# **System Design and Architecture**

## **1. Introduction**

The proposed project, **Infomate**, is a web-based chatbot designed to provide centralized, interactive access to ICT Department information such as faculty details, labs, placements, achievements, and curriculum. The system is built with a **modular architecture** to ensure maintainability, scalability, and ease of updates. Its core objective is to address stakeholder needs—students, admissions, guests, and faculty—by automating repetitive queries and improving information accessibility.

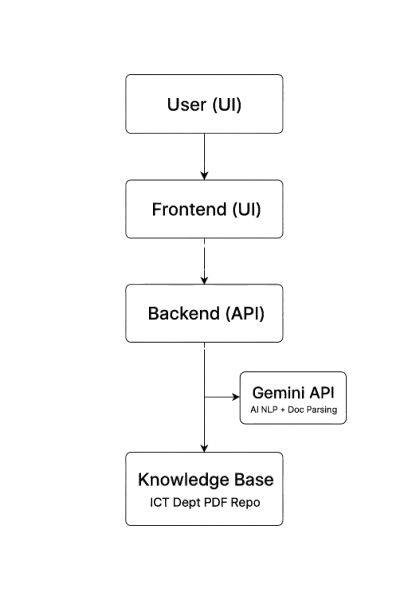
## **2. Modular Design**

The Infomate system is divided into **five major modules**:

1. **Frontend (User Interface)**
   * Built with **React.js**, offering a responsive and interactive interface.
   * Provides a chat-based UI where users can input queries and view responses.
   * Supports accessibility across devices (desktop, tablet, mobile).
2. **Backend (Application Server)**
   * Developed with **Node.js**.
   * Handles user requests, query routing, and communication with APIs.
   * Implements authentication for admin access (PDF updates).
3. **AI Processing Layer (Gemini API Integration)**
   * Responsible for understanding natural language queries.
   * Extracts relevant answers from the ICT Department’s static PDF knowledge base.
   * Ensures responses are accurate, context-aware, and ICT-specific.
4. **Knowledge Base (PDF Repository)**
   * Stores departmental information in structured PDF documents
   * Includes faculty details, labs, achievements, curriculum, and placements.
   * Updating the chatbot requires only replacing the PDF file—making maintenance simple.
5. **Database (Optional Future Expansion)**
   * Initially, no database is required since the chatbot relies on PDF parsing.
   * Future scalability can introduce a **MongoDB** database to store FAQs, logs, and analytics.

### **System Architecture Diagram**

Here’s a high-level architecture



**Justification for Modularity:**

* **Maintainability:** Each module (UI, backend, AI, knowledge base) is independent, allowing updates without disrupting the entire system.
* To address the problem, the following solutions are proposed:
* **Solution 1: Infomate – Web-Based ICT Chatbot (Proposed Project)**
* A **React (frontend)** and **Node.js (backend)** chatbot integrated with **Gemini API** for document parsing.
* Responds to natural language queries by extracting information directly from the ICT Department PDF (curriculum, faculties, placements, labs, achievements).
* Example Queries:
* “Who are the IoT faculty members?”
* “What subjects are in Semester V?”
* “What are the latest placement packages in ICT?”
* **Impact:** Improves accessibility, saves staff time, and enhances student/visitor experience.

## **3. Technology Stack**

| Layer/Module | Technology Choice | Justification |
| --- | --- | --- |
| **Frontend** | React.js | Modern, component-based UI framework; supports responsive design; widely adopted in ICT projects. |
| **Backend** | Node.js | Non-blocking I/O enables handling multiple requests efficiently; lightweight and scalable. |
| **AI Processing** | Gemini API (LLM-based) | Provides robust NLP and document parsing; ensures chatbot understands queries and retrieves answers from PDF. |
| **Knowledge Base** | PDF Repository (static) | Easy maintenance—faculty only need to update one document. |
| **Deployment** | Vercel (frontend), Render (backend) | Cloud hosting ensures scalability, high uptime, and global accessibility. |
| **Future Database** | MongoDB (optional) | Flexible schema for logging queries, analytics, and FAQs if scalability demands it. |

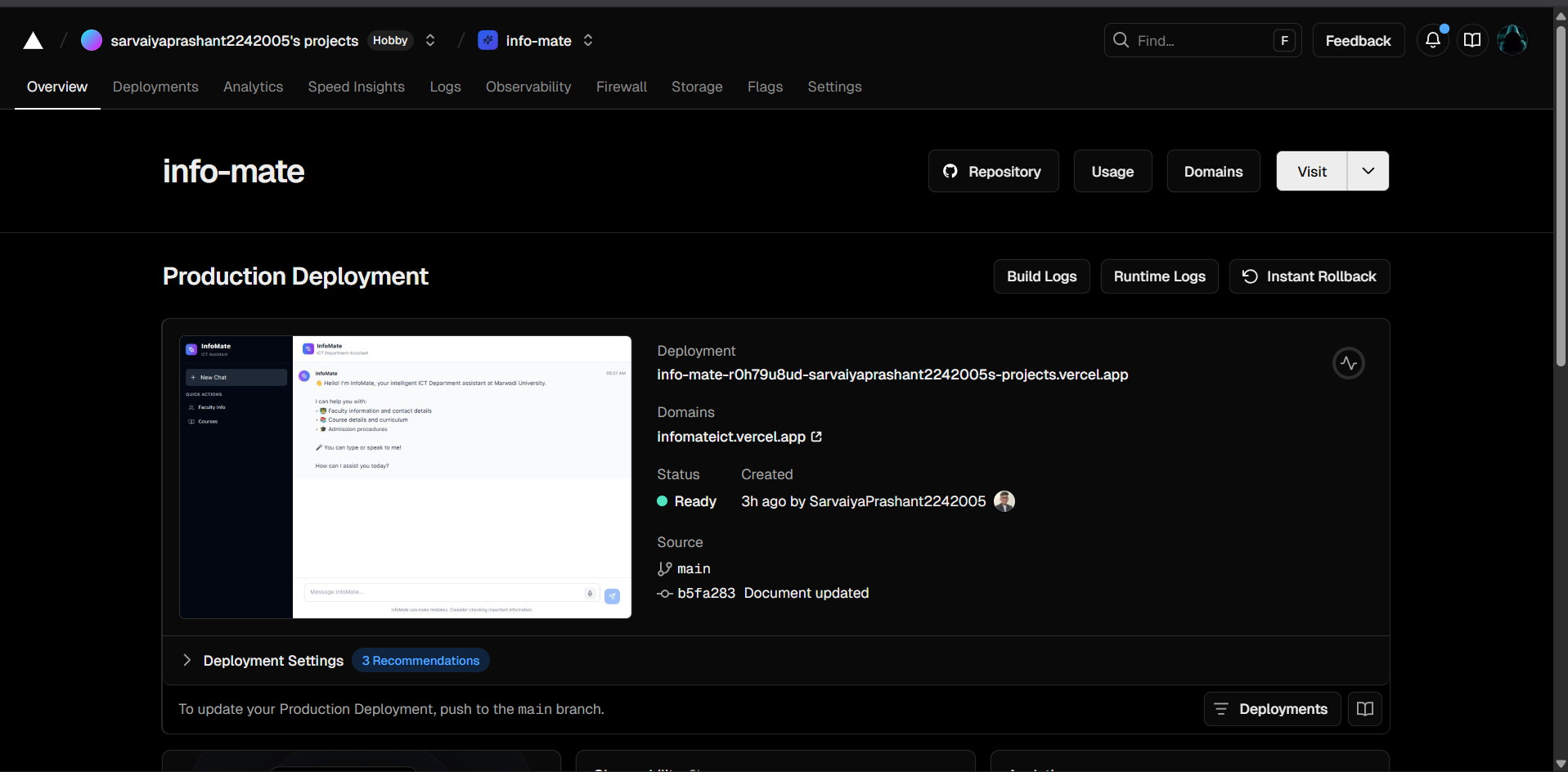
**Credible References:**

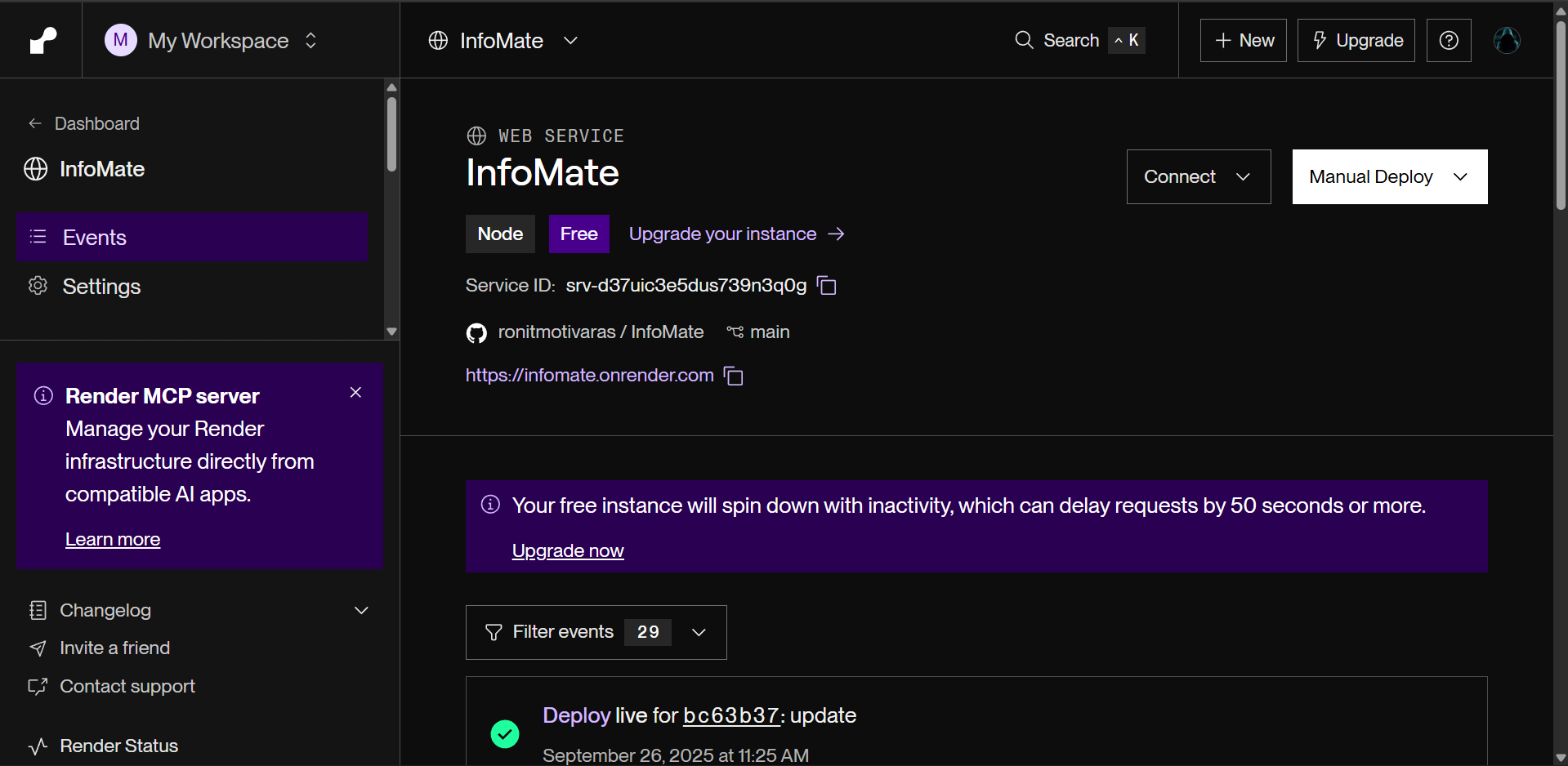
* Node.js is benchmarked for **high scalability in real-time applications** (IEEE Spectrum, 2022).
* React.js remains one of the **most widely used frontend frameworks** in education and enterprise systems (Stack Overflow Developer Survey, 2023).
* AI-driven chatbots are projected to handle **70% of student service queries** by 2026 (Gartner, 2022).

## **4. Scalability Plan**

To ensure Infomate can handle **growing load and users**, scalability strategies are:

1. **Frontend Scalability**
   * Deploy on **Vercel**.



1. **Backend Scalability**
   * Host Node.js backend on **Render.**
2. **AI Processing Scalability**
   * Gemini API usage scaled via request quotas; can integrate **rate-limiting and caching** of frequent queries.
   * Future plan: fine-tune a lightweight local LLM for cost optimization.
3. **Knowledge Base Scalability**
   * Currently static PDF upload; future plan includes **database integration (MongoDB)** for structured queries.
   * Sharding/partitioning possible if dataset grows beyond PDF (e.g., all departments).
4. **Performance Bottlenecks & Solutions**
   * **Database latency** → Add caching layer (Redis).
   * **Network latency** → Deploy servers close to users using **CDNs and edge computing**.
   * **AI response time** → Preprocess and index PDFs for faster retrieval.
5. **Cost & Reliability Considerations**
   * Free/low-cost tiers (Vercel, Render, Gemini API) used initially for affordability.
   * Can migrate to **AWS or GCP** if higher reliability and performance are required.

## **5. Conclusion**

The **Infomate chatbot system design** is modular, scalable, and well-aligned with ICT engineering practices. Its architecture ensures:

* Maintainable modules (UI, backend, AI, knowledge base).
* A robust technology stack (React, Node.js, Gemini API, cloud hosting).
* A future-ready scalability plan that supports horizontal scaling, caching, and multi-department expansion.

This architecture not only meets current stakeholder needs but also provides a **flexible foundation** for expanding Infomate into a **college-wide intelligent assistant**, thereby contributing to the ICT domain’s vision of **AI-powered digital transformation in education**.