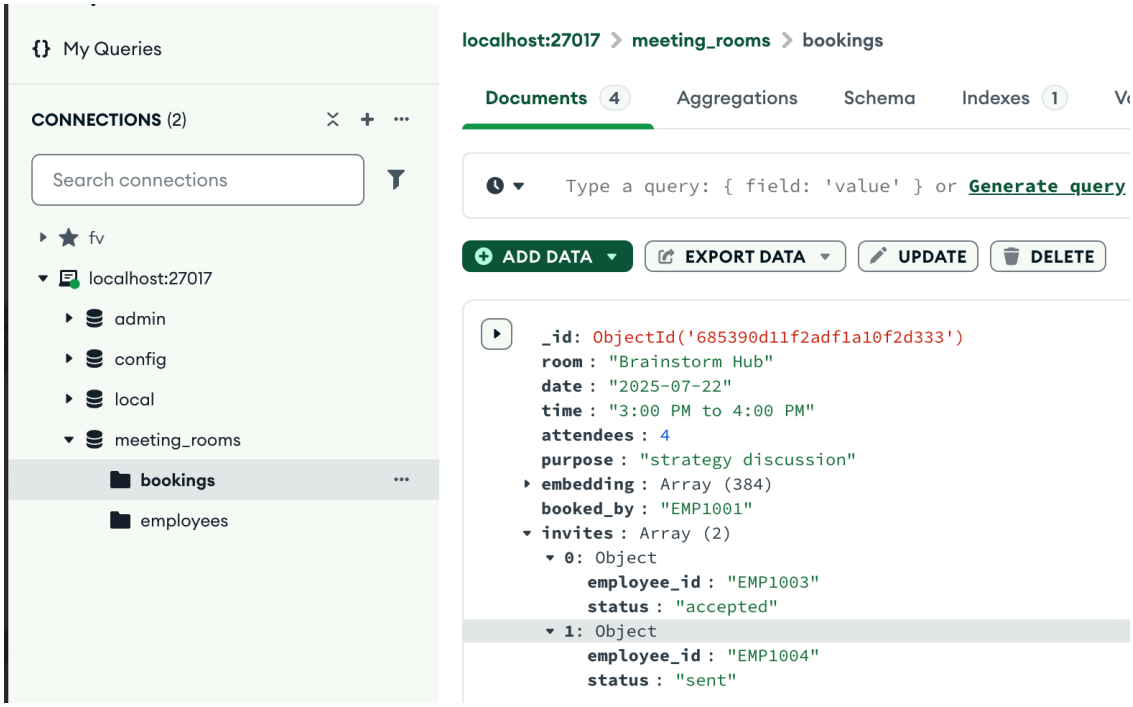
MongoDB Updated

- **Invitees View:**



EMP1003's View of the Dashboard

- Subsequently the invitee can accept or reject the invite and the MongoDB attribute is updated accordingly. This version does not delete the invite and maintains it in the history even if rejected.



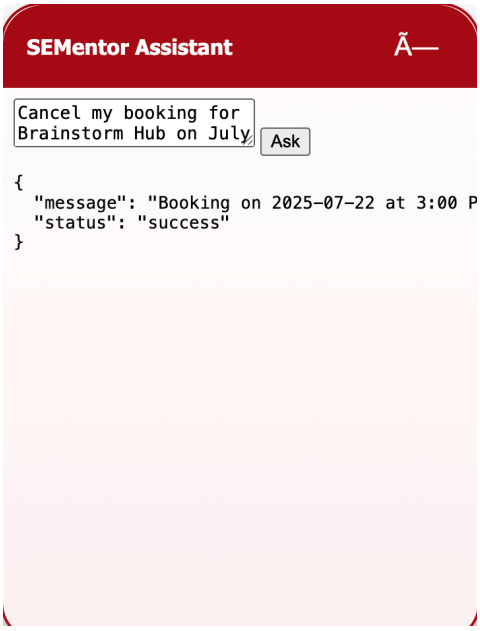
MongoDB Status attribute Updated for EMP1003

6. Cancelling a booking:

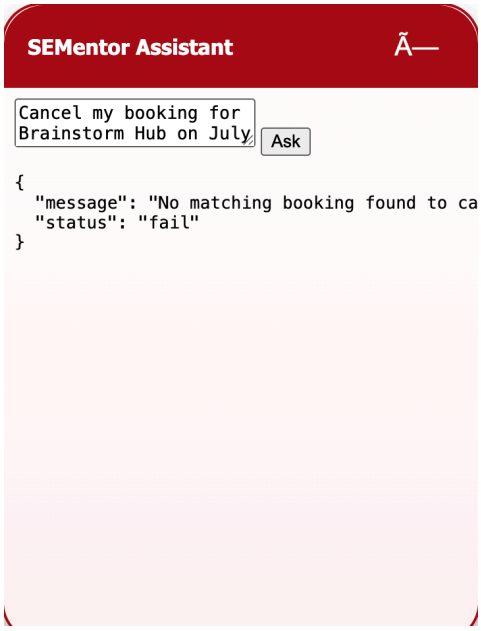
Prompt: (Logged in as EMP1001)

"Cancel my booking for Brainstorm Hub on July 22nd 2025 from 3:00 PM to 4:00 PM."

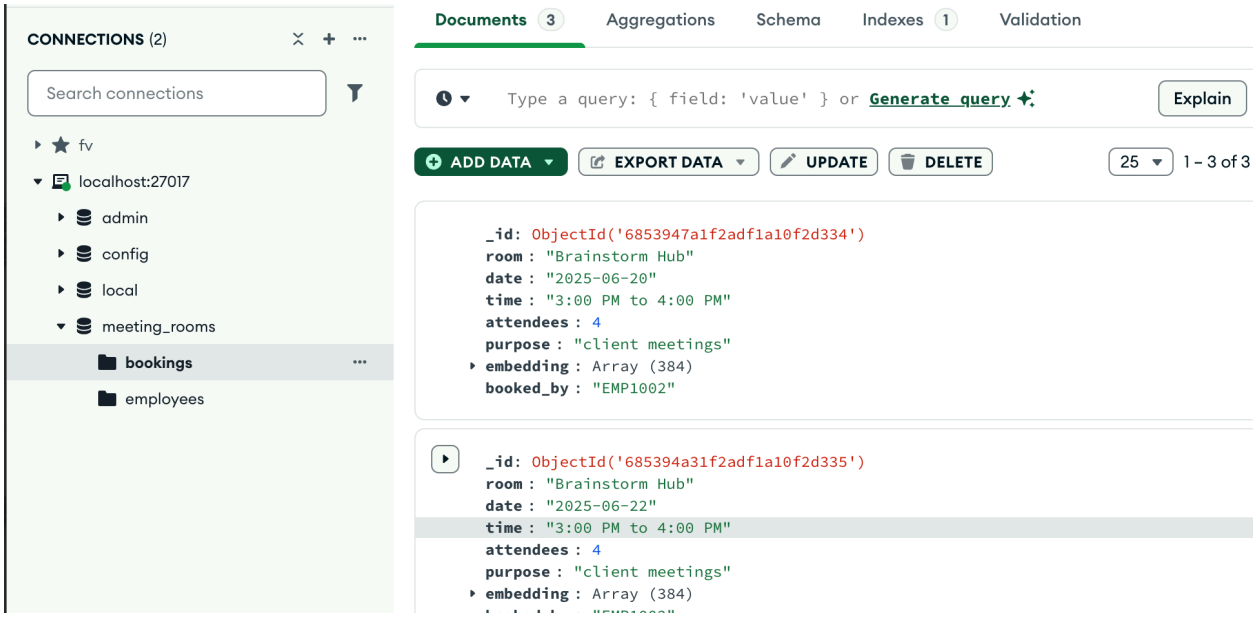
- If booking exists and matches, it is removed.
- If not found, the assistant responds with "No matching booking found to cancel."



Existing booking being cancelled



Booking no longer exists



Booking cancelled and reflected in MongoDB

Conclusion

The SEMentor Meeting Room Booking Portal serves as a focused proof-of-concept demonstrating the integration of AI-powered natural language interfaces with a functional room booking system. Built over the span of a few intensive development sessions, the portal combines a Flask-based backend, a MongoDB document store, and Ollama-powered local LLM inference (via LangChain) to deliver a lightweight, flexible solution for managing organizational meeting spaces.

Despite its deliberately minimal styling and basic structure, the portal successfully delivers key functionality:

- Natural language-driven room booking and cancellation,
- Conflict detection with purpose-based validation,
- Invite-based participation workflows,
- A simple frontend with role-based interaction pathways.

The design intentionally prioritizes clarity and adaptability over polish, making it easy to extend or rework for larger, production-grade systems in the future. While it lacks advanced security, authentication, or admin features, the project captures the spirit of rapid prototyping and offers a compelling example of how LLMs can drive meaningful backend operations without complex NLP pipelines.

Ultimately, the SEMentor portal stands as a successful exploration into AI-augmented scheduling, balancing automation with just enough control and structure to remain interpretable and extensible. The project's crude edges reflect its purpose — to validate architectural decisions, interface design, and model usage — rather than to impress through visuals or deployment maturity.

References

- **Flask Documentation –** https://flask.palletsprojects.com/
- **MongoDB Documentation –** https://www.mongodb.com/docs/manual/
- **LangChain Documentation –** https://python.langchain.com/
- **Ollama: Run open-source LLMs locally** – https://ollama.com