

# Dsu website~report

## 1. Introduction

The Student Portal System is a full-stack web-based application designed to automate student registration, authentication, data storage, and assistance using an AI-powered chatbot. The system consists of three main components: **Frontend, Backend, and an AI Chatbot module**. The platform aims to enhance user interaction by providing automated responses without human intervention, improving responsiveness and reducing manual workload.

The project primarily focuses on core development rather than infrastructure deployment. There is **no budget investment**, and all the development is done using open-source tools and cloud hosting platforms. The system is built to function efficiently for university-level usage, offering user registration, login-based data management, and intelligent chat-based support.

The major objectives of this system include:

- Creating a functional backend for database operations, user authentication, and secure API handling.
- Designing a minimal and responsive frontend UI for user form submission and chatbot interaction.
- Implementing an AI chatbot that can communicate seamlessly and handle student queries 24/7.

This report describes the entire architecture, libraries used, API implementation, model integration, UI behavior, system flow, and future scalability possibilities.

## 2. Frontend Overview

### 2.1 Frontend Purpose

The frontend is the user-facing component that interacts with students to collect input data and display content dynamically. It ensures responsiveness, form validation, API consumption, and chatbot UI integration.

### 2.2 Technologies Used

- **HTML, CSS, JavaScript** (core UI rendering)
- **React.js** for dynamic UI and component-based structure
- **Tailwind CSS** for fast styling and responsiveness
- **Vercel deployment** to host the UI

### 2.3 UI Components

The system includes these core UI components:

## 1. Registration Form

- Fields: Name, Email, Phone Number, Password
- Validations for email format and password length
- On submit, it calls the backend API to store user data

## 2. Login Interface

- Takes email and password
- Verifies credentials via backend API
- Once validated, user is allowed inside the system

## 3. Chatbot User Interface

- Fixed chat window UI on website
- Message input box and display bubbles
- Connects to AI chatbot API
- Responds instantly to student queries

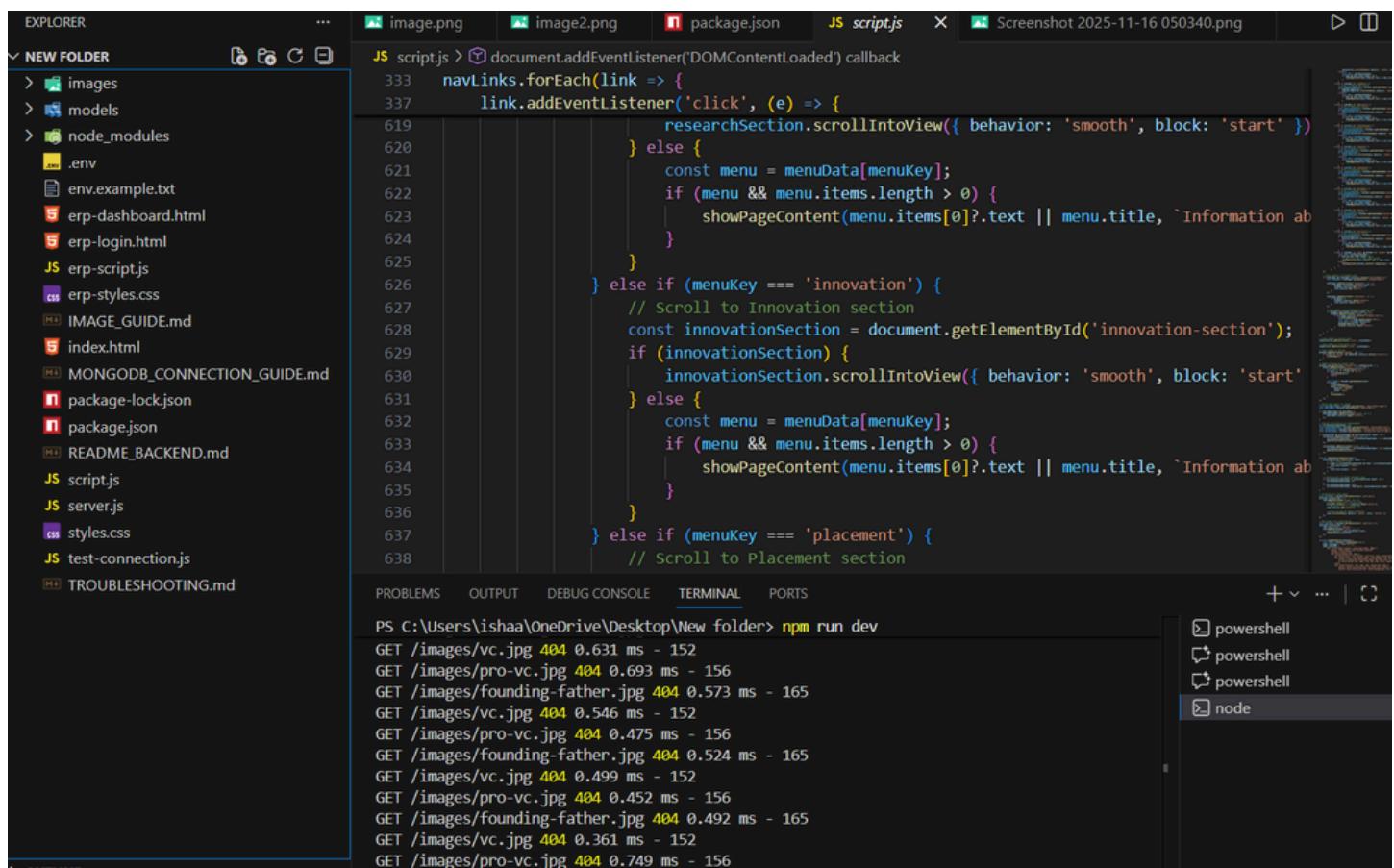
## 2.4 Frontend Working Flow

- User enters details in the form.
- Data is validated on the client side.
- Once validated, data is sent to backend API using fetch() or axios.
- Backend responds with a success/failure message.
- Chatbot UI renders messages and calls chatbot API to generate AI response.

The frontend does not directly deal with data storage; it only transmits data and displays received information from backend and chatbot module.

The screenshot shows the official website of Dayananda Sagar University. At the top, there is a dark header bar with the university's logo and name "DAYANANDA SAGAR UNIVERSITY". A search bar and a "Search" button are located on the right side of the header. Below the header, there is a navigation menu with links for CAREERS, Virtual Tour, Examination (which is underlined), News & Events, Newsletter, Alumni, Blog, ERP Login, and e-Brochure. The main content area features a large banner for "B.Tech Admissions 2026 - 27". The banner includes the university's NAAC accreditation grade "A+", the text "GRADE ACCREDITED BY NAAC", and "Leading & Most Trusted Management College of the Year 2021". It also mentions "Emerging Engineering Institute Research Capabilities 2022" and "BEST EMERGING". On the right side of the banner, there are icons for "APPLY NOW", a phone, a message bubble, and a globe. The background of the banner shows a building at night.

 Undergraduate Programs



```

image.png image2.png package.json JS script.js Screenshot 2025-11-16 050340.png
JS script.js > document.addEventListener('DOMContentLoaded') callback
  333   navLinks.forEach(link => {
  337     link.addEventListener('click', (e) => {
    619       researchSection.scrollIntoView({ behavior: 'smooth', block: 'start' })
    620     } else {
    621       const menu = menuData[menuKey];
    622       if (menu && menu.items.length > 0) {
    623         showPageContent(menu.items[0]?.text || menu.title, `Information ab
    624       }
    625     }
    626   } else if (menuKey === 'innovation') {
    627     // Scroll to Innovation section
    628     const innovationSection = document.getElementById('innovation-section');
    629     if (innovationSection) {
    630       innovationSection.scrollIntoView({ behavior: 'smooth', block: 'start' })
    631     } else {
    632       const menu = menuData[menuKey];
    633       if (menu && menu.items.length > 0) {
    634         showPageContent(menu.items[0]?.text || menu.title, `Information ab
    635       }
    636     }
    637   } else if (menuKey === 'placement') {
    638     // Scroll to Placement section
  }

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS
PS C:\Users\ishaa\OneDrive\Desktop>New folder> npm run dev
GET /images/vc.jpg 404 0.631 ms - 152
GET /images/pro-vc.jpg 404 0.693 ms - 156
GET /images/founding-father.jpg 404 0.573 ms - 165
GET /images/vc.jpg 404 0.546 ms - 152
GET /images/pro-vc.jpg 404 0.475 ms - 156
GET /images/founding-father.jpg 404 0.524 ms - 165
GET /images/vc.jpg 404 0.499 ms - 152
GET /images/pro-vc.jpg 404 0.452 ms - 165
GET /images/founding-father.jpg 404 0.492 ms - 165
GET /images/vc.jpg 404 0.361 ms - 152
GET /images/pro-vc.jpg 404 0.749 ms - 156
  
```

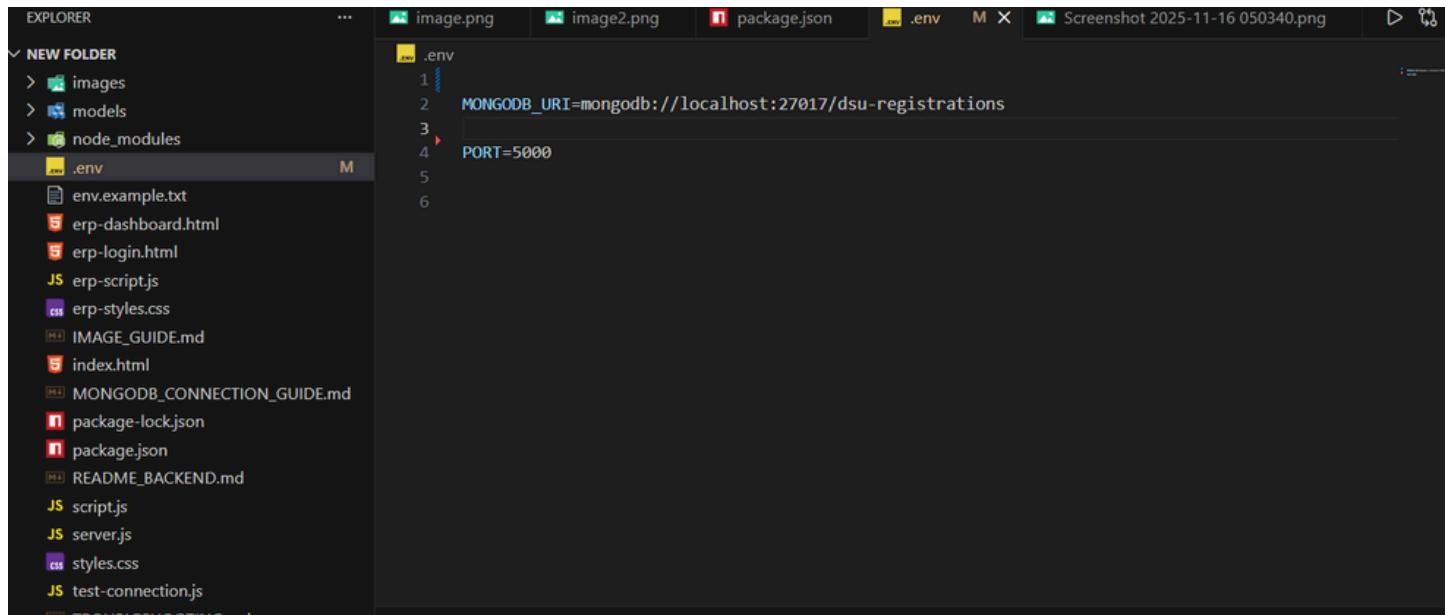
## 3. Backend Architecture

### 3.1 Backend Purpose

The backend manages student information, handles authentication, manages database queries, and exposes REST APIs for AI and frontend connection.

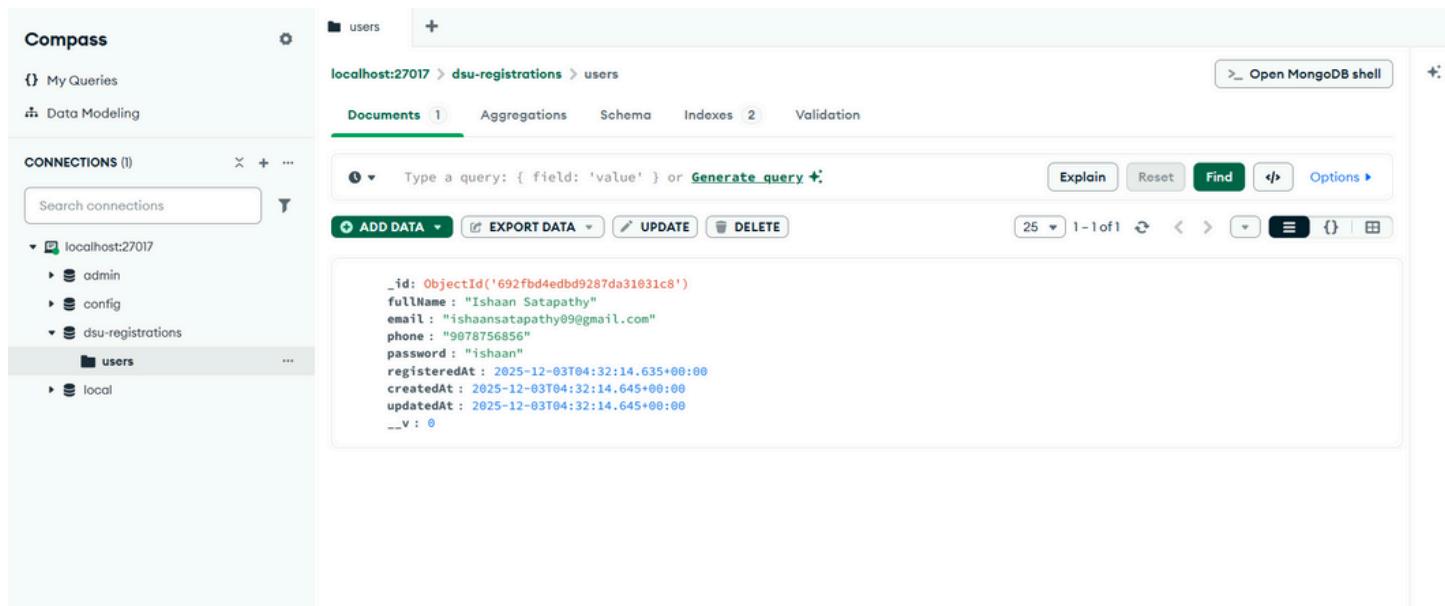
## 3.2 Technologies & Tools Used

- **Node.js** runtime environment
- **Express.js** to build REST APIs
- **MongoDB** as main database
- **Mongoose ORM** for schema and data modeling
- **bcrypt.js** for password hashing
- **JWT (JSON Web Token)** for authentication sessions
- **CORS, dotenv, body-parser** for API security and environment control
- **API hosted on Render / local server**



A screenshot of the VS Code interface showing the Explorer sidebar and the Editor tab. The Explorer sidebar shows a project structure with files like `image.png`, `image2.png`, `package.json`, `.env`, and `Screenshot 2025-11-16 050340.png`. The `.env` file is open in the Editor tab, displaying environment variables:

```
MONGODB_URI=mongodb://localhost:27017/dsu-registrations
PORT=5000
```



A screenshot of the Compass MongoDB interface. The left sidebar shows connections and a list of databases and collections. The main area shows the `users` collection under the `localhost:27017 > dsu-registrations` database. A single document is selected, showing the following data:

```
_id: ObjectId('692fb4edbd9287da31031c8')
fullName : "Ishaan Satapathy"
email : "ishaansatapathy09@gmail.com"
phone : "9078756856"
password : "ishaan"
registeredAt : 2025-12-03T04:32:14.635+00:00
createdAt : 2025-12-03T04:32:14.645+00:00
updatedAt : 2025-12-03T04:32:14.645+00:00
__v : 0
```

## 3.4 Authentication System

Two levels of authentication were implemented:

### 1. Password hashing

- Before saving password, it is converted into hash using bcrypt

- Makes raw password unreadable in DB

## 2. Login session using JWT

- Once user logs in, backend creates a token
- Token is sent back to frontend for future authorization
- Token has expiry time for security

## 3.5 Core Backend API Routes

Route	Method	Purpose
/register	POST	Save new student in DB
/login	POST	Verify user and return JWT
/chat	POST	Receive message → send to AI model → return response

## 3.6 Backend Working Flow

- Registration request received in JSON
- Backend checks if email already exists
- Password is hashed using bcrypt
- Data is saved in MongoDB via Mongoose
- Login request validates credentials
- If correct, JWT token is created and sent back
- AI chatbot request is forwarded to model and reply is returned

## 3.7 Advantages of Backend Design

- Passwords are never stored as raw text
- API endpoints accept only validated JSON
- JWT ensures secure sessions without saving login state in DB
- MongoDB scalability supports future expansion

# 4. AI Chatbot Module

## 4.1 Purpose of AI Chatbot

The chatbot removes the need for manual query handling. It provides automated responses to:

- Admission process questions

- Course details
- Registration help
- General student queries

## 4.2 Integration Approach

- Frontend sends user message to backend /chat
- Backend forwards message to AI model using API call
- AI generates response instantly
- Backend returns AI response to frontend UI

## 4.3 Behavior Characteristics

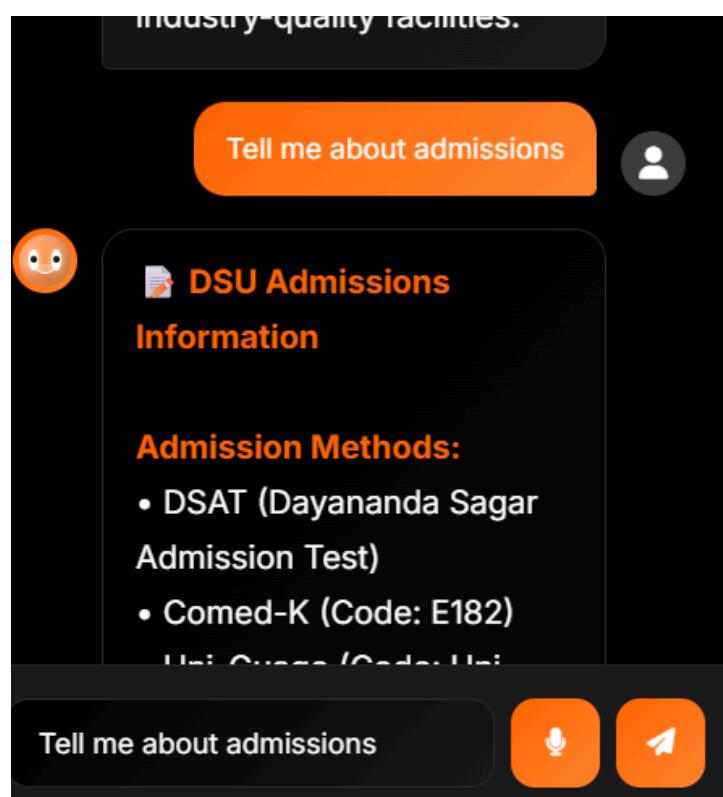
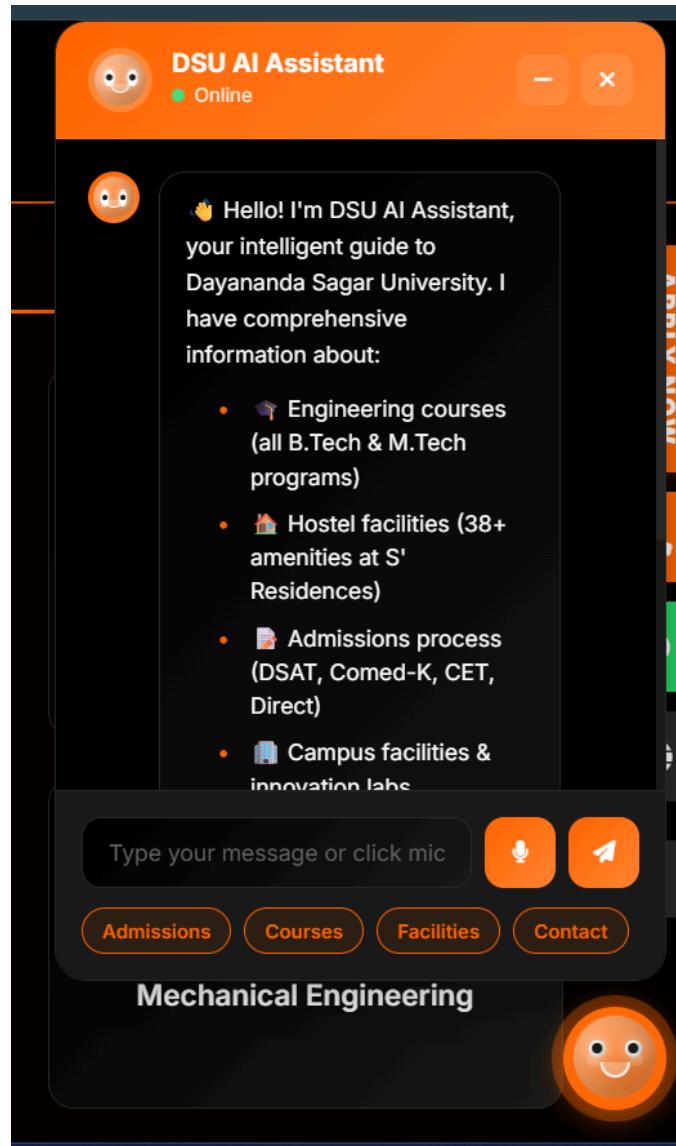
- Always conversational
- Gives short and accurate answers
- No memory retention across chats
- Avoids hallucinated data
- Politely declines unknown institution-specific facts
- Can greet, guide, and assist in form submission

## 4.4 AI Implementation Flow

1. Student types query in chat UI
2. Query goes to backend route
3. Backend calls AI model
4. Reply is received and returned
5. UI displays response bubble

## 4.5 Chatbot Limitations

- Does not fetch live data
- Not connected to external knowledge sources
- Cannot display student-specific stored DB details
- Answers only pre-known/general queries



## 4.6 Benefits

- Instant 24x7 support
- Reduces manual assistance
- Improves student experience
- Can be extended later with live API linking

## 5. System Workflow Diagram (Page 7–8)

1. User opens student portal
2. Portal loads static/dynamic React frontend
3. User registers → backend API → DB storage
4. User logs in → backend auth → JWT token
5. User sends chatbot query → backend → AI → reply

### Overall Data Flow:

User → Frontend UI → Backend API → (MongoDB OR AI Model) → Backend → Frontend UI → Display

## 6. Deployment Architecture (Page 8)

Component	Deployment
Frontend	Vercel
Backend API	Local/Render/Render-like hosting
Database	MongoDB Atlas or local MongoDB
AI Chatbot	API integrated through backend

Deployment ensures separation between modules while keeping costs zero.

## 7. Security Mechanisms (Page 9)

- 1. Database Security**
  - Password hashing
  - Unique email constraint
  - No sensitive data leakage
- 2. API Security**
  - CORS protection
  - dotenv variable protection

- No API keys in frontend
- All AI calls done server-side

### 3. User Security

- JWT session token
- Encrypted passwords

**LINK:** [Dayananda Sagar University | DSU - Bengaluru, India](#)

**GITHUB REPO:** [GitHub - ishaansatapathy/dsuwebsite](#)

Name	USN No	Contribution
Vishal Priyans	ENG24CT0071	Backend API development, authentication logic (JWT & bcrypt), database schema modeling
Ishaan Sachida Satapathy	ENG24CT0042	Frontend development using React.js, UI structuring & API integration
Ashrith Manoj	ENG24CT0032	AI Chatbot integration and response handling pipeline
Bhumika H Bhapri	ENG24CT0002	UI/UX enhancements, testing and documentation
Liya T cariappa	ENG24CT0010	System flow planning, validation testing and report compilation