Honors- Internal Assessment

Name : Swarali Fendar
Roll No. : 13
Sem : VI(3rd year)

Phase 1: Environment Setup
Phase 2: Backend Development (Express.js)
Phase 3: MongoDB Atlas Integration
Phase 4: Email Registration Flow (Nodemailer + Gmail)
Phase 5: JWT Authentication Mechanism
Phase 6: Login Functionality
Phase 7: Protected Route Access Using JWT
Phase 8: Deployment (Nginx + HTTPS via Let's Encrypt)
Phase 9: Final Testing & Demo with Postman

1. Install WSL Ubuntu Terminal-
   Install nginx
   Create a directory for your project-
   mkdir secure-backend-app
   cd secure-backend-app

2. In the directory-
   npm init -y
   Install express mongoose nodemailer jsonwebtoken bcrypt dotenv
   In the directory have the folder structure as -

```
secure-backend-app/
├── controllers/
│   └── authController.js
├── models/
│   └── User.js
├── routes/
│   └── authRoutes.js
├── middlewares/
│   └── authMiddleware.js
├── .env
├── server.js
├── package.json
```

   To open the folder in VS code, install WSL Remote and open it using the command-
   code .

3. MongoDB Atlas -
   Create cluster and connect string-
   Save the following details in the .env file as-

# MongoDB Atlas Connection String
MONGO_URI=mongodb+srv://<yourMongoUser>:<yourMongoPassword>@cluster0.mongodb.net/secureApp

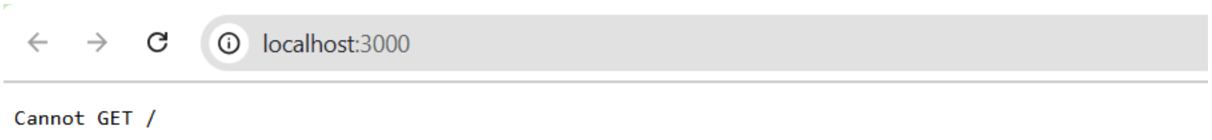
# JWT Secret Key for Token Generation
JWT_SECRET=yourStrongJwtSecretKey

# Gmail Credentials for Nodemailer (Use App Password here)
EMAIL_USER=yourEmail@gmail.com
EMAIL_PASS=your16CharAppPassword

Select the option Allow Access from Anywhere
Generate the JWT_Secret key by running the command-
openssl rand -base64 32

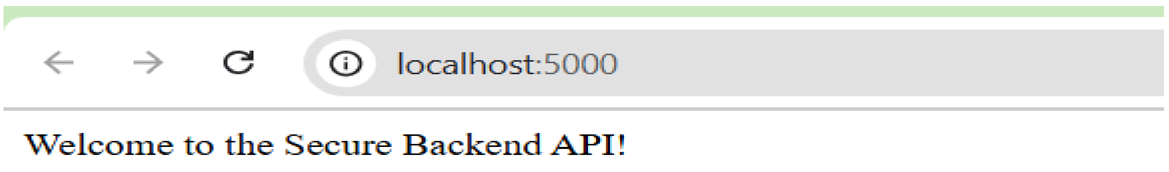
4. Run the command -
   npm start
   Key challenges-
   -Configure the Code of the server.js
   -Check the package.json file for the start script and the "main"
   -Make sure to install all the dependencies such as node in the Ubuntu terminal path not in the Windows, also install dotenv, express, etc.

5. I configure an error first the server run on port 3000



```
Cannot GET /
```

After all issues are solved it finally run on port 5000



Welcome to the Secure Backend API!

This is the initial step to successfully run the server by Express.js server.

6. Install Nodemailer -
   npm install nodemailer

Add code for the folders-
controllers, middlewares, models, routes with the files as authController.js, authMiddleware.js, User.js, authRoutes.js

authController.js

```
const User = require('../models/User');
const jwt = require('jsonwebtoken');
const bcrypt = require('bcryptjs');
const nodemailer = require('nodemailer');

const JWT_SECRET = process.env.JWT_SECRET;

// Nodemailer transporter (Gmail / SMTP setup)
const transporter = nodemailer.createTransport({
    service: 'Gmail',  // OR use 'smtp.ethereal.email' / custom
SMTP settings
    auth: {
        user: process.env.EMAIL_USER,  // Your Gmail address
        pass: process.env.EMAIL_PASS   // App password (not your
real password)
    }
});

console.log('EMAIL_USER:', process.env.EMAIL_USER);
console.log('EMAIL_PASS:', process.env.EMAIL_PASS ? 'Loaded' :
'Missing');

// Register User
exports.registerUser = async (req, res) => {
    try {
        const { email, password } = req.body;
        const existingUser = await User.findOne({ email });
        if (existingUser) {
            return res.status(400).json({ message: 'User already
exists' });
        }

        const hashedPassword = await bcrypt.hash(password, 10);
        const newUser = new User({ email, password:
hashedPassword, verified: false });

        await newUser.save();

        // console.log('JWT_SECRET:',JWT_SECRET);
        const token = jwt.sign({ userId: newUser._id },
JWT_SECRET, { expiresIn: '1h' });
        // console.log("Verification Token:",token);
        console.log('Verification Token:', token);

        const verificationLink =
`${process.env.BASE_URL}/api/auth/verify-registration?token=${tok
en}`;

        // Send verification email
        try{
        await transporter.sendMail({
            from: `"Secure Backend App"
<${process.env.EMAIL_USER}>`,
            to: newUser.email,
            subject: 'Verify your Email',
            html: `<h3>Email Verification</h3><p>Click <a
href="${verificationLink}">here</a> to verify your email.</p>`
        });
           console.log("Verification email sent");
        } catch(err) {
            console.error("Error sending email:",err);
        }

        res.status(201).json({ message: 'User registered. Please
verify your email.' });
    } catch (error) {
        console.error('Registration error:', error);
        res.status(500).json({ message: 'Registration failed.'
});
```

```
    }
};

// Verify Registration
exports.verifyRegistration = async (req, res) => {
    try {
        const { token } = req.query;

        const decoded = jwt.verify(token, JWT_SECRET);
        const user = await User.findById(decoded.userId);

        if (!user) {
            return res.status(400).json({ message: 'Invalid
token' });
        }

        if (user.verified) {
            return res.status(400).json({ message: 'User already
verified' });
        }

        user.verified = true;
        await user.save();

        res.status(200).json({ message: 'Email verified
successfully' });
    } catch (error) {
        console.error('Verification error:', error);
        res.status(500).json({ message: 'Email verification
failed.' });
    }
};

// Login User
exports.loginUser = async (req, res) => {
    try {
        const { email, password } = req.body;

        const user = await User.findOne({ email });
        if (!user) {
            return res.status(400).json({ message: 'Invalid
credentials' });
        }
        if (!user.verified) {
            return res.status(400).json({ message: 'Please verify
your email before logging in' });
        }

        const isPasswordValid = await bcrypt.compare(password,
user.password);
        if (!isPasswordValid) {
            return res.status(400).json({ message: 'Invalid
credentials' });
        }

        const token = jwt.sign({ userId: user._id }, JWT_SECRET,
{ expiresIn: '1h' });

        res.status(200).json({ token });
    } catch (error) {
        console.error('Login error:', error);
        res.status(500).json({ message: 'Login failed.' });
    }
};

// Protected Route
exports.protected = (req, res) => {
    res.status(200).json({ message: 'Protected route accessed
successfully' });
};

exports.generateTokenForUser = async (req, res) => {
    const { userId } = req.params;

    try {
        const token = jwt.sign({ userId },
process.env.JWT_SECRET, { expiresIn: '1h' });
        console.log('Generated Token:', token);

        res.status(200).json({ token });
    } catch (error) {
        console.error('Token Generation Error:', error);
        res.status(500).json({ message: 'Failed to generate
token.' });
    }
```

```
};
```

authMiddleware.js

```
const jwt = require('jsonwebtoken');

const authenticateToken = (req, res, next) => {
  const authHeader = req.headers.authorization;
  if (!authHeader) {
    return res.status(401).json({ message: 'Authorization header
is missing' });
  }

  const token = authHeader.split(' ')[1]; // Bearer <token>
  if (!token) {
    return res.status(401).json({ message: 'Token is missing' });
  }

  jwt.verify(token, process.env.JWT_SECRET, (err, user) => {
    if (err) {
      console.log(err)
      return res.status(403).json({ message: 'Invalid token' });
    }
    req.user = user; //  Store the user data from the token in
the request
    next(); //  Pass control to the next middleware or route
handler
  });
};

module.exports = authenticateToken;
```

User.js

```
const mongoose = require('mongoose');

const userSchema = new mongoose.Schema({
    email: { type: String, required: true, unique: true },
    password: { type: String, required: true },
    verified: { type: Boolean, default: false },
    verificationToken: {type:String},
    tokenExpiry: Date
});

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

[authRoutes.js]

```
const express = require('express');
const { registerUser, verifyRegistration, loginUser, protected,
generateTokenForUser } = require('../controllers/authController');
// const { registerUser, loginUser } =
require('../controllers/authController');
const authMiddleware = require('../middlewares/authMiddleware');

const router = express.Router();

router.post('/register', registerUser);
router.get('/verify-registration', verifyRegistration);
router.post('/login', loginUser);
router.get('/protected', authMiddleware, protected);
router.get('/generate-token/:userId', generateTokenForUser);

module.exports = router;
```

[server.js]

```
require('dotenv').config();
const express = require('express');
const mongoose = require('mongoose');
// const dotenv = require('dotenv');
const authRoutes = require('./routes/authRoutes'); // Assuming you have
this

// dotenv.config();

const app = express();
const PORT = process.env.PORT || 5001;
```

```
// app.listen(PORT, () => console.log(`Server running on port
${PORT}`));

// Connect to MongoDB
mongoose.connect(process.env.MONGODB_URI, {
  useNewUrlParser: true,
  useUnifiedTopology: true,
}).then(() => {
  console.log('Connected to MongoDB');
}).catch(err => {
  console.error('Error connecting to MongoDB:', err);
});

const user = require('./models/User');

// Middleware
app.use(express.json());

// Routes
app.use('/api/auth', authRoutes);

// *** ADD THIS ROUTE HANDLER ***
app.get('/', (req, res) => {
  res.send('Welcome to the Secure Backend API!'); // Or any other
response
});

app.listen(PORT, () => {
  console.log(`Server is running on port ${PORT}`);
});

const jwt = require('jsonwebtoken');

app.post('/api/login', (req, res) => {
  const user = { id: 1, username: 'testuser' };  // Simulated user data
  const token = jwt.sign(user, process.env.JWT_SECRET, { expiresIn:
'1h' });
  res.json({ token });
});
```

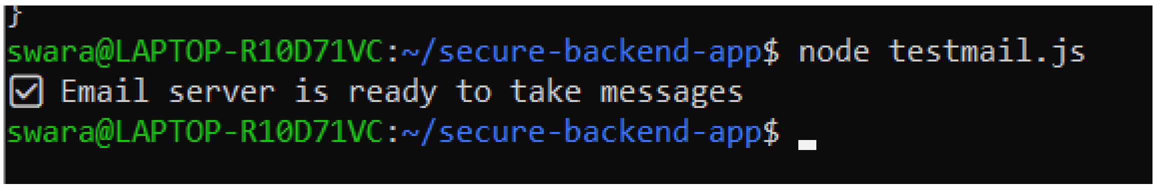
7. JWT Authentication Mechanism:

```
secure-backend-app@1.0.0 /home/swara/secure-backend-app
├── bcrypt@5.1.1
├── dotenv@16.5.0
├── express@4.21.2
├── jsonwebtoken@9.0.2
├── mongoose@8.14.2
├── nodemailer@6.10.1
└── nodemon@3.1.10
```

This is the installed dependencies on the directory by running the command- **npm list**

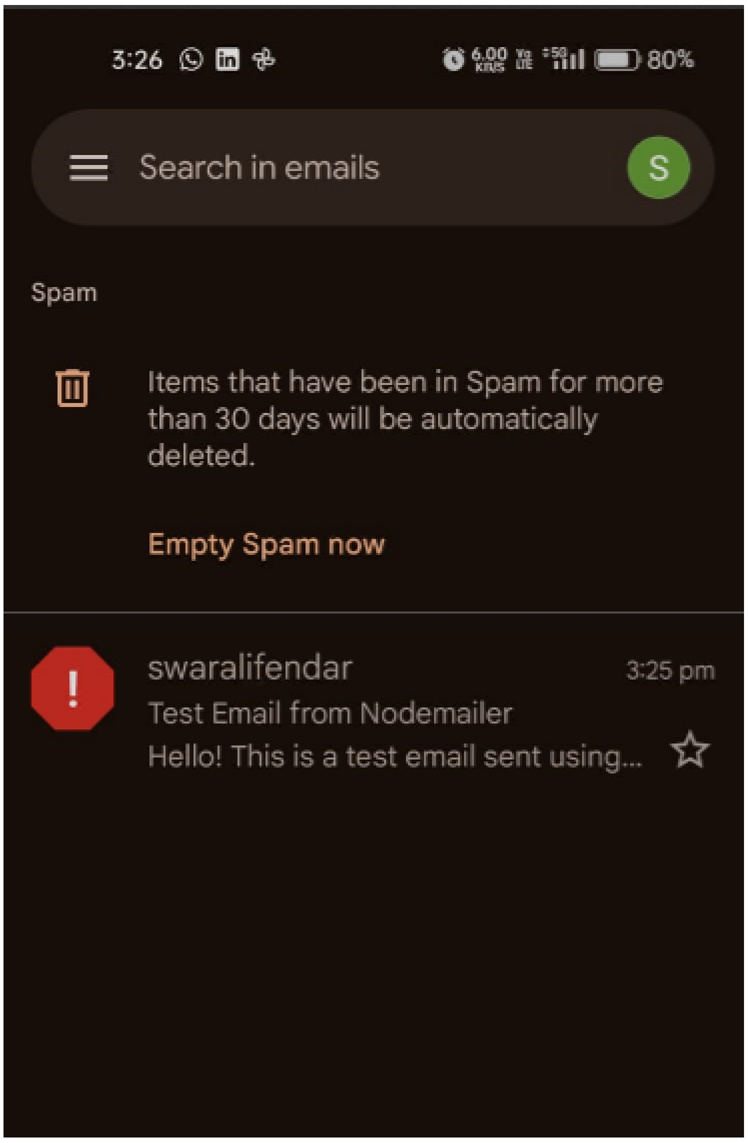
As, we will require jsonwebtoken for the JWT Authentication

8. I have created a test file to check the nodemailer so I have run it and the output is as follows-



```
}
swara@LAPTOP-R10D71VC:~/secure-backend-app$ node testmail.js
☑ Email server is ready to take messages
swara@LAPTOP-R10D71VC:~/secure-backend-app$ 
```

9. The mail is successfully sent to the recipient-



3:26

80%

Search in emails

S

Spam

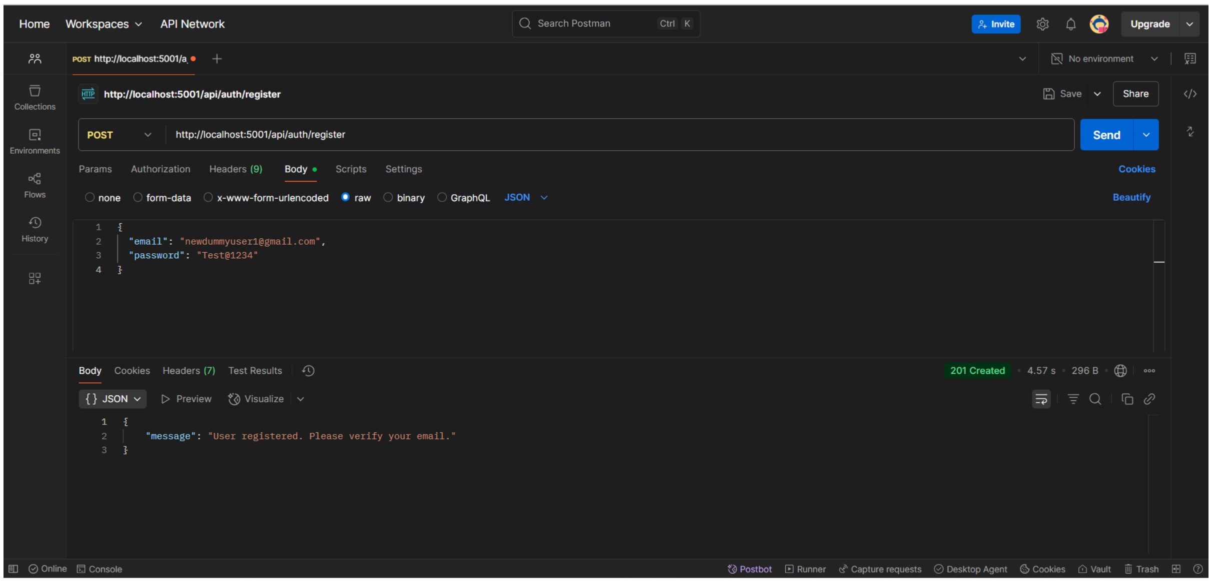
Items that have been in Spam for more than 30 days will be automatically deleted.

Empty Spam now

swaralifendar 3:25 pm

Test Email from Nodemailer

Hello! This is a test email sent using...



POST http://localhost:5001/api/auth/register

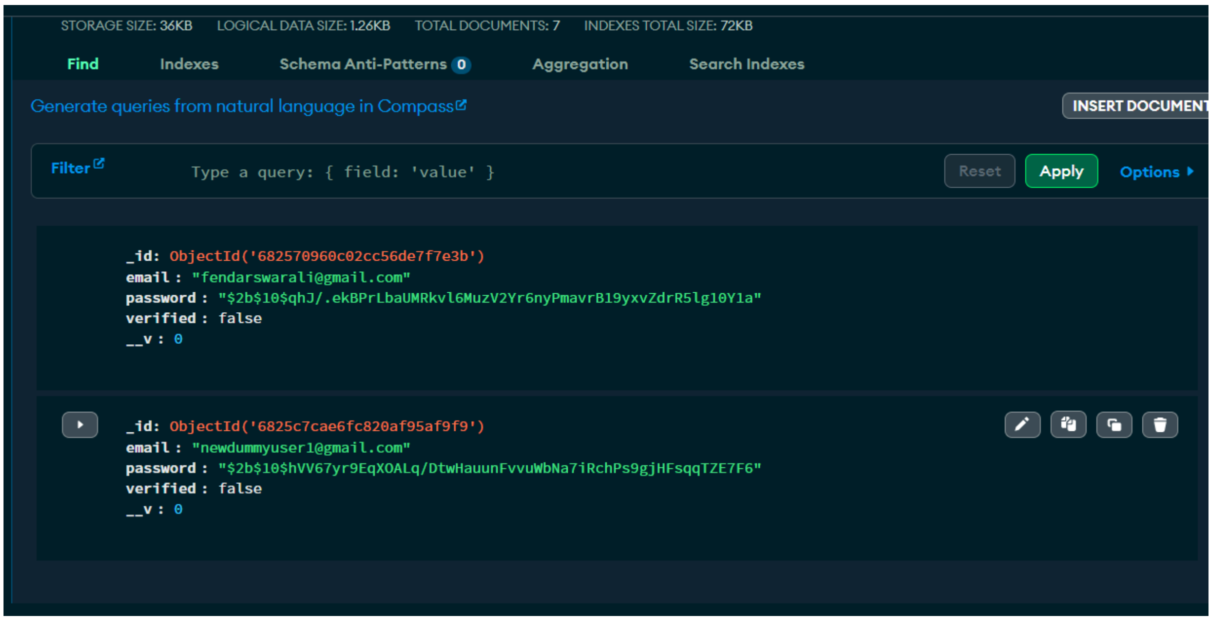
Params | Authorization | Headers (9) | Body | Scripts | Settings

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

```
{
  "email": "newdummyuser1@gmail.com",
  "password": "Test@1234"
}
```

Body | Cookies | Headers (7) | Test Results — 201 Created · 4.57 s · 296 B

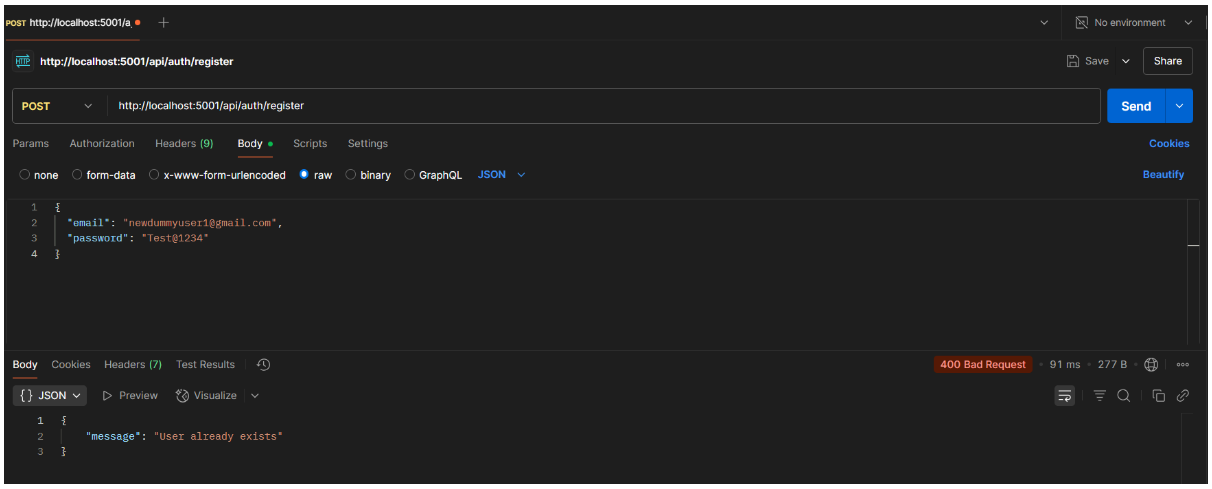
```
{
    "message": "User registered. Please verify your email."
}
```



STORAGE SIZE: 36KB LOGICAL DATA SIZE: 1.26KB TOTAL DOCUMENTS: 7 INDEXES TOTAL SIZE: 72KB

Type a query: { field: 'value' }

```
_id: ObjectId('682570960c02cc56de7f7e3b')
email : "fendarswarali@gmail.com"
password : "$2b$10$qhJ/.ekBPrLbaUMRkvl6MuzV2Yr6nyPmavrB19yxvZdrR5lg10Y1a"
verified : false
__v : 0

_id: ObjectId('6825c7cae6fc820af95af9f9')
email : "newdummyuser1@gmail.com"
password : "$2b$10$hVV67yr9EqXOALq/DtwHauunFvvuWbNa7iRchPs9gjHFsqqTZE7F6"
verified : false
__v : 0
```



POST http://localhost:5001/api/auth/register

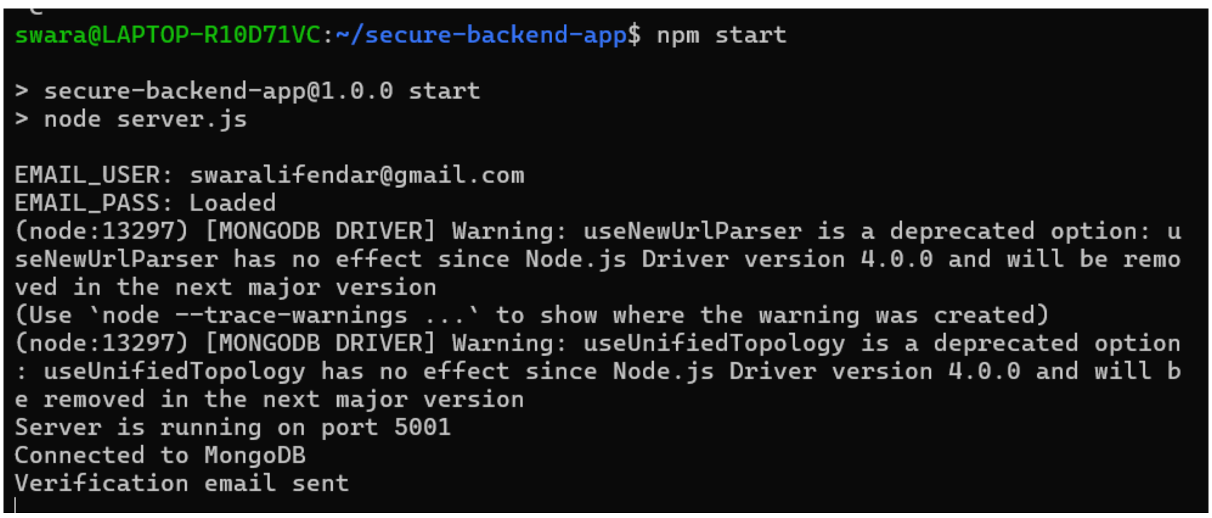
Params | Authorization | Headers (9) | Body | Scripts | Settings

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

```
{
  "email": "newdummyuser1@gmail.com",
  "password": "Test@1234"
}
```

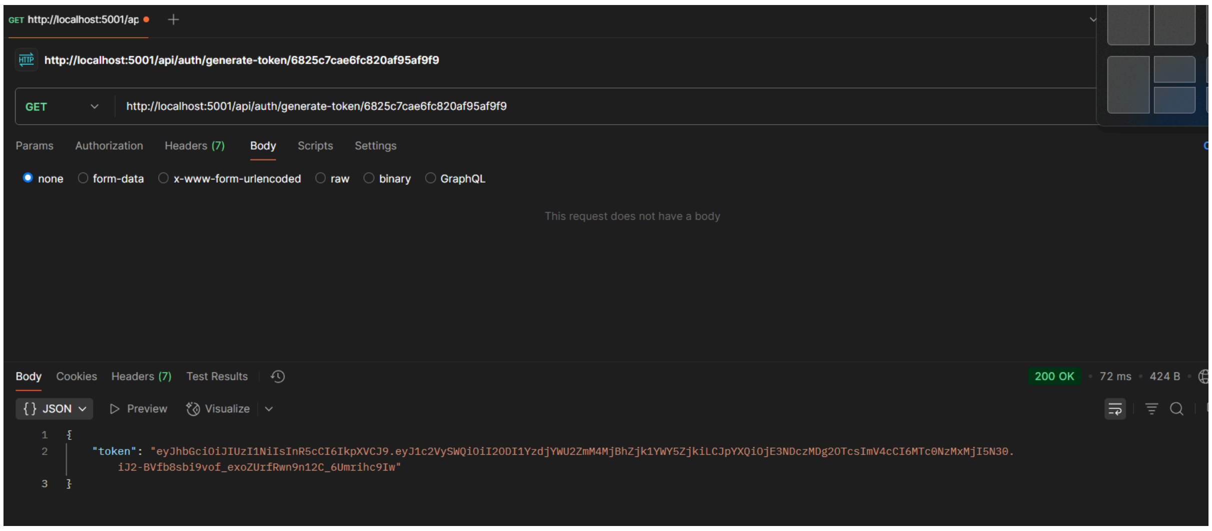
Body | Cookies | Headers (7) | Test Results — 400 Bad Request · 91 ms · 277 B

```
{
    "message": "User already exists"
}
```



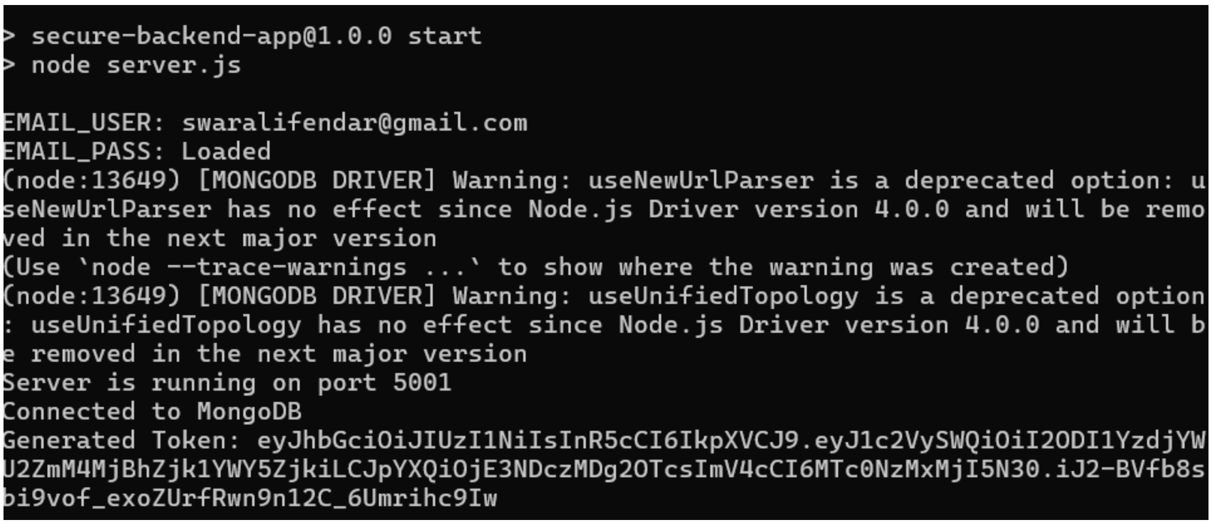
```
swara@LAPTOP-R10D71VC:~/secure-backend-app$ npm start

> secure-backend-app@1.0.0 start
> node server.js

EMAIL_USER: swaralifendar@gmail.com
EMAIL_PASS: Loaded
(node:13297) [MONGODB DRIVER] Warning: useNewUrlParser is a deprecated option: u
seNewUrlParser has no effect since Node.js Driver version 4.0.0 and will be remo
ved in the next major version
(Use `node --trace-warnings ...` to show where the warning was created)
(node:13297) [MONGODB DRIVER] Warning: useUnifiedTopology is a deprecated option
: useUnifiedTopology has no effect since Node.js Driver version 4.0.0 and will b
e removed in the next major version
Server is running on port 5001
Connected to MongoDB
Verification email sent
```

Token generation-



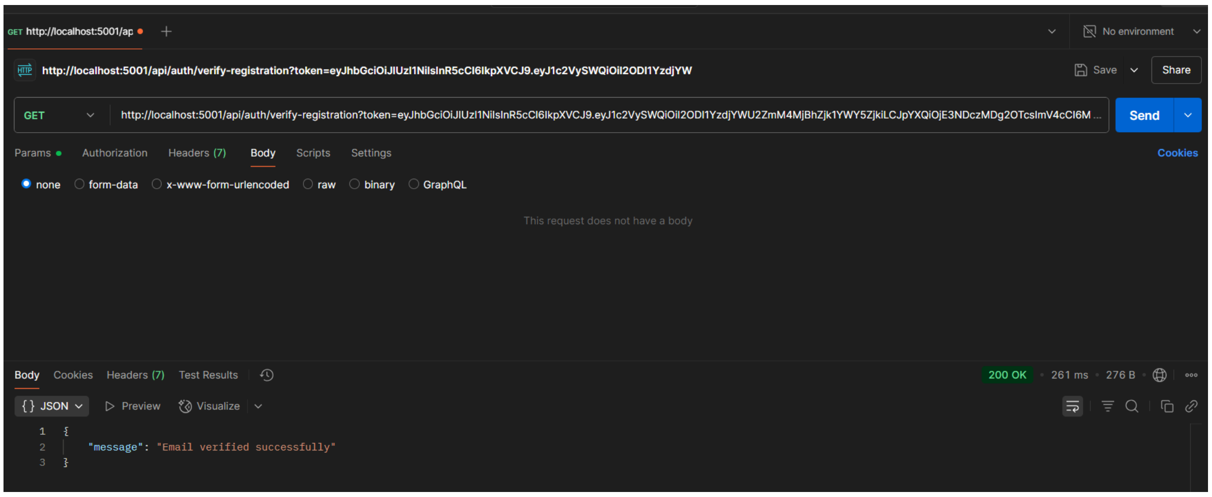
GET http://localhost:5001/api/auth/generate-token/6825c7cae6fc820af95af9f9

```
{
    "token": "eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJ1c2VySWQiOiI2ODI1YzdjYWU2ZmM4MjBhZjk1YWY5ZjkiLCJpYXQiOjE3NDczMDg2OTcsImV4cCI6MTc0NzMxMjI5N30.
        iJ2-BVfb8sbi9vof_exoZUrfRwn9n12C_6Umrihc9Iw"
}
```



```
> secure-backend-app@1.0.0 start
> node server.js

EMAIL_USER: swaralifendar@gmail.com
EMAIL_PASS: Loaded
(node:13649) [MONGODB DRIVER] Warning: useNewUrlParser is a deprecated option: u
seNewUrlParser has no effect since Node.js Driver version 4.0.0 and will be remo
ved in the next major version
(Use `node --trace-warnings ...` to show where the warning was created)
(node:13649) [MONGODB DRIVER] Warning: useUnifiedTopology is a deprecated option
: useUnifiedTopology has no effect since Node.js Driver version 4.0.0 and will b
e removed in the next major version
Server is running on port 5001
Connected to MongoDB
Generated Token: eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJ1c2VySWQiOiI2ODI1YzdjYW
U2ZmM4MjBhZjk1YWY5ZjkiLCJpYXQiOjE3NDczMDg2OTcsImV4cCI6MTc0NzMxMjI5N30.iJ2-BVfb8s
bi9vof_exoZUrfRwn9n12C_6Umrihc9Iw
```

Verification by token-

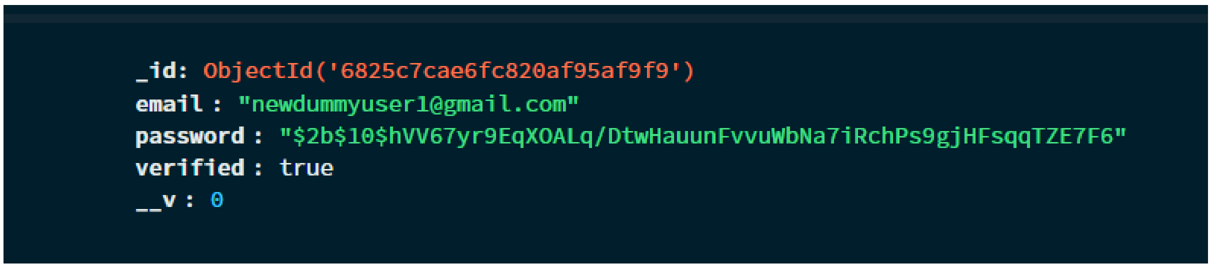


GET http://localhost:5001/api/auth/verify-registration?token=eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJ1c2VySWQiOiI2ODI1YzdjYWU2ZmM4MjBhZjk1YWY5ZjkiLCJpYXQiOjE3NDczMDg2OTcsImV4cCI6M ...

Body: none — This request does not have a body

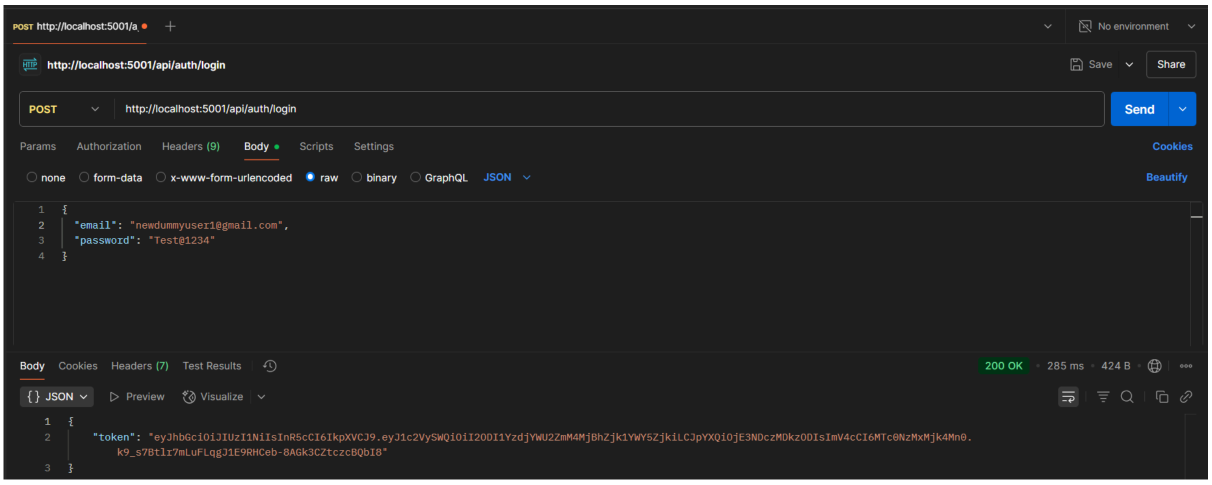
200 OK · 261 ms · 276 B

```
{
    "message": "Email verified successfully"
}
```



```
_id: ObjectId('6825c7cae6fc820af95af9f9')
email : "newdummyuser1@gmail.com"
password : "$2b$10$hVV67yr9EqXOALq/DtwHauunFvvuWbNa7iRchPs9gjHFsqqTZE7F6"
verified : true
__v : 0
```

Login -



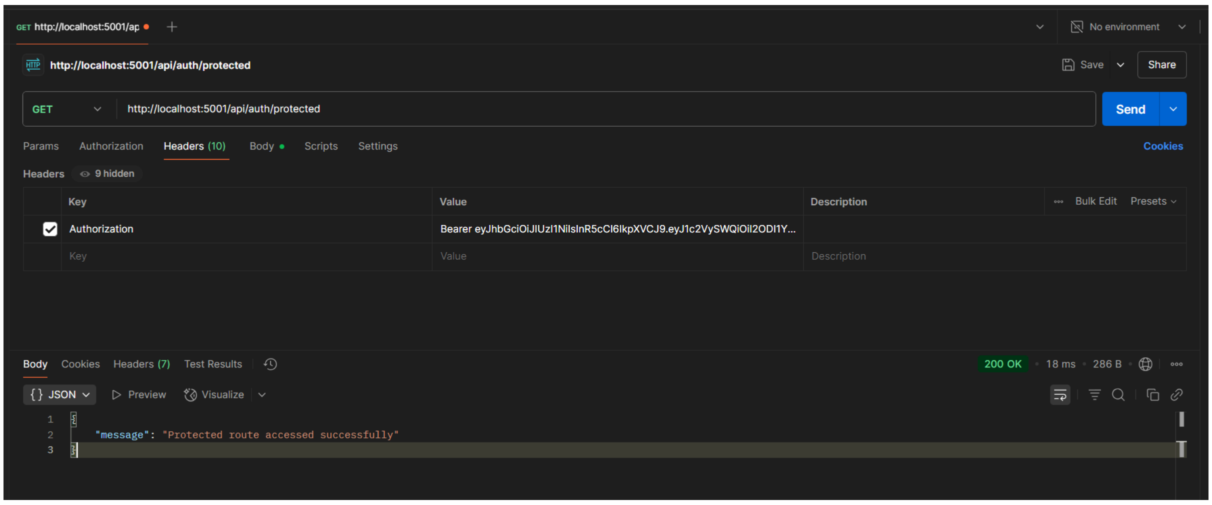
POST http://localhost:5001/api/auth/login

```
{
    "email": "newdummyuser1@gmail.com",
    "password": "Test@1234"
}
```

200 OK · 285 ms · 424 B

```
{
    "token": "eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJ1c2VySWQiOiI2ODI1YzdjYWU2ZmM4MjBhZjk1YWY5ZjkiLCJpYXQiOjE3NDczMDkzODIsImV4cCI6MTc0NzMxMjk4Mn0.
        k9_s7Btlr7mLuFLqgJ1E9RHCeb-8AGk3CZtczcBQbI8"
}
```

Protected - Route -


GET http://localhost:5001/api/auth/protected

	Key	Value	Description
✓	Authorization	Bearer eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJ1c2VySWQiOiI2ODI1Y...	

200 OK · 18 ms · 286 B

```
{
    "message": "Protected route accessed successfully"
}
```

This is the summary-

[POST] /api/auth/register ➔ User created

|
▼

[GET] /api/auth/verify-registration?token=... ➔ User verified

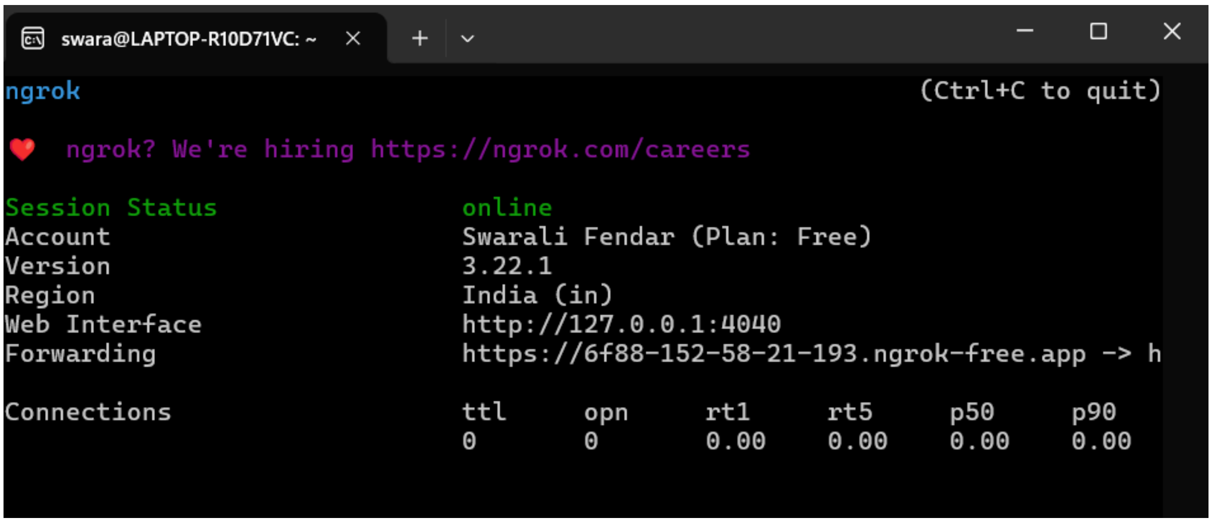
|
▼

[POST] /api/auth/login ➔ Get JWT Token

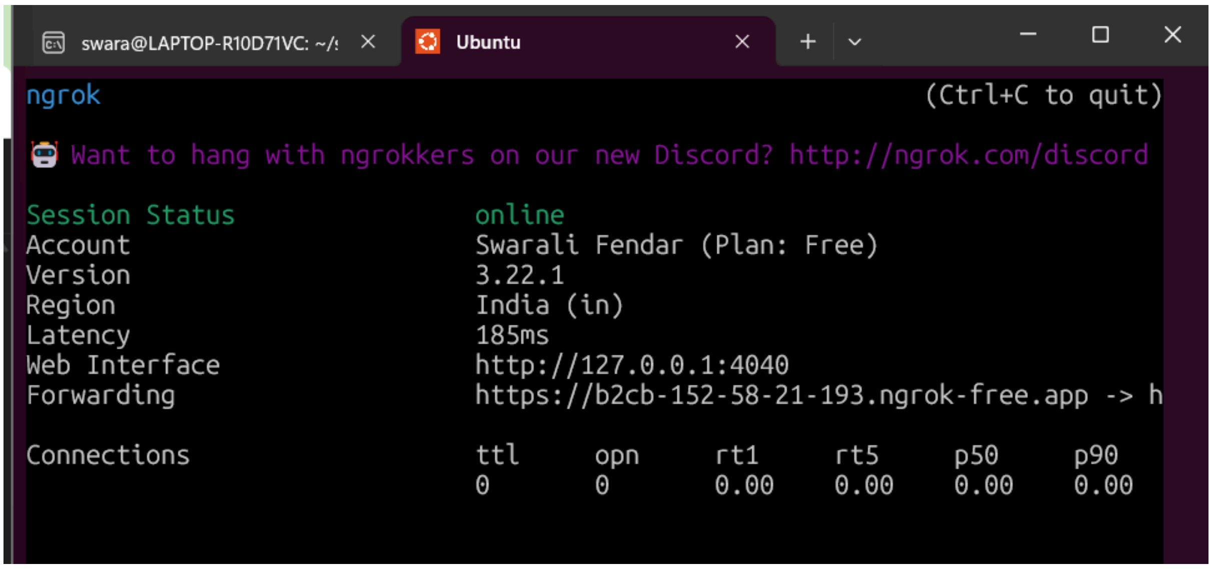
|
▼

[GET] /api/auth/protected with Bearer Token ➔ Access granted

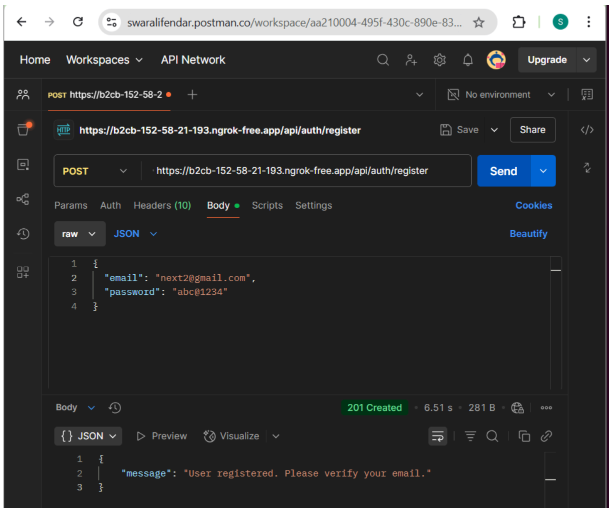
Ngrok installation and configuring the authtoken -



```
ngrok                                                       (Ctrl+C to quit)

❤️ ngrok? We're hiring https://ngrok.com/careers

Session Status                online
Account                       Swarali Fendar (Plan: Free)
Version                       3.22.1
Region                        India (in)
Web Interface                 http://127.0.0.1:4040
Forwarding                    https://6f88-152-58-21-193.ngrok-free.app -> h

Connections                   ttl     opn     rt1     rt5     p50     p90
                              0       0       0.00    0.00    0.00    0.00
```



```
ngrok                                                       (Ctrl+C to quit)

🤖 Want to hang with ngrokkers on our new Discord? http://ngrok.com/discord

Session Status                online
Account                       Swarali Fendar (Plan: Free)
Version                       3.22.1
Region                        India (in)
Latency                       185ms
Web Interface                 http://127.0.0.1:4040
Forwarding                    https://b2cb-152-58-21-193.ngrok-free.app -> h

Connections                   ttl     opn     rt1     rt5     p50     p90
                              0       0       0.00    0.00    0.00    0.00
```



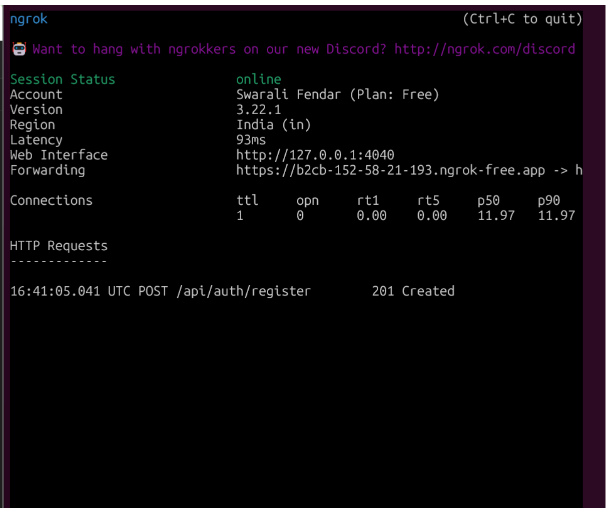
https://b2cb-152-58-21-193.ngrok-free.app/api/auth/register

POST https://b2cb-152-58-21-193.ngrok-free.app/api/auth/register

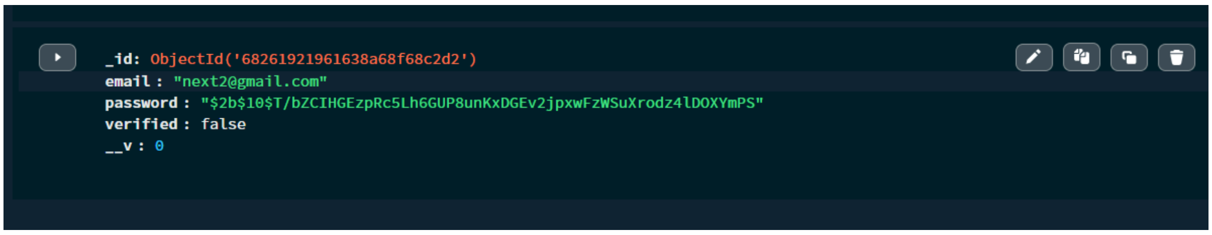
```
{
  "email": "next2@gmail.com",
  "password": "abc@1234"
}
```

201 Created · 6.51 s · 281 B

```
{
    "message": "User registered. Please verify your email."
}
```



```
ngrok                                                       (Ctrl+C to quit)

🤖 Want to hang with ngrokkers on our new Discord? http://ngrok.com/discord

Session Status                online
Account                       Swarali Fendar (Plan: Free)
Version                       3.22.1
Region                        India (in)
Latency                       93ms
Web Interface                 http://127.0.0.1:4040
Forwarding                    https://b2cb-152-58-21-193.ngrok-free.app -> h

Connections                   ttl     opn     rt1     rt5     p50     p90
                              1       0       0.00    0.00    11.97   11.97

HTTP Requests
-------------

16:41:05.041 UTC POST /api/auth/register        201 Created
```



```
_id: ObjectId('68261921961638a68f68c2d2')
email : "next2@gmail.com"
password : "$2b$10$T/bZCIHGEzpRc5Lh6GUP8unKxDGEv2jpxwFzWSuXrodz4lDOXYmPS"
verified : false
__v : 0
```

Phase : 1 - Email-Based Password Reset Flow
Phase : 2 - Rate Limiting with express-rate-limit
Phase : 3 - CLI Testing Tool
Phase : 4 - Unit Testing with Jest or Mocha
Phase : 5 - Demonstration using Postman & CLI Tool

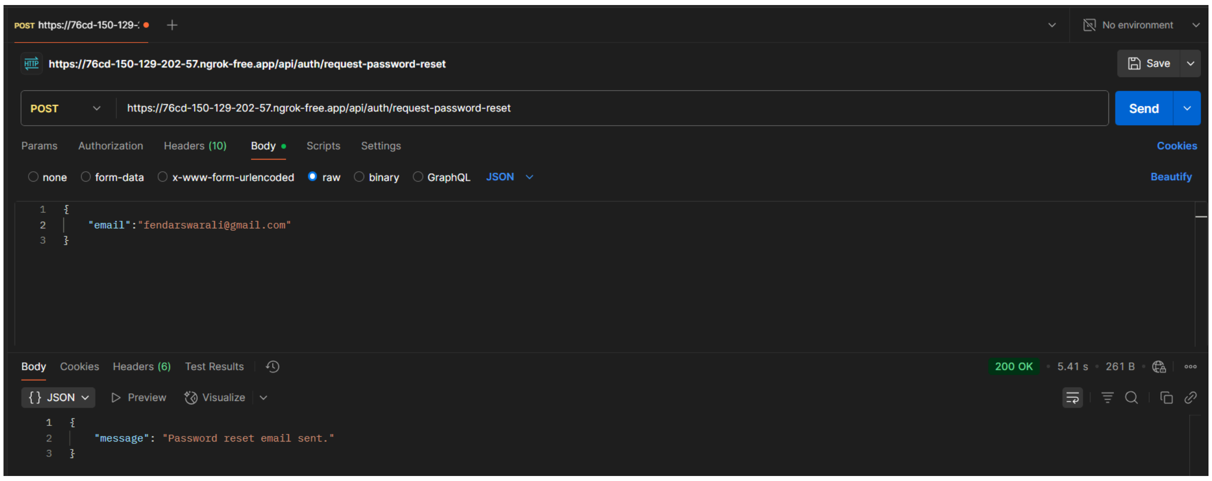
Folder Structure -

```
/secure-backend-app
│
├── controllers/
│   ├── authController.js
│   └── rateLimitController.js (optional)
│
├── middlewares/
│   ├── authMiddleware.js
│   └── rateLimitMiddleware.js
│
├── models/
│   ├── User.js
│   └── LoginAttempt.js (if storing in DB)
│
├── routes/
│   └── authRoutes.js
│
├── cli-tool.js    <-- CLI script for rate limit testing
├── tests/         <-- For Jest & Supertest tests
│   └── auth.test.js
│
├── .env
└── server.js
```

Flow -

Feature	**How to Demo**
Password Reset	Use Postman ➔ Request reset ➔ Get email ➔ Use token to reset password
Rate Limiting	Use Postman ➔ Try rapid requests ➔ See 429 error
Block Multiple Logins	Login ➔ Try new login ➔ Show token invalidation
CLI tool	Run CLI ➔ Show failed login flooding simulation
Unit Tests	`npm test` ➔ Show test pass/fail status

Register-password

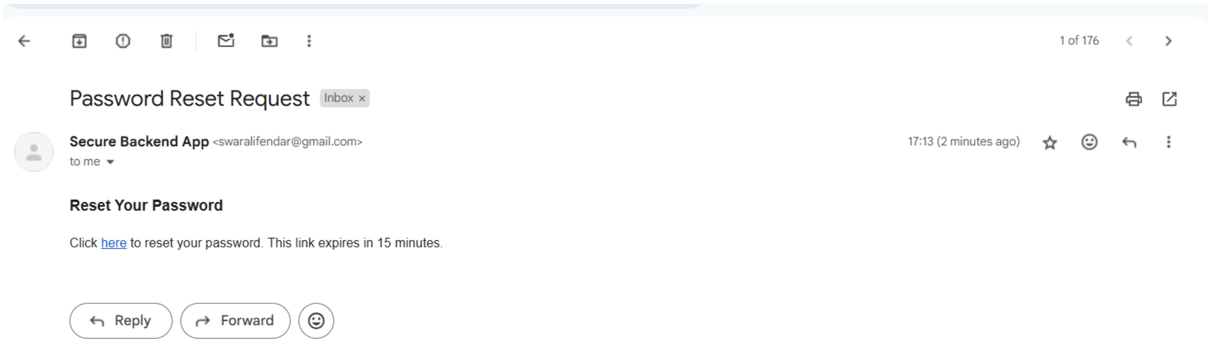

POST https://76cd-150-129-202-57.ngrok-free.app/api/auth/request-password-reset

Body: raw JSON

```
{
    "email":"fendarswarali@gmail.com"
}
```

200 OK · 5.41 s · 261 B

```
{
    "message": "Password reset email sent."
}
```

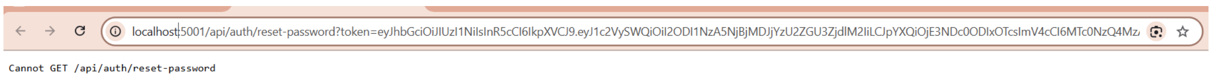

Password Reset Request

Secure Backend App <swaralifendar@gmail.com>
to me

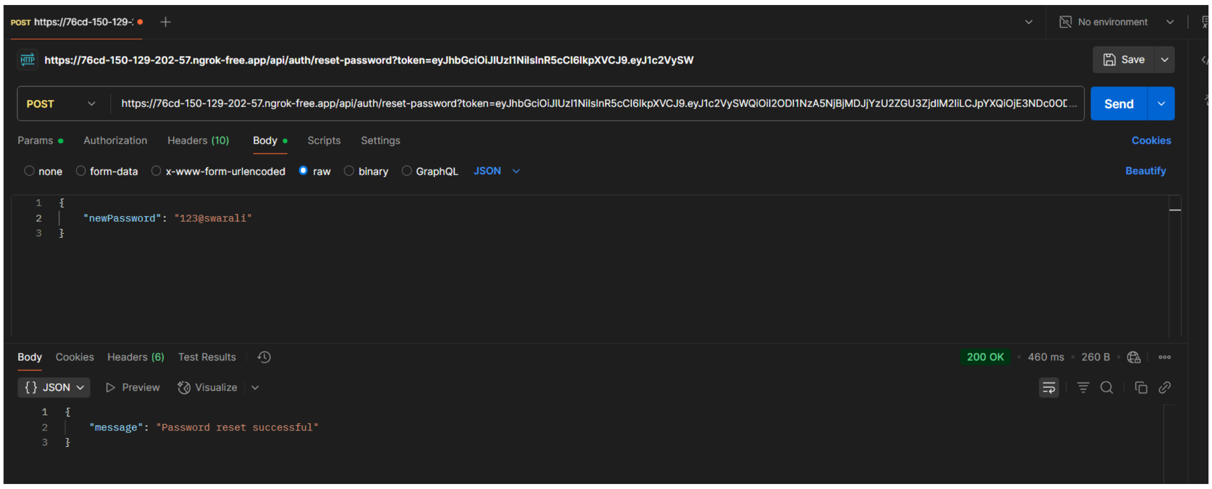
17:13 (2 minutes ago)

Reset Your Password

Click here to reset your password. This link expires in 15 minutes.


localhost:5001/api/auth/reset-password?token=eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJ1c2VySWQiOiI2ODI1NzA5NjBjMDJjYzU2ZGU3ZjdlM2IiLCJpYXQiOjE3NDc0ODIxOTcsImV4cCI6MTc0NzQ4Mz

```
Cannot GET /api/auth/reset-password
```

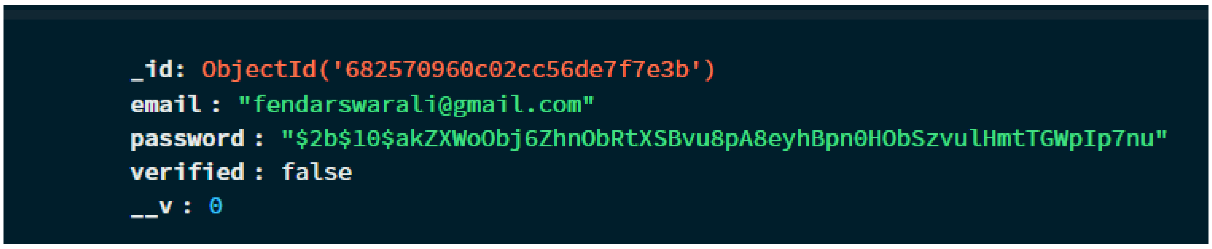

POST https://76cd-150-129-202-57.ngrok-free.app/api/auth/reset-password?token=eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJ1c2VySWQiOiI2ODI1NzA5NjBjMDJjYzU2ZGU3ZjdlM2IiLCJpYXQiOjE3NDc0OD...

Body: raw JSON

```
{
    "newPassword": "123@swarali"
}
```

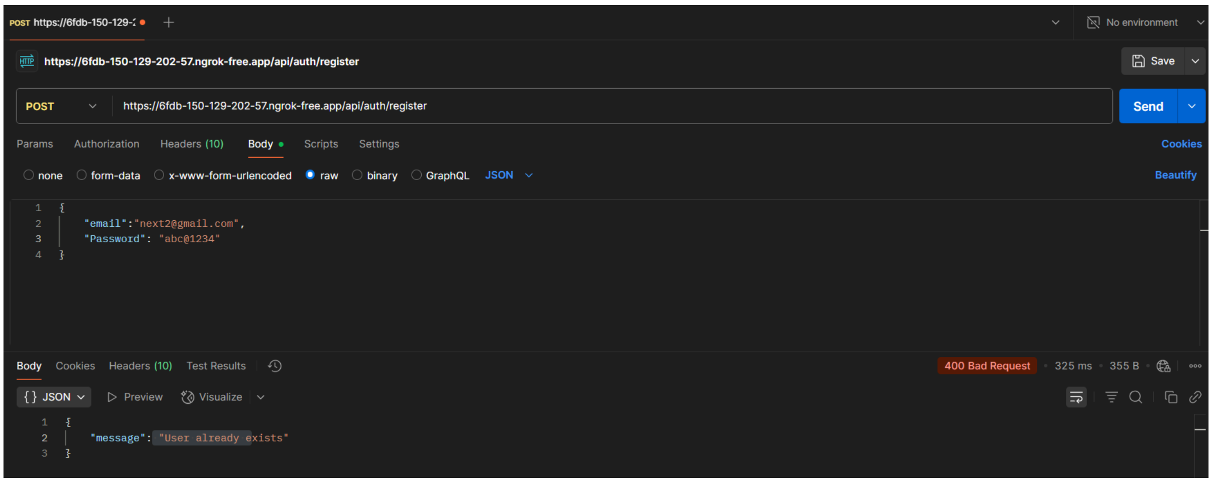
200 OK · 460 ms · 260 B

```
{
    "message": "Password reset successful"
}
```



```
_id: ObjectId('682570960c02cc56de7f7e3b')
email : "fendarswarali@gmail.com"
password : "$2b$10$akZXWoObj6ZhnObRtXSBvu8pA8eyhBpn0HObSzvulHmtTGWpIp7nu"
verified : false
__v : 0
```

Rate - limit :



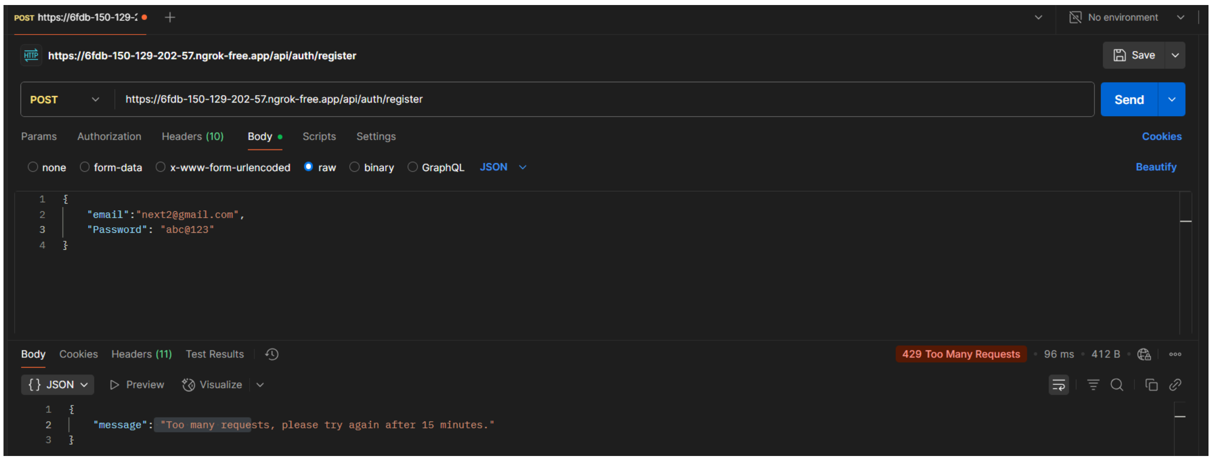
POST https://6fdb-150-129-202-57.ngrok-free.app/api/auth/register

Body: raw JSON

```
{
  "email":"next2@gmail.com",
  "Password": "abc@1234"
}
```

400 Bad Request · 325 ms · 355 B

```
{
  "message": "User already exists"
}
```



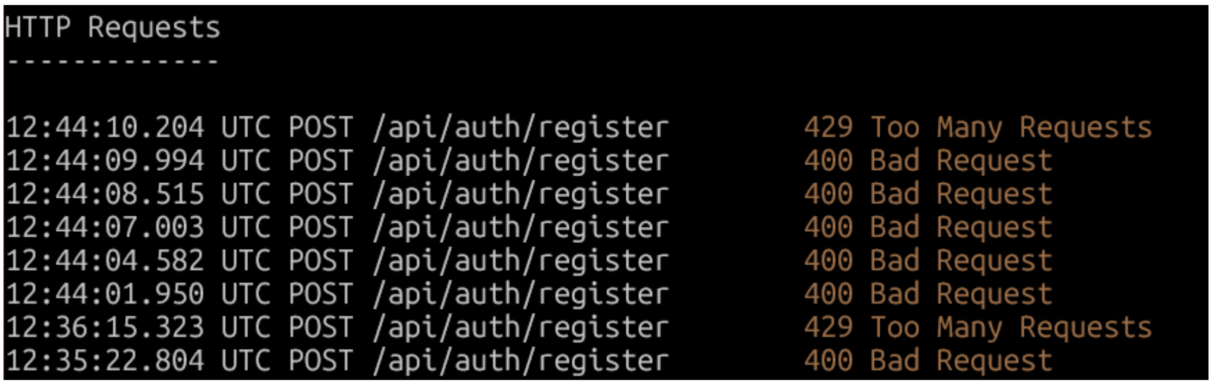
POST https://6fdb-150-129-202-57.ngrok-free.app/api/auth/register

Body: raw JSON

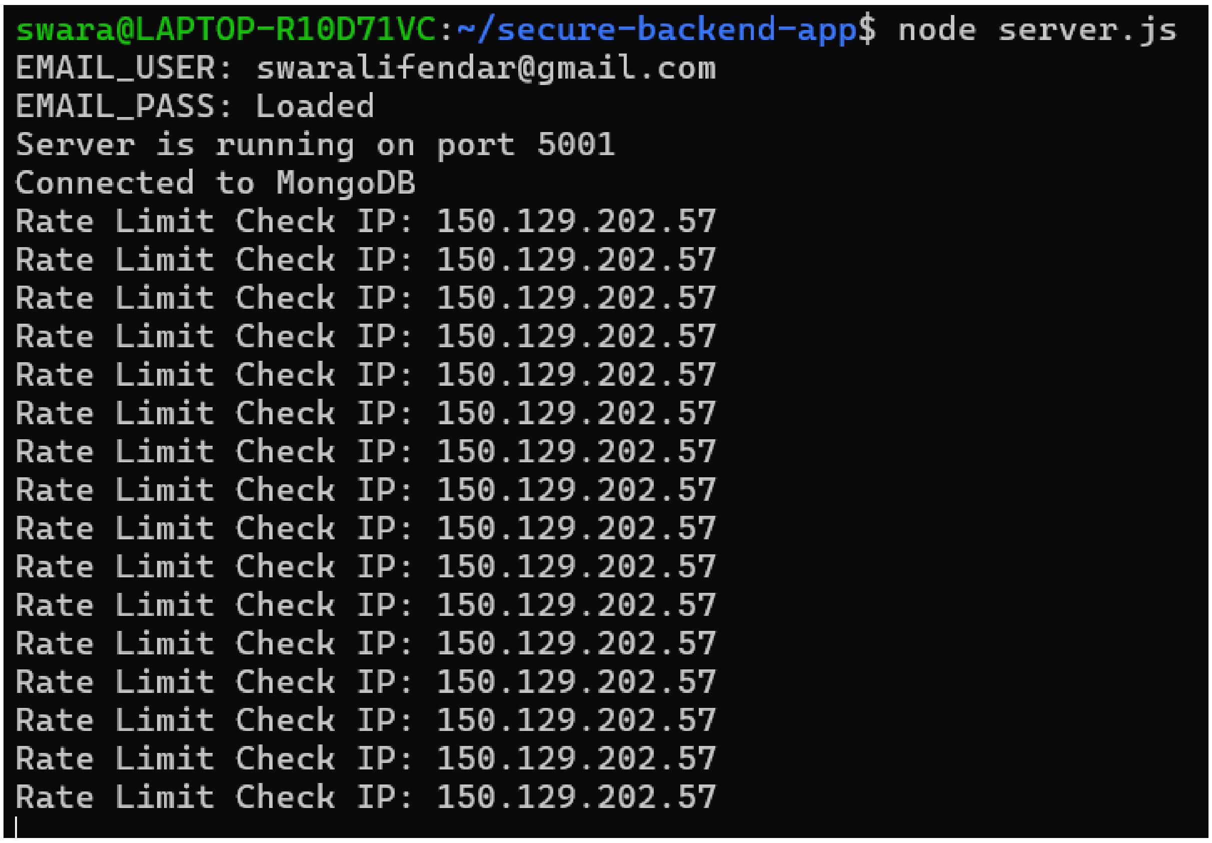
```
{
  "email":"next2@gmail.com",
  "Password": "abc@123"
}
```

429 Too Many Requests · 96 ms · 412 B

```
{
  "message": "Too many requests, please try again after 15 minutes."
}
```

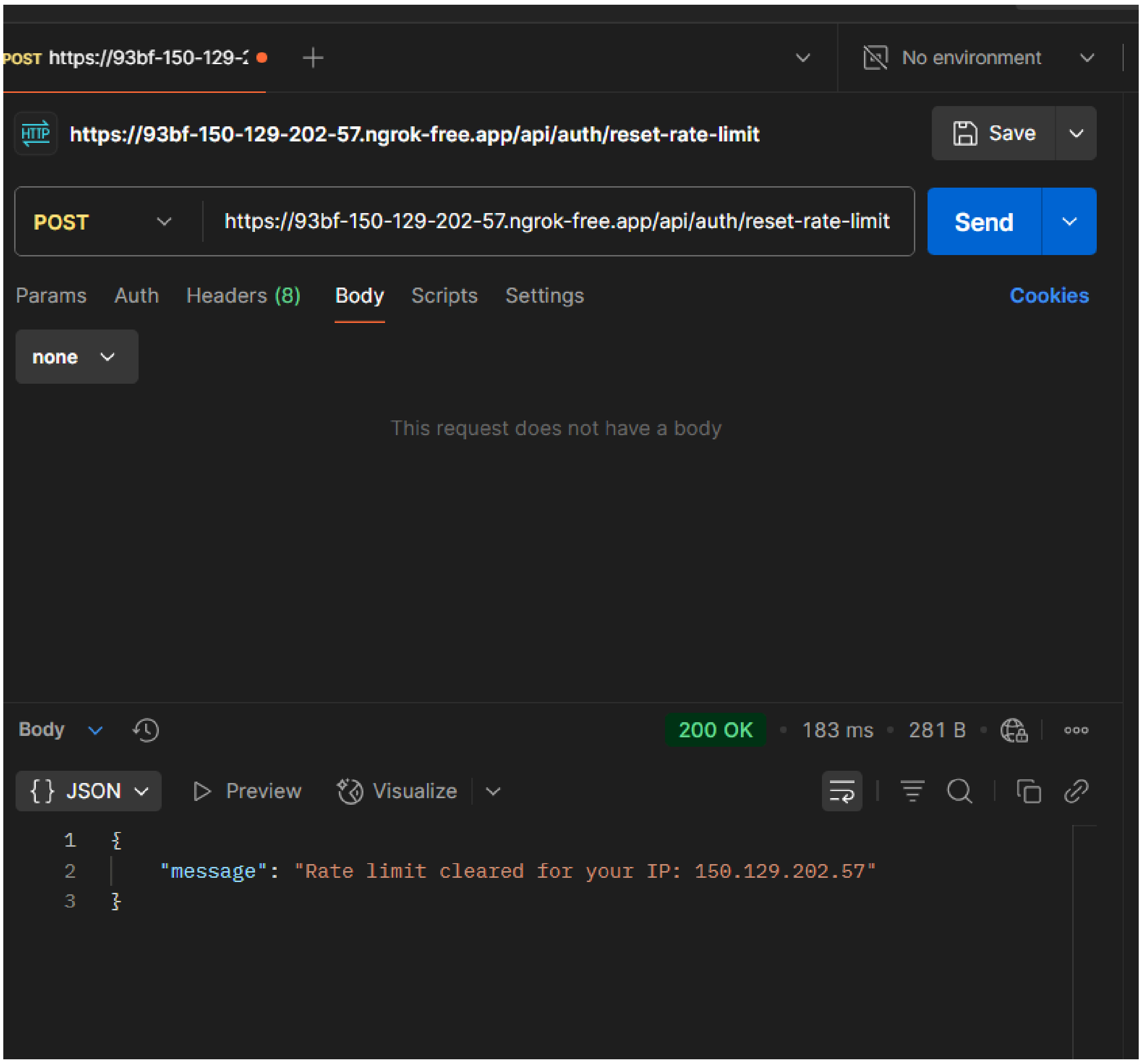


```
HTTP Requests
-------------

12:44:10.204 UTC POST /api/auth/register        429 Too Many Requests
12:44:09.994 UTC POST /api/auth/register        400 Bad Request
12:44:08.515 UTC POST /api/auth/register        400 Bad Request
12:44:07.003 UTC POST /api/auth/register        400 Bad Request
12:44:04.582 UTC POST /api/auth/register        400 Bad Request
12:44:01.950 UTC POST /api/auth/register        400 Bad Request
12:36:15.323 UTC POST /api/auth/register        429 Too Many Requests
12:35:22.804 UTC POST /api/auth/register        400 Bad Request
```



```
swara@LAPTOP-R10D71VC:~/secure-backend-app$ node server.js
EMAIL_USER: swaralifendar@gmail.com
EMAIL_PASS: Loaded
Server is running on port 5001
Connected to MongoDB
Rate Limit Check IP: 150.129.202.57
Rate Limit Check IP: 150.129.202.57
Rate Limit Check IP: 150.129.202.57
Rate Limit Check IP: 150.129.202.57
Rate Limit Check IP: 150.129.202.57
Rate Limit Check IP: 150.129.202.57
Rate Limit Check IP: 150.129.202.57
Rate Limit Check IP: 150.129.202.57
Rate Limit Check IP: 150.129.202.57
Rate Limit Check IP: 150.129.202.57
Rate Limit Check IP: 150.129.202.57
Rate Limit Check IP: 150.129.202.57
Rate Limit Check IP: 150.129.202.57
Rate Limit Check IP: 150.129.202.57
Rate Limit Check IP: 150.129.202.57
Rate Limit Check IP: 150.129.202.57
```

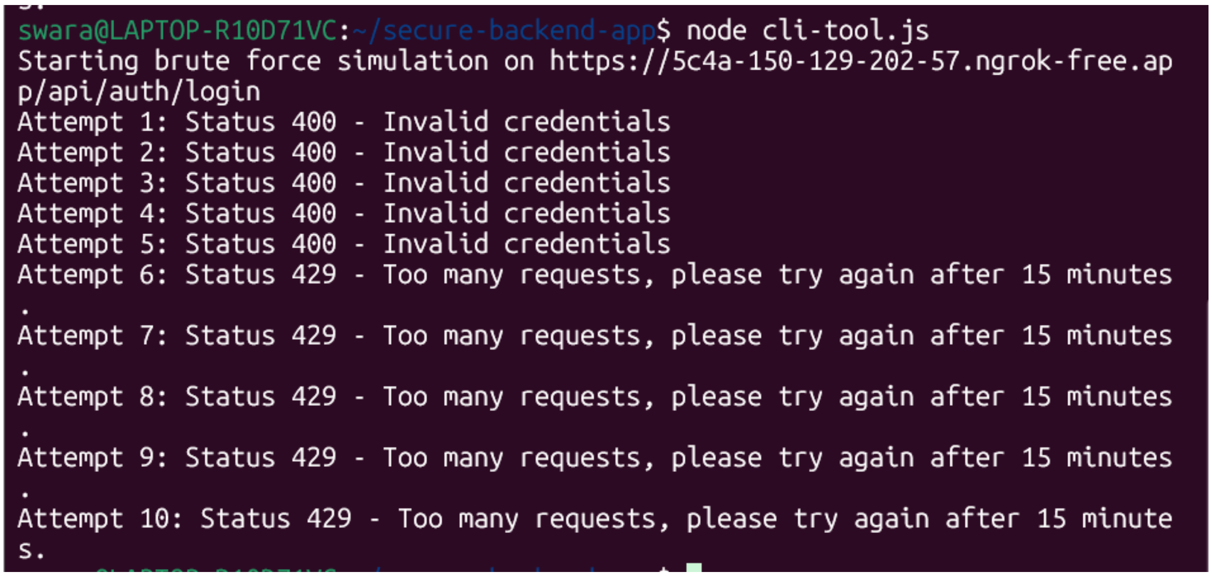
Reset Rate Limit



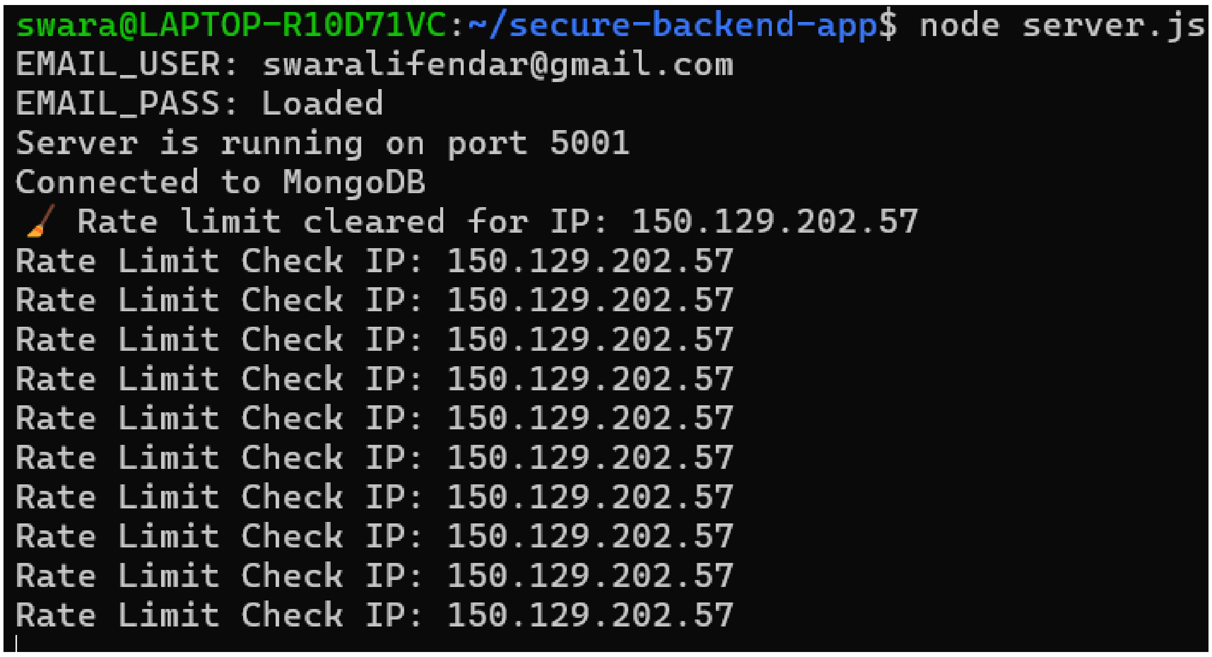
https://93bf-150-129-202-57.ngrok-free.app/api/auth/reset-rate-limit

POST https://93bf-150-129-202-57.ngrok-free.app/api/auth/reset-rate-limit

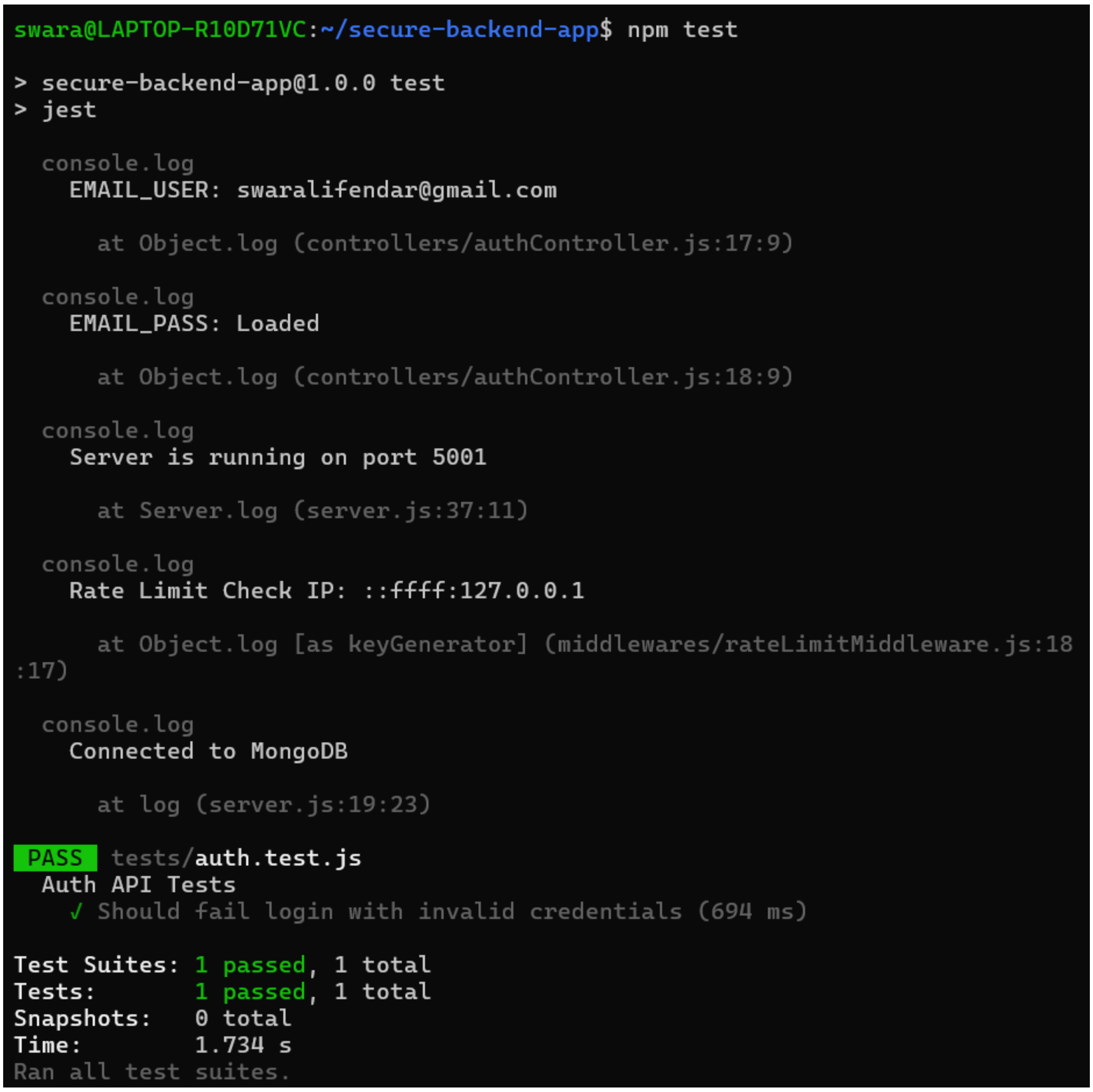
```
{
    "message": "Rate limit cleared for your IP: 150.129.202.57"
}
```



```
swara@LAPTOP-R10D71VC:~/secure-backend-app$ node cli-tool.js
Starting brute force simulation on https://5c4a-150-129-202-57.ngrok-free.ap
p/api/auth/login
Attempt 1: Status 400 - Invalid credentials
Attempt 2: Status 400 - Invalid credentials
Attempt 3: Status 400 - Invalid credentials
Attempt 4: Status 400 - Invalid credentials
Attempt 5: Status 400 - Invalid credentials
Attempt 6: Status 429 - Too many requests, please try again after 15 minutes
.
Attempt 7: Status 429 - Too many requests, please try again after 15 minutes
.
Attempt 8: Status 429 - Too many requests, please try again after 15 minutes
.
Attempt 9: Status 429 - Too many requests, please try again after 15 minutes
.
Attempt 10: Status 429 - Too many requests, please try again after 15 minute
s.
```



```
swara@LAPTOP-R10D71VC:~/secure-backend-app$ node server.js
EMAIL_USER: swaralifendar@gmail.com
EMAIL_PASS: Loaded
Server is running on port 5001
Connected to MongoDB
🧹 Rate limit cleared for IP: 150.129.202.57
Rate Limit Check IP: 150.129.202.57
Rate Limit Check IP: 150.129.202.57
Rate Limit Check IP: 150.129.202.57
Rate Limit Check IP: 150.129.202.57
Rate Limit Check IP: 150.129.202.57
Rate Limit Check IP: 150.129.202.57
Rate Limit Check IP: 150.129.202.57
Rate Limit Check IP: 150.129.202.57
Rate Limit Check IP: 150.129.202.57
Rate Limit Check IP: 150.129.202.57
```



```
swara@LAPTOP-R10D71VC:~/secure-backend-app$ npm test

> secure-backend-app@1.0.0 test
> jest

  console.log
    EMAIL_USER: swaralifendar@gmail.com

      at Object.log (controllers/authController.js:17:9)

  console.log
    EMAIL_PASS: Loaded

      at Object.log (controllers/authController.js:18:9)

  console.log
    Server is running on port 5001

      at Server.log (server.js:37:11)

  console.log
    Rate Limit Check IP: ::ffff:127.0.0.1

      at Object.log [as keyGenerator] (middlewares/rateLimitMiddleware.js:18
:17)

  console.log
    Connected to MongoDB

      at log (server.js:19:23)

 PASS  tests/auth.test.js
  Auth API Tests
    √ Should fail login with invalid credentials (694 ms)

Test Suites: 1 passed, 1 total
Tests:       1 passed, 1 total
Snapshots:   0 total
Time:        1.734 s
Ran all test suites.
```