A
Summer Internship Report
On
"Backend with Node JS"

(Summer Internship - II)

Prepared by

Vraj Desai (21IT032)

Under the Supervision of

Prof. Dhaval Patel

Submitted to

Charotar University of Science & Technology (CHARUSAT)
for the Partial Fulfillment of the Requirements for the
Degree of Bachelor of Technology (B.Tech.)
for Semester 7

Submitted at

CHARUSAT
CHAROTAR UNIVERSITY OF SCIENCE AND TECHNOLOGY
Accredited with Grade A+ by NAAC
Accredited with Grade A by KCG

SMT. KUNDANBEN DINSHA PATEL DEPARTMENT OF INFORMATION TECHNOLOGY
Chandu Bhai S. Patel Institute of Technology (CSPIT)
Faculty of Technology & Engineering (FTE), CHARUSAT
At: Changa, Dist.: Anand, Pin: 388421.
July 2024

CHARUSAT
CHAROTAR UNIVERSITY OF SCIENCE AND TECHNOLOGY

Accredited with Grade A+ by NAAC
Accredited with Grade A by KCG

CERTIFICATE

This is to certify that the report entitled "**Backend with Node JS**" is a bonafied work carried out by **Vraj Desai (21IT032)** under the guidance and supervision of **Prof. Dhaval Patel and Mrs. Priya Soni** for the subject **Summer Internship – II** of **7th Semester** of Bachelor of Technology in **Information Technology** at Chandu Bhai S. Patel Institute of Technology (CSPIT), Faculty of Technology & Engineering (FTE) – CHARUSAT, Gujarat.

To the best of my knowledge and belief, this work embodies the work of candidate himself, has duly been completed, and fulfills the requirement of the ordinance relating to the B.Tech. Degree of the University and is up to the standard in respect of content, presentation and language for being referred by the examiner(s).

Under the supervision of,

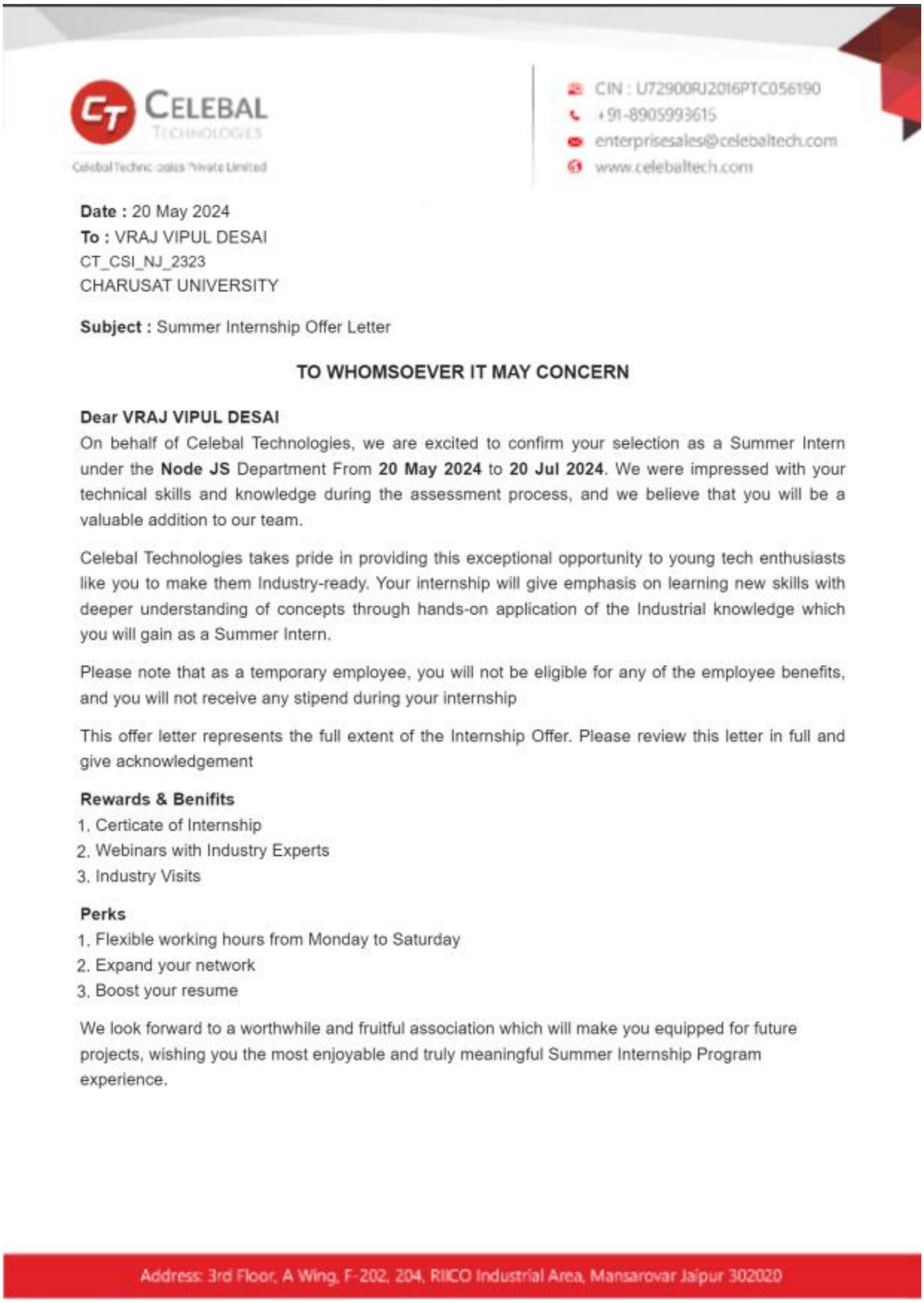
WEETECH SOLUTION PVT. LTD
[illegible]
Auth Sign.

Prof.Dhaval Patel
Assistant Professor
Smt. Kundanben Dinsha Patel Department of Information Technology
CSPIT, FTE, CHARUSAT, Changa, Gujarat

Mrs. Priya Soni
HR Manager
HR Department
Celebal Technologies

Dr. Parth Shah
Head of Department (IT)
CHARUSAT, Changa, Gujarat.

Chandu Bhai S. Patel Institute of Technology (CSPIT)
Faculty of Technology & Engineering (FTE), CHARUSAT
At: Changa, Ta. Petlad, Dist. Anand, Pin: 388421. Gujarat



CELEBAL TECHNOLOGIES

Celebal Technologies Private Limited

CIN : U72900RJ2016PTC056190
+91-8905993615
enterprisesales@celebaltech.com
www.celebaltech.com

Date : 20 May 2024
To : VRAJ VIPUL DESAI
CT_CSI_NJ_2323
CHARUSAT UNIVERSITY

Subject : Summer Internship Offer Letter

TO WHOMSOEVER IT MAY CONCERN

Dear VRAJ VIPUL DESAI
On behalf of Celebal Technologies, we are excited to confirm your selection as a Summer Intern under the **Node JS** Department From **20 May 2024** to **20 Jul 2024**. We were impressed with your technical skills and knowledge during the assessment process, and we believe that you will be a valuable addition to our team.

Celebal Technologies takes pride in providing this exceptional opportunity to young tech enthusiasts like you to make them Industry-ready. Your internship will give emphasis on learning new skills with deeper understanding of concepts through hands-on application of the Industrial knowledge which you will gain as a Summer Intern.

Please note that as a temporary employee, you will not be eligible for any of the employee benefits, and you will not receive any stipend during your internship

This offer letter represents the full extent of the Internship Offer. Please review this letter in full and give acknowledgement

Rewards & Benifits
1. Certicate of Internship
2. Webinars with Industry Experts
3. Industry Visits

Perks
1. Flexible working hours from Monday to Saturday
2. Expand your network
3. Boost your resume

We look forward to a worthwhile and fruitful association which will make you equipped for future projects, wishing you the most enjoyable and truly meaningful Summer Internship Program experience.

Address: 3rd Floor, A Wing, F-202, 204, RIICO Industrial Area, Mansarovar Jaipur 302020

ACKNOWLEDGEMENT

- I am privileged to have this opportunity to express my gratitude and acknowledge everyone’s never ending support and valuable contributions for this technology training.

- My sincere thanks go to Principal Sir Dr. Trushit Upadhyay, and Head of the department Dr. Parth Shah who provided us an opportunity to work on a summer internship technology training and to be able to present the same.

- Last but not the least , I would like to thank my friends and family for supporting me spiritually throughout this technology training and for always being a constant source of inspiration. We also place on record, our sense of gratitude to one and all, who directly or indirectly, have lent their hand in this venture.

ABSTRACT

During my internship, I focused on developing backend solutions using Node.js. The primary goal of this project was to create robust and efficient APIs for an e-commerce website. This involved several key stages, starting with gaining a solid understanding of Node.js fundamentals, including core modules and npm. I then delved into asynchronous programming, which is crucial for handling multiple tasks concurrently in a non-blocking manner.

My work also included learning and utilizing Express.js for building server-side applications, and integrating MongoDB for database management. I developed RESTful APIs, which facilitated smooth communication between the client and server, and implemented authentication and security measures to protect user data.

Additionally, I explored advanced features of Express.js to enhance the performance and scalability of the applications. By the end of the internship, I had successfully developed a comprehensive backend system for an e-commerce website, demonstrating my proficiency in Node.js and my ability to create secure, efficient, and scalable web applications.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

DESCRIPTION OF COMPANY/ORGANIZATION

Celebal Technologies is a premier provider of innovative technology solutions, recognized globally as the AI Partner of the Year. With a deep expertise in various industries, Celebal Technologies leverages cutting-edge technologies to address complex business challenges, empowering enterprises to achieve digital transformation. Their solutions are designed to enhance operational efficiency, boost revenue, and ensure robust security and compliance. By partnering with leading Fortune 500 companies, particularly in the manufacturing sector, Celebal Technologies helps businesses streamline operations and improve supply chain management through modern cloud technologies like Azure and Databricks.

The company offers a range of user-centric products that revolutionize business processes. These include the Enterprise Knowledge Advisor (EKA) powered by OpenAI, an AI Tutor for educational purposes, and the Intelligent Enterprise Data Lake with industry-specific KPIs. Additionally, Celebal Technologies provides solutions like the SAP Chatbot for quick ERP/CRM access, an Industrial IoT predictive maintenance solution, a Digital Twin for decision-making optimization, and Sustainability 2.0 for managing carbon footprints. Their Automated Invoice Processing, also powered by AI, exemplifies their commitment to integrating advanced technologies into practical business applications.

Celebal Technologies stands as a trusted technological partner for successful companies and multinational enterprises. By establishing modern business practices and leveraging technological advancements, they transform innovative ideas into reality. With a global presence and a strong focus on industry-specific solutions, Celebal Technologies continues to deliver enduring results, driving the digital transformation journey for businesses worldwide.

CHAPTER 1 INTRODUCTION

1.1 INTERNSHIP OBJECTIVE:

The primary objective of this internship is to develop a comprehensive backend system for an e-commerce platform using Node.js. This system aims to enhance the efficiency and security of online transactions, improve user authentication, and ensure seamless integration with a MongoDB database. Additionally, the internship focuses on implementing advanced Express.js features and robust RESTful APIs to facilitate smooth client-server communication. The goal is to integrate these backend solutions into a scalable and secure application framework, enabling efficient handling of real-time user data and transactions. This project bridges theoretical knowledge of Node.js with practical applications, contributing to more robust and scalable e-commerce solutions.

Key Points:

1. Develop a comprehensive backend system using Node.js.
2. Enhance efficiency and security in online transactions.
3. Implement user authentication and secure data handling.
4. Utilize advanced Express.js features and MongoDB integration.
5. Create scalable and robust RESTful APIs for client-server communication.
6. Apply Node.js concepts to practical e-commerce applications.

Table 1.2: Overview of Internship Activities:

	Date	Day	Name of Topic
Week 1	20/05/24	Monday	1. What is Node.js? 2. Installation and Setup 3. Your First Node.js Application 4. Node.js Architecture 5. The Node.js Ecosystem
	22/05/24	Wednesday	1. Basic JavaScript Refresher 2. Node.js REPL 3. Modules in Node.js 4. Creating a Simple Server 5. Understanding package.json
	24/05/24	Friday	Assignment - Set up Node.js and create your first application.
	25/05/24	Saturday	Set up the final project structure and initialize Node.js with Express.js.
Week 2	27/05/24	Monday	1. File System Module 2. HTTP Module 3. Events Module 4. Util Module 5. Path Module
	29/05/24	Wednesday	1. NPM Basics 2. Installing Packages 3. Creating a Package 4. Version Management 5. NPM Scripts
	31/05/24	Friday	Assignment - Build a file management tool using core modules.
	01/06/24	Saturday	Utilized core modules and set up basic npm packages for the final project
Week 3	03/06/24	Monday	1. Callback Functions 2. Handling Errors in Callbacks 3. Promises 4. Async/Await 5. Event Loop
	05/06/24	Wednesday	1. Handling Asynchronous Operations 2. File System with Promises 3. Creating Promises 4. Promise Chaining

			5. Error Handling in Async/Await
	07/06/24	Friday	Assignment - Convert callback-based code to Promises and Async/Await.
	08/06/24	Saturday	Implemented Promises and async/await for database interactions in the project
Week 4	10/06/24	Monday	1. What is Express.js? 2. Setting Up an Express Server 3. Routing 4. Middleware 5. Handling Requests and Responses
	12/06/24	Wednesday	1. Query Parameters and URL Parameters 2. Static Files 3. Template Engines 4. Express Router 5. Basic Express.js Security
	14/06/24	Friday	Assignment - Create a basic web server with Express.js.
	15/06/240	Saturday	Set up an Express server with routing and middleware for user authentication in the project
Week 5	17/06/24	Monday	1. Introduction to Databases 2. Using MongoDB with Node.js and CRUD Operations 3. Relational Databases and Node.js 4. Database Schema Design
	19/06/24	Wednesday	1. Data Validation and Sanitization 2. Connecting to a Database 3. Migrations and Seeding 4. Advanced Query Techniques
	21/06/24	Friday	Assignment - Build a CRUD application with MongoDB
	22/06/24	Saturday	Implemented MongoDB with CRUD operations for product management
Week 6	24/06/24	Monday	1. Introduction to REST 2. Setting Up a REST API 3. Middleware for REST APIs 4. Authentication in APIs
	26/06/24	Wednesday	1. Error Handling in APIs

			2. Documenting APIs 3. Testing APIs 4. Versioning APIs
	28/06/24	Friday	Assignment - Develop a RESTful API.
	29/06/24	Saturday	Developed and tested RESTful API endpoints for product and user data
Week 7	01/07/24	Monday	1. User Authentication 2. Hashing and Salting Passwords 3. JSON Web Tokens (JWT) 4. OAuth and Social Login 5. Session Management
	03/07/24	Wednesday	1. HTTPS and SSL/TLS 2. Security Best Practices 3. Preventing Common Attacks 4. Rate Limiting and Throttling 5. Dependency Security
	05/07/24	Friday	Assignment - Implement JWT-based authentication in your API.
	06/07/24	Saturday	Implemented JWT-based authentication and apply security best practices
Week 8	08/07/24	Monday	1. Advanced Routing 2. Error Handling in Express 3. Performance Optimization 4. Middleware Stacks
	10/07/24	Wednesday	1. Express and WebSocket Integration 2. File Uploads 3. Server 4. Express App Structure
	12/0724	Friday	Assignment - Enhance your Express.js application.
	13/07/24	Saturday	Enhanced Express.js application with advanced routing and error handling
Week 9	15/07/24 – 19/07/24	Monday - Saturday	1. Implemented the user shopping cart functionality (add to cart, remove from cart, view cart). 2. Developed the checkout process including order placement and payment integration.

			3. Conducted comprehensive testing and debugging of the entire application using Postman

CHAPTER 2 TOOLS AND TECHNOLOGIES

2.1 INTRODUCTION TO NODE.JS:

- Node.js is a high-performance, event-driven JavaScript runtime that executes code server-side. It is built on the V8 JavaScript engine, which is known for its speed and efficiency.
- Node.js is widely used for developing scalable network applications and is ideal for building backend services such as APIs and microservices.
- The platform supports non-blocking, asynchronous programming, which enhances performance and handles multiple operations concurrently.
- npm (Node Package Manager) is a key component, providing access to a vast ecosystem of libraries and modules that facilitate the development process.

2.1.1 Asynchronous Programming in Node.js:

- Asynchronous programming allows for the execution of multiple tasks simultaneously without waiting for each task to complete before moving on to the next.
- Node.js utilizes callback functions, promises, and async/await syntax to manage asynchronous operations efficiently.
- This approach is essential for handling I/O operations, such as database queries and file system access, in a non-blocking manner, improving the application's responsiveness.

2.1.2 Core Modules and npm:

- Node.js comes with several core modules like http, fs, path, and url, which provide essential functionalities for server-side development.
- npm enables developers to easily install, update, and manage third-party packages, enhancing productivity and reducing development time.
- Commonly used npm packages include express for web application frameworks, mongoose for MongoDB interactions, and jsonwebtoken for implementing authentication.

2.2 INTRODUCTION TO EXPRESS.JS:

- Express.js is a minimal and flexible Node.js web application framework that provides a robust set of features for building web and mobile applications.
- It simplifies server creation and routing, making it easier to build RESTful APIs and handle HTTP requests.
- Express.js supports middleware, which are functions that execute during the lifecycle

of a request to the server, allowing for modular and reusable code.

2.2.1 RESTful API Development:

- REST (Representational State Transfer) is an architectural style for designing networked applications. RESTful APIs use HTTP methods such as GET, POST, PUT, and DELETE to perform CRUD (Create, Read, Update, Delete) operations.
- Express.js provides methods to define routes and their handlers, making it straightforward to implement RESTful APIs.
- Middleware functions in Express.js can handle authentication, validation, error handling, and other processing tasks, enhancing the API's functionality and security.

2.2.2 Advanced Features of Express.js:

- Express.js supports template engines like Pug and EJS, which facilitate dynamic HTML generation.
- It allows for the organization of routes into separate modules, improving code maintainability.
- Middleware such as body-parser and cors help in processing request bodies and handling cross-origin resource sharing, respectively.

2.3 WORKING WITH DATABASES:

- MongoDB is a NoSQL database that stores data in JSON-like documents, providing flexibility and scalability for modern applications.
- Mongoose is an Object Data Modeling (ODM) library for MongoDB and Node.js, offering a straightforward schema-based solution to model application data.

2.3.1 MongoDB:

- MongoDB is designed for high availability and horizontal scaling, making it suitable for applications with large amounts of data and high throughput.
- It supports a rich query language and features like indexing, aggregation, and geospatial queries.

2.3.2 Mongoose:

- Mongoose simplifies data validation, casting, and business logic by defining schemas and models.
- It provides middleware hooks for pre and post-processing during document creation, updates, and deletion, enabling complex data workflows.

2.4 AUTHENTICATION AND SECURITY:

- Authentication and security are critical aspects of backend development to ensure that only authorized users can access the application and their data is protected.

2.4.1 Authentication:

- JWT (JSON Web Tokens) are commonly used for stateless authentication. They encode user information and are signed to ensure integrity.
- Passport.js is an authentication middleware for Node.js, supporting various strategies like local, OAuth, and JWT-based authentication.

2.4.2 Security Best Practices:

- Implementing HTTPS to encrypt data in transit.
- Using environment variables to manage sensitive configuration data.
- Regularly updating dependencies to mitigate security vulnerabilities.
- Implementing rate limiting and input validation to prevent attacks such as DDoS and SQL injection.

2.5 SUMMARY:

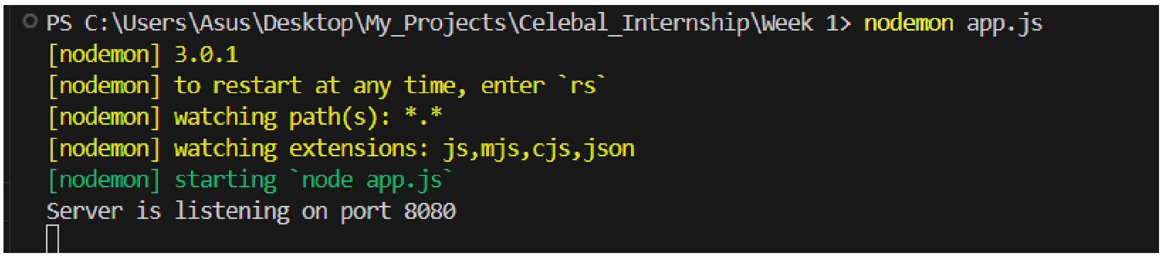
- The tools and technologies discussed provide a comprehensive framework for developing, securing, and maintaining robust backend systems using Node.js.
- Understanding the principles of asynchronous programming, efficient database management with MongoDB, and secure authentication methods is crucial for building scalable web applications.
- The integration of Express.js and Mongoose streamlines the development process, enabling developers to focus on business logic and application features.
- This chapter covers the essential concepts and practices that form the backbone of backend development, ensuring the delivery of high-quality and reliable web services.

CHAPTER 3 TASK DESCRIPTION

Task 3.1: Set up Node.js and create your first application

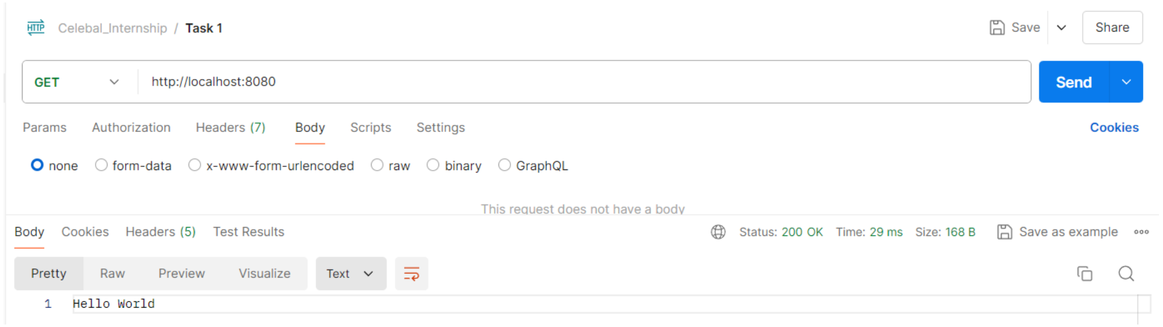
Task Description: Install Node.js and build a simple "Hello World" application. Understand the basics of Node.js runtime and execute your application.

Output:



```
PS C:\Users\Asus\Desktop\My_Projects\Celebal_Internship\Week 1> nodemon app.js
[nodemon] 3.0.1
[nodemon] to restart at any time, enter `rs`
[nodemon] watching path(s): *.*
[nodemon] watching extensions: js,mjs,cjs,json
[nodemon] starting `node app.js`
Server is listening on port 8080
```

Figure 3.1.1) Task 1 console output



Celebal_Internship / Task 1

GET http://localhost:8080

Body: [✓ none] form-data x-www-form-urlencoded raw binary GraphQL

This request does not have a body

Status: 200 OK Time: 29 ms Size: 168 B

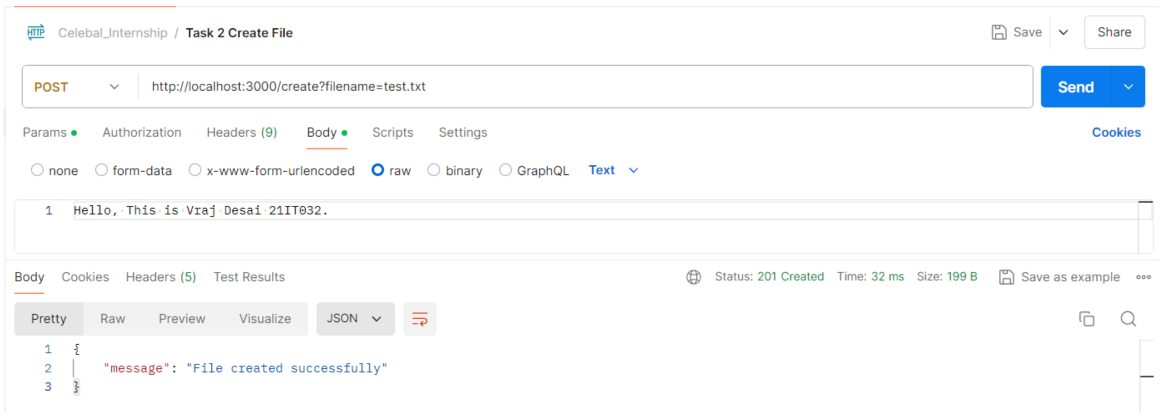
```
Hello World
```

Figure 3.1.2) Task 1 localhost response output

Task 3.2: Build a file management tool using core modules

Task Description: Utilize Node.js core modules such as File System, Path, and HTTP to create a simple file management tool that can create, read, and delete files.

Output:



Celebal_Internship / Task 2 Create File

POST http://localhost:3000/create?filename=test.txt

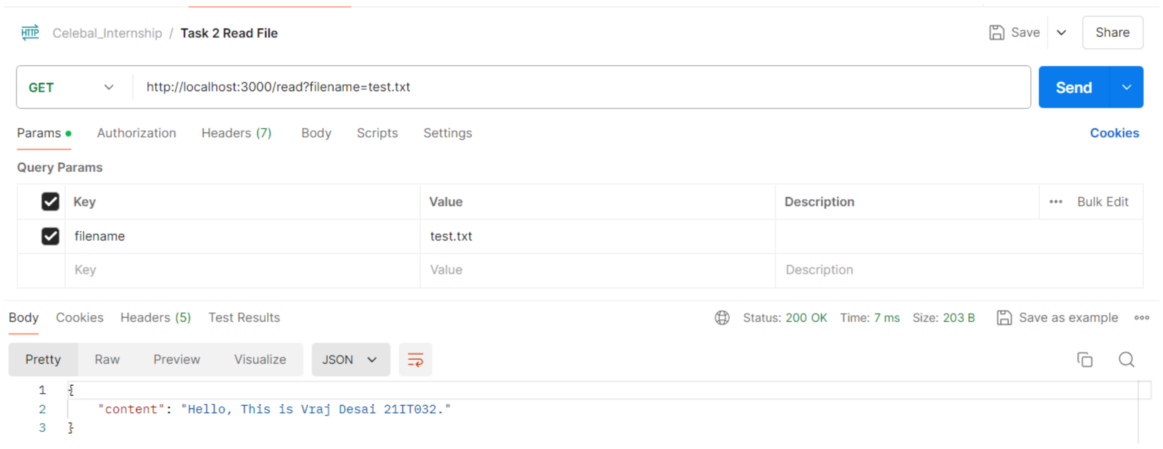
Body: none form-data x-www-form-urlencoded [✓ raw] binary GraphQL Text

```
Hello, This is Vraj Desai 21IT032.
```

Status: 201 Created Time: 32 ms Size: 199 B

```
{
    "message": "File created successfully"
}
```

Figure 3.2.1) Create a File



Celebal_Internship / Task 2 Read File

GET http://localhost:3000/read?filename=test.txt

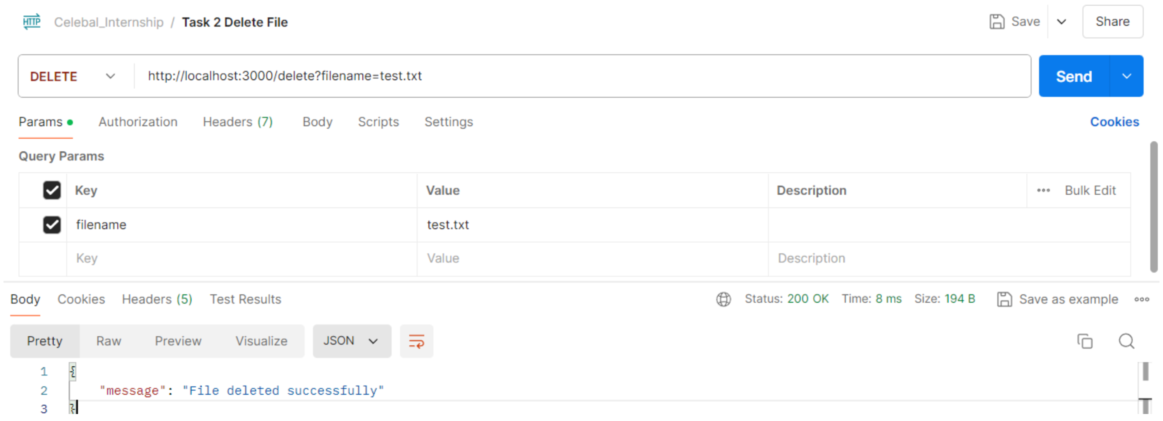
Query Params

Key	Value	Description
filename	test.txt	

Status: 200 OK Time: 7 ms Size: 203 B

```
{
    "content": "Hello, This is Vraj Desai 21IT032."
}
```

Figure 3.2.2) Read a File



Celebal_Internship / Task 2 Delete File

DELETE http://localhost:3000/delete?filename=test.txt

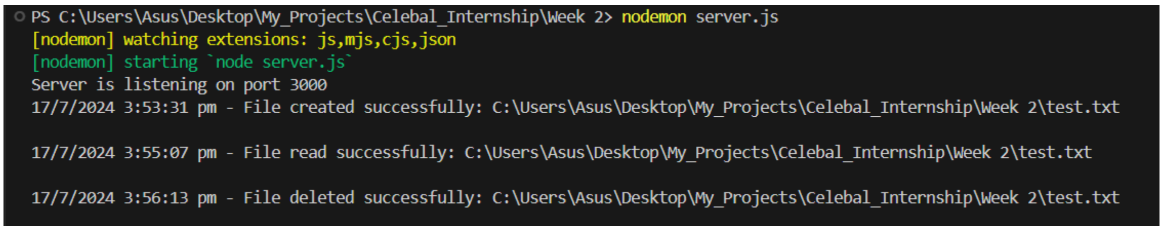
Query Params

Key	Value	Description
filename	test.txt	

Status: 200 OK Time: 8 ms Size: 194 B

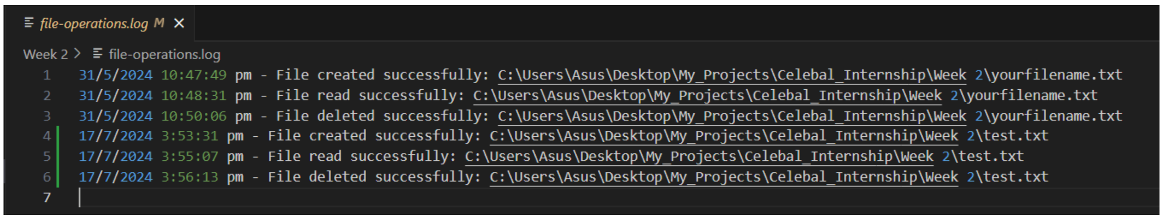
```
{
    "message": "File deleted successfully"
}
```

Figure 3.2.3) Delete a File



```
PS C:\Users\Asus\Desktop\My_Projects\Celebal_Internship\Week 2> nodemon server.js
[nodemon] watching extensions: js,mjs,cjs,json
[nodemon] starting `node server.js`
Server is listening on port 3000
17/7/2024 3:53:31 pm - File created successfully: C:\Users\Asus\Desktop\My_Projects\Celebal_Internship\Week 2\test.txt

17/7/2024 3:55:07 pm - File read successfully: C:\Users\Asus\Desktop\My_Projects\Celebal_Internship\Week 2\test.txt

17/7/2024 3:56:13 pm - File deleted successfully: C:\Users\Asus\Desktop\My_Projects\Celebal_Internship\Week 2\test.txt
```

Figure 3.2.4) Console output after all of the operations



file-operations.log

Week 2 > file-operations.log

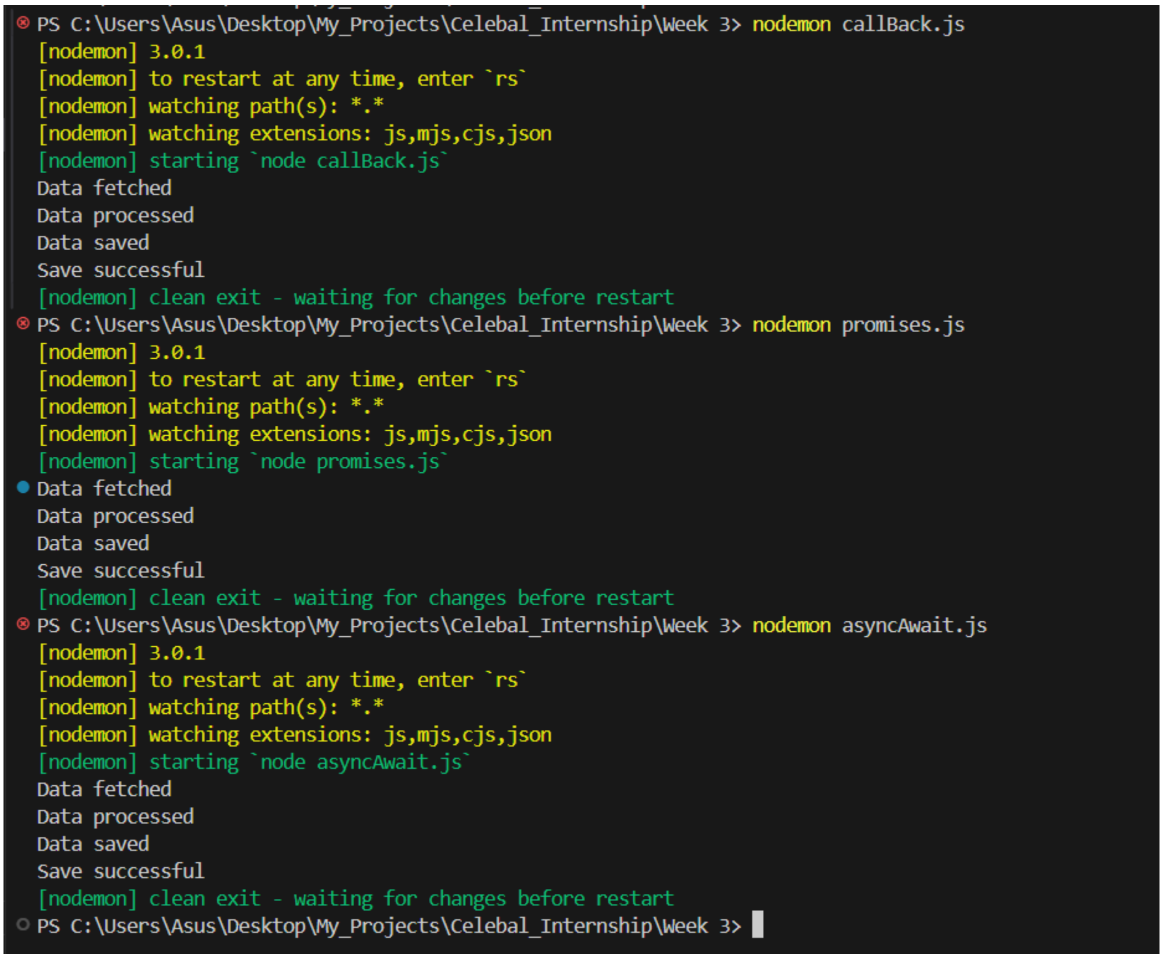
```
31/5/2024 10:47:49 pm - File created successfully: C:\Users\Asus\Desktop\My_Projects\Celebal_Internship\Week 2\yourfilename.txt
31/5/2024 10:48:31 pm - File read successfully: C:\Users\Asus\Desktop\My_Projects\Celebal_Internship\Week 2\yourfilename.txt
31/5/2024 10:50:06 pm - File deleted successfully: C:\Users\Asus\Desktop\My_Projects\Celebal_Internship\Week 2\yourfilename.txt
17/7/2024 3:53:31 pm - File created successfully: C:\Users\Asus\Desktop\My_Projects\Celebal_Internship\Week 2\test.txt
17/7/2024 3:55:07 pm - File read successfully: C:\Users\Asus\Desktop\My_Projects\Celebal_Internship\Week 2\test.txt
17/7/2024 3:56:13 pm - File deleted successfully: C:\Users\Asus\Desktop\My_Projects\Celebal_Internship\Week 2\test.txt
```

Figure 3.2.5) Creating a log file to save all of the operations

Task 3.3: Convert callback-based code to Promises and Async/Await

Task Description: Refactor an existing piece of code that uses callbacks for async operations to use Promises and Async/Await for better readability and error handling.

Output:



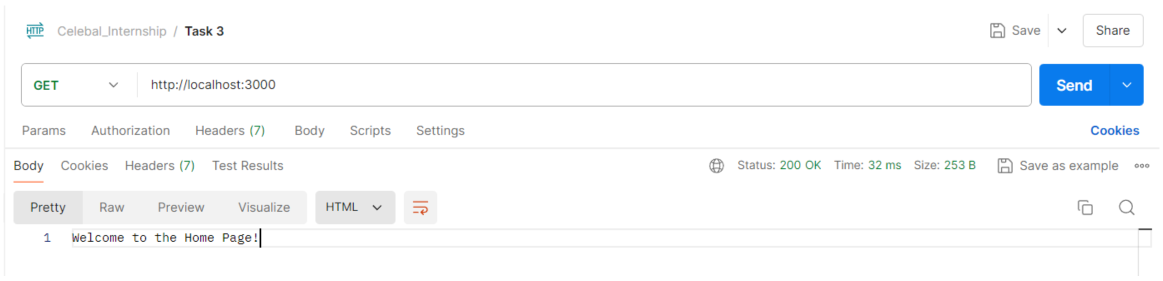
```
PS C:\Users\Asus\Desktop\My_Projects\Celebal_Internship\Week 3> nodemon callBack.js
[nodemon] 3.0.1
[nodemon] to restart at any time, enter `rs`
[nodemon] watching path(s): *.*
[nodemon] watching extensions: js,mjs,cjs,json
[nodemon] starting `node callBack.js`
Data fetched
Data processed
Data saved
Save successful
[nodemon] clean exit - waiting for changes before restart
PS C:\Users\Asus\Desktop\My_Projects\Celebal_Internship\Week 3> nodemon promises.js
[nodemon] 3.0.1
[nodemon] to restart at any time, enter `rs`
[nodemon] watching path(s): *.*
[nodemon] watching extensions: js,mjs,cjs,json
[nodemon] starting `node promises.js`
Data fetched
Data processed
Data saved
Save successful
[nodemon] clean exit - waiting for changes before restart
PS C:\Users\Asus\Desktop\My_Projects\Celebal_Internship\Week 3> nodemon asyncAwait.js
[nodemon] 3.0.1
[nodemon] to restart at any time, enter `rs`
[nodemon] watching path(s): *.*
[nodemon] watching extensions: js,mjs,cjs,json
[nodemon] starting `node asyncAwait.js`
Data fetched
Data processed
Data saved
Save successful
[nodemon] clean exit - waiting for changes before restart
PS C:\Users\Asus\Desktop\My_Projects\Celebal_Internship\Week 3>
```

Figure 3.3.1) Implementation and output of Callback-based code, Promises-based code and Async-Await-based code respectively

Task 3.4: Create a basic web server with Express.js

Task Description: Set up a simple web server using Express.js that can handle basic routing and middleware. Implement routes to respond to at least two different endpoints.

Output:



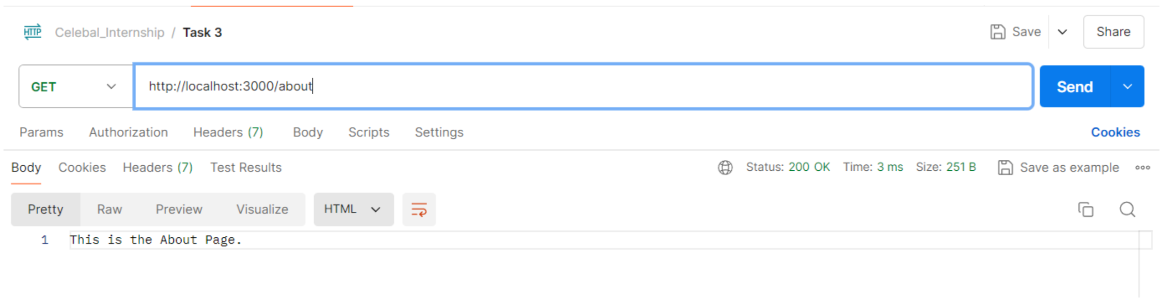
Celebal_Internship / Task 3

GET http://localhost:3000

Status: 200 OK Time: 32 ms Size: 253 B

```
Welcome to the Home Page!
```

Figure 3.4.1) Home Page

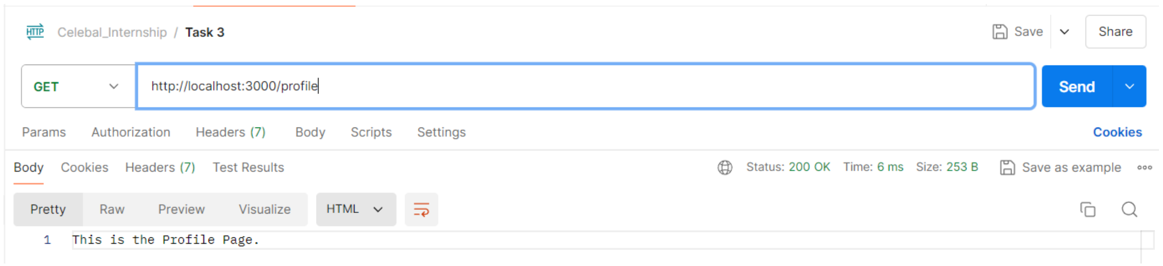


http://localhost:3000/about

Status: 200 OK Time: 3 ms Size: 251 B

```
This is the About Page.
```

Figure 3.4.2) About Page

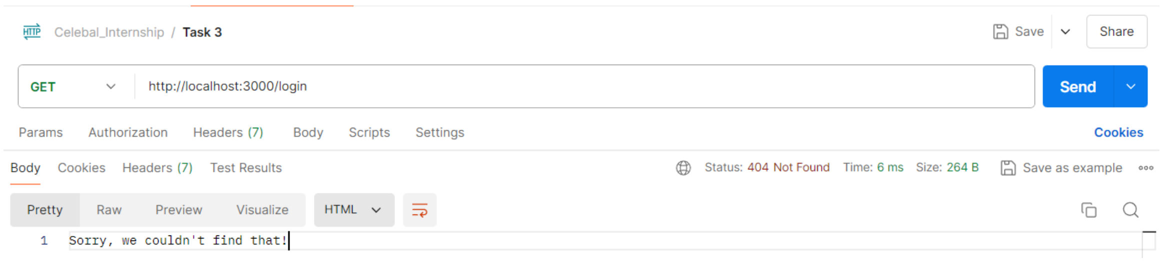


http://localhost:3000/profile

Status: 200 OK Time: 6 ms Size: 253 B

```
This is the Profile Page.
```

Figure 3.4.3) Profile Page

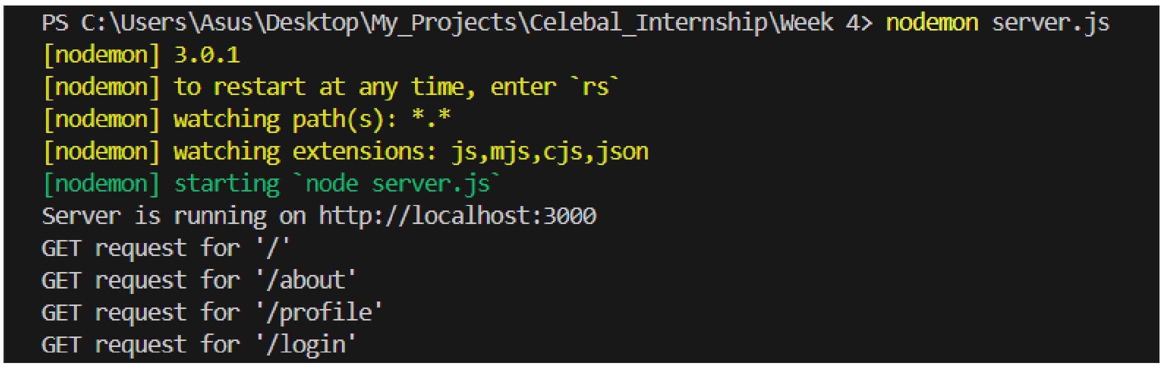


http://localhost:3000/login

Status: 404 Not Found Time: 6 ms Size: 264 B

```
Sorry, we couldn't find that!
```

Figure 3.4.4) Middleware to handle Error page



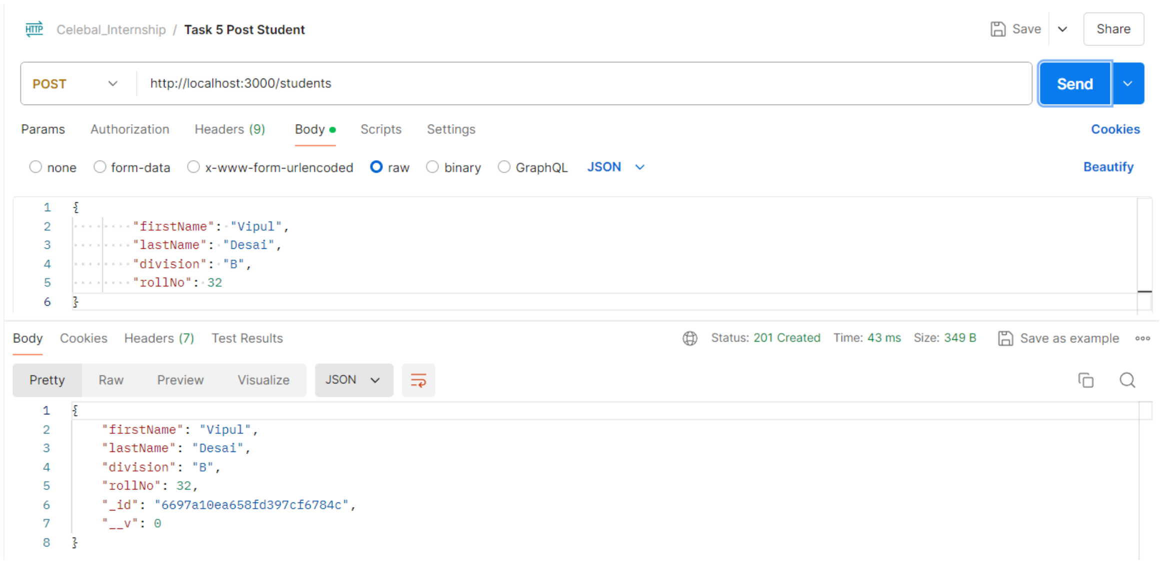
```
PS C:\Users\Asus\Desktop\My_Projects\Celebal_Internship\Week 4> nodemon server.js
[nodemon] 3.0.1
[nodemon] to restart at any time, enter `rs`
[nodemon] watching path(s): *.*
[nodemon] watching extensions: js,mjs,cjs,json
[nodemon] starting `node server.js`
Server is running on http://localhost:3000
GET request for '/'
GET request for '/about'
GET request for '/profile'
GET request for '/login'
```

Figure 3.4.5) Console output after each request

Task 3.5: Build a CRUD application with MongoDB

Task Description: Develop a simple application to Create, Read, Update, and Delete (CRUD) entries in a MongoDB database using Mongoose.

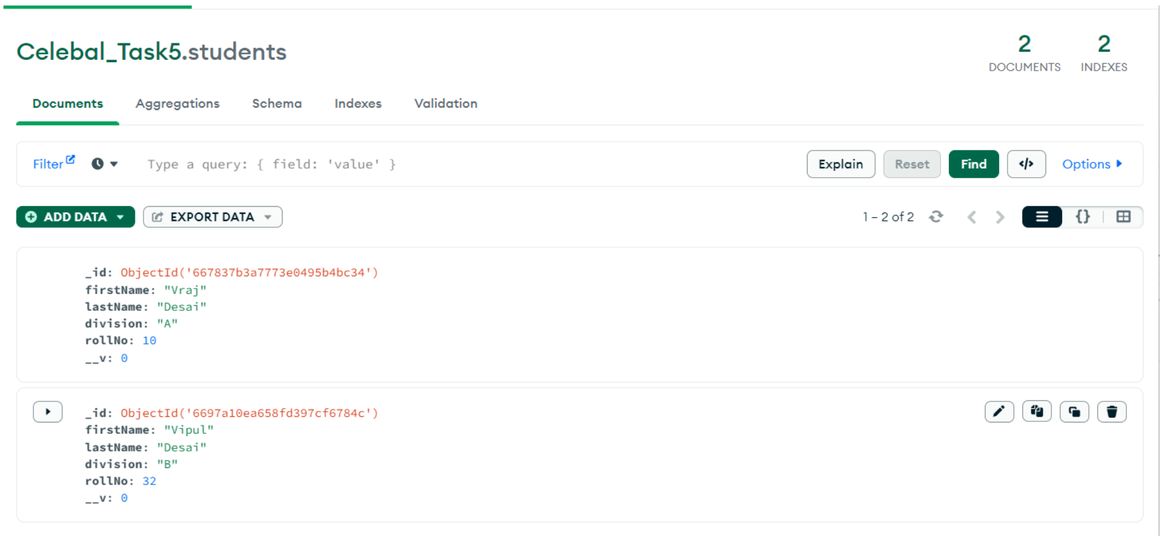
Output:



```
{
    "firstName": "Vipul",
    "lastName": "Desai",
    "division": "B",
    "rollNo": 32
}
```

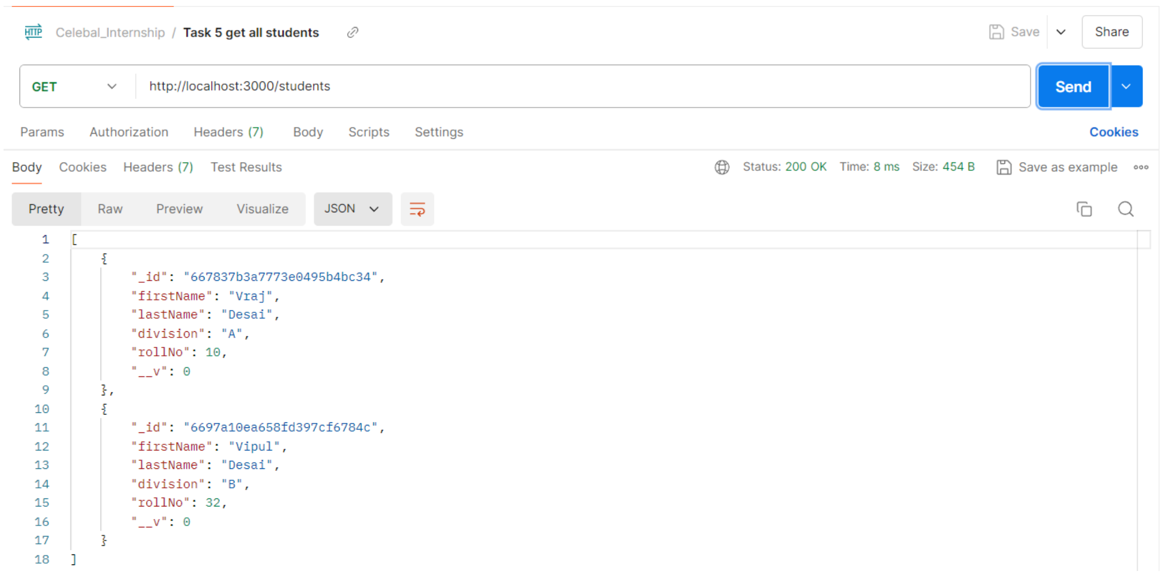
```
{
    "firstName": "Vipul",
    "lastName": "Desai",
    "division": "B",
    "rollNo": 32,
    "_id": "6697a10ea658fd397cf6784c",
    "__v": 0
}
```

Figure 3.5.1) Entering the student data



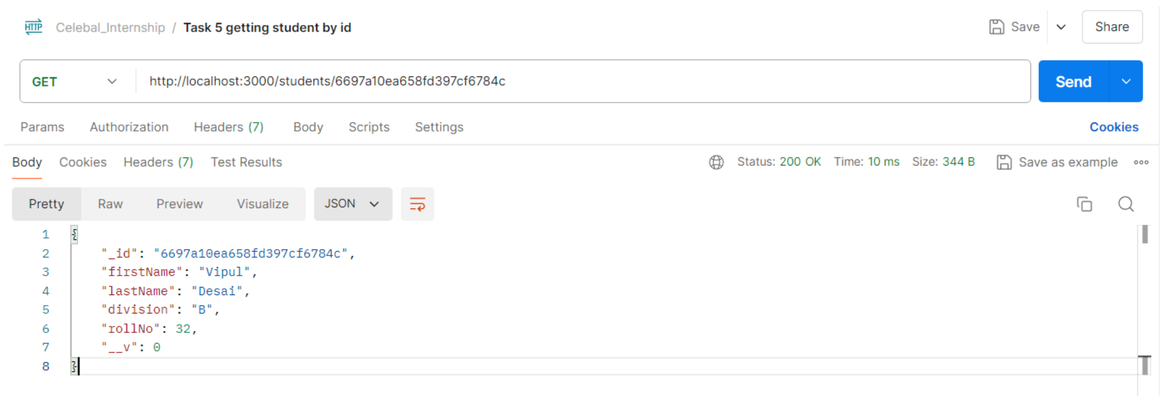
```
_id: ObjectId('667837b3a7773e0495b4bc34')
firstName: "Vraj"
lastName: "Desai"
division: "A"
rollNo: 10
__v: 0

_id: ObjectId('6697a10ea658fd397cf6784c')
firstName: "Vipul"
lastName: "Desai"
division: "B"
rollNo: 32
__v: 0
```

Figure 3.5.2) Entry of new data in the database



```
[
    {
        "_id": "667837b3a7773e0495b4bc34",
        "firstName": "Vraj",
        "lastName": "Desai",
        "division": "A",
        "rollNo": 10,
        "__v": 0
    },
    {
        "_id": "6697a10ea658fd397cf6784c",
        "firstName": "Vipul",
        "lastName": "Desai",
        "division": "B",
        "rollNo": 32,
        "__v": 0
    }
]
```

Figure 3.5.3) Getting all the students data



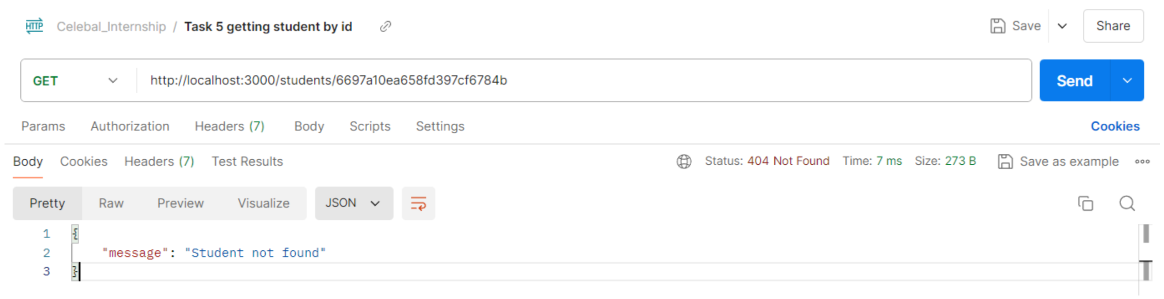
Celebal_Internship / Task 5 getting student by id

GET http://localhost:3000/students/6697a10ea658fd397cf6784c

Status: 200 OK Time: 10 ms Size: 344 B

```
{
    "_id": "6697a10ea658fd397cf6784c",
    "firstName": "Vipul",
    "lastName": "Desai",
    "division": "B",
    "rollNo": 32,
    "__v": 0
}
```

Figure 3.5.4) Getting Student data by their id



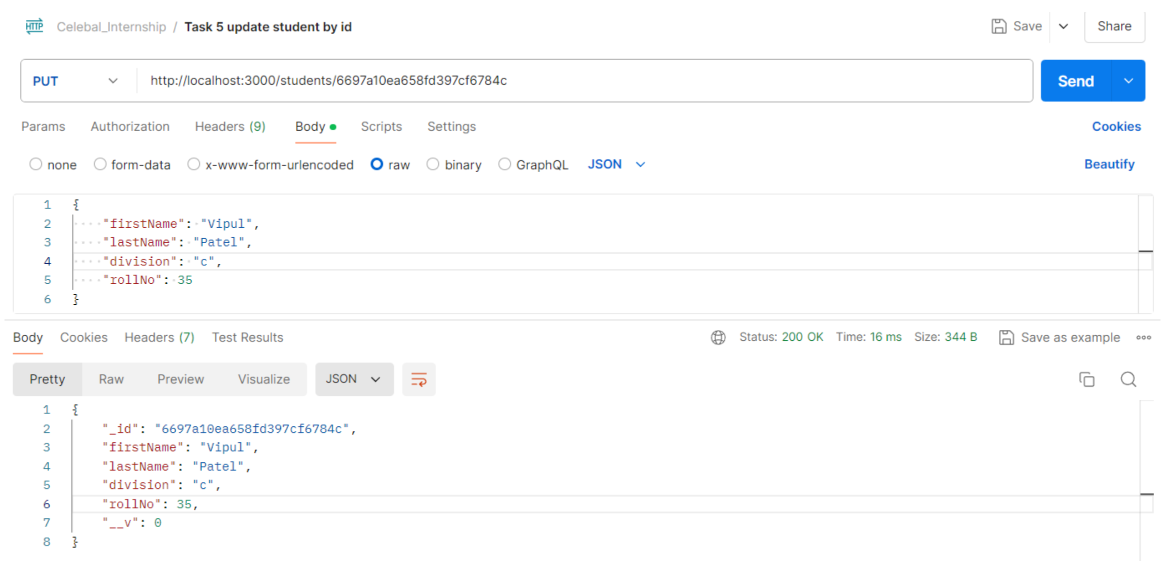
Celebal_Internship / Task 5 getting student by id

GET http://localhost:3000/students/6697a10ea658fd397cf6784b

Status: 404 Not Found Time: 7 ms Size: 273 B

```
{
    "message": "Student not found"
}
```

Figure 3.5.5) Error Page if id is not matched with that of in the database



Celebal_Internship / Task 5 update student by id

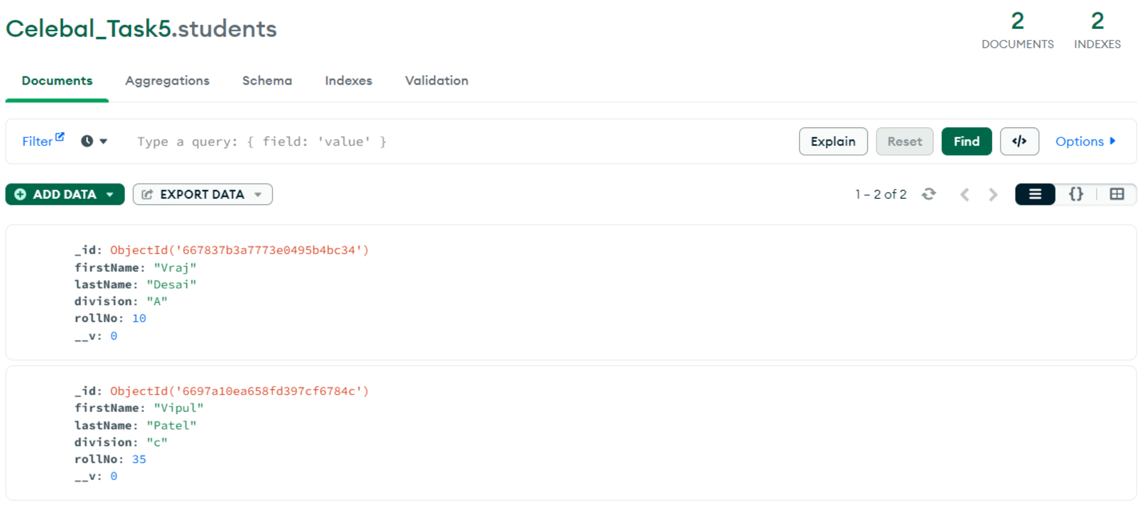
PUT http://localhost:3000/students/6697a10ea658fd397cf6784c

```
{
    "firstName": "Vipul",
    "lastName": "Patel",
    "division": "c",
    "rollNo": 35
}
```

Status: 200 OK Time: 16 ms Size: 344 B

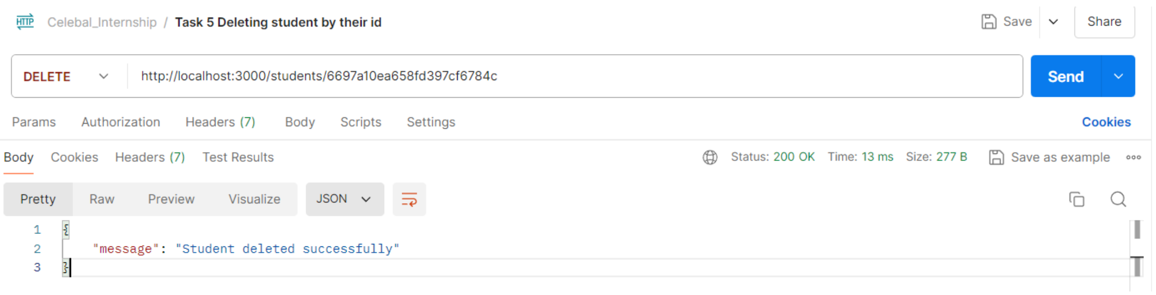
```
{
    "_id": "6697a10ea658fd397cf6784c",
    "firstName": "Vipul",
    "lastName": "Patel",
    "division": "c",
    "rollNo": 35,
    "__v": 0
}
```

Figure 3.5.6) Update student data by their id



```
_id: ObjectId('667837b3a7773e0495b4bc34')
firstName: "Vraj"
lastName: "Desai"
division: "A"
rollNo: 10
__v: 0

_id: ObjectId('6697a10ea658fd397cf6784c')
firstName: "Vipul"
lastName: "Patel"
division: "c"
rollNo: 35
__v: 0
```

Figure 3.5.7) Updated data in the database

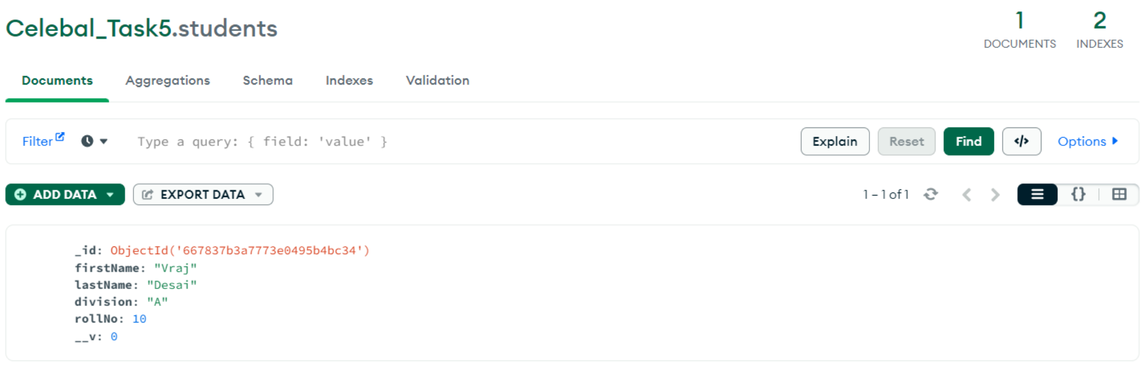


DELETE http://localhost:3000/students/6697a10ea658fd397cf6784c

Status: 200 OK Time: 13 ms Size: 277 B

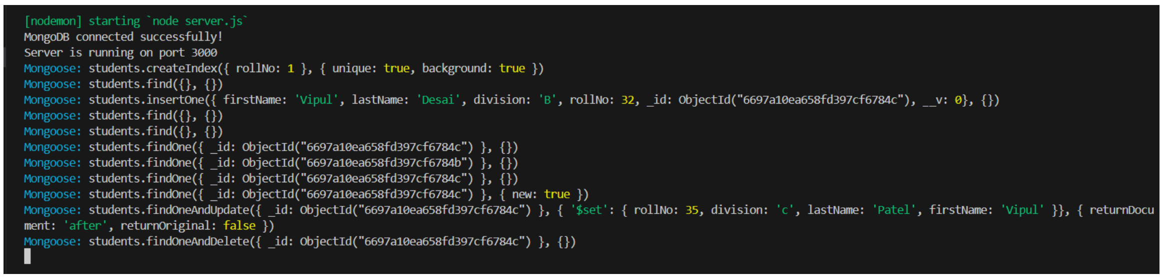
```
{
    "message": "Student deleted successfully"
}
```

Figure 3.5.8) Deleting Student by their id



```
_id: ObjectId('667837b3a7773e0495b4bc34')
firstName: "Vraj"
lastName: "Desai"
division: "A"
rollNo: 10
__v: 0
```

Figure 3.5.9) Student deleted from the database



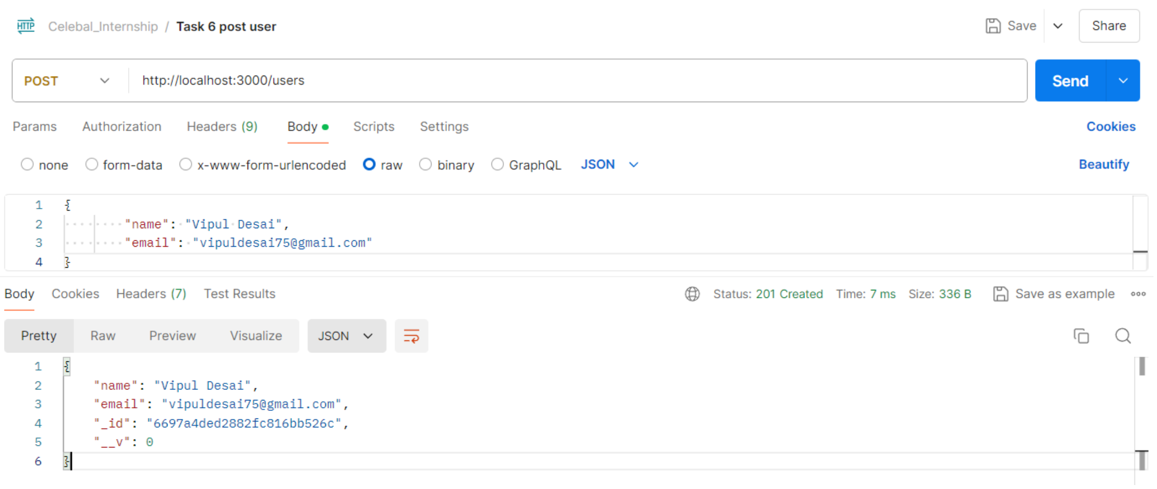
```
[nodemon] starting `node server.js`
MongoDB connected successfully!
Server is running on port 3000
Mongoose: students.createIndex({ rollNo: 1 }, { unique: true, background: true })
Mongoose: students.find({}, {})
Mongoose: students.insertOne({ firstName: 'Vipul', lastName: 'Desai', division: 'B', rollNo: 32, _id: ObjectId("6697a10ea658fd397cf6784c"), __v: 0}, {})
Mongoose: students.find({}, {})
Mongoose: students.find({}, {})
Mongoose: students.findOne({ _id: ObjectId("6697a10ea658fd397cf6784c") }, {})
Mongoose: students.findOne({ _id: ObjectId("6697a10ea658fd397cf6784b") }, {})
Mongoose: students.findOne({ _id: ObjectId("6697a10ea658fd397cf6784c") }, {})
Mongoose: students.findOne({ _id: ObjectId("6697a10ea658fd397cf6784c") }, { new: true })
Mongoose: students.findOneAndUpdate({ _id: ObjectId("6697a10ea658fd397cf6784c") }, { '$set': { rollNo: 35, division: 'c', lastName: 'Patel', firstName: 'Vipul' }}, { returnDocu
ment: 'after', returnOriginal: false })
Mongoose: students.findOneAndDelete({ _id: ObjectId("6697a10ea658fd397cf6784c") }, {})
```

Figure 3.5.10) Console out after all of the operations

Task 3.6: Develop a RESTful API

Task Description: Create a RESTful API using Node.js and Express that supports basic CRUD operations on a resource (like users, products, etc.)

Output:



POST http://localhost:3000/users

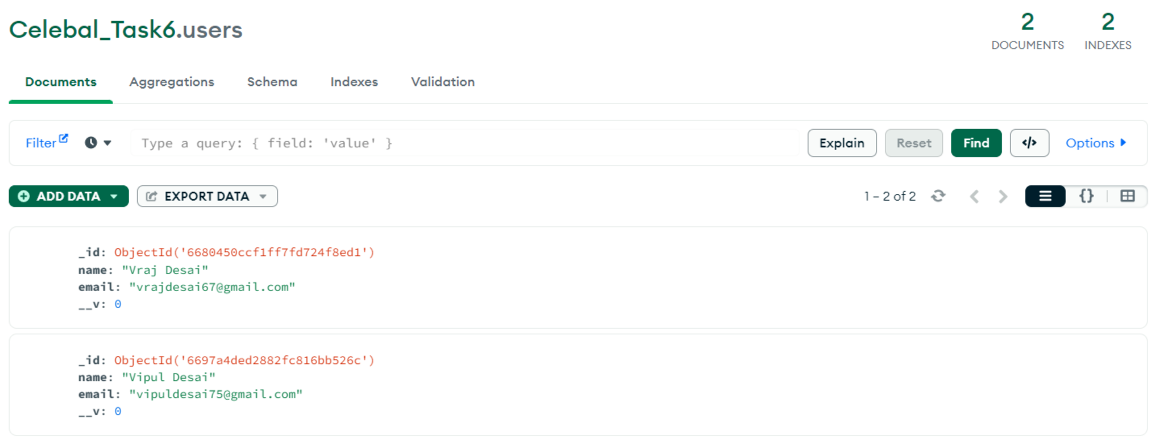
none form-data x-www-form-urlencoded [✓ raw] binary GraphQL JSON

```
{
    "name": "Vipul Desai",
    "email": "vipuldesai75@gmail.com"
}
```

Body Cookies Headers (7) Test Results — Status: 201 Created Time: 7 ms Size: 336 B

```
{
    "name": "Vipul Desai",
    "email": "vipuldesai75@gmail.com",
    "_id": "6697a4ded2882fc816bb526c",
    "__v": 0
}
```

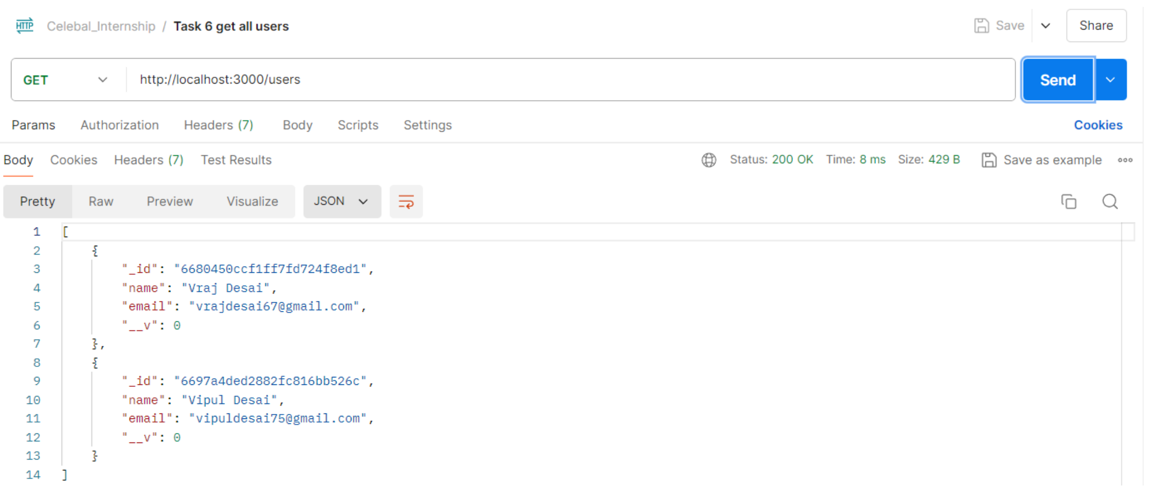
Figure 3.6.1) Adding a new user



Celebal_Task6.users — 2 DOCUMENTS, 2 INDEXES

```
_id: ObjectId('6680450ccf1ff7fd724f8ed1')
name: "Vraj Desai"
email: "vrajdesai67@gmail.com"
__v: 0

_id: ObjectId('6697a4ded2882fc816bb526c')
name: "Vipul Desai"
email: "vipuldesai75@gmail.com"
__v: 0
```

Figure 3.6.2) user added in the database

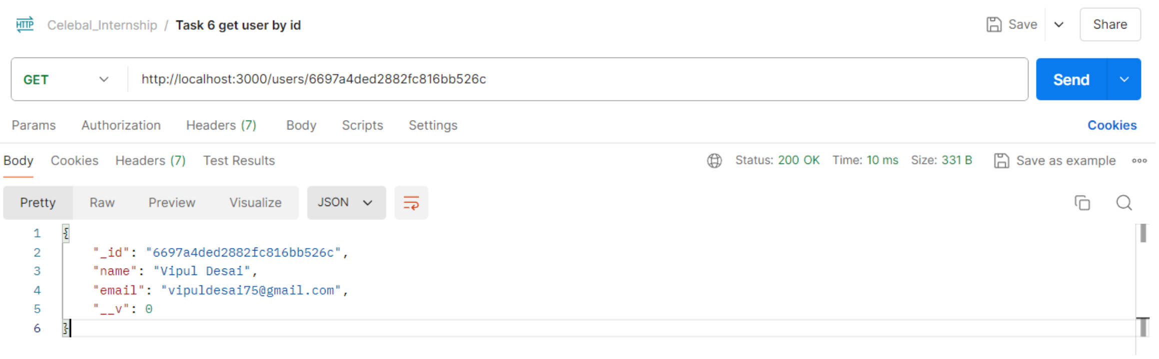


GET http://localhost:3000/users

Body Cookies Headers (7) Test Results — Status: 200 OK Time: 8 ms Size: 429 B

```
[
    {
        "_id": "6680450ccf1ff7fd724f8ed1",
        "name": "Vraj Desai",
        "email": "vrajdesai67@gmail.com",
        "__v": 0
    },
    {
        "_id": "6697a4ded2882fc816bb526c",
        "name": "Vipul Desai",
        "email": "vipuldesai75@gmail.com",
        "__v": 0
    }
]
```

Figure 3.6.3) Getting all of the users

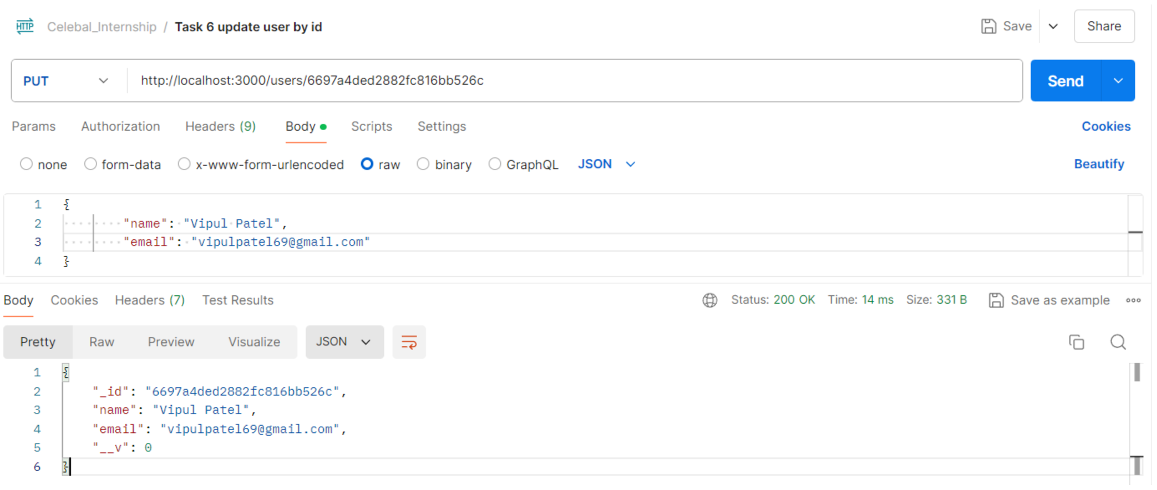


GET http://localhost:3000/users/6697a4ded2882fc816bb526c

Status: 200 OK Time: 10 ms Size: 331 B

```
{
    "_id": "6697a4ded2882fc816bb526c",
    "name": "Vipul Desai",
    "email": "vipuldesai75@gmail.com",
    "__v": 0
}
```

Figure 3.6.4) Getting user by id



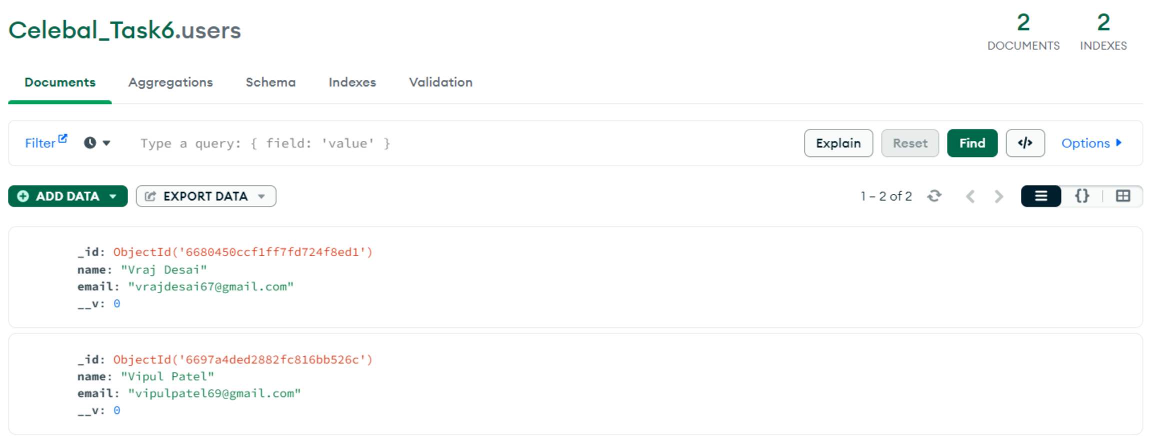
PUT http://localhost:3000/users/6697a4ded2882fc816bb526c

```
{
    "name": "Vipul Patel",
    "email": "vipulpatel69@gmail.com"
}
```

Status: 200 OK Time: 14 ms Size: 331 B

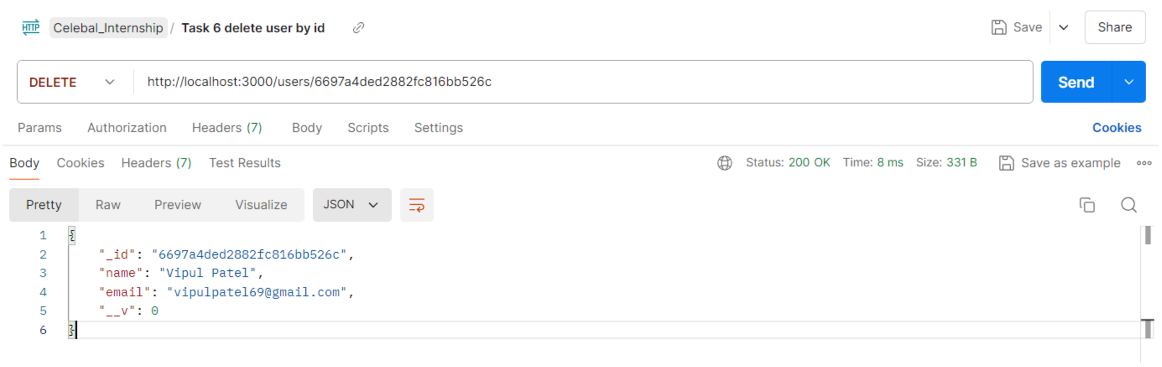
```
{
    "_id": "6697a4ded2882fc816bb526c",
    "name": "Vipul Patel",
    "email": "vipulpatel69@gmail.com",
    "__v": 0
}
```

Figure 3.6.5) Update user by id



```
_id: ObjectId('6680450ccf1ff7fd724f8ed1')
name: "Vraj Desai"
email: "vrajdesai67@gmail.com"
__v: 0

_id: ObjectId('6697a4ded2882fc816bb526c')
name: "Vipul Patel"
email: "vipulpatel69@gmail.com"
__v: 0
```

Figure 3.6.6) User updated in the database

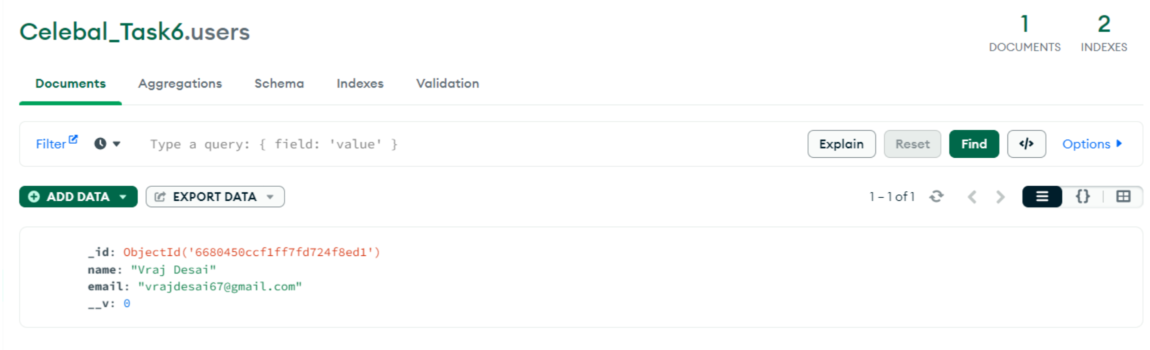


DELETE http://localhost:3000/users/6697a4ded2882fc816bb526c

Status: 200 OK Time: 8 ms Size: 331 B

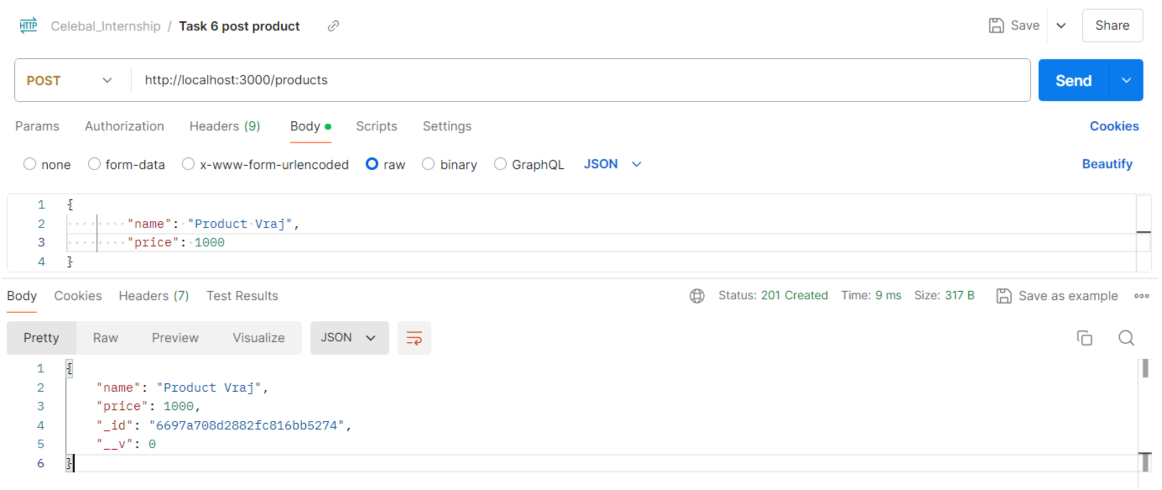
```
{
    "_id": "6697a4ded2882fc816bb526c",
    "name": "Vipul Patel",
    "email": "vipulpatel69@gmail.com",
    "__v": 0
}
```

Figure 3.6.7) Deleting user by their id



```
_id: ObjectId('6680450ccf1ff7fd724f8ed1')
name: "Vraj Desai"
email: "vrajdesai67@gmail.com"
__v: 0
```

Figure 3.6.8) User deleted from the database



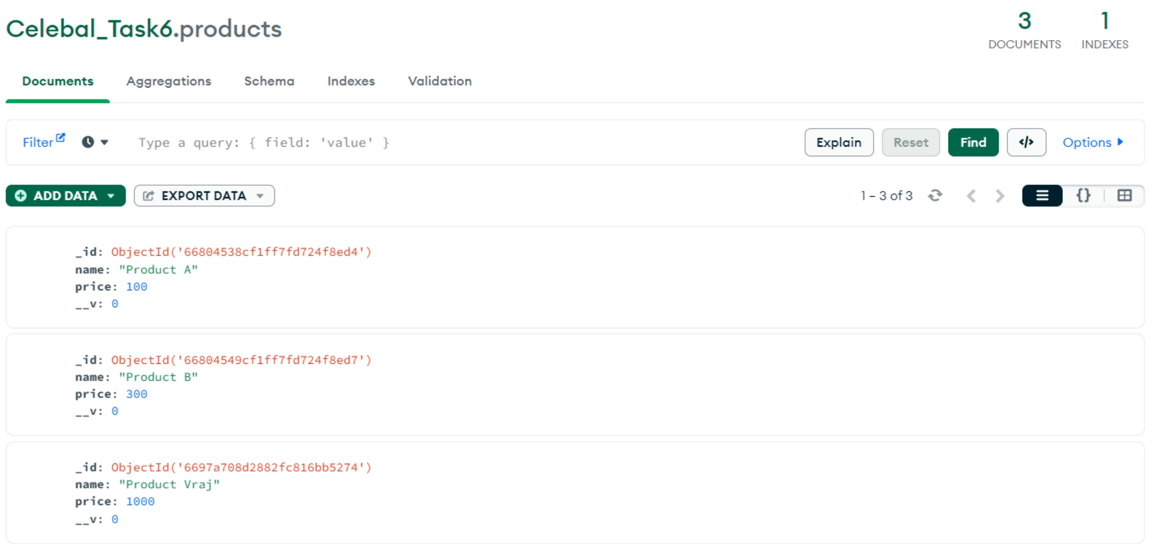
POST http://localhost:3000/products

```
{
    "name": "Product Vraj",
    "price": 1000
}
```

Status: 201 Created Time: 9 ms Size: 317 B

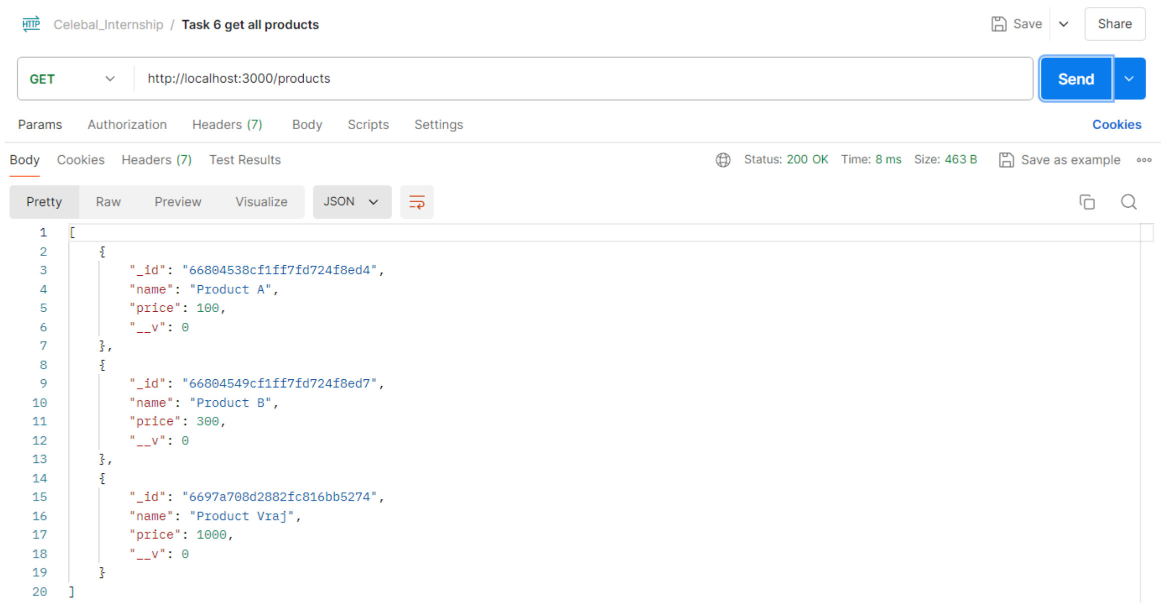
```
{
    "name": "Product Vraj",
    "price": 1000,
    "_id": "6697a708d2882fc816bb5274",
    "__v": 0
}
```

Figure 3.6.9) Adding a new product



```
_id: ObjectId('66804538cf1ff7fd724f8ed4')
name: "Product A"
price: 100
__v: 0

_id: ObjectId('66804549cf1ff7fd724f8ed7')
name: "Product B"
price: 300
__v: 0

_id: ObjectId('6697a708d2882fc816bb5274')
name: "Product Vraj"
price: 1000
__v: 0
```

Figure 3.6.10) Product added in the database

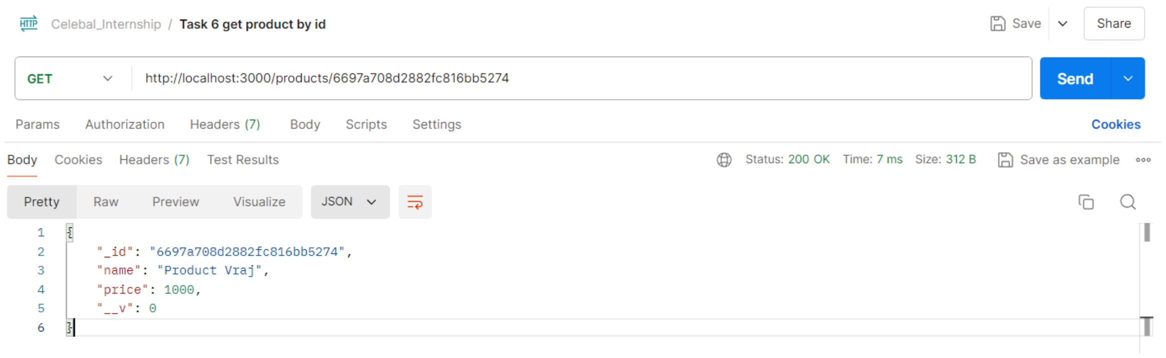


GET http://localhost:3000/products

Status: 200 OK Time: 8 ms Size: 463 B

```
[
    {
        "_id": "66804538cf1ff7fd724f8ed4",
        "name": "Product A",
        "price": 100,
        "__v": 0
    },
    {
        "_id": "66804549cf1ff7fd724f8ed7",
        "name": "Product B",
        "price": 300,
        "__v": 0
    },
    {
        "_id": "6697a708d2882fc816bb5274",
        "name": "Product Vraj",
        "price": 1000,
        "__v": 0
    }
]
```

Figure 3.6.11) Getting all of the products

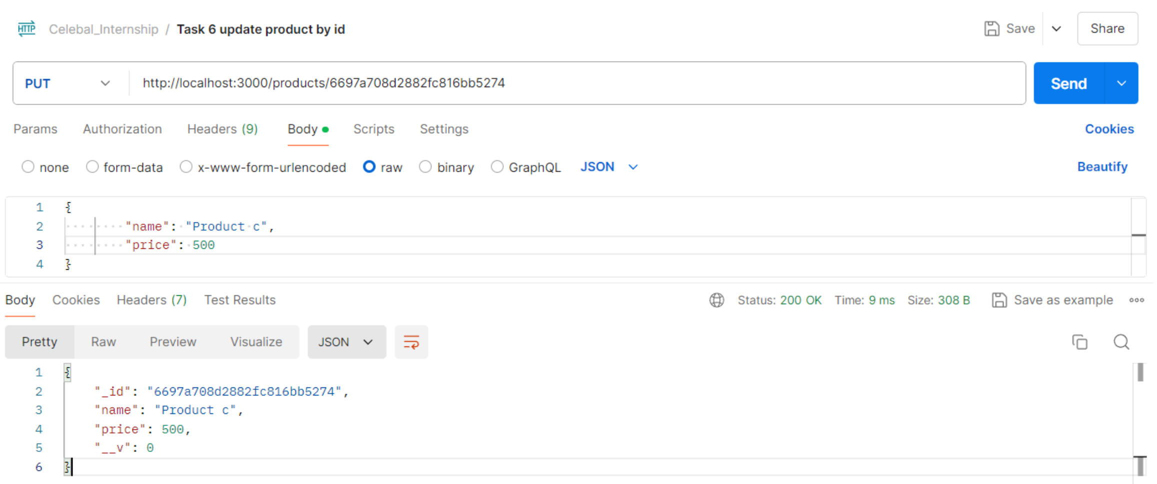


GET http://localhost:3000/products/6697a708d2882fc816bb5274

Status: 200 OK Time: 7 ms Size: 312 B

```
{
    "_id": "6697a708d2882fc816bb5274",
    "name": "Product Vraj",
    "price": 1000,
    "__v": 0
}
```

Figure 3.6.12) Getting Product by id



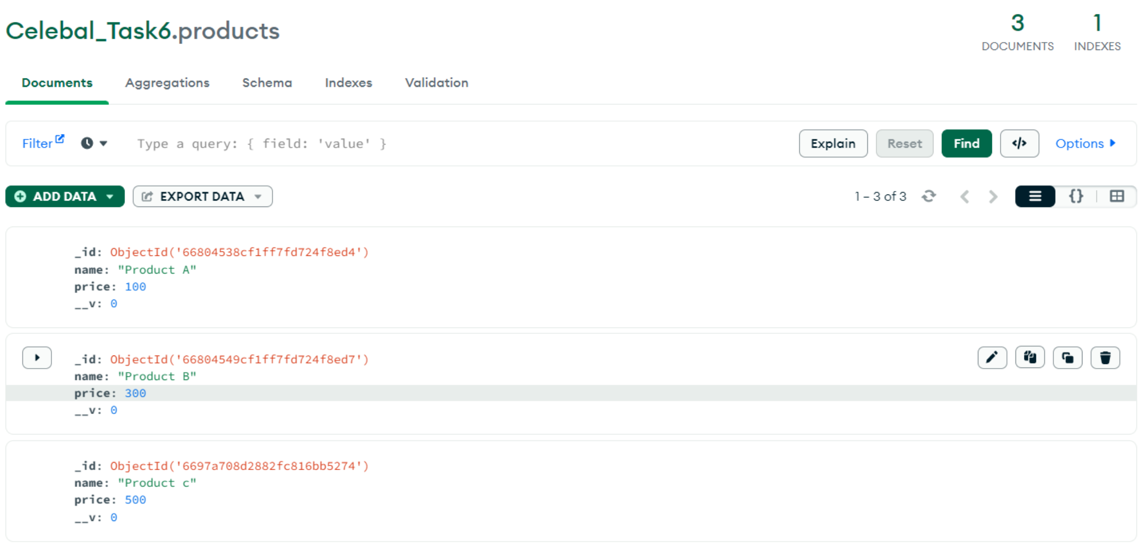
PUT http://localhost:3000/products/6697a708d2882fc816bb5274

```
{
    "name": "Product c",
    "price": 500
}
```

Status: 200 OK Time: 9 ms Size: 308 B

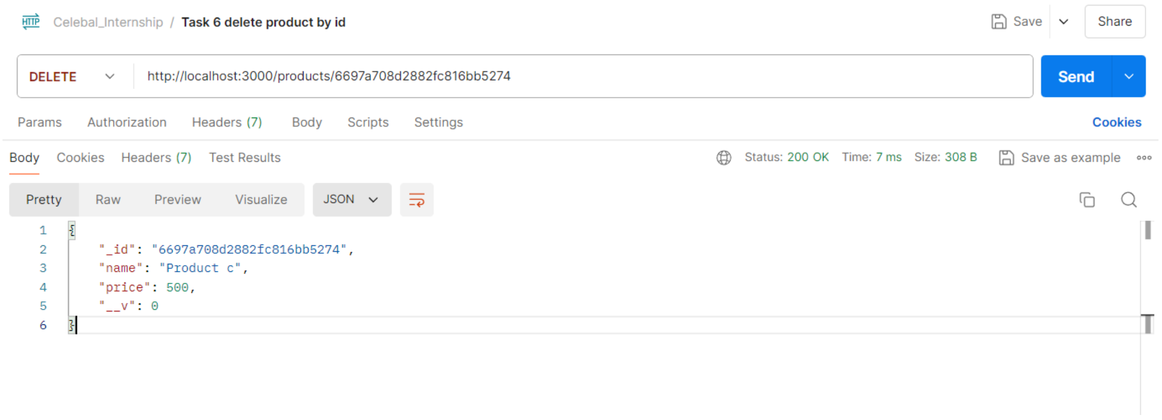
```
{
    "_id": "6697a708d2882fc816bb5274",
    "name": "Product c",
    "price": 500,
    "__v": 0
}
```

Figure 3.6.13) Updating product by id



```
_id: ObjectId('66804538cf1ff7fd724f8ed4')
name: "Product A"
price: 100
__v: 0

_id: ObjectId('66804549cf1ff7fd724f8ed7')
name: "Product B"
price: 300
__v: 0

_id: ObjectId('6697a708d2882fc816bb5274')
name: "Product c"
price: 500
__v: 0
```

Figure 3.6.14) Product updated in the database

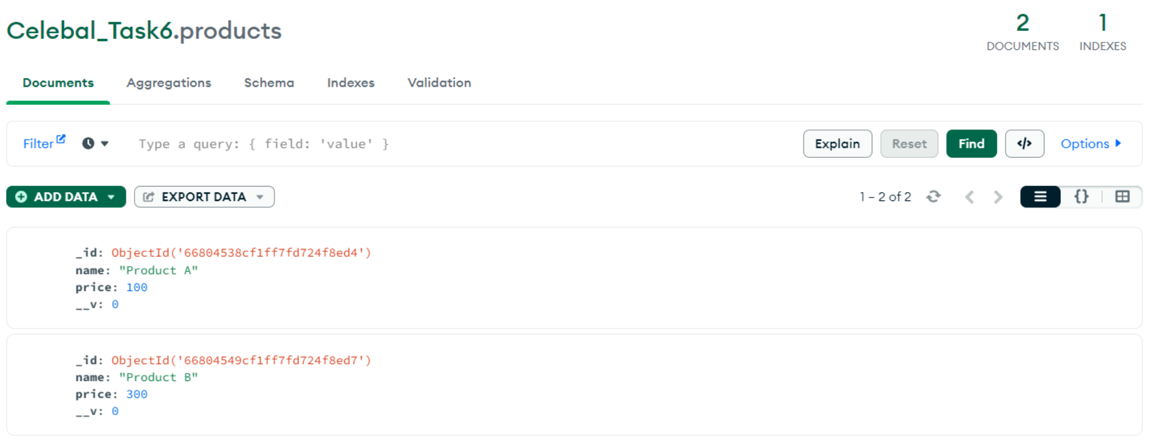


DELETE http://localhost:3000/products/6697a708d2882fc816bb5274

Status: 200 OK Time: 7 ms Size: 308 B

```
{
    "_id": "6697a708d2882fc816bb5274",
    "name": "Product c",
    "price": 500,
    "__v": 0
}
```

Figure 3.6.15) Deleting product by id



```
_id: ObjectId('66804538cf1ff7fd724f8ed4')
name: "Product A"
price: 100
__v: 0
```

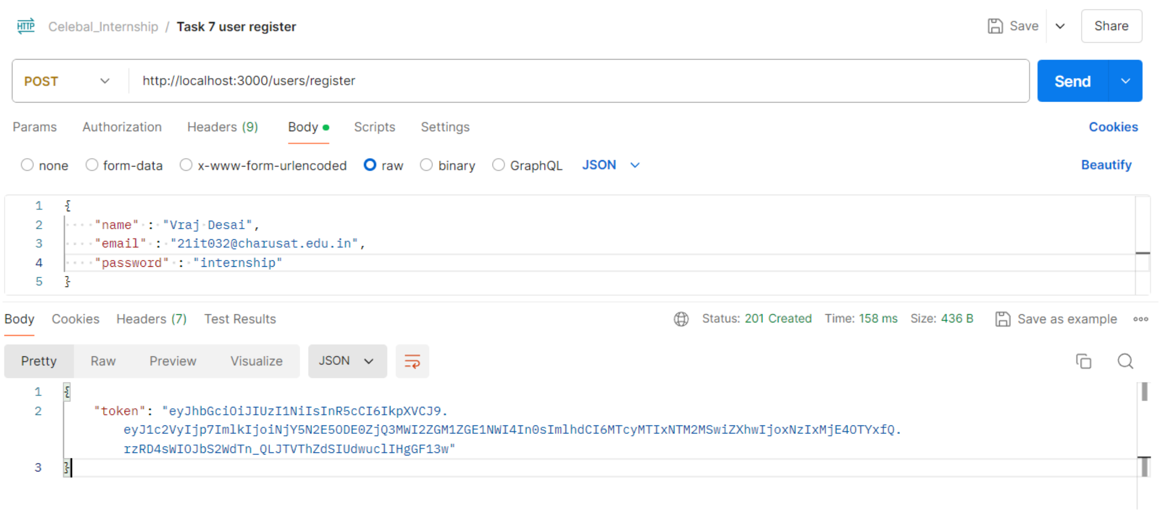
```
_id: ObjectId('66804549cf1ff7fd724f8ed7')
name: "Product B"
price: 300
__v: 0
```

Figure 3.6.16) Product deleted from the database

Task 3.7: Implement JWT-based authentication in your API

Task Description: Add JSON Web Token (JWT) authentication to your existing RESTful API. Ensure secure handling of tokens and implement a protected route.

Output:



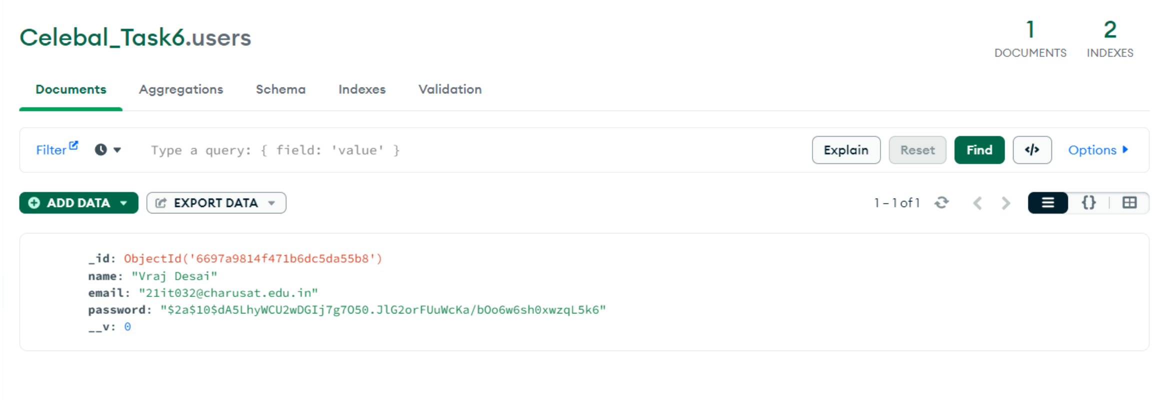
POST http://localhost:3000/users/register

```
{
    "name" : "Vraj Desai",
    "email" : "21it032@charusat.edu.in",
    "password" : "internship"
}
```

Status: 201 Created Time: 158 ms Size: 436 B

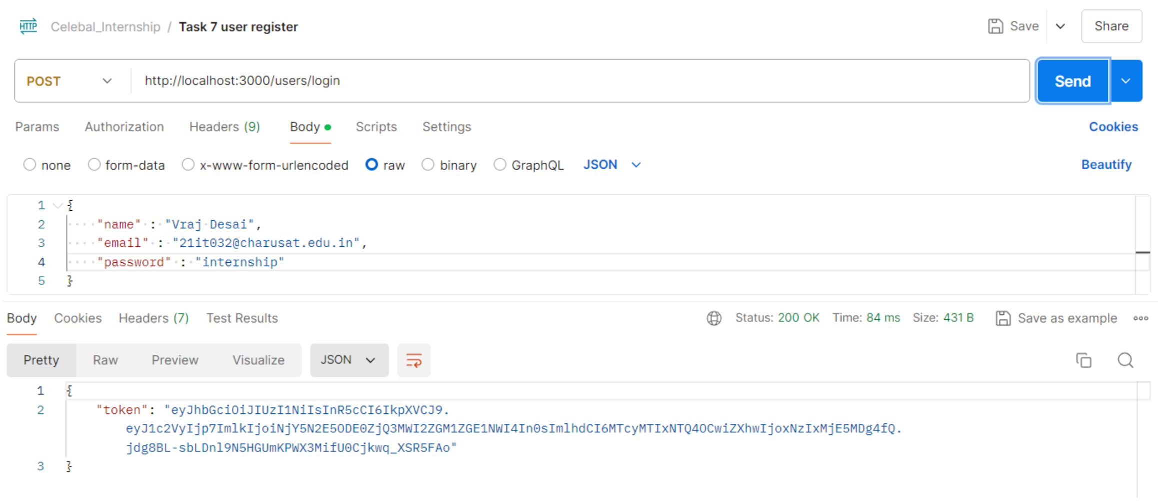
```
{
    "token": "eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.
        eyJ1c2VyIjp7ImlkIjoiNjY5N2E5ODE0ZjQ3MWI2ZGM1ZGE1NWI4In0sImlhdCI6MTcyMTIxNTM2MSwiZXhwIjoxNzIxMjE4OTYxfQ.
        rzRD4sWIOJbS2WdTn_QLJTVThZdSIUdwuclIHgGF13w"
}
```

Figure 3.7.1) a new user is successfully registered and JWT token is returned in the backend



```
_id: ObjectId('6697a9814f471b6dc5da55b8')
name: "Vraj Desai"
email: "21it032@charusat.edu.in"
password: "$2a$10$dA5LhyWCU2wDGIj7g7O50.JlG2orFUuWcKa/bOo6w6sh0xwzqL5k6"
__v: 0
```

Figure 3.7.2) Registered user in the database in which his password is encrypted due to bcrypt library



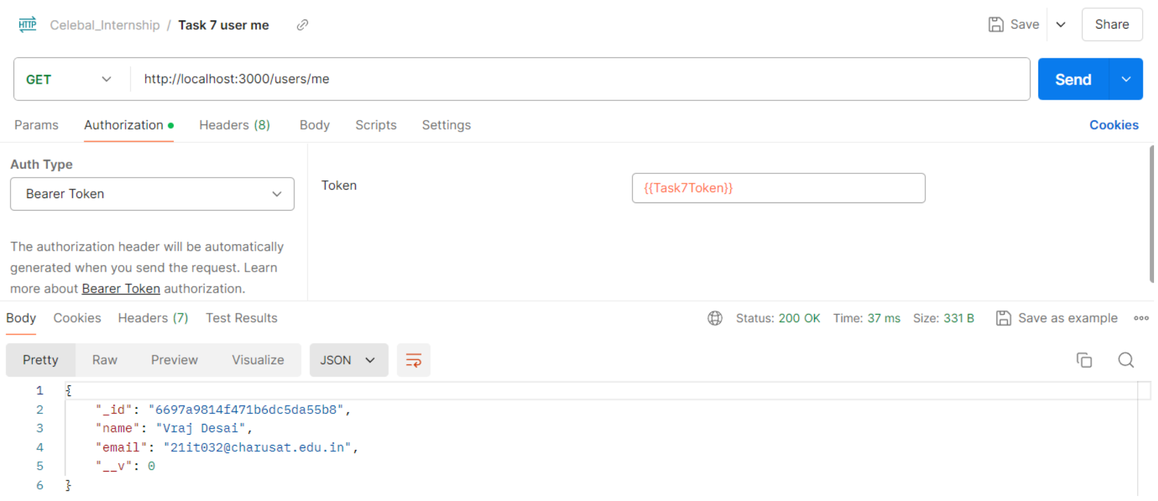
POST http://localhost:3000/users/login

```
{
    "name" : "Vraj Desai",
    "email" : "21it032@charusat.edu.in",
    "password" : "internship"
}
```

Status: 200 OK Time: 84 ms Size: 431 B

```
{
    "token": "eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.
        eyJ1c2VyIjp7ImlkIjoiNjY5N2E5ODE0ZjQ3MWI2ZGM1ZGE1NWI4In0sImlhdCI6MTcyMTIxNTQ4OCwiZXhwIjoxNzIxMjE5MDg4fQ.
        jdg8BL-sbLDnl9N5HGUmKPWX3MifU0Cjkwq_XSR5FAo"
}
```

Figure 3.7.3) User login successful and JWT token is returned



GET http://localhost:3000/users/me

Auth Type: Bearer Token

Token: {{Task7Token}}

The authorization header will be automatically generated when you send the request. Learn more about Bearer Token authorization.

Status: 200 OK Time: 37 ms Size: 331 B

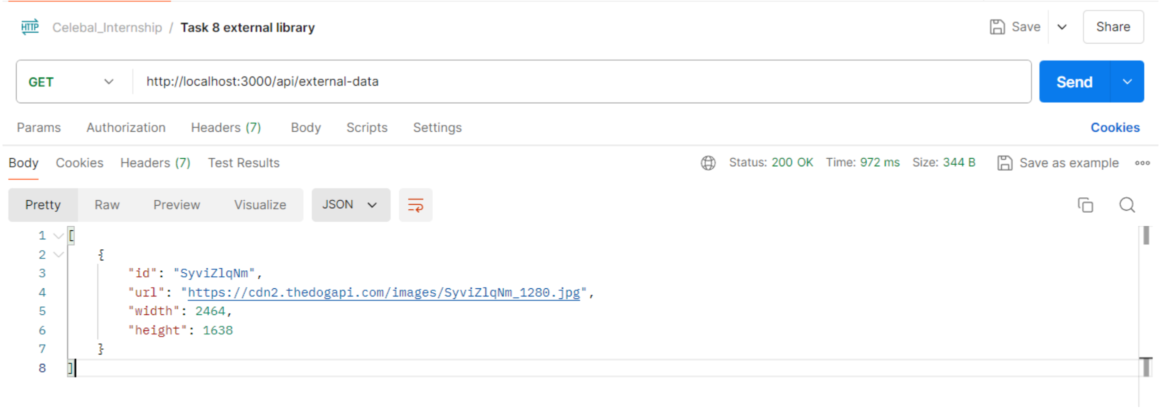
```
{
    "_id": "6697a9814f471b6dc5da55b8",
    "name": "Vraj Desai",
    "email": "21it032@charusat.edu.in",
    "__v": 0
}
```

Figure 3.7.4) Checking if the user is logged in or not

Task 3.8: Enhance your Express.js application

Task Description: Add advanced features to your Express.js application such as file upload, error handling, or integrating a third-party API.

Output:



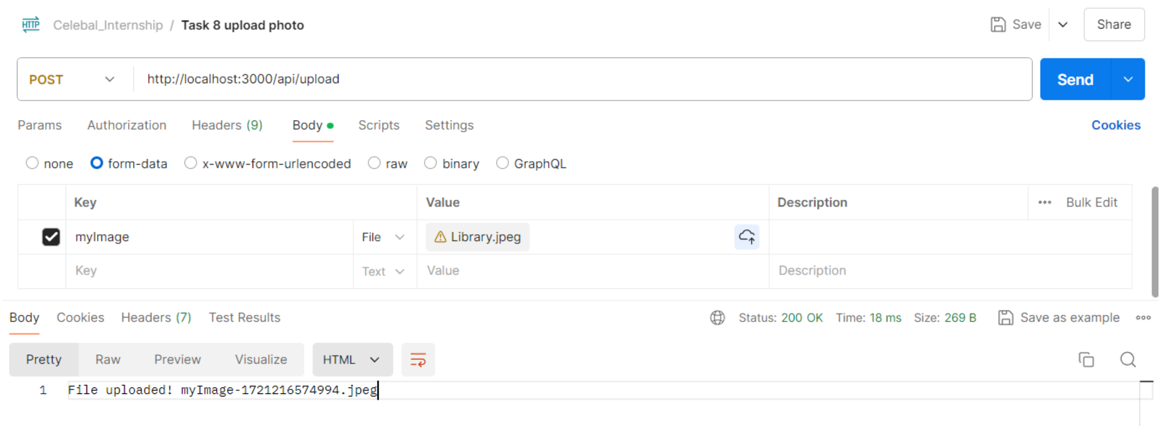
Celebal_Internship / Task 8 external library

GET http://localhost:3000/api/external-data

Status: 200 OK Time: 972 ms Size: 344 B

```
[
    {
        "id": "SyviZlqNm",
        "url": "https://cdn2.thedogapi.com/images/SyviZlqNm_1280.jpg",
        "width": 2464,
        "height": 1638
    }
]
```

Figure 3.8.1) Using third party library "Axios" to fetch the api for image



Celebal_Internship / Task 8 upload photo

POST http://localhost:3000/api/upload

Body: form-data

	Key		Value	Description
✓	myImage	File	Library.jpeg	
	Key	Text	Value	Description

Status: 200 OK Time: 18 ms Size: 269 B

```
File uploaded! myImage-1721216574994.jpeg
```

Figure 3.8.2) photo upload successfully

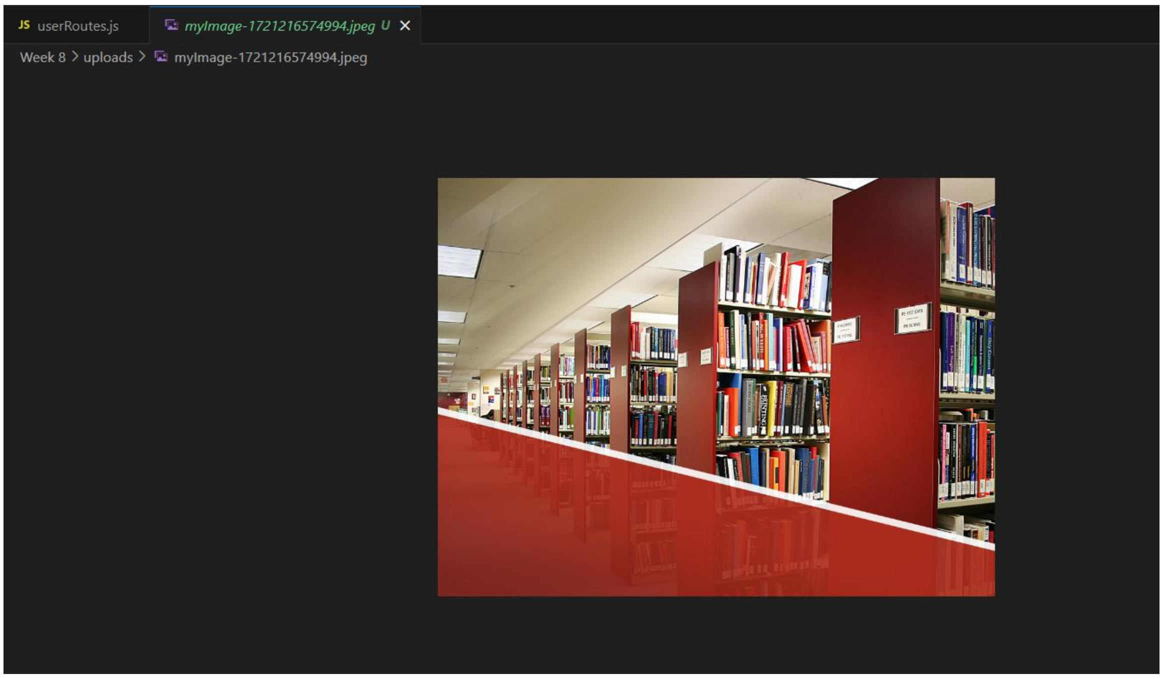
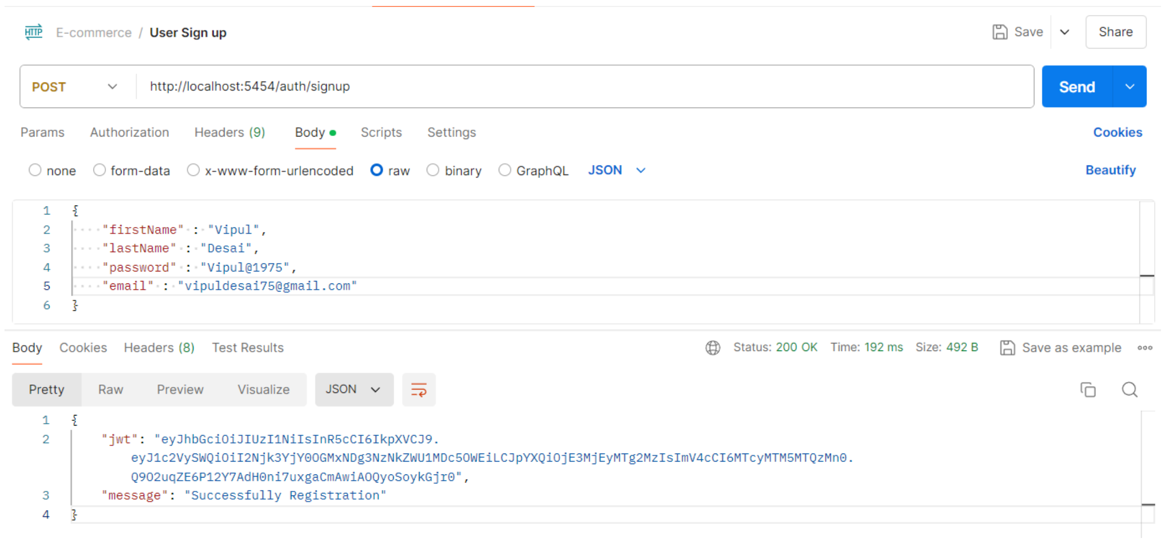

userRoutes.js
myImage-1721216574994.jpeg U
Week 8 > uploads > myImage-1721216574994.jpeg

Figure 3.8.3) Uploaded photo

Task 3.9: E-commerce Website Backend

Task Description: Develop the backend for an e-commerce website using Node.js, Express.js, and MongoDB. Implement features like user authentication, product management, shopping cart functionality, and checkout process.

Output:



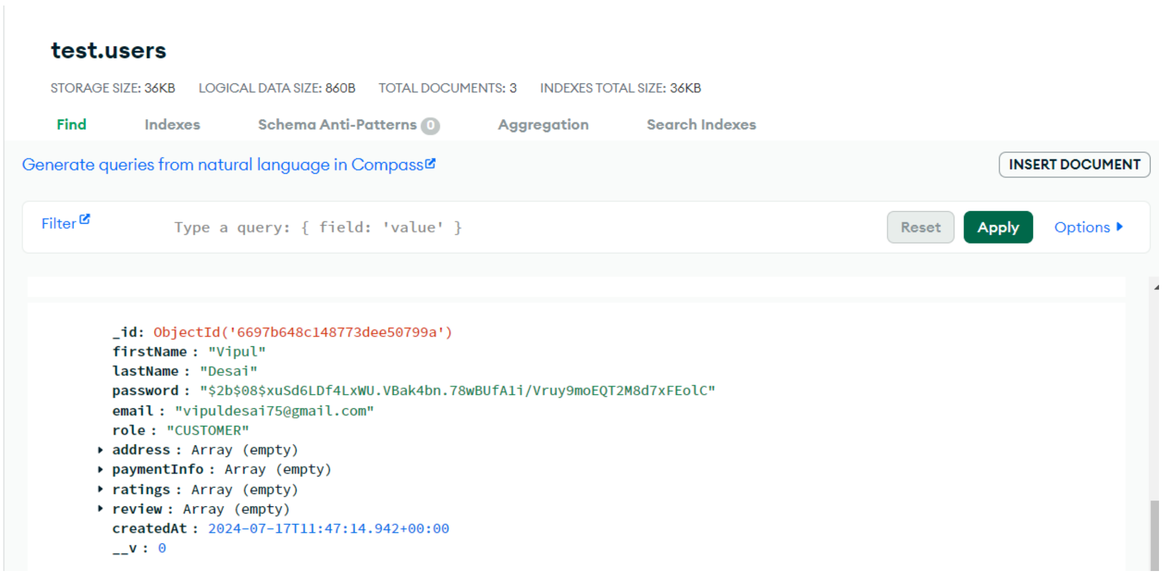
```
POST http://localhost:5454/auth/signup
```

```
{
    "firstName" : "Vipul",
    "lastName" : "Desai",
    "password" : "Vipul@1975",
    "email" : "vipuldesai75@gmail.com"
}
```

Status: 200 OK   Time: 192 ms   Size: 492 B

```
{
    "jwt": "eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.
        eyJ1c2VySWQiOiI2Njk3YjY0OGMxNDg3NzNkZWU1MDc5OWEiLCJpYXQiOjE3MjEyMTg2MzIsImV4cCI6MTcyMTM5MTQzMn0.
        Q9O2uqZE6P12Y7AdH0ni7uxgaCmAwiAOQyoSoykGjr0",
    "message": "Successfully Registration"
}
```

Figure 3.9.1) a new user is registered

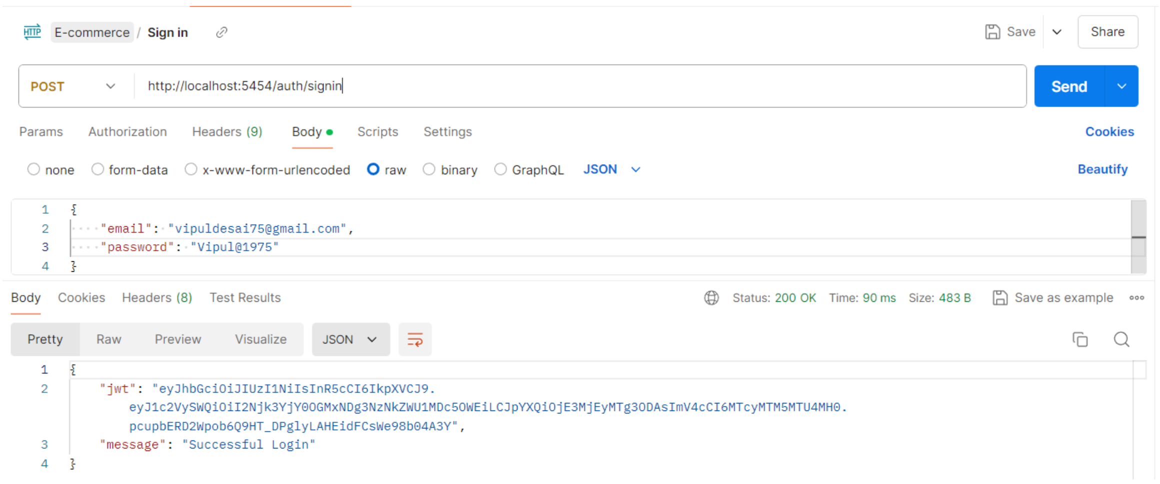


test.users

STORAGE SIZE: 36KB   LOGICAL DATA SIZE: 860B   TOTAL DOCUMENTS: 3   INDEXES TOTAL SIZE: 36KB

```
  _id: ObjectId('6697b648c148773dee50799a')
  firstName : "Vipul"
  lastName : "Desai"
  password : "$2b$08$xuSd6LDf4LxWU.VBak4bn.78wBUfA1i/Vruy9moEQT2M8d7xFEolC"
  email : "vipuldesai75@gmail.com"
  role : "CUSTOMER"
▸ address : Array (empty)
▸ paymentInfo : Array (empty)
▸ ratings : Array (empty)
▸ review : Array (empty)
  createdAt : 2024-07-17T11:47:14.942+00:00
  __v : 0
```

Figure 3.9.2) user is inserted in the database



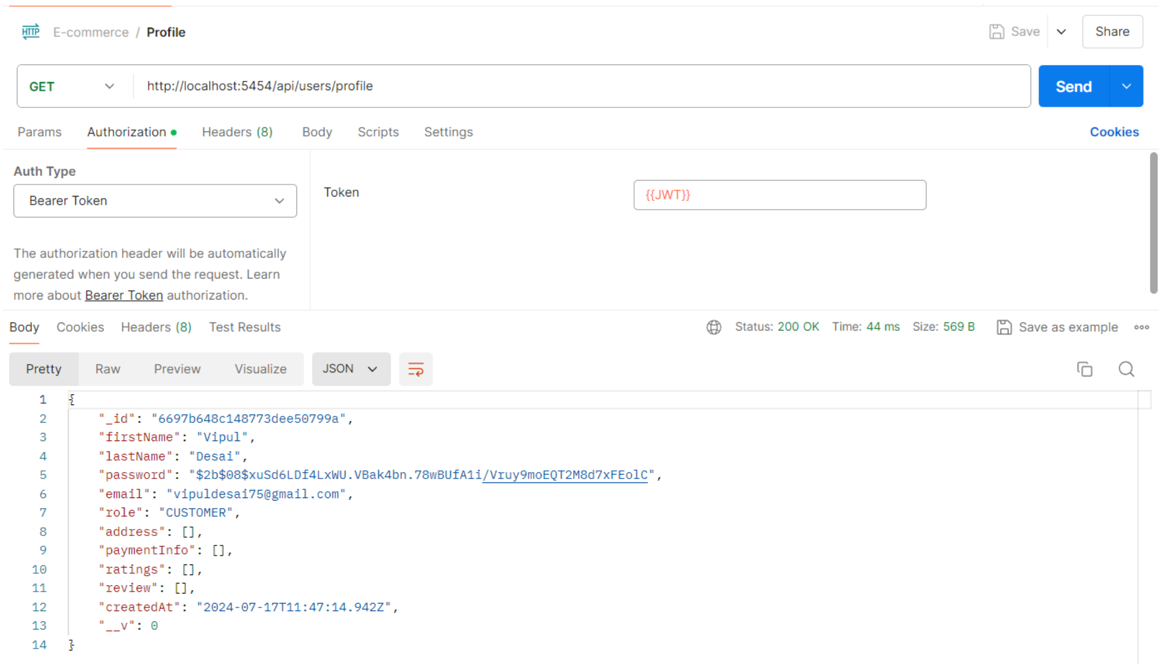
POST http://localhost:5454/auth/signin

```
{
    "email": "vipuldesai75@gmail.com",
    "password": "Vipul@1975"
}
```

Status: 200 OK Time: 90 ms Size: 483 B

```
{
    "jwt": "eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.
        eyJ1c2VySWQiOiI2Njk3YjY0OGMxNDg3NzNkZWU1MDc5OWEiLCJpYXQiOjE3MjEyMTg3ODAsImV4cCI6MTcyMTM5MTU4MH0.
        pcupbERD2Wpob6Q9HT_DPglyLAHEidFCsWe98b04A3Y",
    "message": "Successful Login"
}
```

Figure 3.9.3) User is successfully logged in



GET http://localhost:5454/api/users/profile

Auth Type: Bearer Token

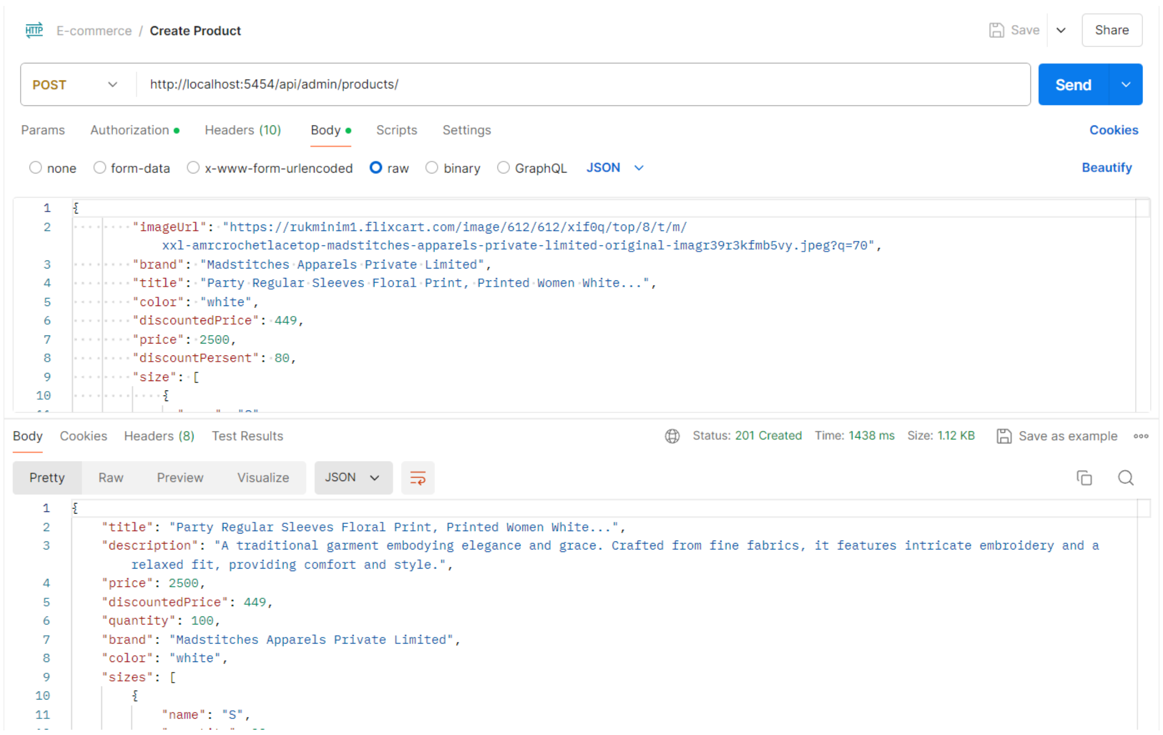
Token: {{JWT}}

The authorization header will be automatically generated when you send the request. Learn more about Bearer Token authorization.

Status: 200 OK Time: 44 ms Size: 569 B

```
{
    "_id": "6697b648c148773dee50799a",
    "firstName": "Vipul",
    "lastName": "Desai",
    "password": "$2b$08$xuSd6LDf4LxWU.VBak4bn.78wBUfA1i/Vruy9moEQT2M8d7xFEolC",
    "email": "vipuldesai75@gmail.com",
    "role": "CUSTOMER",
    "address": [],
    "paymentInfo": [],
    "ratings": [],
    "review": [],
    "createdAt": "2024-07-17T11:47:14.942Z",
    "__v": 0
}
```

Figure 3.9.4) Getting the user profile



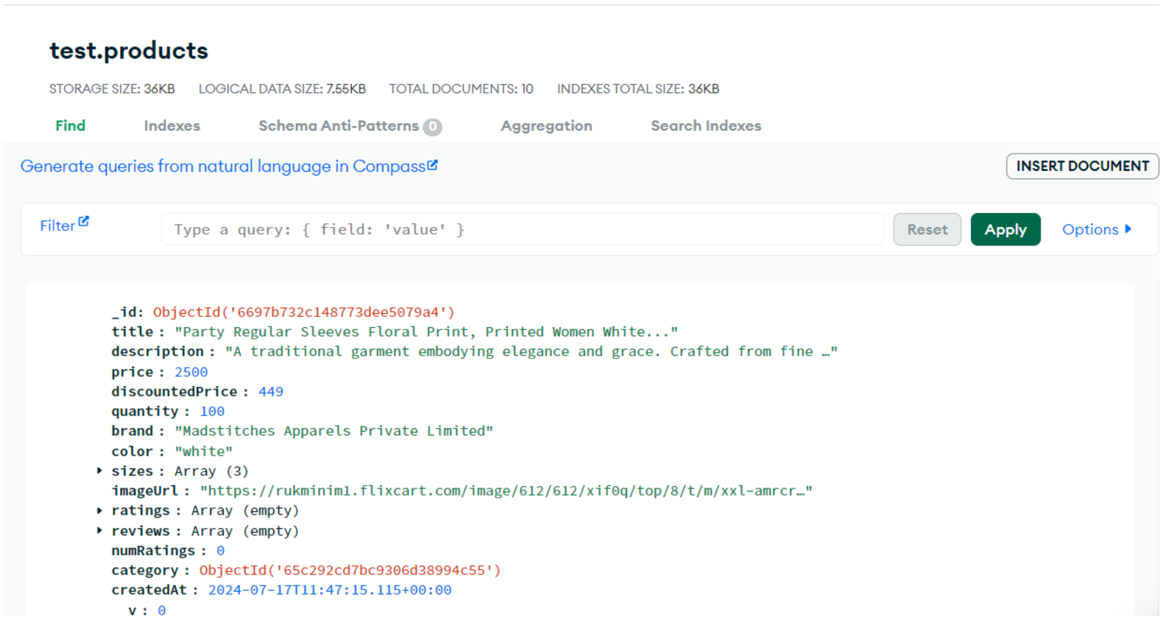
POST http://localhost:5454/api/admin/products/

```
{
    "imageUrl": "https://rukminim1.flixcart.com/image/612/612/xif0q/top/8/t/m/
        xxl-amrcrochetlacetop-madstitches-apparels-private-limited-original-imagr39r3kfmb5vy.jpeg?q=70",
    "brand": "Madstitches Apparels Private Limited",
    "title": "Party Regular Sleeves Floral Print, Printed Women White...",
    "color": "white",
    "discountedPrice": 449,
    "price": 2500,
    "discountPersent": 80,
    "size": [
        {
```

Status: 201 Created Time: 1438 ms Size: 1.12 KB

```
{
    "title": "Party Regular Sleeves Floral Print, Printed Women White...",
    "description": "A traditional garment embodying elegance and grace. Crafted from fine fabrics, it features intricate embroidery and a
        relaxed fit, providing comfort and style.",
    "price": 2500,
    "discountedPrice": 449,
    "quantity": 100,
    "brand": "Madstitches Apparels Private Limited",
    "color": "white",
    "sizes": [
        {
            "name": "S",
```

Figure 3.9.5) Creating a new product

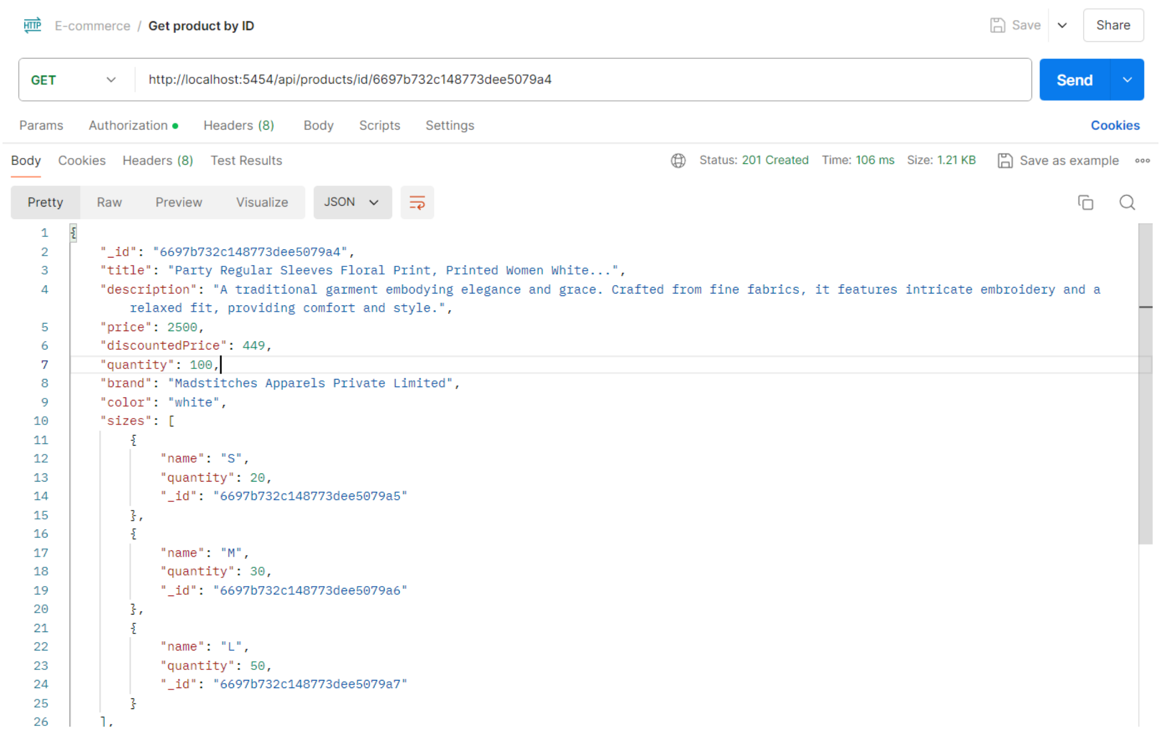


test.products

STORAGE SIZE: 36KB LOGICAL DATA SIZE: 7.55KB TOTAL DOCUMENTS: 10 INDEXES TOTAL SIZE: 36KB

```
_id: ObjectId('6697b732c148773dee5079a4')
title : "Party Regular Sleeves Floral Print, Printed Women White..."
description : "A traditional garment embodying elegance and grace. Crafted from fine …"
price : 2500
discountedPrice : 449
quantity : 100
brand : "Madstitches Apparels Private Limited"
color : "white"
sizes : Array (3)
imageUrl : "https://rukminim1.flixcart.com/image/612/612/xif0q/top/8/t/m/xxl-amrcr…"
ratings : Array (empty)
reviews : Array (empty)
numRatings : 0
category : ObjectId('65c292cd7bc9306d38994c55')
createdAt : 2024-07-17T11:47:15.115+00:00
  v : 0
```

Figure 3.9.6) New product is added in the database



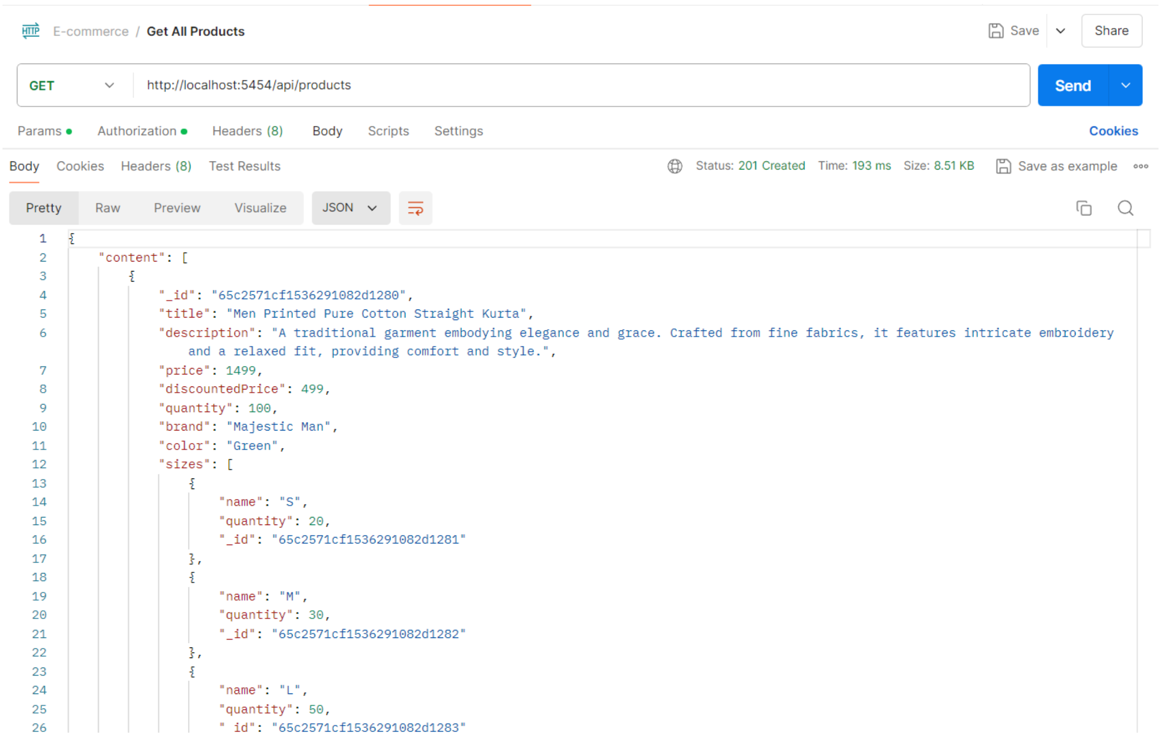
GET http://localhost:5454/api/products/id/6697b732c148773dee5079a4

Status: 201 Created Time: 106 ms Size: 1.21 KB

JSON

```
{
    "_id": "6697b732c148773dee5079a4",
    "title": "Party Regular Sleeves Floral Print, Printed Women White...",
    "description": "A traditional garment embodying elegance and grace. Crafted from fine fabrics, it features intricate embroidery and a
        relaxed fit, providing comfort and style.",
    "price": 2500,
    "discountedPrice": 449,
    "quantity": 100,
    "brand": "Madstitches Apparels Private Limited",
    "color": "white",
    "sizes": [
        {
            "name": "S",
            "quantity": 20,
            "_id": "6697b732c148773dee5079a5"
        },
        {
            "name": "M",
            "quantity": 30,
            "_id": "6697b732c148773dee5079a6"
        },
        {
            "name": "L",
            "quantity": 50,
            "_id": "6697b732c148773dee5079a7"
        }
    ],
```

Figure 3.9.7) Getting the product by id



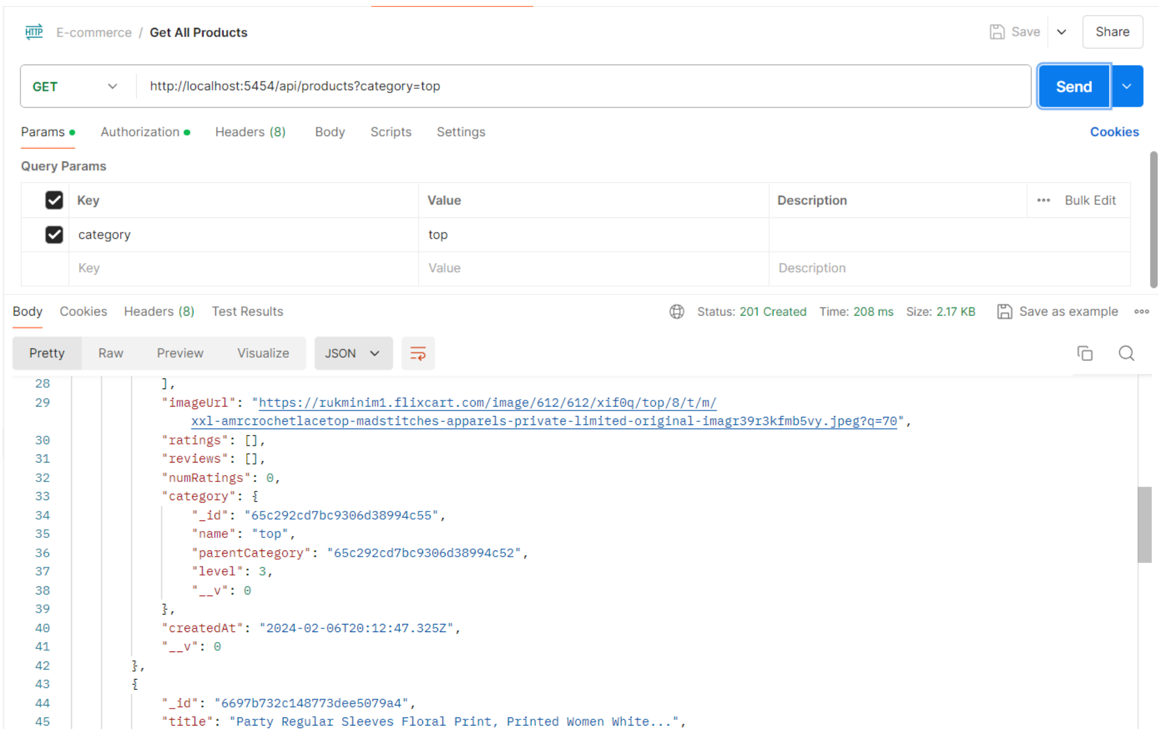
GET http://localhost:5454/api/products

Status: 201 Created Time: 193 ms Size: 8.51 KB

JSON

```
{
    "content": [
        {
            "_id": "65c2571cf1536291082d1280",
            "title": "Men Printed Pure Cotton Straight Kurta",
            "description": "A traditional garment embodying elegance and grace. Crafted from fine fabrics, it features intricate embroidery
                and a relaxed fit, providing comfort and style.",
            "price": 1499,
            "discountedPrice": 499,
            "quantity": 100,
            "brand": "Majestic Man",
            "color": "Green",
            "sizes": [
                {
                    "name": "S",
                    "quantity": 20,
                    "_id": "65c2571cf1536291082d1281"
                },
                {
                    "name": "M",
                    "quantity": 30,
                    "_id": "65c2571cf1536291082d1282"
                },
                {
                    "name": "L",
                    "quantity": 50,
                    "_id": "65c2571cf1536291082d1283"
```

Figure 3.9.8) Getting all of the products



GET http://localhost:5454/api/products?category=top

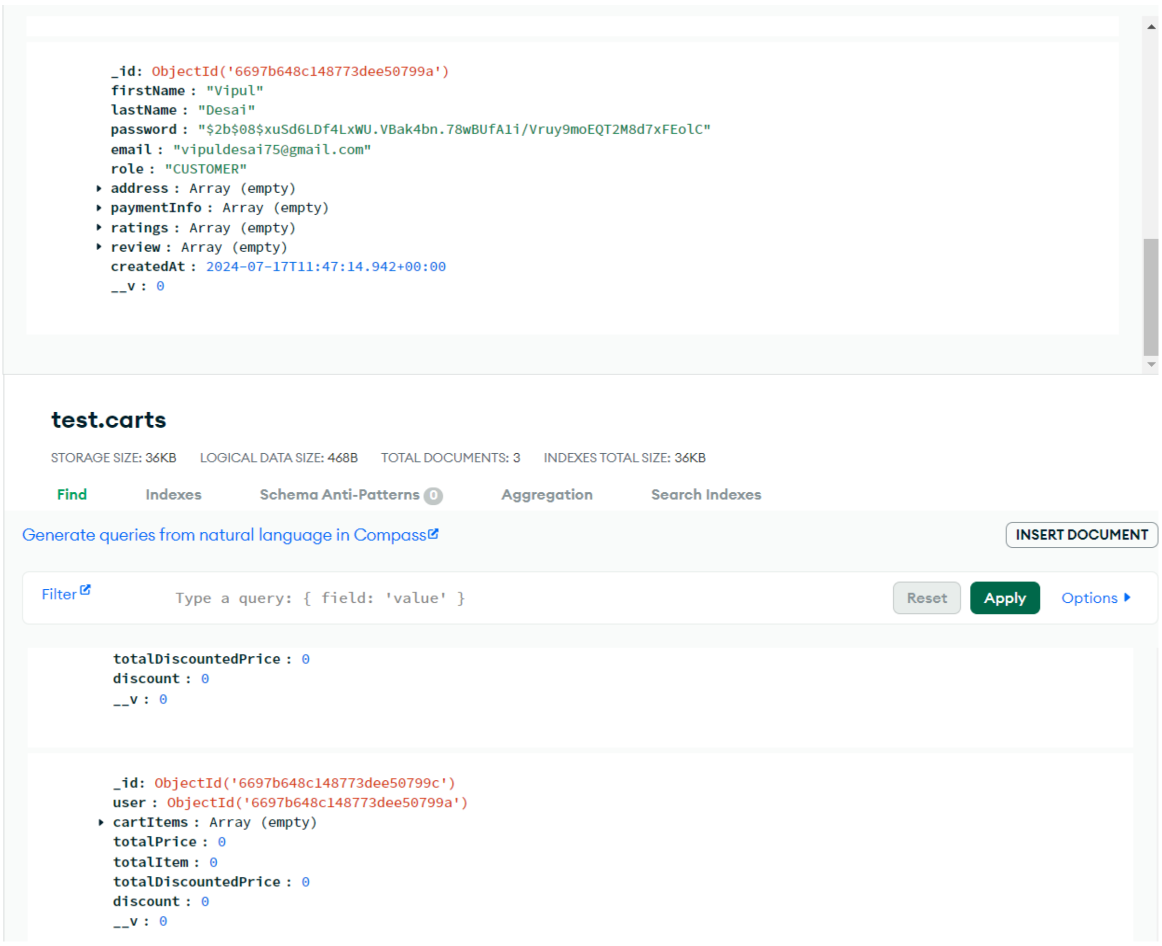
Query Params

	Key	Value	Description
✓	category	top	
	Key	Value	Description

Status: 201 Created Time: 208 ms Size: 2.17 KB

```
        ],
        "imageUrl": "https://rukminim1.flixcart.com/image/612/612/xif0q/top/8/t/m/
            xxl-amrcrochetlacetop-madstitches-apparels-private-limited-original-imagr39r3kfmb5vy.jpeg?q=70",
        "ratings": [],
        "reviews": [],
        "numRatings": 0,
        "category": {
            "_id": "65c292cd7bc9306d38994c55",
            "name": "top",
            "parentCategory": "65c292cd7bc9306d38994c52",
            "level": 3,
            "__v": 0
        },
        "createdAt": "2024-02-06T20:12:47.325Z",
        "__v": 0
    },
    {
        "_id": "6697b732c148773dee5079a4",
        "title": "Party Regular Sleeves Floral Print, Printed Women White...",
```

Figure 3.9.9) Getting a specific kind of product by searching and filtering



```
_id: ObjectId('6697b648c148773dee50799a')
firstName : "Vipul"
lastName : "Desai"
password : "$2b$08$xuSd6LDf4LxWU.VBak4bn.78wBUfA1i/Vruy9moEQT2M8d7xFEolC"
email : "vipuldesai75@gmail.com"
role : "CUSTOMER"
address : Array (empty)
paymentInfo : Array (empty)
ratings : Array (empty)
review : Array (empty)
createdAt : 2024-07-17T11:47:14.942+00:00
__v : 0
```

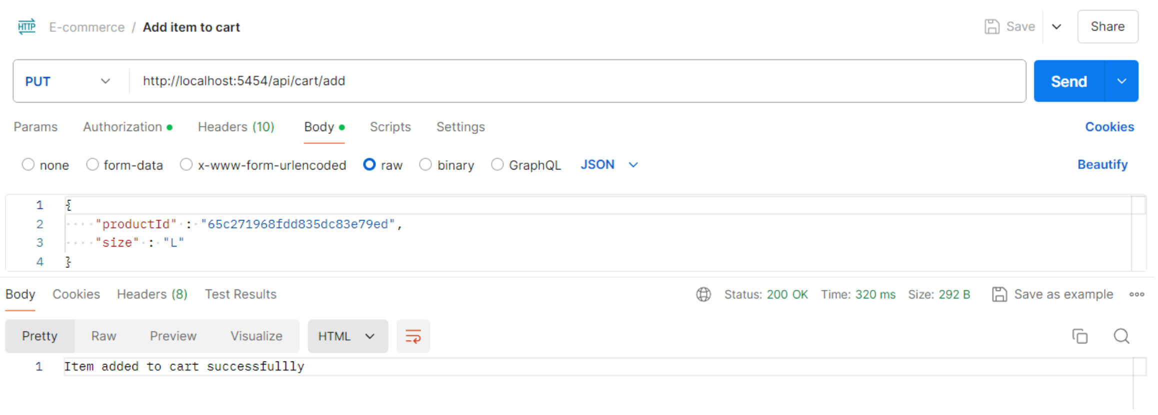
test.carts

STORAGE SIZE: 36KB LOGICAL DATA SIZE: 468B TOTAL DOCUMENTS: 3 INDEXES TOTAL SIZE: 36KB

```
totalDiscountedPrice : 0
discount : 0
__v : 0
```

```
_id: ObjectId('6697b648c148773dee50799c')
user : ObjectId('6697b648c148773dee50799a')
cartItems : Array (empty)
totalPrice : 0
totalItem : 0
totalDiscountedPrice : 0
discount : 0
__v : 0
```

Figure 3.9.10) As soon as the user registers, an empty cart is created which can be verified from the user's id and cart's user id



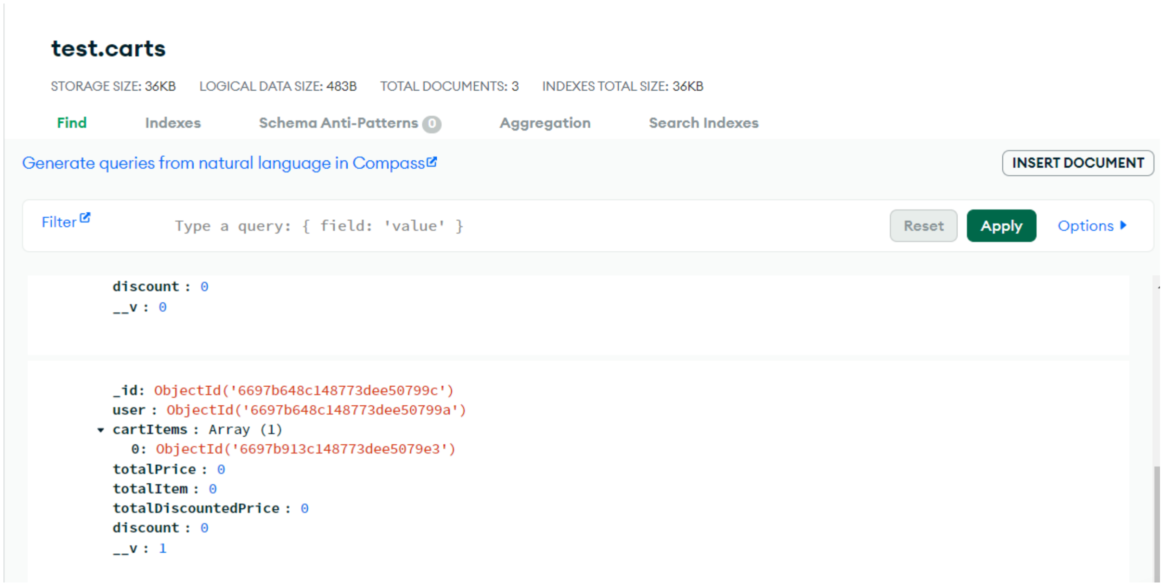
PUT http://localhost:5454/api/cart/add

```
{
    "productId" : "65c271968fdd835dc83e79ed",
    "size" : "L"
}
```

Status: 200 OK Time: 320 ms Size: 292 B

```
Item added to cart successfullly
```

Figure 3.9.11) Adding product to the cart using product's id



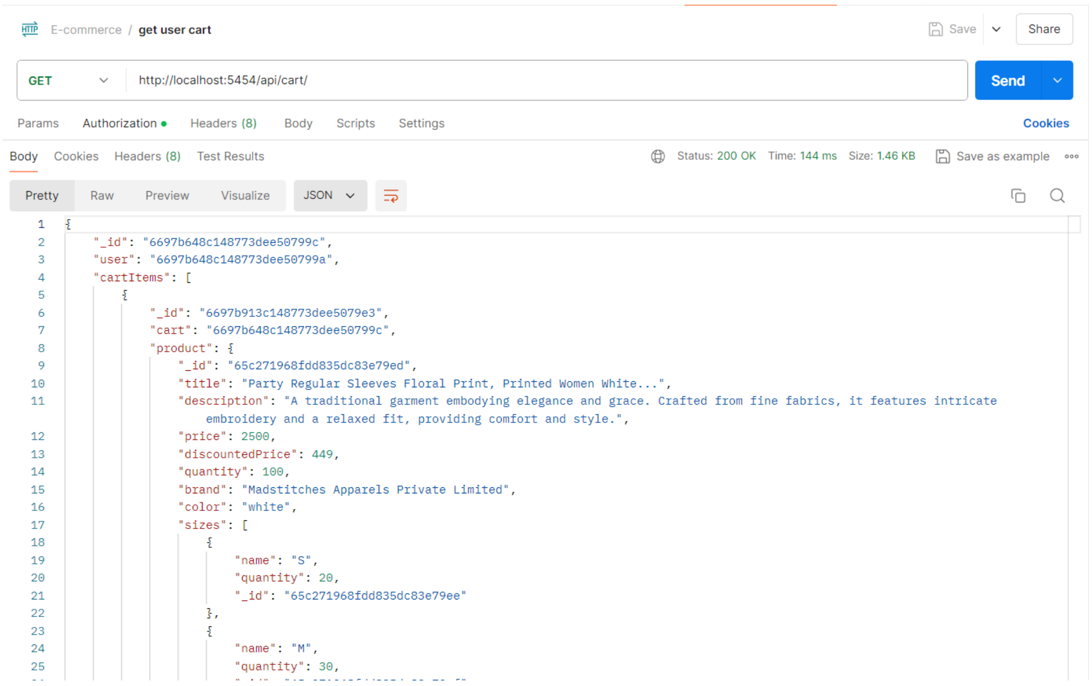
test.carts

STORAGE SIZE: 36KB LOGICAL DATA SIZE: 483B TOTAL DOCUMENTS: 3 INDEXES TOTAL SIZE: 36KB

Type a query: { field: 'value' }

```
discount : 0
__v : 0

_id: ObjectId('6697b648c148773dee50799c')
user : ObjectId('6697b648c148773dee50799a')
cartItems : Array (1)
  0: ObjectId('6697b913c148773dee5079e3')
totalPrice : 0
totalItem : 0
totalDiscountedPrice : 0
discount : 0
__v : 1
```

Figure 3.9.12) A new item is added into the user's cart

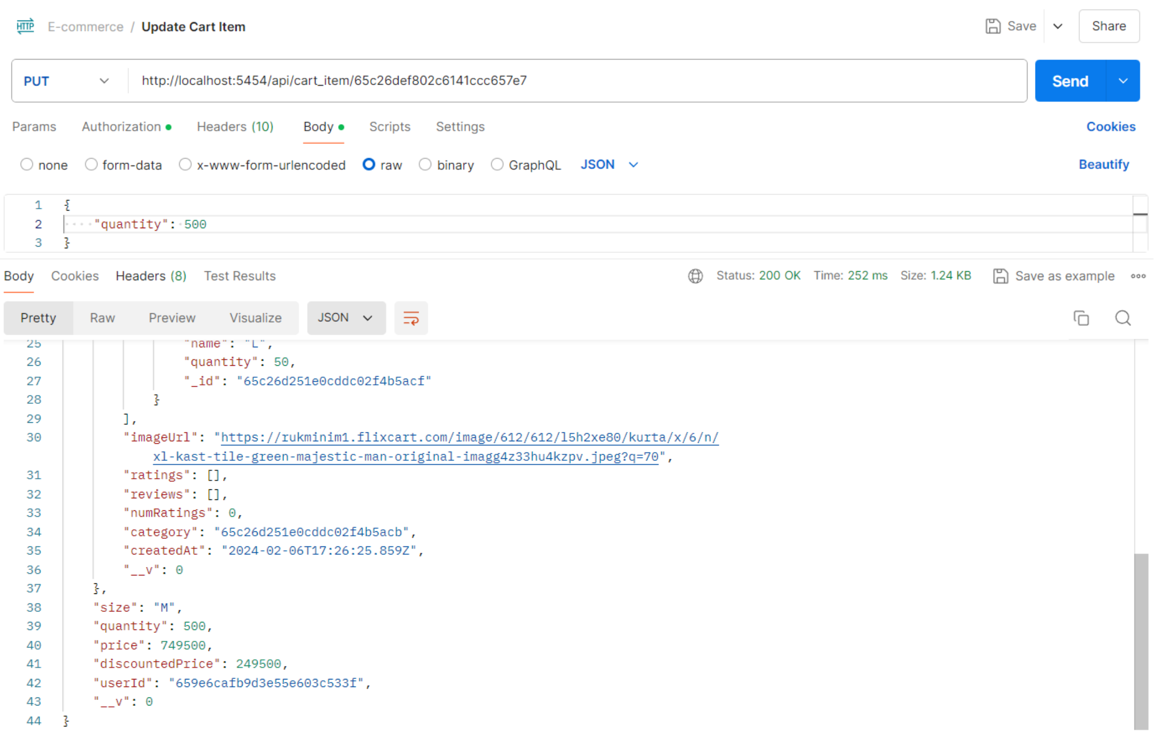


GET http://localhost:5454/api/cart/

Status: 200 OK Time: 144 ms Size: 1.46 KB

```
{
    "_id": "6697b648c148773dee50799c",
    "user": "6697b648c148773dee50799a",
    "cartItems": [
        {
            "_id": "6697b913c148773dee5079e3",
            "cart": "6697b648c148773dee50799c",
            "product": {
                "_id": "65c271968fdd835dc83e79ed",
                "title": "Party Regular Sleeves Floral Print, Printed Women White...",
                "description": "A traditional garment embodying elegance and grace. Crafted from fine fabrics, it features intricate
                    embroidery and a relaxed fit, providing comfort and style.",
                "price": 2500,
                "discountedPrice": 449,
                "quantity": 100,
                "brand": "Madstitches Apparels Private Limited",
                "color": "white",
                "sizes": [
                    {
                        "name": "S",
                        "quantity": 20,
                        "_id": "65c271968fdd835dc83e79ee"
                    },
                    {
                        "name": "M",
                        "quantity": 30,
```

Figure 3.9.13) Getting the user cart



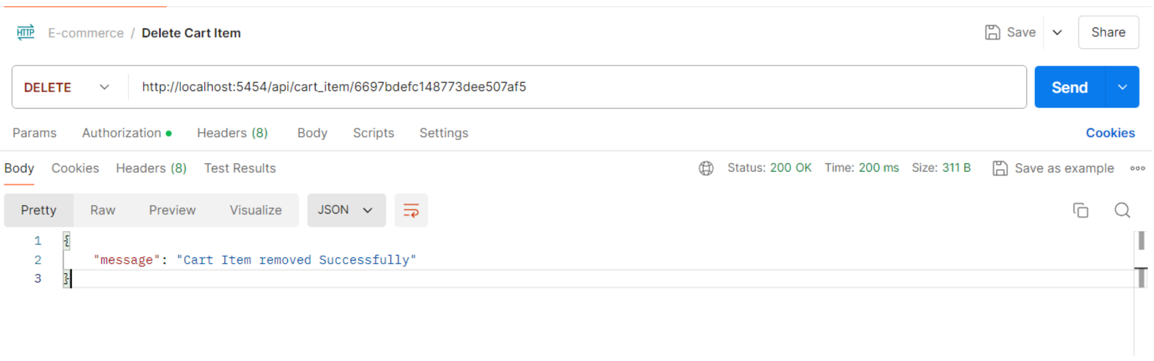
PUT http://localhost:5454/api/cart_item/65c26def802c6141ccc657e7

```
{
    "quantity": 500
}
```

Status: 200 OK Time: 252 ms Size: 1.24 KB

```
                    "name": "L",
                    "quantity": 50,
                    "_id": "65c26d251e0cddc02f4b5acf"
                }
            ],
            "imageUrl": "https://rukminim1.flixcart.com/image/612/612/l5h2xe80/kurta/x/6/n/
                xl-kast-tile-green-majestic-man-original-imagg4z33hu4kzpv.jpeg?q=70",
            "ratings": [],
            "reviews": [],
            "numRatings": 0,
            "category": "65c26d251e0cddc02f4b5acb",
            "createdAt": "2024-02-06T17:26:25.859Z",
            "__v": 0
        },
        "size": "M",
        "quantity": 500,
        "price": 749500,
        "discountedPrice": 249500,
        "userId": "659e6cafb9d3e55e603c533f",
        "__v": 0
}
```

Figure 3.9.14) Updating the cart item

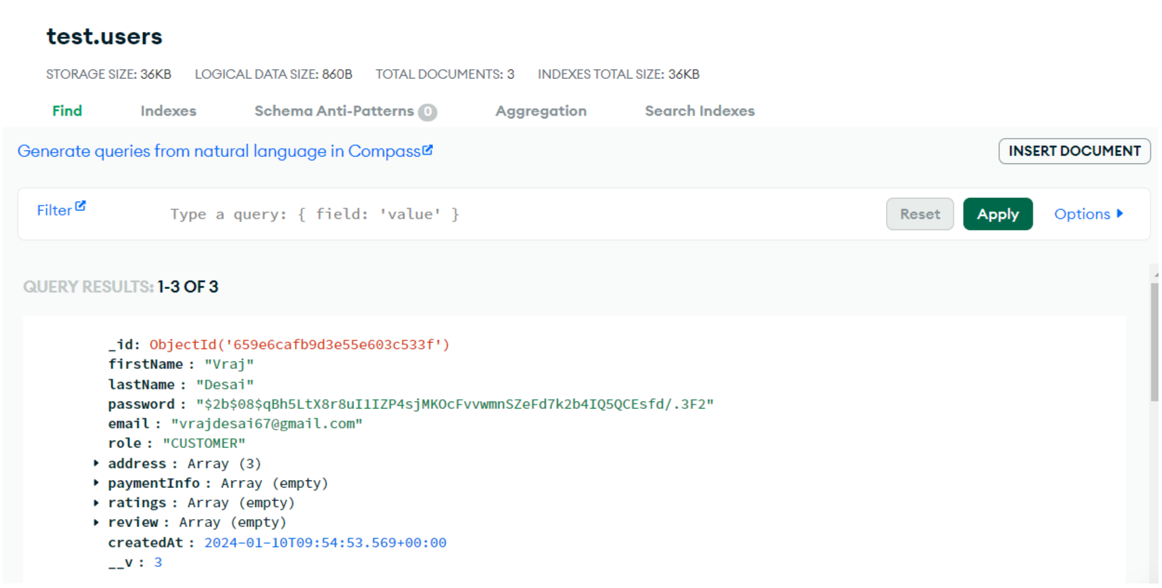


DELETE http://localhost:5454/api/cart_item/6697bdefc148773dee507af5

Status: 200 OK Time: 200 ms Size: 311 B

```
{
    "message": "Cart Item removed Successfully"
}
```

Figure 3.9.15) Deleting item from the cart



test.users

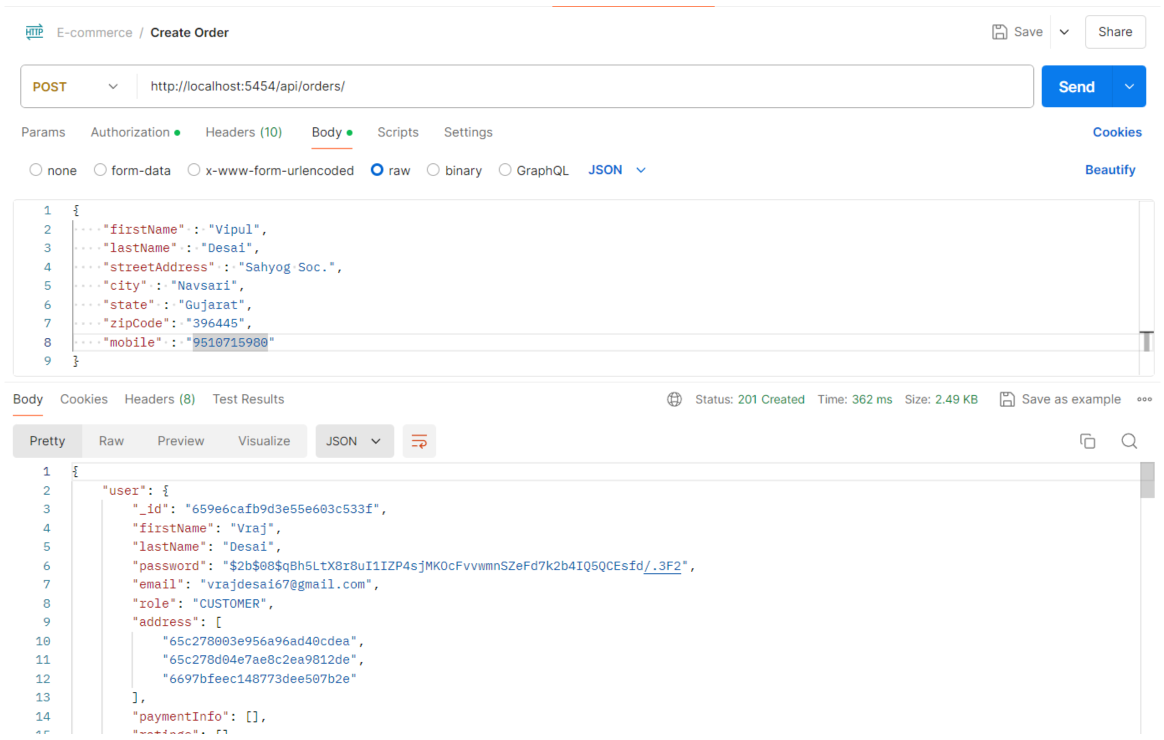
STORAGE SIZE: 36KB LOGICAL DATA SIZE: 860B TOTAL DOCUMENTS: 3 INDEXES TOTAL SIZE: 36KB

Type a query: { field: 'value' }

QUERY RESULTS: 1-3 OF 3

```
_id: ObjectId('659e6cafb9d3e55e603c533f')
firstName : "Vraj"
lastName : "Desai"
password : "$2b$08$qBh5LtX8r8uI1IZP4sjMKOcFvvwmnSZeFd7k2b4IQ5QCEsfd/.3F2"
email : "vrajdesai67@gmail.com"
role : "CUSTOMER"
▸ address : Array (3)
▸ paymentInfo : Array (empty)
▸ ratings : Array (empty)
▸ review : Array (empty)
createdAt : 2024-01-10T09:54:53.569+00:00
__v : 3
```

Figure 3.9.16) Item has been deleted from database



POST http://localhost:5454/api/orders/

Body: ○ none ○ form-data ○ x-www-form-urlencoded ● raw ○ binary ○ GraphQL JSON

```
{
    "firstName" : "Vipul",
    "lastName" : "Desai",
    "streetAddress" : "Sahyog Soc.",
    "city" : "Navsari",
    "state" : "Gujarat",
    "zipCode":"396445",
    "mobile" : "9510715980"
}
```

Body Cookies Headers (8) Test Results — Status: 201 Created Time: 362 ms Size: 2.49 KB

JSON

```
{
    "user": {
        "_id": "659e6cafb9d3e55e603c533f",
        "firstName": "Vraj",
        "lastName": "Desai",
        "password": "$2b$08$qBh5LtX8r8uI1IZP4sjMKOcFvvwmnSZeFd7k2b4IQ5QCEsfd/.3F2",
        "email": "vrajdesai67@gmail.com",
        "role": "CUSTOMER",
        "address": [
            "65c278003e956a96ad40cdea",
            "65c278d04e7ae8c2ea9812de",
            "6697bfeec148773dee507b2e"
        ],
        "paymentInfo": [],
```

Figure 3.9.17) Creating order

CHAPTER 4 LEARNING EXPERIENCE

4.1 KNOWLEDGE ACQUIRED/SKILLS LEARNED IN BACKEND DEVELOPMENT:

When diving into backend development, you gain a wide array of knowledge and skills that are crucial for building, maintaining, and scaling server-side applications. Here are some key areas and skills acquired:

Fundamentals of Backend Development:

- **Understanding Backend Architecture**: Learned core concepts such as client-server architecture, RESTful services, and API development.
- **Database Management**: Gained skills in designing and managing databases, including CRUD operations, data modeling, and schema design.
- **Security Practices**: Learned techniques to secure applications, such as authentication, authorization, and data encryption.

Server-Side Programming:

- **Node.js**: Mastered this runtime environment for executing JavaScript code server-side, leveraging its asynchronous nature and extensive libraries.
- **Express.js**: Gained expertise in building RESTful APIs with Express, managing routes, middleware, and handling HTTP requests and responses.

Authentication and Authorization:

- **JWT (JSON Web Tokens)**: Learned to implement user authentication and authorization using JWT for secure and stateless sessions.
- **OAuth2 and Passport.js**: Gained understanding of OAuth2 protocols and how to integrate third-party authentication providers using Passport.js.

Database Interaction:

- **MongoDB**: Acquired skills in using MongoDB for NoSQL database management, including document-oriented data modeling and aggregation pipelines.
- **Mongoose**: Mastered Mongoose for schema-based data validation, relationships, and middleware for MongoDB.

Middleware and Event Handling:

- **Event-Driven Architecture**: Learned to implement event-driven programming for decoupling application components and handling asynchronous operations using EventEmitter.
- **Middleware Functions**: Gained skills in creating and utilizing middleware for tasks like logging, error handling, and request parsing.

File Handling and Storage:

- **File Uploads**: Learned to handle file uploads using libraries like Multer, including processing, storing, and validating uploaded files.
- **File System Operations**: Gained expertise in performing file system operations such as reading, writing, and deleting files using Node.js built-in modules.

API Development and Testing:

- **RESTful API Design**: Learned to design and implement RESTful APIs, adhering to best practices and principles for scalable and maintainable code.
- **Postman**: Gained proficiency in using Postman for API testing, including creating requests, automating tests, and validating responses.

Debugging and Optimization:

- **Debugging**: Enhanced proficiency in debugging backend code using tools like Node.js debugger and console logging.
- **Performance Optimization**: Learned to optimize server performance through techniques like caching, query optimization, and load balancing.

4.2 REAL-TIME APPLICABILITY OF TECHNOLOGIES LEARNED IN BACKEND DEVELOPMENT:

The technologies and skills learned in backend development have wide-ranging real-time applications across various domains. Here are some examples of how these can be applied:

1. User Authentication and Security:

- **E-commerce Platforms**: Implementing secure user authentication and authorization for online shopping platforms.
- **Banking Systems**: Securing user accounts and transactions with robust authentication mechanisms.

2. Database Management:

- **Content Management Systems (CMS)**: Managing and retrieving content efficiently using NoSQL databases like MongoDB.
- **Customer Relationship Management (CRM)**: Storing and managing customer data to provide personalized services.

3. File Handling:

- **Social Media**: Handling user uploads, such as profile pictures, posts, and media content.
- **Document Management Systems**: Enabling secure upload, storage, and retrieval of documents.

4. API Development:

- **Microservices Architecture**: Building and deploying microservices to handle

specific business logic within a larger application.
- **Third-Party Integrations**: Developing APIs to integrate with external services like payment gateways, email services, and social media platforms.

5. Real-Time Applications:
- **Chat Applications**: Implementing real-time communication features using WebSockets and event-driven architecture.
- **Real-Time Analytics**: Providing live data insights and updates for dashboards and monitoring systems.

6. Scalability and Performance:
- **High-Traffic Websites**: Ensuring scalability and performance for websites with a high volume of concurrent users.
- **Load Balancing**: Distributing incoming traffic across multiple servers to optimize resource usage and performance.

7. Event-Driven Systems:
- **Notification Systems**: Implementing event-driven notifications for real-time alerts and updates.
- **Background Processing**: Handling background tasks and processing jobs asynchronously to improve application responsiveness.

8. DevOps and Deployment:
- **Continuous Integration/Continuous Deployment (CI/CD)**: Automating the deployment process to ensure seamless updates and integration.
- **Containerization**: Using Docker to containerize applications for consistent deployment across different environments.

CHAPTER 5 CONCLUSION

Backend development is a critical component of modern software applications, providing the necessary infrastructure and services that support front-end functionality. Throughout this internship, a solid foundation in backend development was established, covering essential topics such as user authentication, database management, file handling, API development, real-time applications, scalability, event-driven systems, and DevOps practices. These skills are essential for creating secure, scalable, and efficient backend systems that can handle a wide range of real-world applications.

The knowledge and expertise gained during this internship are applicable across various domains, including e-commerce, banking, social media, and content management. For instance, implementing secure user authentication and authorization is crucial for protecting user data in online platforms, while efficient database management ensures seamless content retrieval and personalized services. Additionally, the ability to develop robust APIs facilitates integration with third-party services, enhancing the overall functionality of applications.

Backend development, leveraging technologies such as Node.js, Express.js, and MongoDB, provides a powerful platform for building innovative and responsive applications. The skills acquired in handling real-time communication, managing background processes, and ensuring performance and scalability are invaluable for developing high-traffic websites and real-time analytics systems. By mastering these technologies, one can significantly contribute to the creation of intelligent, data-driven applications that drive efficiency and progress in various industries. Continuous learning and staying updated with the latest advancements in backend development will further enhance these skills and open up exciting career opportunities in the dynamic field of software engineering.

REFERNCES:

RESEARCH PAPER:

[1] Richards, M., & Murach, J. (2020). Murach's Node.js. Murach & Associates.

[2] Dayley, B., & Dayley, B. (2017). Node.js, MongoDB, and AngularJS Web Development (2nd ed.). Addison-Wesley.

[3] Soshnikov, I. (2019). Advanced Node.js Development: Master Node.js by Building Real-World Applications. Packt Publishing.

[4] Date, C. J. (2019). An Introduction to Database Systems (8th ed.). Addison-Wesley.

[5] Freeman, E., & Robson, E. (2015). Head First Design Patterns. O'Reilly Media.

[6] Vohra, D. (2018). Pro RESTful APIs: Design, Build and Integrate with REST, JSON, XML and JAX-RS. Apress.

WEB REFERENCE:

1. https://expressjs.com/
2. https://nodejs.org/en/docs/
3. https://docs.mongodb.com/
4. https://jwt.io/introduction/
5. https://swagger.io/docs/
6. https://www.rabbitmq.com/tutorials/tutorial-one-javascript.html
7. https://www.youtube.com/watch?v=gnsO8-xJ8rs (Node.js Crash Course)
8. https://www.youtube.com/watch?v=7S_tz1z_5bA (Express.js Tutorial)
9. https://www.youtube.com/watch?v=9zUHg7xjIqQ (MongoDB Tutorial)
10. https://www.youtube.com/watch?v=6YqPq8MtSC8 (JWT Authentication in Node.js)