SHRI VILEPARLE KELAVANI MANDAL'S
DWARKADAS J. SANGHVI COLLEGE OF ENGINEERING
(Autonomous College Affiliated to the University of Mumbai)
NAAC ACCREDITED with "A" GRADE (CGPA : 3.18)

ACADEMIC YEAR: 2024-2025 SAP: 60003220045

COURSE CODE: DJS22ITL604 **DATE: 11-02-2025**

COURSE NAME: Full Stack Web Development Laboratory **CLASS: TYBTech**

NAME: Anish Sharma **DIV: IT1-1** **ROLL: I011**

DEPARTMENT OF INFORMATION TECHNOLOGY

EXPERIMENT NO. 03

CO/LO: CO1-Develop a full stack web application.

AIM / OBJECTIVE: Creating RESTful APIs with Express.js: Create a MongoDB database, design basic schemas, and execute CRUD operations using the MongoDB driver.

THEORY:

Introduction to RESTful APIs

A RESTful API (Representational State Transfer) allows communication between client and server over HTTP using standard methods such as GET, POST, PUT, and DELETE. Express.js, a popular Node.js framework, simplifies the creation of these APIs.

Introduction to MongoDB

MongoDB is a NoSQL database that stores data in JSON-like documents. It is highly scalable and allows flexible schema design.

Key Concepts:

- **Express.js: A minimal and flexible web application framework for Node.js.**
- **MongoDB Driver: A Node.js library that enables database operations.**
- **Mongoose: An ODM (Object Data Modeling) library for MongoDB that simplifies schema creation.**
- **CRUD Operations: Create, Read, Update, and Delete functions for managing data.**

PROCEDURE

Step 1: Set Up the Project

1. **Install Node.js from** https://nodejs.org/.
2. **Create a new directory and initialize a project:**

```
mkdir restful-api && cd restful-api npm
init -y
```

3. **Install dependencies: npm install express mongoose cors body-parser dotenv**

Step 2: Configure Express.js Server

1. **Create a file server.js and add:**

```
const express = require('express'); const
mongoose = require('mongoose'); const
cors = require('cors'); const bodyParser =
require('body-parser');
require('dotenv').config();

const app = express();
app.use(cors());
app.use(bodyParser.json());

mongoose.connect(process.env.MONGO_URI, {
   useNewUrlParser: true, useUnifiedTopology:
   true
}).then(() => console.log('MongoDB Connected'))
.catch(err => console.log(err));

const PORT = process.env.PORT || 5000;
app.listen(PORT, () => console.log(`Server running on port ${PORT}`));
```

Step 3: Define MongoDB Schema and Model

1. **Create a folder models and add a file User.js:**

```
const mongoose = require('mongoose');
const UserSchema = new mongoose.Schema({
  name: String, email: String, age: Number
});

module.exports = mongoose.model('User', UserSchema);
```

Step 4: Implement CRUD Routes

```
const express = require('express'); const User = require('../models/User'); const router = express.Router();

// Create User
router.post('/users', async (req, res) => { const user = new User(req.body);
  await user.save();
  res.send(user);
});

// Read Users
router.get('/users', async (req, res) => {
  const users = await User.find();
  res.send(users);
});

// Update User
router.put('/users/:id', async (req, res) => { const user = await User.findByIdAndUpdate(req.params.id, req.body, { new: true }); res.send(user);
});

// Delete User
router.delete('/users/:id', async (req, res) => {

  await User.findByIdAndDelete(req.params.id);
  res.send({ message: 'User deleted' });
});
```

```
module.exports = router;
```

Step 5: Integrate Routes with Express.js

1. Modify `server.js`:

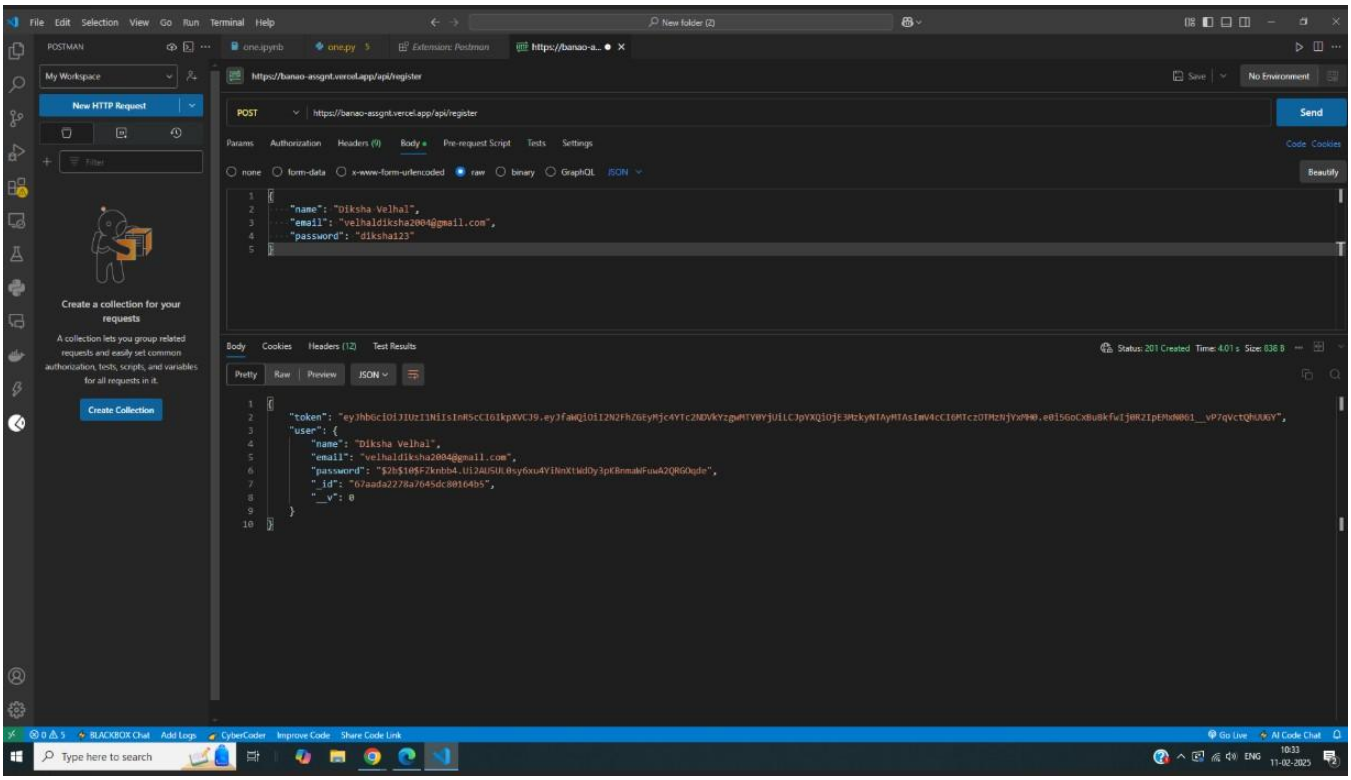
```
const userRoutes = require('./routes/userRoutes'); app.use('/api',
userRoutes);
```

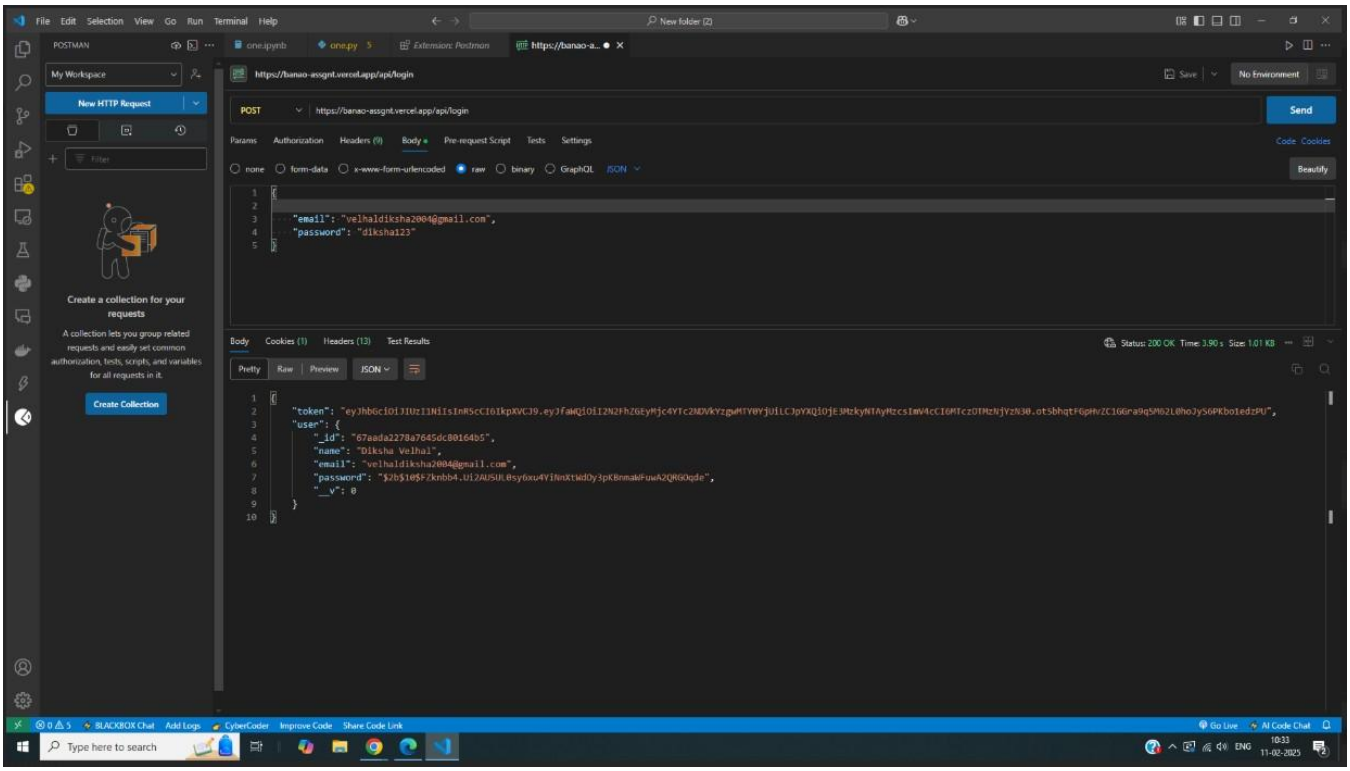
Step 6: Start the Server

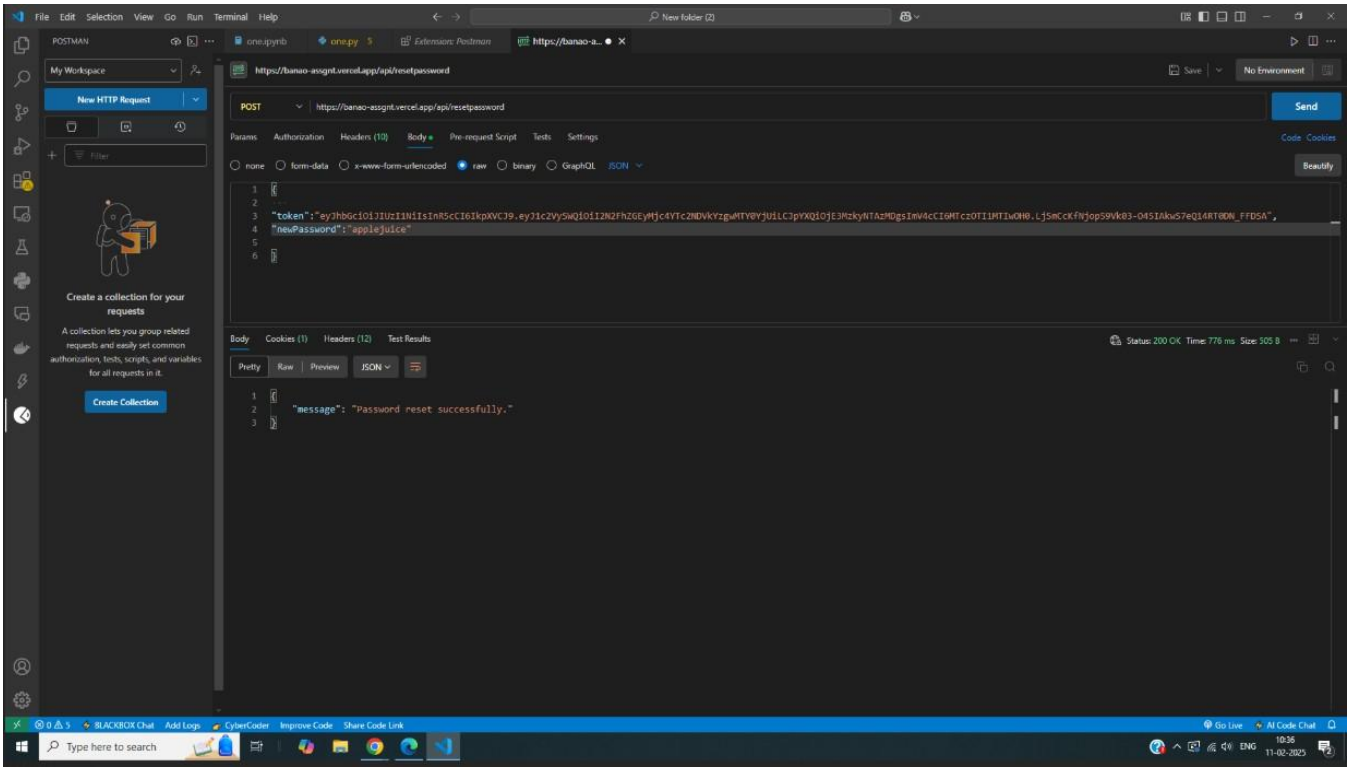
Run the following command:

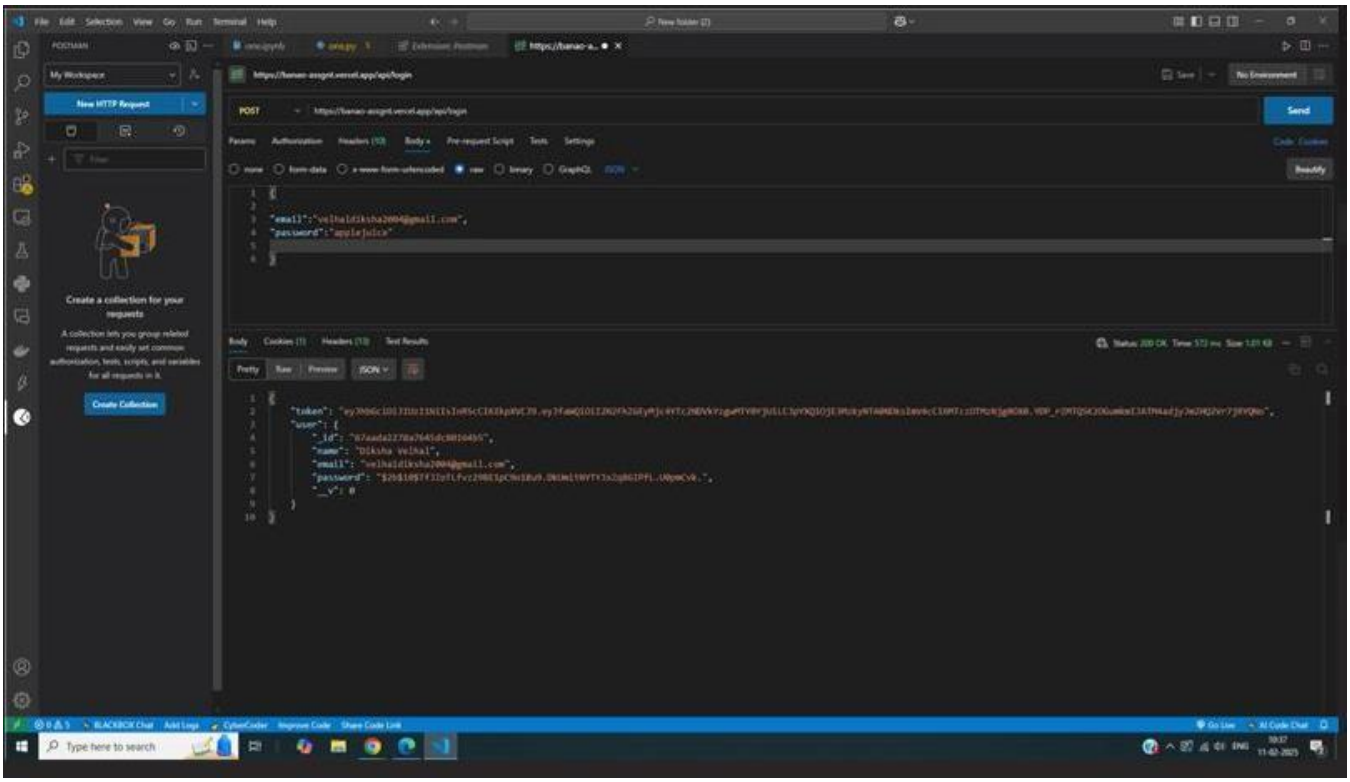
```
node server.js
```

Your API will be available at `http://localhost:5000/api/users`.




POST
https://banao-assgnt.vercel.app/api/login
{
"email": "velhaldiksha2004@gmail.com",
"password": "diksha123"
}
Status: 200 OK
{
"token": "eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.ey3faWQiOiI2N2FhZGEyMjc4YTc2NDVkYzgwMTY0YjUiLCJpYXQiOjE3MzkyNTAyMzcsImV4cCI6MTczOTMzNjYzN30.otSbhqtFGpHvZC1GGra9q5M02L0hoJyS6PKbo1edzPU",
"user": {
"_id": "67aada2278a7645dc80164b5",
"name": "Diksha Velhal",
"email": "velhaldiksha2004@gmail.com",
"password": "$2b$10$F2knbb4.Ui2AU5ULBsy6xu4YiNnXtWdOy3pKBnmaWFuwA2QR6Oqde",
"__v": 0
}
}


POST
https://banao-assgnt.vercel.app/api/resetpassword
{
"token":"eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJ1c2VySWQiOiI2N2FhZGEyMjc4YTc2NDVkYzgwMTY0YjUiLCJpYXQiOjE3MzkyNTAzMDgsImV4cCI6MTczOTI1MTIwOH0.LjSmCcKfNjopS9Vk03-O4SIAkwS7eQ14RT0DN_FFDSA",
"newPassword":"applejuice"
}
Status: 200 OK
{
"message": "Password reset successfully."
}



```
_id: ObjectId('67aada2278a7645dc80164b5')
name : "Diksha Velhal"
email : "velhaldiksha2004@gmail.com"
password : "$2b$10$TfJZoTLfvz29BE1pC9o1Bu9.DkUmitNYTYJxZq8GIPfL.U0pmCvk."
__v : 0
```

CONCLUSION: In this experiment, we created RESTful APIs with Express.js: Create a MongoDB database, design basic schemas, and execute CRUD operations using the MongoDB driver.

BOOKS AND WEB RESOURCES:

[1] MDN Web Docs - https://developer.mozilla.org/en-US/docs/Learn/Server-side/Express_Nodejs

[2] MongoDB Documentation - https://www.mongodb.com/docs/manual/ [3] Express.js Guide - https://expressjs.com/en/guide/routing.html