**PROJECT REPORT**

REST API Development with Django and PostgreSQL

**Introduction:**

The goal of this project was to build a REST API using the Django framework, integrating it with a PostgreSQL database, and implementing CRUD functionality for a resource of our choice. The API provides operations to create, read, update, and delete data related to users in a college management system.

**Project Overview:**

1. **Technology Stack:**

- **Django:** Used as the web framework for API development.

- **PostgreSQL:** Chosen as the database management system for data storage.

- **Django REST Framework:** Employed for building the REST API.

- **Bootstrap and CSS:** Utilized for creating a user-friendly front-end interface.

- **JavaScript and AJAX:** Implemented for client-side functionality and asynchronous communication.

2. **Accomplishments:**

- Designed and implemented the database schema using Django models.

- Created API endpoints for CRUD operations on user data.

- Integrated PostgreSQL database with Django for data persistence.

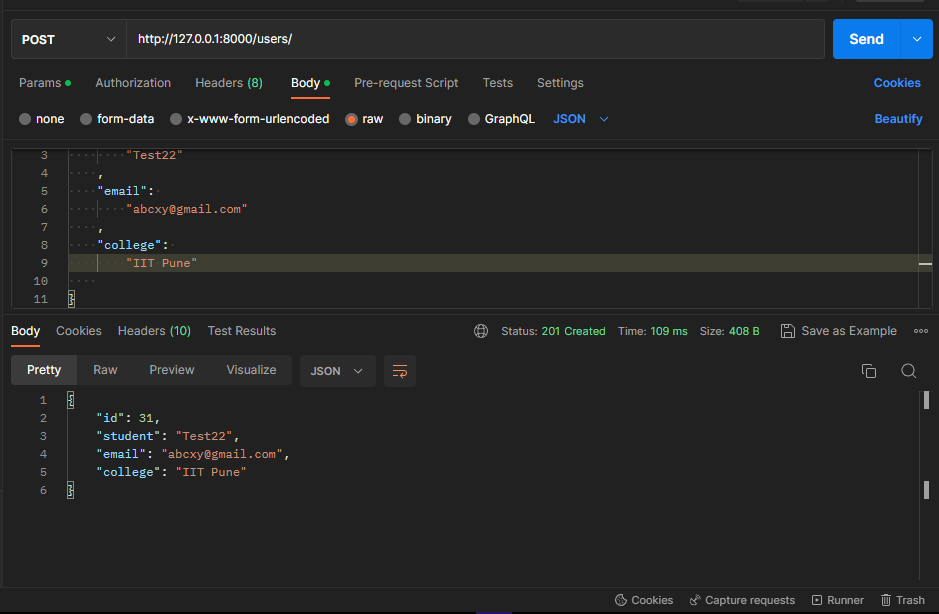
- Implemented form validation and serialization using Django REST Framework.

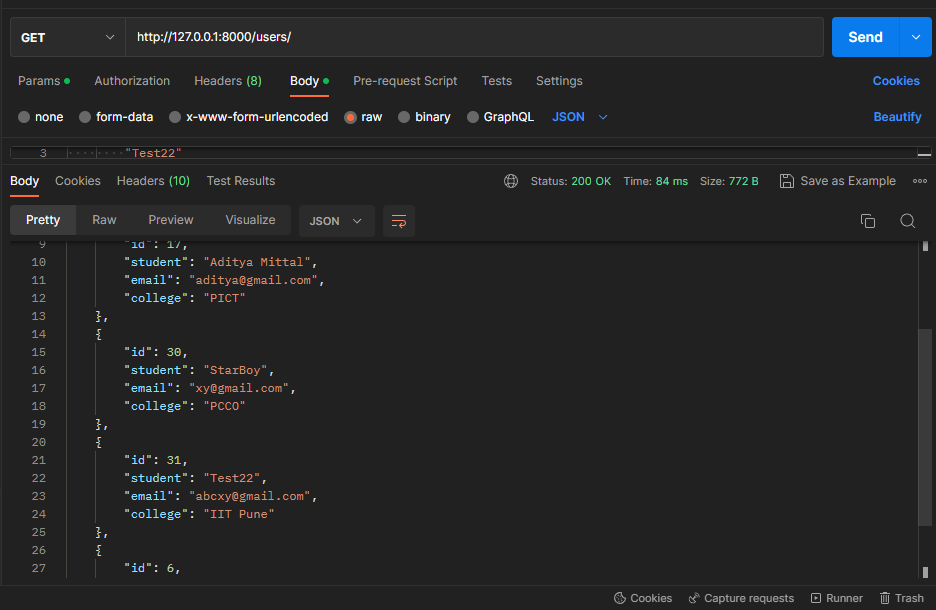
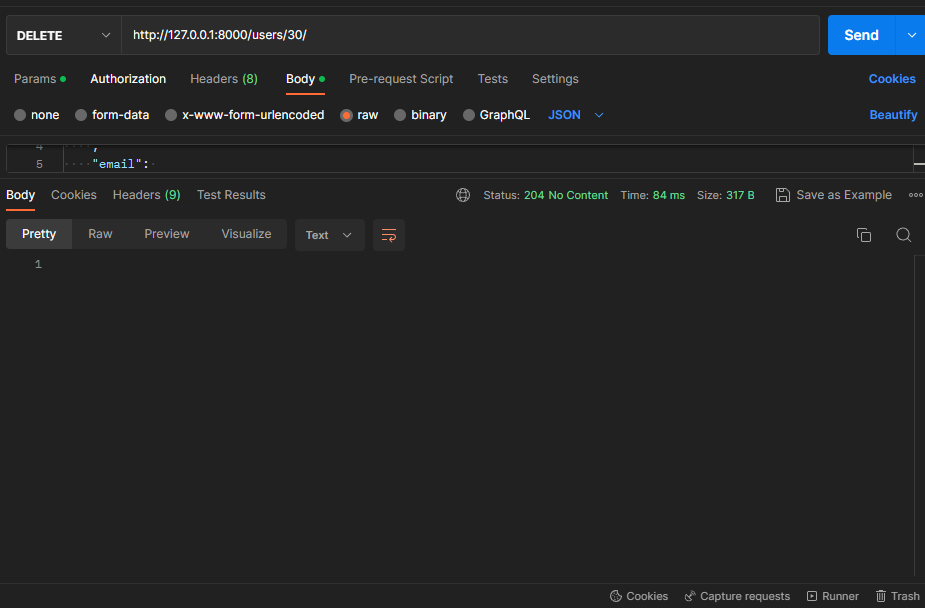
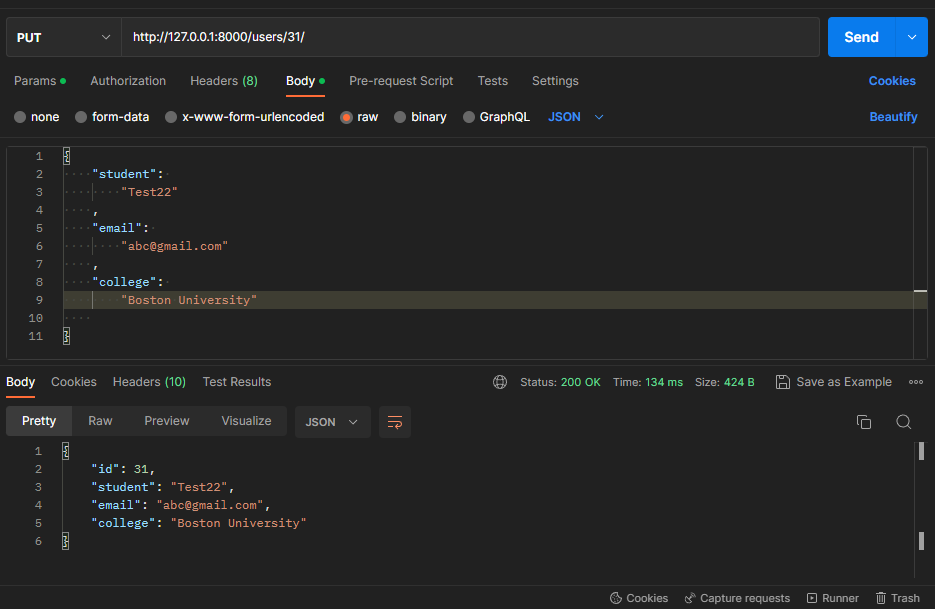
- Developed a responsive front-end interface using Bootstrap and CSS.

- Utilized JavaScript and AJAX for dynamic content updates and user interactions.

**Testing on Postman:**

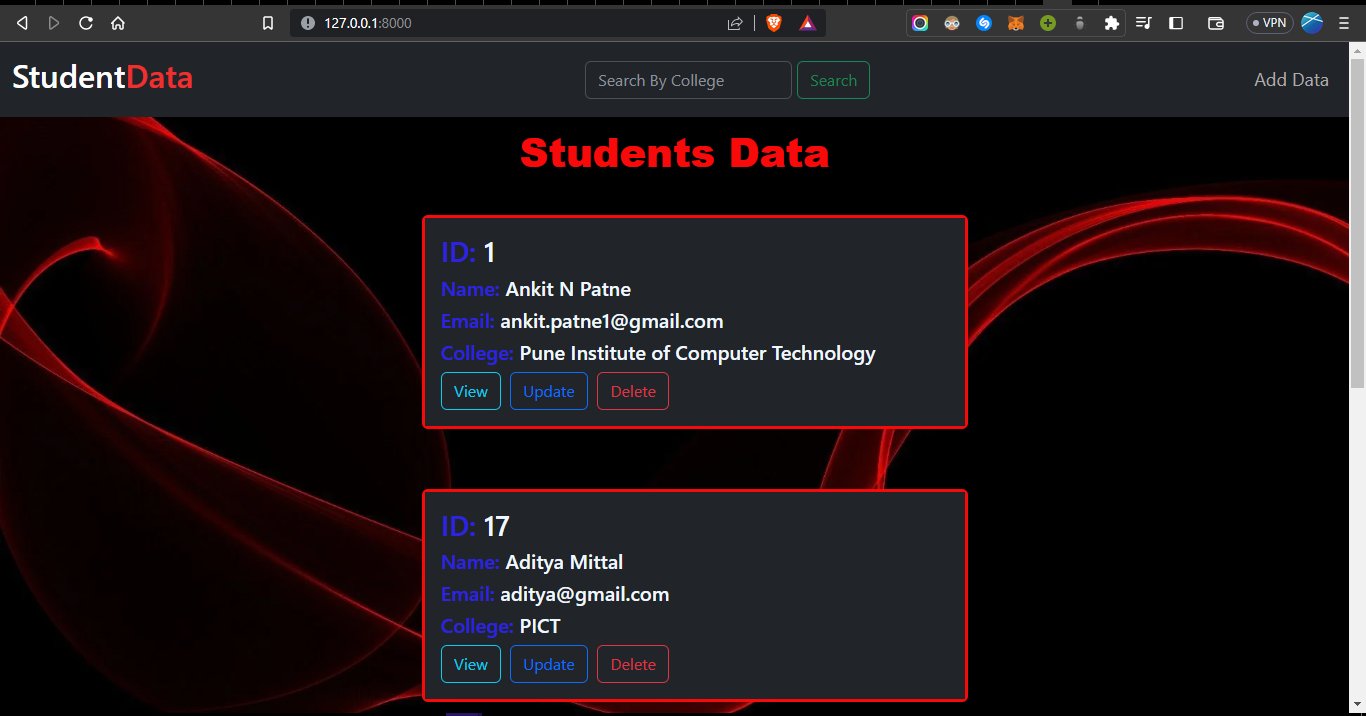
1. **POST:**

****

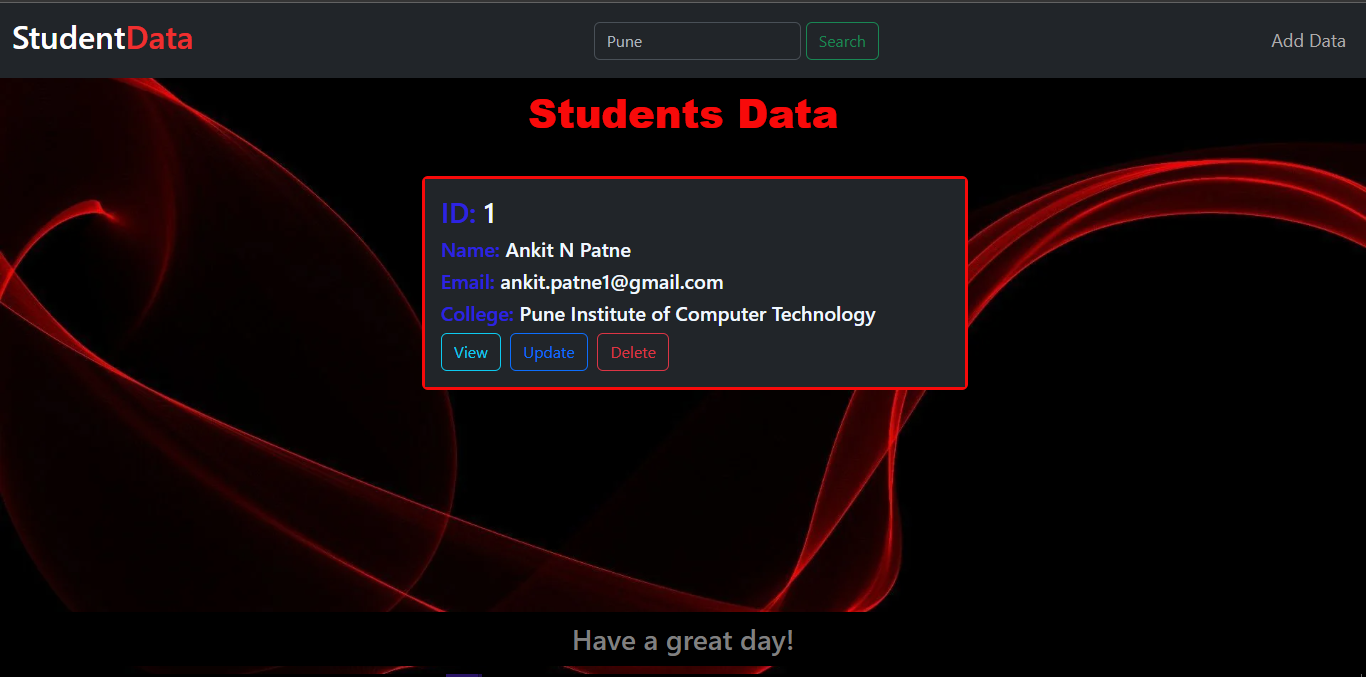
1. **GET:** ****
2. **DELETE:** 
3. **PUT:** ****

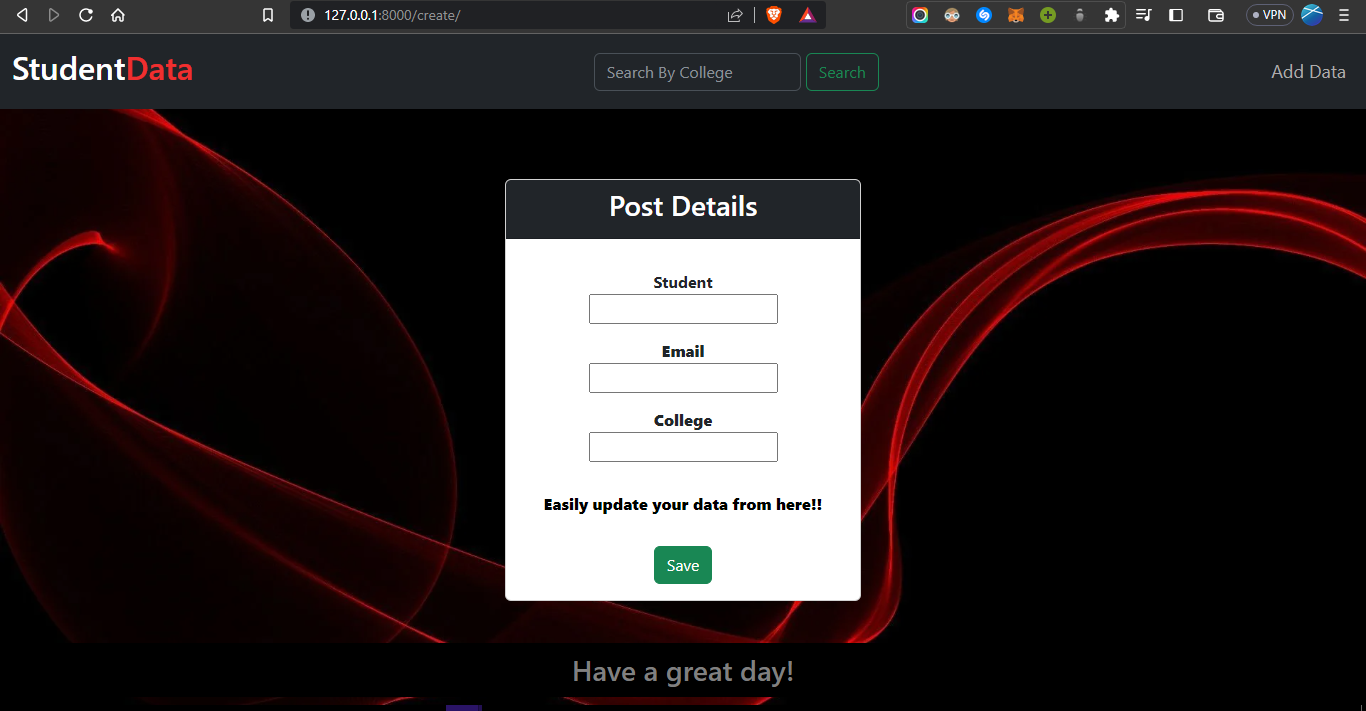
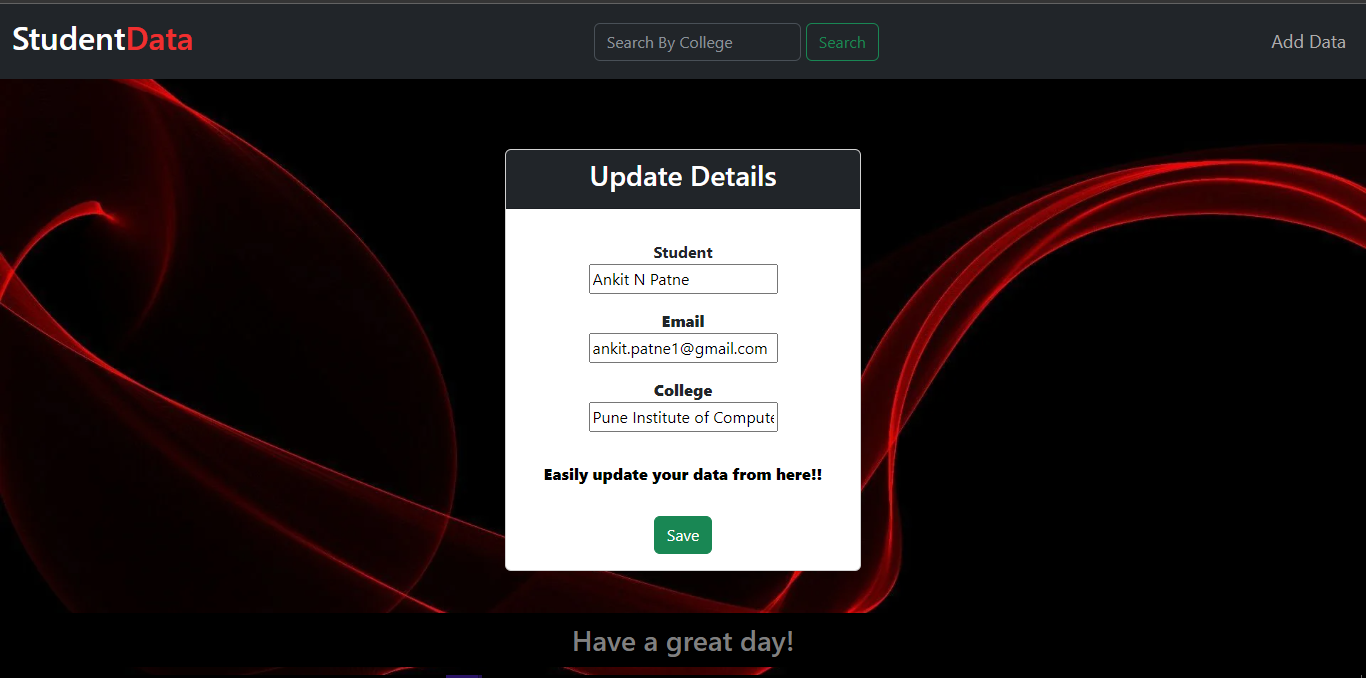
**Implementation on front-end:**

1. **Displaying the complete data:**

****

1. **Search function:**

****

1. **Adding data:** ****
2. **Updating data:**
3. **View specific data: A screenshot of a computer

   Description automatically generated**

**Challenges Faced and Solutions:**

1. **Learning Curve:** The initial challenge was familiarizing myself with the Django framework, Django REST Framework, and integrating it with PostgreSQL. Overcoming this challenge involved reading some documentation and watching tutorials.

2. **API Testing:** Testing the API endpoints using tools like Postman posed a challenge. To overcome this, I referred to tutorials on YouTube to understand how to write effective tests. Regular testing and debugging helped identify and resolve any issues in the API implementation.

3. **Front-End Development:** Creating a user-friendly front-end interface using Bootstrap and CSS required attention to detail and an understanding of responsive design principles. I overcame this challenge by following Bootstrap documentation and examples, experimenting with different layouts, and seeking inspiration from existing templates and designs.

4. **Client-Side Functionality:** Implementing dynamic behaviour and interactivity on the front-end using JavaScript and AJAX was another challenge. Since I wasn’t much familiar with AJAX, this challenge took a bit longer than expected. I tackled this by breaking down the functionality into smaller tasks, referring to JavaScript and AJAX documentation, and gradually building the required features step by step. Additionally, seeking help from online communities and ChatGPT proved beneficial for resolving specific issues.

**Conclusion:**

In this project, I successfully built a REST API using Django and PostgreSQL, providing CRUD functionality for user data. I overcame various challenges by leveraging documentation, seeking help from online communities, and persistently testing and debugging the application. The project not only enhanced my understanding of Django, Django REST Framework, and front-end development but also strengthened my debugging skills. Moving forward, I will continue to refine and expand the project, incorporating additional features and improvements based on various feedback and requirements.