CSIS SmartAssist ChatBot

An Intelligent Information & Resource Management Assistant

Project Proposal — Hackenza 2026

1. Problem Statement

The CSIS department at BITS Pilani handles a high volume of recurring queries from students and faculty regarding academic policies, room and lab bookings, and departmental circulars. These requests are currently managed through fragmented channels like email threads, notice boards, and verbal communication which lead to delays and other inefficiencies.

There is no unified, intelligent interface that can: (a) answer document-grounded questions instantly, (b) facilitate lab and room bookings in real time, and (c) streamline administrative approval workflows. CSIS SmartAssist is designed to solve all three problems within a single, role-aware chatbot platform.

2. Proposed Solution

CSIS SmartAssist is a role-based intelligent chatbot portal tailored for the BITS Pilani CSIS department. It combines Retrieval-Augmented Generation (RAG) for document-grounded Q&A with an agentic booking workflow and a secure, email-driven approval system. The platform is accessible via web browser and authenticates users automatically based on their institutional Gmail accounts.

3. Core Features

3.1 Intelligent Information Retrieval (RAG)

- **Document-Grounded Q&A**
  - Answers queries about TA forms, reimbursement workflows, fee structures, and academic policies using only institution-approved documents (PDFs, Excel, Word files).
  - Responses are strictly grounded in uploaded documents, the bot will not hallucinate or speculate beyond its knowledge base.
- **Deep Citation Linking**
  - Every answer includes a direct reference to the source document and the specific page or section used, allowing users to verify information instantly.
- **Knowledge Gap Detection**
  - Tracks questions the system cannot answer and flags them as 'missing documentation' alerts for administrators, ensuring the knowledge base stays current.

3.2 Smart Resource Management (Agentic Booking)

- **Natural Language Booking**
  - Users can request rooms or labs via plain English chat (e.g., "Is Lab 101 free tomorrow at 2 PM?"). The bot interprets intent, checks availability, and confirms bookings — all without leaving the chat interface.
- **Real-Time Availability Check**
  - The bot interfaces with Google Calendar to verify slot availability instantly and surface up-to-date occupancy data.
- **Proactive Conflict Resolution**
  - If a requested slot is unavailable, the bot automatically surfaces the next available time slots or alternative rooms with similar capacity, ranked by proximity and availability.
- **Automated Booking Requests**
  - Once a user confirms a slot, the bot drafts and submits a formal booking request to the CSIS office on the user's behalf.
- **Waitlist Visibility**
  - Before confirming a booking, users can view a live waitlist showing all pending requests for the requested room, giving them context on demand and queue position.

3.3 Administrative & Approval Workflow

- **One-Click Email Approvals**
  - Administrators receive a notification email containing secure, unique approve/reject links. Decisions can be made directly from the email without logging into the portal.
- **Real-Time Status Notifications**
  - Once an administrator acts on a request, the applicant is automatically notified via email, including any reason-for-rejection remarks provided by the admin.
- **Role-Based Document Filtering**
  - Sensitive documents (such as faculty-only circulars or administrative memos) are accessible only to users with the appropriate role, enforced at the retrieval layer.

3.4 Role-Based User Authentication

Authentication is handled automatically via institutional Gmail. There are three user roles:

Role	Authentication Method	Key Permissions
Student	Any BITS Gmail not in faculty or admin list defaults to Student.	Information retrieval (RAG Q&A), room availability checks.
Faculty	Gmail matched against pre-loaded CSIS faculty list (publicly available on BITS website).	All student permissions + ability to request room bookings through the chatbot.
Admin	Gmail matched against a fixed, pre-loaded list of admin accounts.	All faculty permissions + approve/reject booking requests, manage knowledge base documents.

4. Technology Stack

To ensure a user-friendly and interactive interface, we'll be using Next.js on the front-end, which builds on top of React and simplifies complex UIs into reusable components. Next.js also gives us server-side rendering out of the box, which improves performance and makes the app more scalable .Tailwind CSS will handle styling, letting us move fast without getting bogged down writing custom CSS from scratch.

For authentication, we'll use NextAuth.js with Google OAuth. This means users simply sign in with their institutional Gmail account, and the portal automatically determines their role (student, faculty, or admin) by checking their email against pre-loaded lists. There's no separate login system to build or maintain, and it keeps things secure without extra overhead.

Since Next.js lets us write API routes alongside the frontend code, we don't need a separate backend service. All the server-side logic including handling booking requests, serving the chatbot, managing approval workflows will live in Next.js API routes running on Node.js. This keeps the entire project in one repo and makes deployment significantly simpler for a 24-hour timeline.

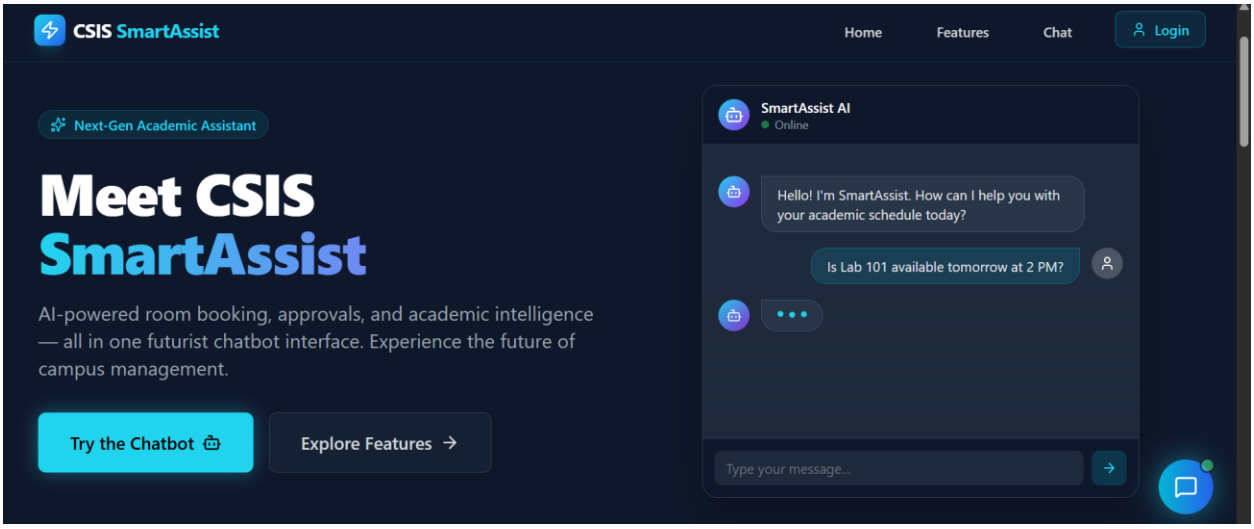
The intelligence of the chatbot is powered by OpenAI's GPT-4o-mini combined with LangChain.js to handle the Retrieval-Augmented Generation (RAG) pipeline. When a user asks a question, LangChain searches a vector database for the most relevant chunks from CSIS documents, and passes them to the model as context, ensuring answers are grounded in real institutional content rather than general knowledge. Document embeddings will be stored in Pinecone, a managed vector database with a free tier that requires zero infrastructure setup.

For document ingestion, LangChain's built-in loaders handle PDFs, Word documents, and Excel files, which covers the full range of formats that CSIS documents typically come in. Room and lab availability is managed through the Google Calendar API, which lets us check real-time slot availability and create booking events programmatically. Nodemailer over Gmail SMTP handles all outgoing emails .

The entire application will be deployed on Vercel, which has native support for Next.js and lets us go from code to live URL in minutes.

5. Interface Model

https://hackenza-2026.vercel.app/



6. Limitations & Scope

- The knowledge base for the hackathon submission will be seeded with a representative sample of CSIS documents; a production deployment would require a full ingestion pipeline for all departmental documents.
- Google Calendar integration will use a shared calendar managed by the CSIS admin account. Real-world deployment would require calendar access delegation per room.
- The faculty email list will be pre-loaded as a static JSON file for the hackathon; a production version would sync with the BITS LDAP/directory.
- One-click email approval links will use time-limited signed tokens (JWT) for security; the hackathon version will use 24-hour expiry tokens.