A Project Report On

**InfoMate : An AI powered Information Assistant for ICT Department**

**Submitted by:**

*Prashant Sarvaiya– 92200133001*

*Ronit Motivaras – 92420133001*

**Guided by**: Prof. Chandrasinh Parmar

A Project Report Submitted to

Marwadi University in Partial Fulfillment of the Requirements for the B. Tech in Information Communication Technology



**MARWADI UNIVERSITY**

Rajkot- Morbi Road, At & Po. Gauridad, Rajkot-

360003, Gujarat, India.

Contents

[A Project Report On i](#_Toc209860375)

[**InfoMate : An AI powered Information Assistant for ICT Department** i](#_Toc209860376)

[**1. Technical Report** 1](#_Toc209860377)

[**Introduction** 1](#_Toc209860378)

[**System Design & Architecture** 1](#_Toc209860379)

[Implementation Highlights 1](#_Toc209860380)

[**Key Outcomes** 2](#_Toc209860381)

[**2. User Manual** 2](#_Toc209860382)

[**Getting Started** 2](#_Toc209860383)

[Primary Use Case Example 2](#_Toc209860384)

[**Troubleshooting** 2](#_Toc209860385)

[**Chatbot Flow Diagram** 2](#_Toc209860386)

[**3. Code Documentation** 3](#_Toc209860387)

[**Codebase Overview** 3](#_Toc209860388)

[**Dependencies** 3](#_Toc209860389)

[**Documentation Practices** 5](#_Toc209860390)

[**Repository Note** 5](#_Toc209860391)

[Project Screenshots : 5](#_Toc209860392)

## **1. Technical Report**

### **Introduction**

Infomate is a web-based chatbot developed for the ICT Department. It provides centralized, interactive access to departmental information such as faculty details, laboratories, curriculum, placements, and achievements. The system was designed to address challenges faced by students, parents, and visitors who often struggle to retrieve relevant information from lengthy PDF documents. By automating repetitive queries, Infomate reduces the workload on faculty and administrative staff while improving accessibility and transparency.

### **System Design & Architecture**

The system is modular, consisting of five main components:

1. **Frontend (User Interface)** – Built with **React.js**, providing a responsive, chat-based interface for user queries.
2. **Backend (Application Server)** – Developed in **Node.js**, handling request routing, authentication, and communication with APIs.
3. **AI Processing Layer** – Integrated with **Gemini API**, enabling natural language understanding and document-based knowledge retrieval.
4. **Knowledge Base** – Departmental PDF repository storing details on curriculum, faculty, placements, and labs.
5. **Deployment Layer** – Hosted on **Vercel** (frontend) and **Render** (backend) for global accessibility and scalability.

**System Architecture Diagram (textual representation):**

User → React Frontend → Node.js Backend → Gemini API → PDF Repository

### Implementation Highlights

* Modular design ensures maintainability and scalability.
* No database dependency in the initial phase; chatbot retrieves answers directly from departmental PDFs.
* Admins can update chatbot knowledge by replacing the PDF file.
* Deployment on Vercel and Render ensures automatic scaling and high uptime.

### **Key Outcomes**

* Reduced repetitive queries for faculty and staff.
* Enhanced student engagement during admissions through quick, accurate responses.
* Improved transparency and accessibility of departmental data.
* Scalable design for future expansion into other departments or multilingual support.

## **2. User Manual**

### **Getting Started**

1. Open Infomate in any modern web browser (desktop or mobile).
2. Enter a query in the chatbot window (e.g., “What subjects are in Semester V?”).
3. The system retrieves and displays the relevant answer instantly.

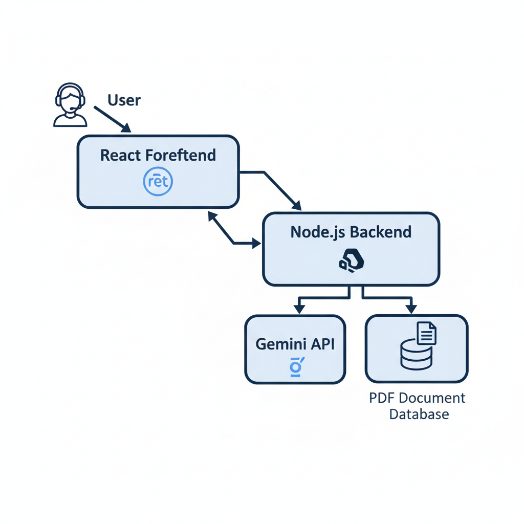
### Primary Use Case Example

**Query:** “Who are the IoT faculty members?”  
**Response:** Infomate fetches names and details of IoT-related faculty from the departmental PDF.

### **Troubleshooting**

* If no response is returned:
  + Rephrase the query using simpler keywords.
  + Ensure the departmental PDF is updated.
  + Check internet connection and refresh the page.

### **Chatbot Flow Diagram**



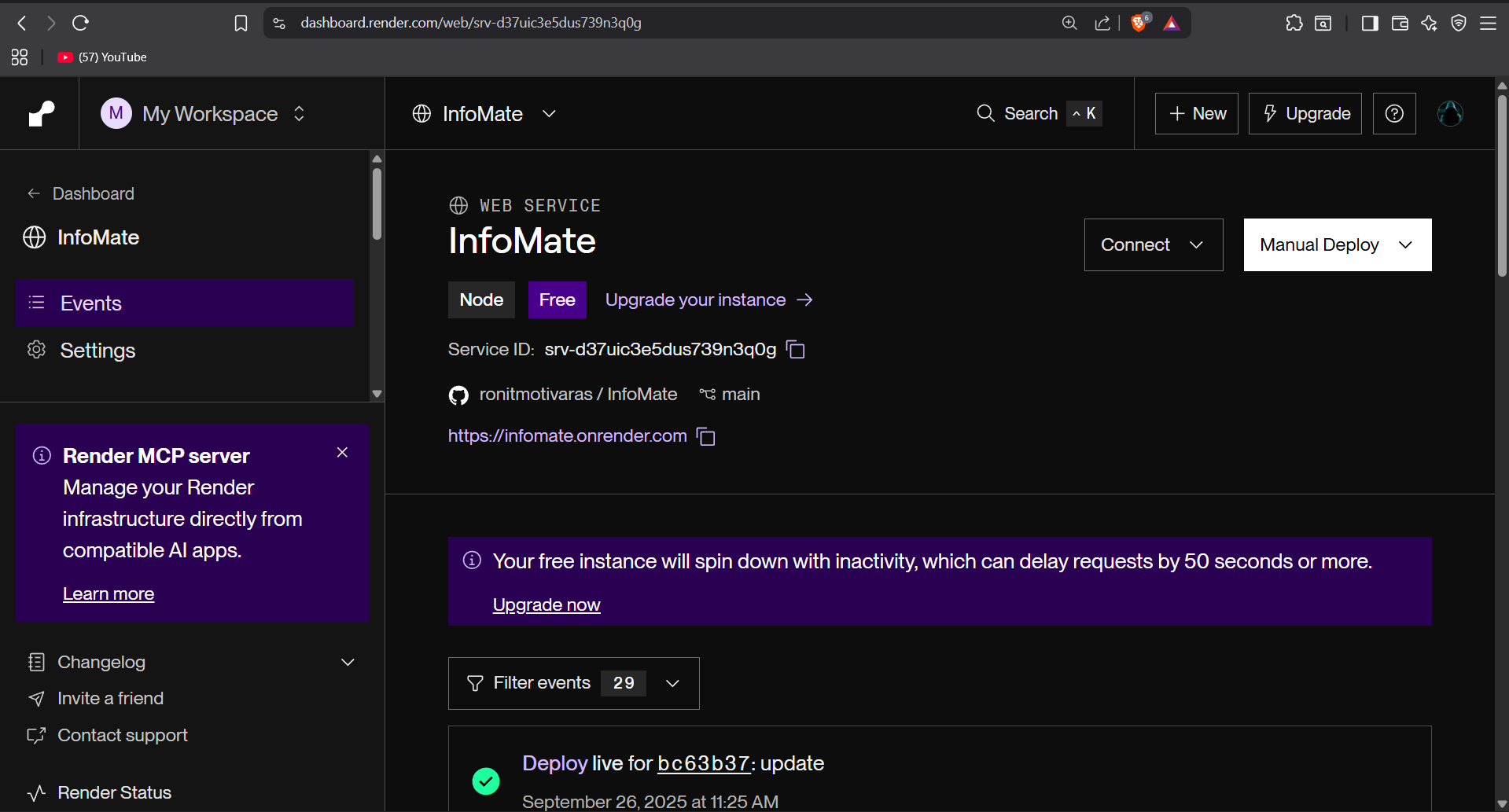
## **3. Code Documentation**

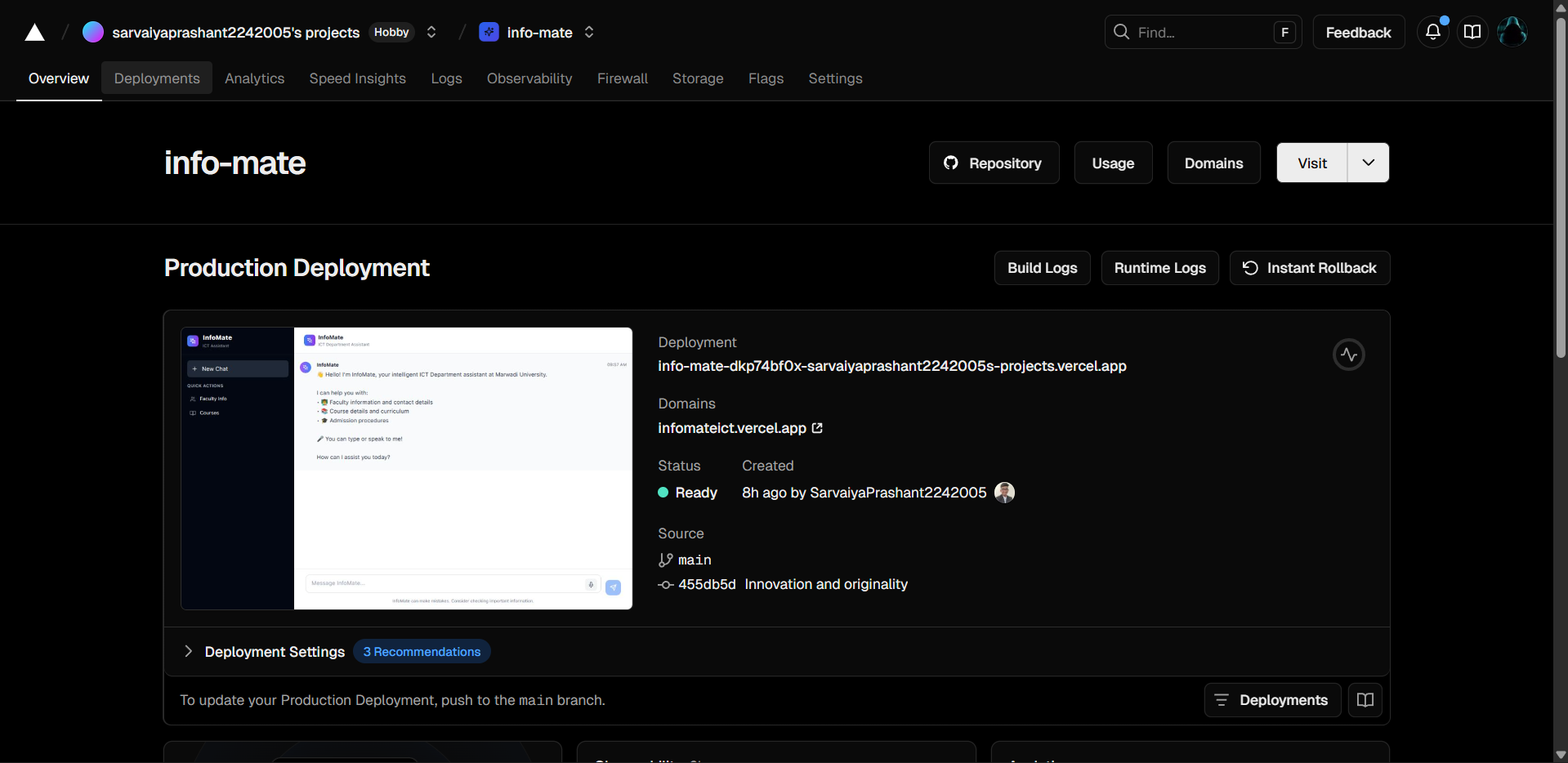
### **Codebase Overview**

* **Frontend (React.js)**
  + index.js: Main chat UI.
  + infomate.js: Handles input/output display.
* **Backend (Node.js)**
  + index.js: Configures Express server and routes.
  + routes/chat.js: API endpoints for chatbot requests.
  + .env: Communicates with Gemini API.
* **AI Integration**
  + pdfloader.js: Handles query parsing and PDF-based knowledge extraction.

### **Dependencies**

* **Frontend**:
* "@testing-library/jest-dom": "^5.17.0",
* "@testing-library/react": "^13.4.0",
* "@testing-library/user-event": "^13.5.0",
* "firebase": "^10.12.2",
* "lucide-react": "^0.263.1",
* "react": "^18.3.1",
* "react-dom": "^18.3.1",
* "react-markdown": "^10.1.0",
* "react-scripts": "5.0.1",
* "web-vitals": "^2.1.4"
* "@testing-library/jest-dom": "^5.17.0",
* "@testing-library/react": "^13.4.0",
* "@testing-library/user-event": "^13.5.0",
* "firebase": "^10.12.2",
* "lucide-react": "^0.263.1",
* "react": "^18.3.1",
* "react-dom": "^18.3.1",
* "react-markdown": "^10.1.0",
* "react-scripts": "5.0.1",
* "web-vitals": "^2.1.4"
* **Backend**:
* "@google/genai": "^1.20.0",
* "@google/generative-ai": "^0.20.0",
* "@huggingface/transformers": "^3.7.3",
* "body-parser": "^1.20.2",
* "cors": "^2.8.5",
* "dotenv": "^16.4.5",
* "express": "^4.19.2",
* "pdf-parse": "^1.1.1"
* **Hosting**: Vercel (frontend), Render (backend).





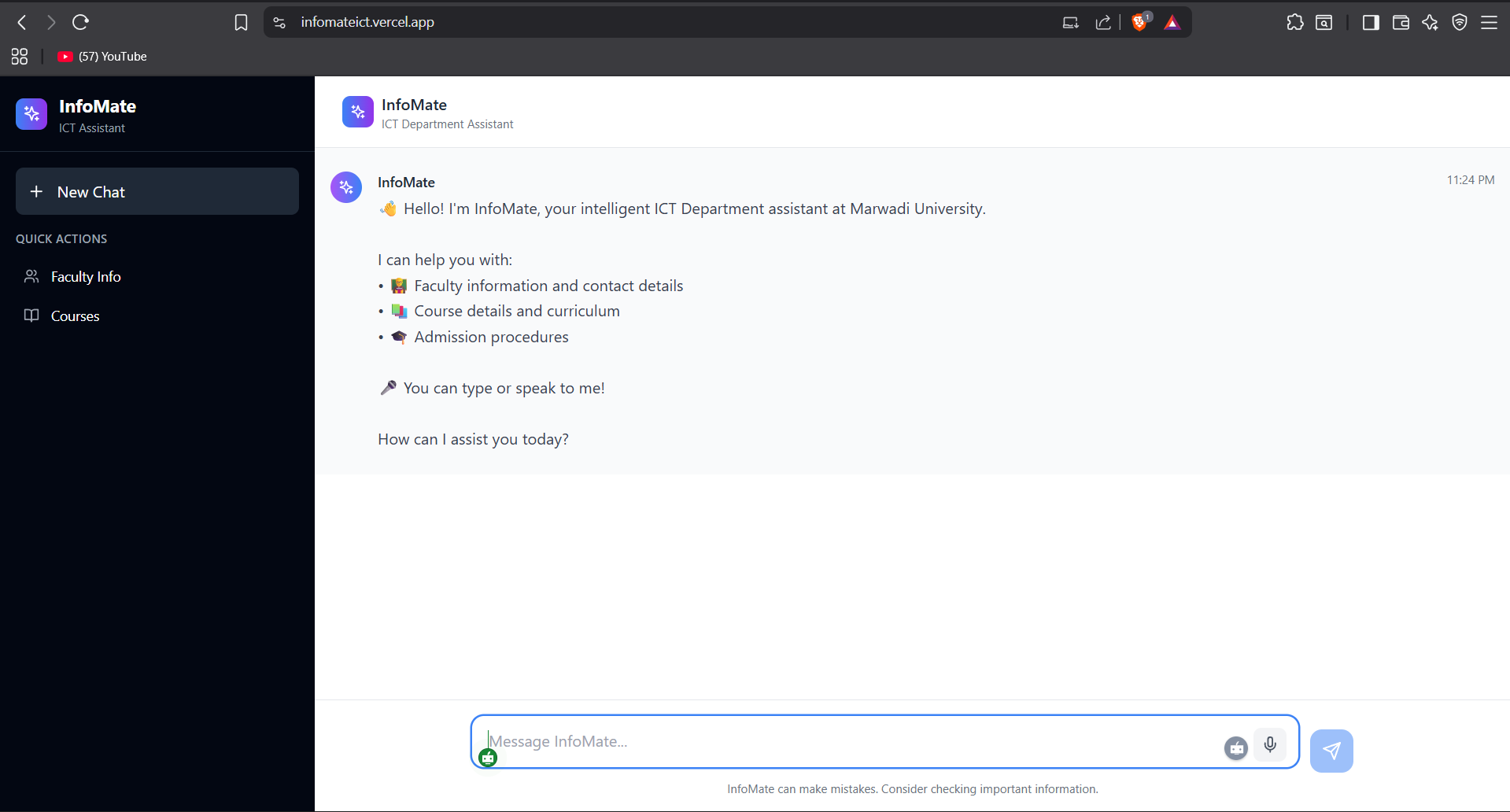
### **Documentation Practices**

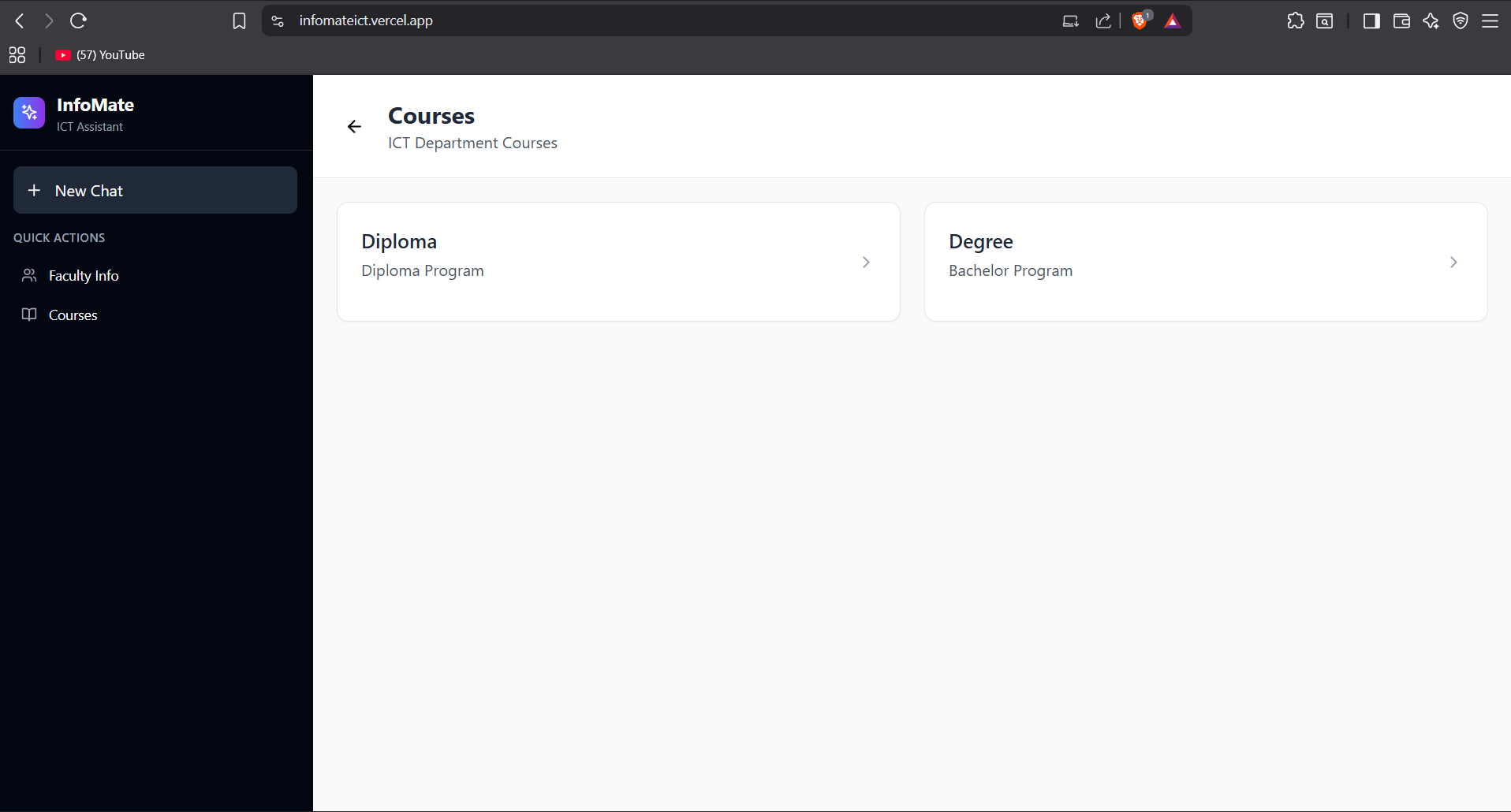
* **React Components** include inline comments explaining props and states.
* **Node.js Functions** documented with JSDoc format.   
  Example:
* /\*\*
* \* Handles incoming chat requests.
* \* @param {string} message - User input query.
* \* @returns {string} - Chatbot response.
* \*/

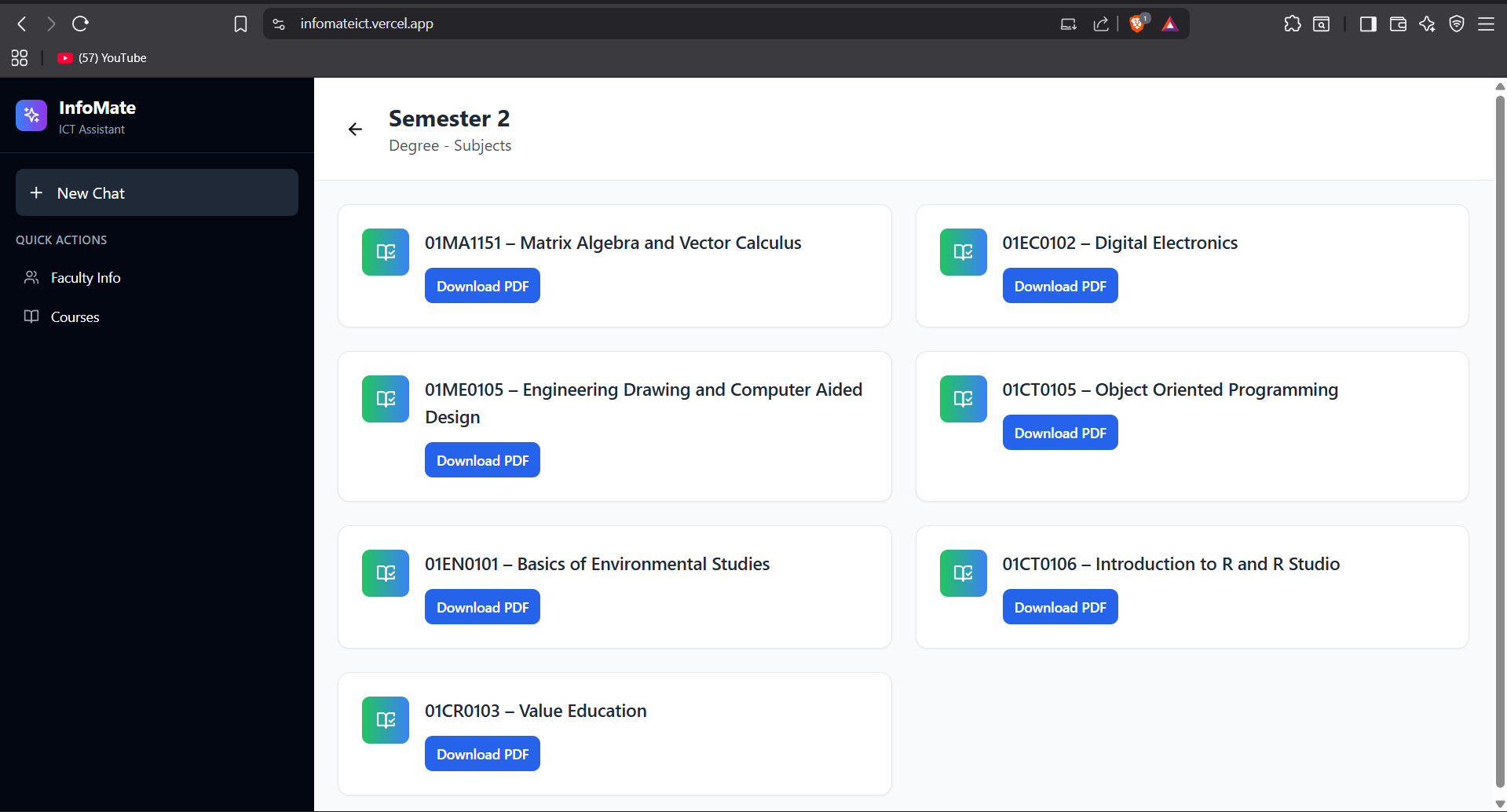
### **Repository Note**

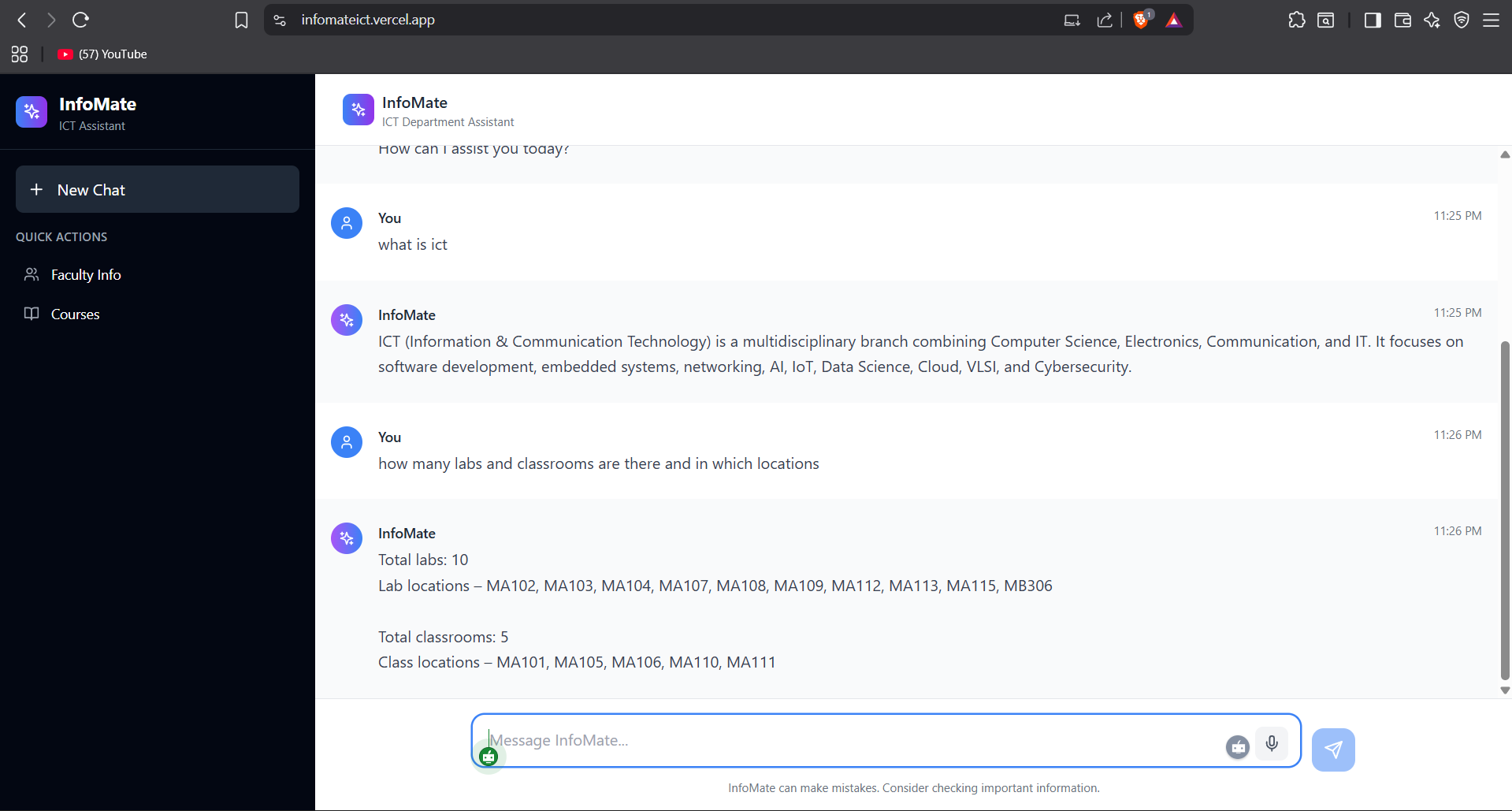
* Complete codebase stored in GitHub.
* README.md includes setup instructions, environment variable configuration, and deployment details.

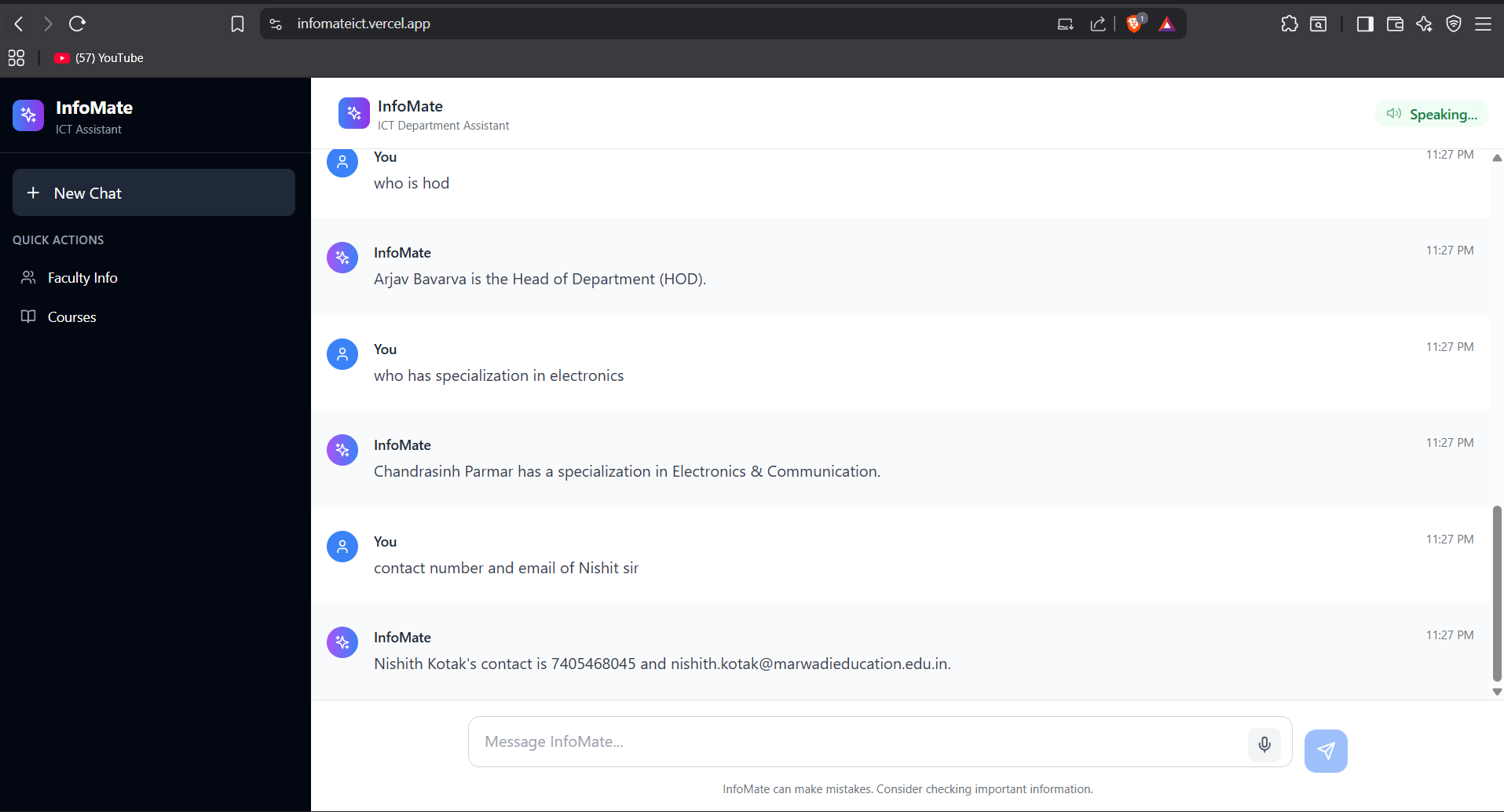
### **Project Screenshots :**

****

****

****

****

****