Implement the JWT base authentication and role base authorization

Muhmmad Yaqoob
2021-CS-118

Introduction

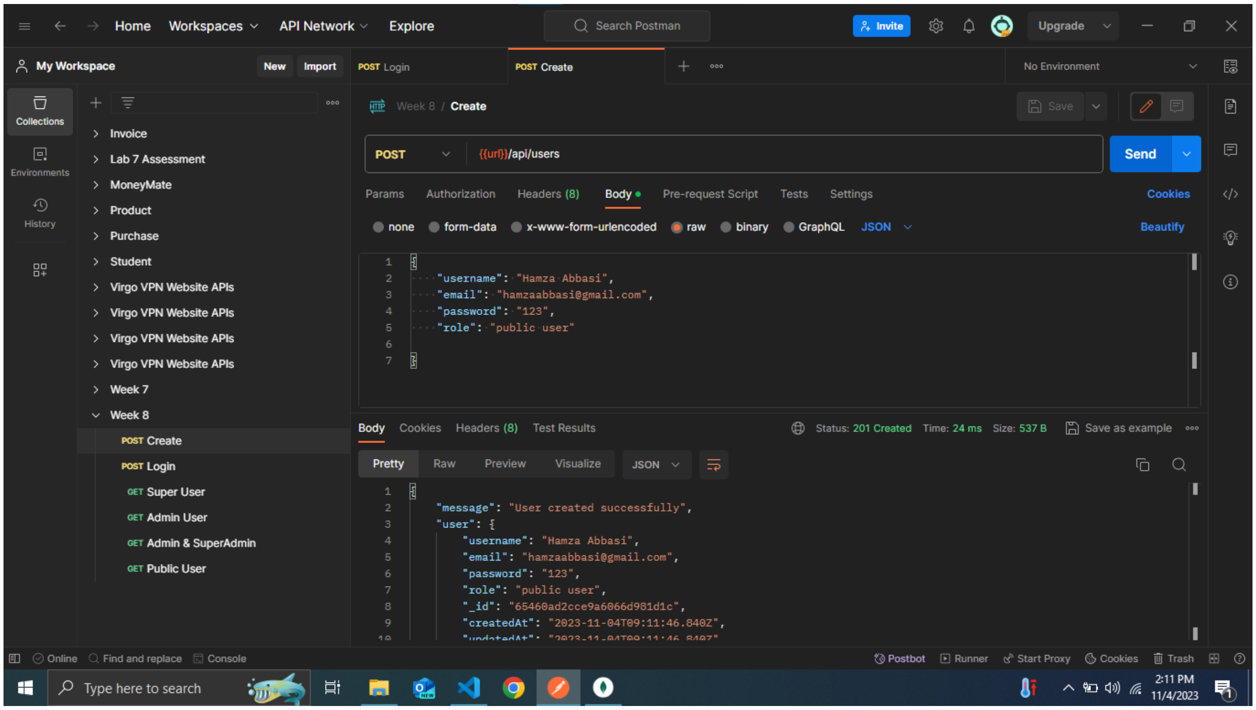
In today's digital age, securing web applications is of paramount importance. One of the key aspects of application security is user authentication and authorization. JSON Web Tokens (JWTs) provide a robust and efficient way to authenticate users and grant them specific permissions based on their roles. In this report, I will demonstrate how to build a web application with JWT-based authentication and role-based authorization.

Directory Structure

The project's directory structure is as follows:

```
lab_8_home_task"/

 controllers/
    userController.js

 routes/
    userRoutes.js

 middleware/
    authMiddleware.js

 models/
    User.js

 config/
    config.js

 utils/
    db.js
    auth_middleware.js
    roles_middleware.js

 app.js
 server.js
 package.json
```

API Operations



POST {{url}}/api/users

Send

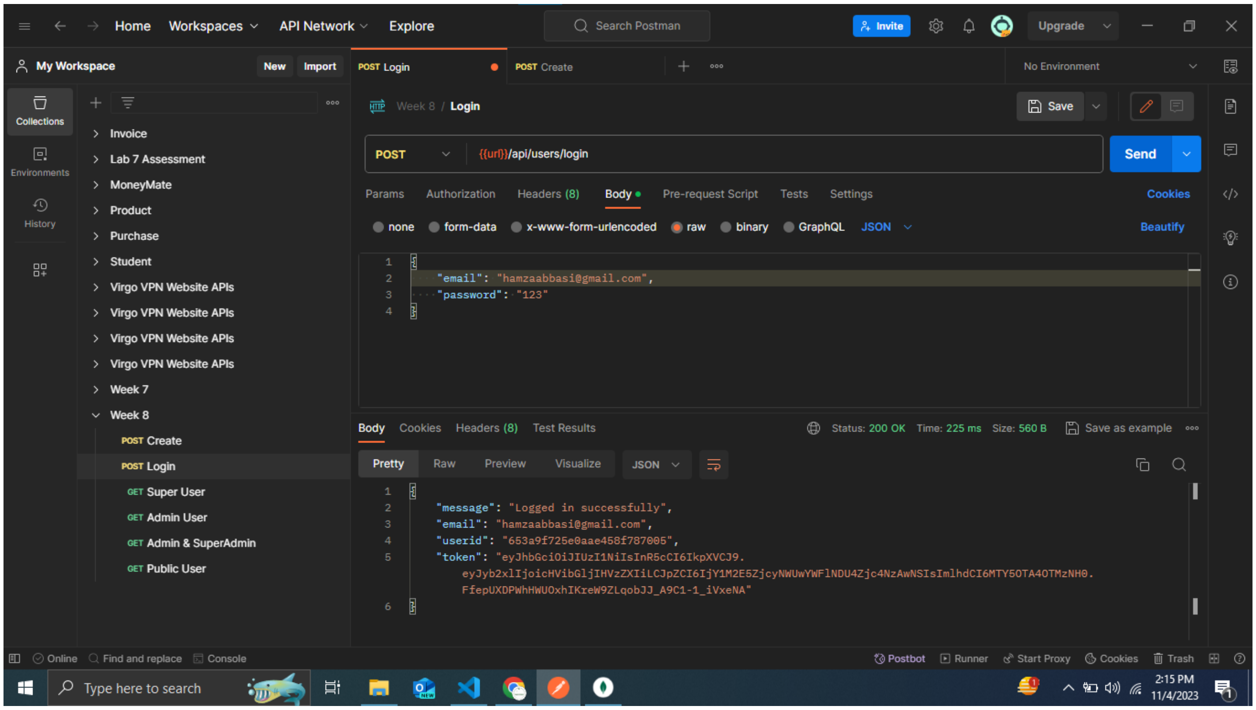
○ none ○ form-data ○ x-www-form-urlencoded ● raw ○ binary ○ GraphQL JSON

```
{
    "username": "Hamza Abbasi",
    "email": "hamzaabbasi@gmail.com",
    "password": "123",
    "role": "public user"

}
```

Status: 201 Created Time: 24 ms Size: 537 B

```
{
    "message": "User created successfully",
    "user": {
        "username": "Hamza Abbasi",
        "email": "hamzaabbasi@gmail.com",
        "password": "123",
        "role": "public user",
        "_id": "65460ad2cce9a6066d981d1c",
        "createdAt": "2023-11-04T09:11:46.840Z",
        "updatedAt": "2023-11-04T09:11:46.840Z"
```

Figure 1: Screenshot of Create User API Call



POST {{url}}/api/users/login

Send

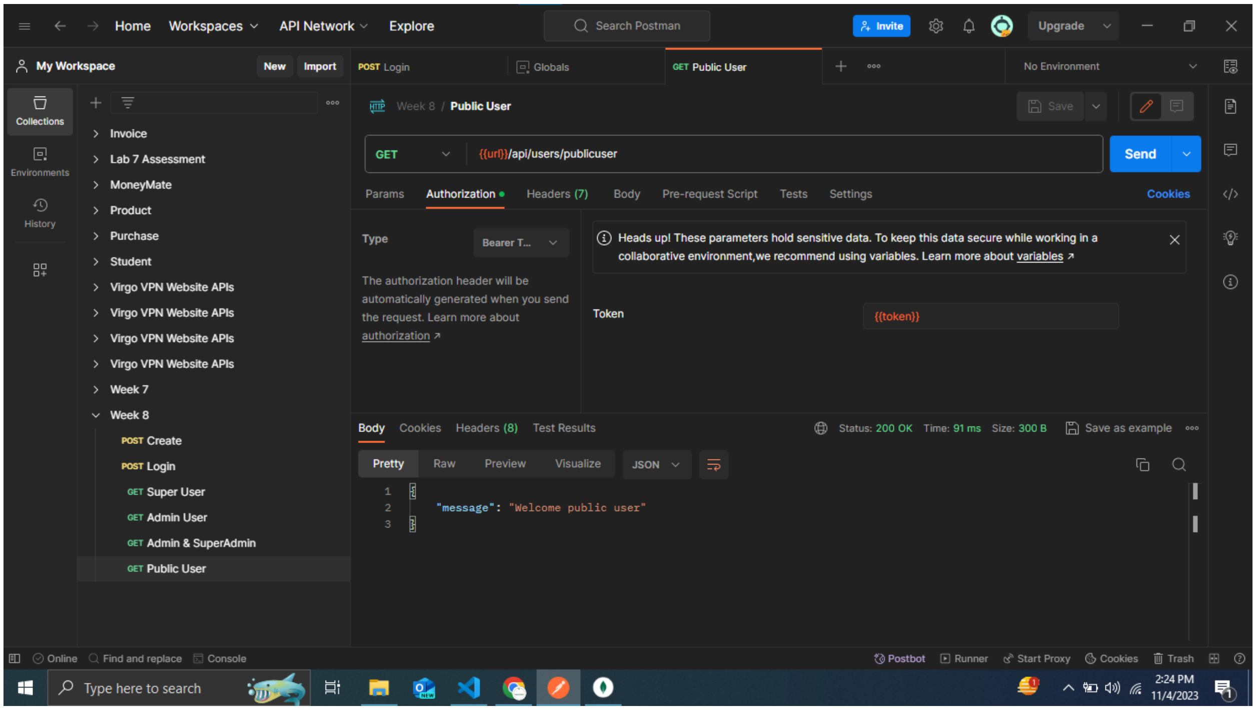
○ none ○ form-data ○ x-www-form-urlencoded ● raw ○ binary ○ GraphQL JSON

```
{
    "email": "hamzaabbasi@gmail.com",
    "password": "123"
}
```

Status: 200 OK Time: 225 ms Size: 560 B

```
{
    "message": "Logged in successfully",
    "email": "hamzaabbasi@gmail.com",
    "userid": "653a9f725e0aae458f787005",
    "token": "eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.
        eyJyb2xlIjoicHVibGljIHVzZXIiLCJpZCI6IjY1M2E5ZjcyNWUwYWFlNDU4Zjc4NzAwNSIsImlhdCI6MTY5OTA4OTMzNH0.
        FfepUXDPWhHWUOxhIKreW9ZLqobJJ_A9C1-1_iVxeNA"
}
```

Figure 2: Screenshot of Login User API Call



GET {{url}}/api/users/publicuser

Type Bearer T...

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

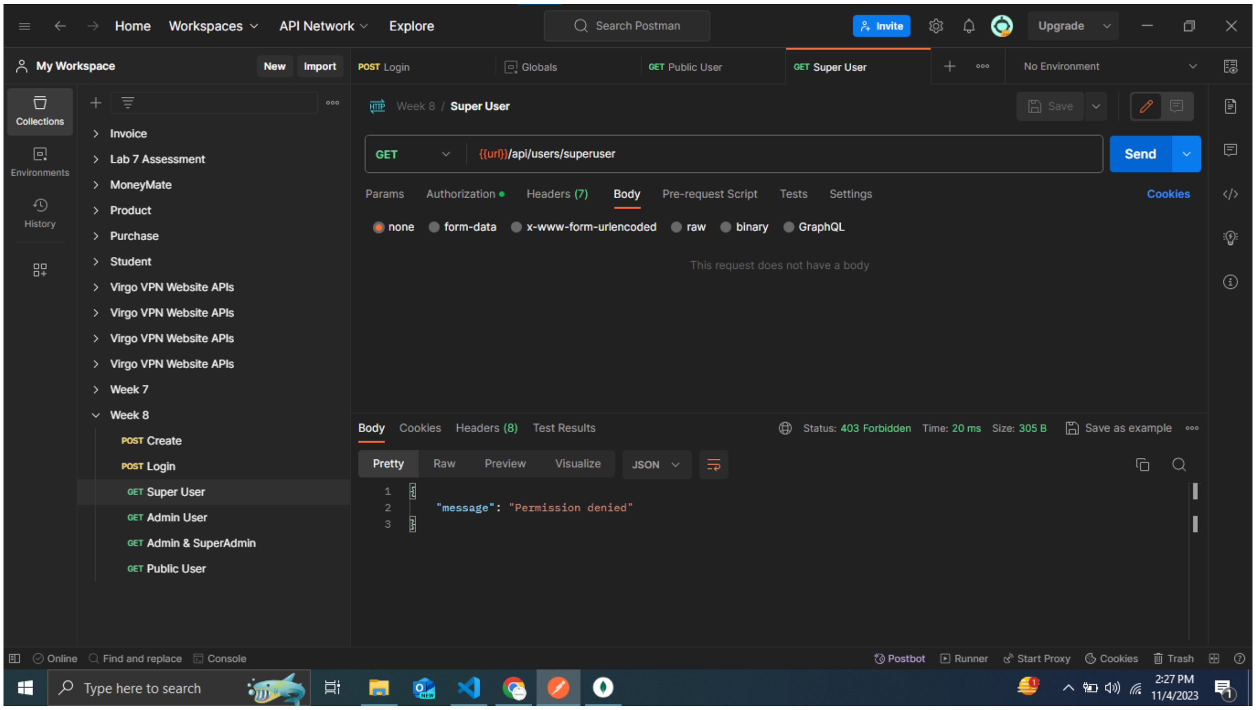
Heads up! These parameters hold sensitive data. To keep this data secure while working in a collaborative environment,we recommend using variables. Learn more about variables ↗

Token {{token}}

Status: 200 OK Time: 91 ms Size: 300 B

```
{
    "message": "Welcome public user"
}
```

Figure 3: Screenshot of Public User Login by validating his role



GET {{url}}/api/users/superuser

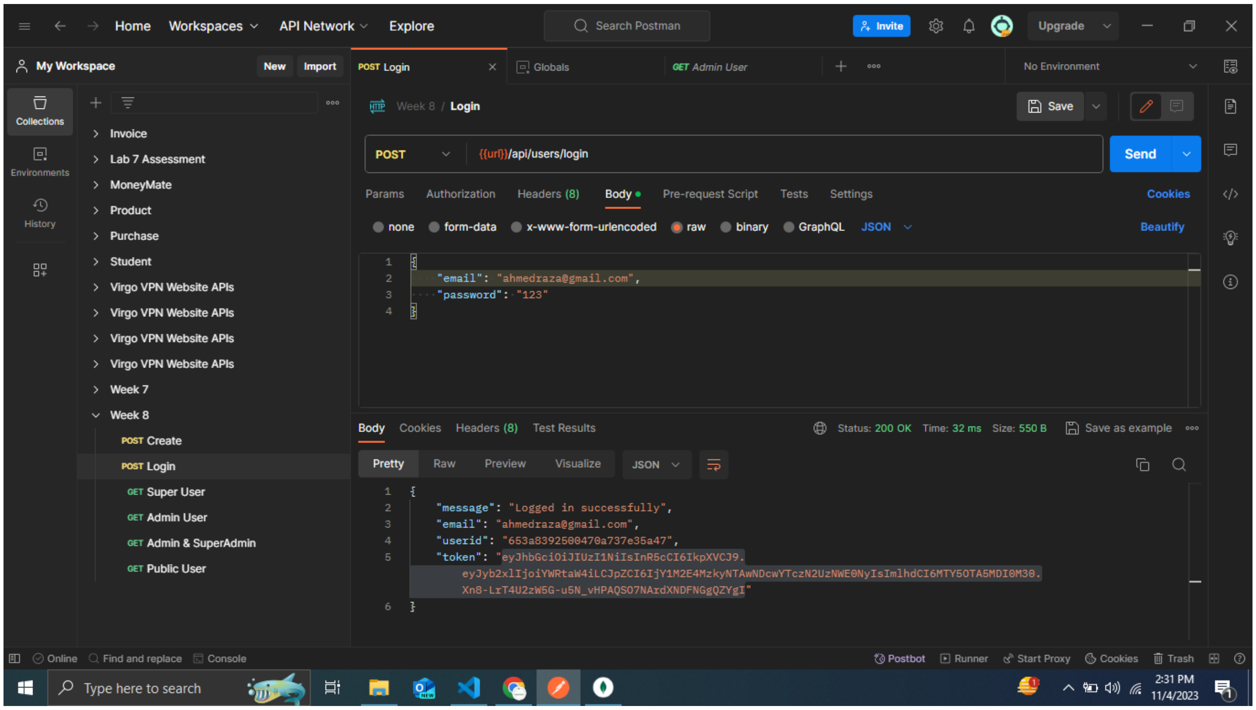
● none ○ form-data ○ x-www-form-urlencoded ○ raw ○ binary ○ GraphQL

This request does not have a body

Status: 403 Forbidden Time: 20 ms Size: 305 B

```
{
    "message": "Permission denied"
}
```

Figure 4: Screenshot of Public User When he tried to access Super User roll



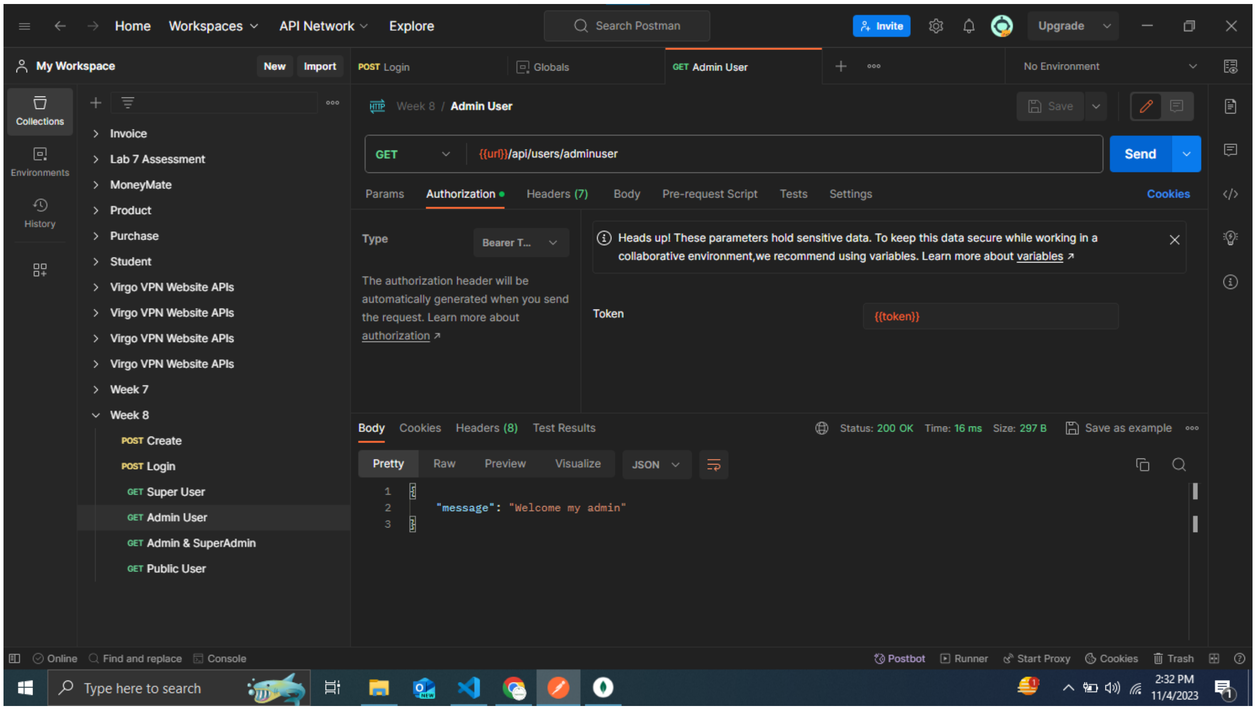
POST {{url}}/api/users/login

```
{
    "email": "ahmedraza@gmail.com",
    "password": "123"
}
```

Status: 200 OK Time: 32 ms Size: 550 B

```
{
    "message": "Logged in successfully",
    "email": "ahmedraza@gmail.com",
    "userid": "653a8392500470a737e35a47",
    "token": "eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.
        eyJyb2xlIjoiYWRtaW4iLCJpZCI6IjY1M2E4MzkyNTAwNDcwYTczN2UzNWE0NyIsImlhdCI6MTY5OTA5MDI0M30.
        Xn8-LrT4U2zW5G-u5N_vHPAQSO7NArdXNDFNGgQZYgI"
}
```

Figure 5: Screenshot of Admin Login API Call



GET {{url}}/api/users/adminuser

Type: Bearer T...

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

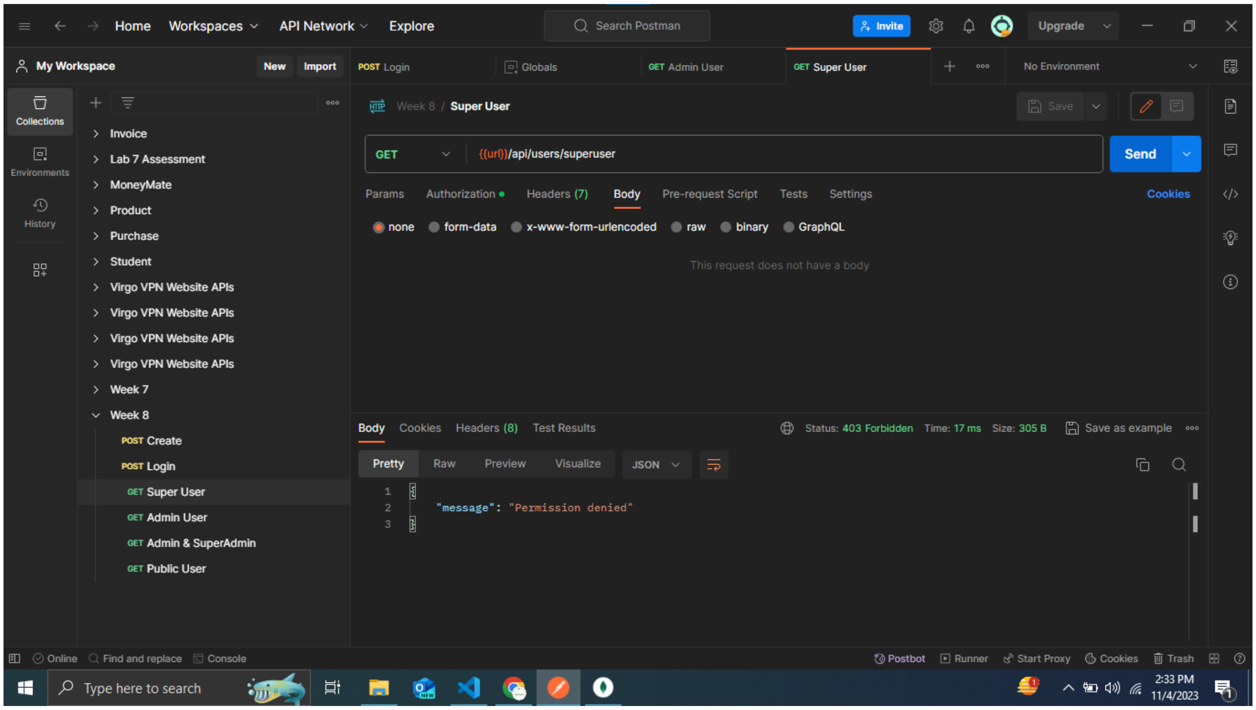
Heads up! These parameters hold sensitive data. To keep this data secure while working in a collaborative environment,we recommend using variables. Learn more about variables ↗

Token: {{token}}

Status: 200 OK Time: 16 ms Size: 297 B

```
{
    "message": "Welcome my admin"
}
```

Figure 6: Screenshot of Admin Authentication



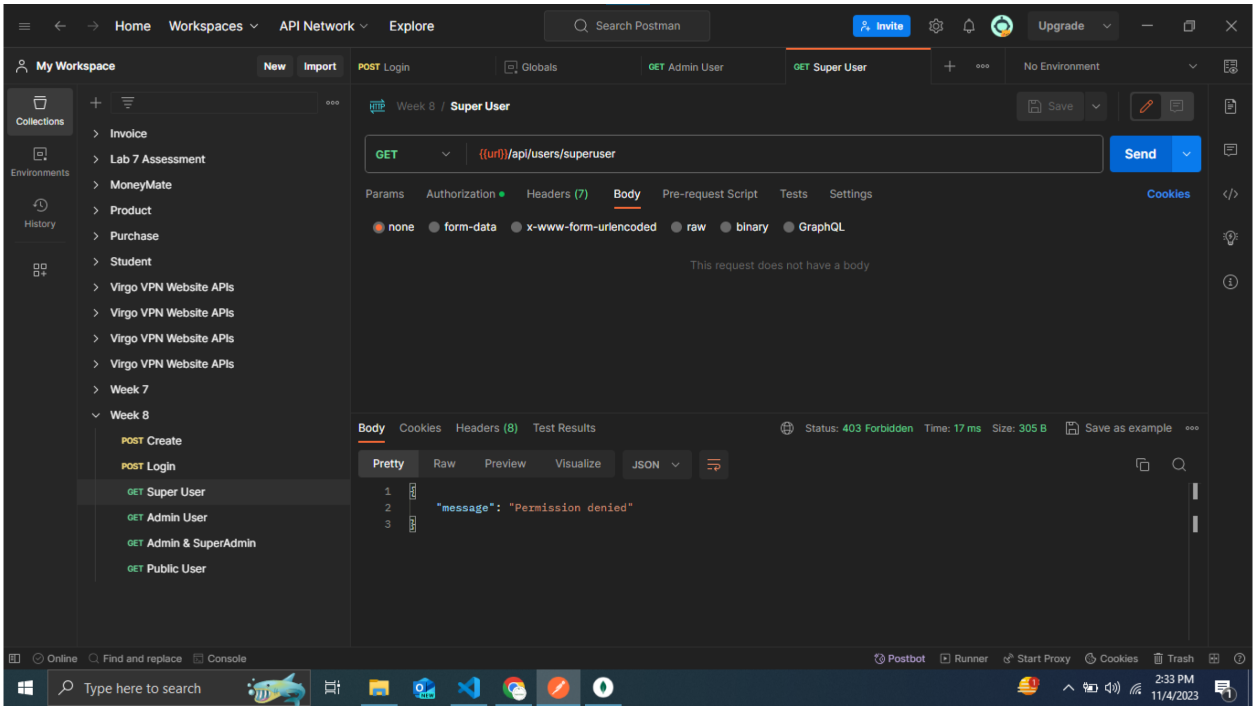
GET {{url}}/api/users/superuser

This request does not have a body

Status: 403 Forbidden Time: 17 ms Size: 305 B

```
{
    "message": "Permission denied"
}
```

Figure 7: Screenshot of Admin When he tries to login as Super Admin



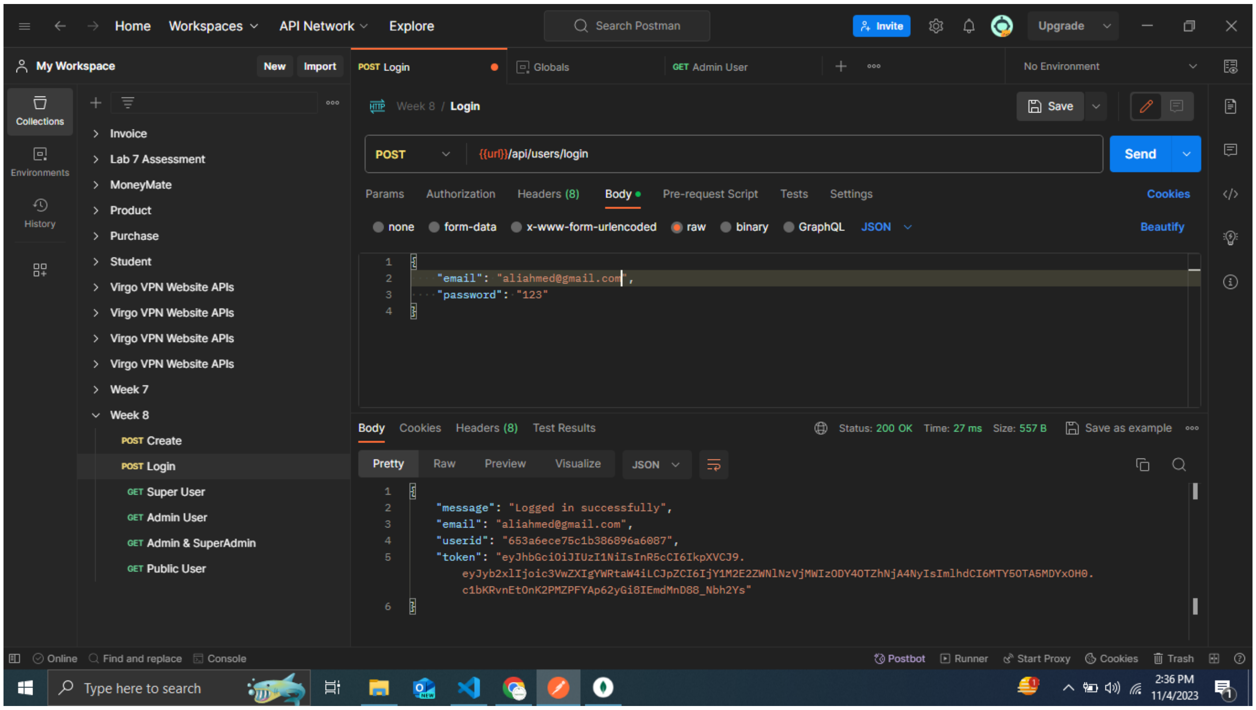
GET {{url}}/api/users/superuser

This request does not have a body

Status: 403 Forbidden Time: 17 ms Size: 305 B

```
{
    "message": "Permission denied"
}
```

Figure 8: Screenshot of Admin/Superuser combined endpoint API Call



POST {{url}}/api/users/login

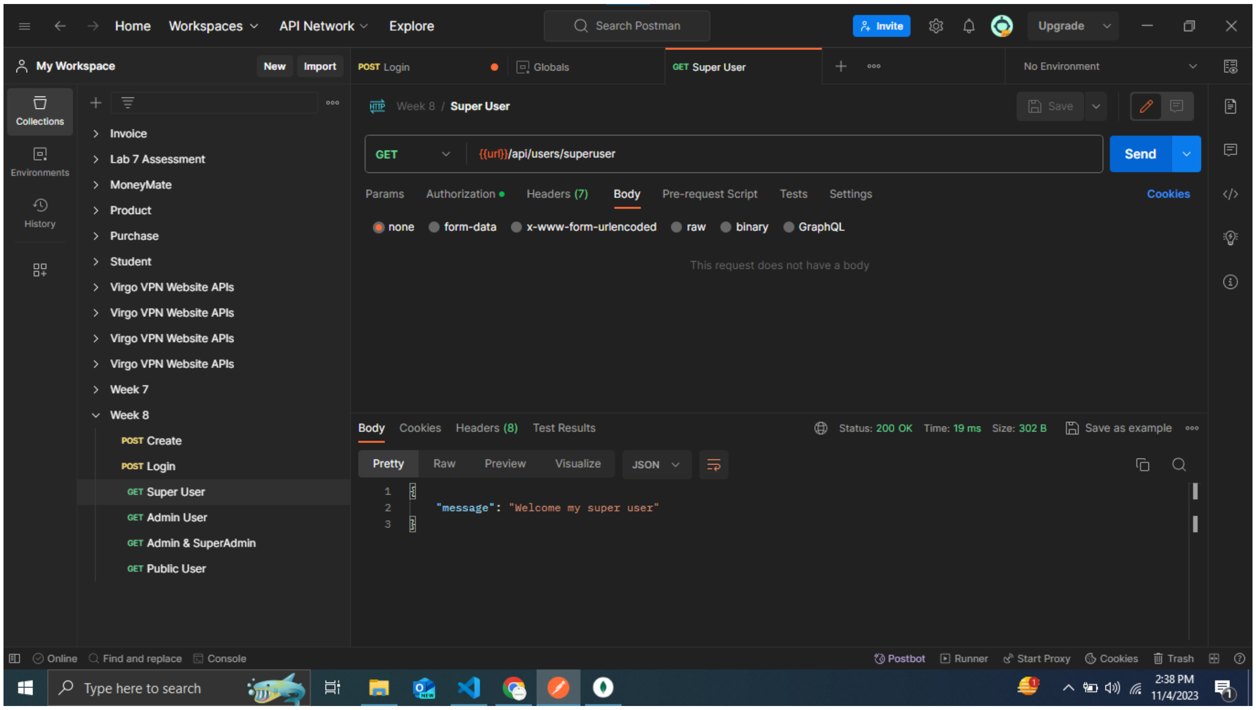
○ none ○ form-data ○ x-www-form-urlencoded ● raw ○ binary ○ GraphQL JSON

```
{
    "email": "aliahmed@gmail.com",
    "password": "123"
}
```

Body Cookies Headers (8) Test Results Status: 200 OK Time: 27 ms Size: 557 B

```
{
    "message": "Logged in successfully",
    "email": "aliahmed@gmail.com",
    "userid": "653a6ece75c1b386896a6087",
    "token": "eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.
        eyJyb2xlIjoic3VwZXIgYWRtaW4iLCJpZCI6IjY1M2E2ZWNlNzVjMWIzODY4OTZhNjA4NyIsImlhdCI6MTY5OTA5MDYxOH0.
        c1bKRvnEtOnK2PMZPFYAp62yGi8IEmdMnD88_Nbh2Ys"
}
```

Figure 9: Screenshot of Superuser Login endpoint API Call



GET {{url}}/api/users/superuser

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

This request does not have a body

Body Cookies Headers (8) Test Results Status: 200 OK Time: 19 ms Size: 302 B

```
{
    "message": "Welcome my super user"
}
```

Figure 10: Screenshot of Superuser Authentication endpoint API Call

User Model

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

const userSchema = mongoose.Schema(
  {
    username: String,
    email: String,
    password: String,
    role: String,
  },
  { timestamps: true }
);

module.exports = mongoose.model("Users", userSchema);
```

User Controller

```
const User = require("../models/userModel");
const jwt = require("jsonwebtoken"); // Import the jwt library

// Create a User
async function createUser(req, res) {
  try {
    const newUser = await User.create(req.body);
    res.status(201).json({
      message: "User created successfully",
      user: newUser,
    });
  } catch (error) {
    res.status(500).json({ error: error.message });
  }
}

// Get all Users
async function getAllUsers(req, res) {
  try {
    const users = await User.find();
    res.json(users);
  } catch (err) {
    res.status(500).json({ error: err.message });
  }
}
// Login User
async function login(req, res) {
  const { email, password } = req.body;
  try {
    const user = await User.findOne({ email });
    if (!user) return res.status(404).json({ error: "User not found" });
    if (user.password != password)
      return res.status(401).json({ error: "Invalid credentails" });
    return res.status(200).json({
      message: "Logged in successfully",
      email: email,
      fullname: user.fullname,
      userid: user.id,
      token: GenerateToken(user),
    });
  } catch (err) {
    return res.status(500).json({ message: err });
  }
}

// Super Admin
async function superUser(req, res) {
  res.json({ message: "Welcome my super user" });
}

// Admin
async function adminUser(req, res) {
  res.json({ message: "Welcome my admin" });
}

// Admin or Super Admin
async function admin_superadmin(req, res) {
  res.json({ message: "Welcome admin/super admin" });
}

// public user
async function publicUser(req, res) {
  res.json({ message: "Welcome public user" });
}

function GenerateToken(user) {
  const payload = {
    role: user.role,
    id: user._id,
  };
  const token = jwt.sign(payload, "adsfasdfjkh$#asdfasdf.adsfxc");
  return token;
}

module.exports = {
  createUser,
  login,
  adminUser,
  admin_superadmin,
  publicUser,
  superUser,
};
```

User Routes

```
const express = require("express");
const router = express.Router();
const userController = require(”../controllers/userController”);
const validateToken = require(”../utils/auth/auth_middleware”);
const requireRoles = require(”../utils/auth/roles_middleware”);

// create a user
router.post(”/users”, userController.createUser);

// Login User
router.post(”/users/login”, userController.login);

//super user
router.get(
  ”/users/superuser”,
  validateToken,
  requireRoles([”super admin”]),
  userController.superUser
);

//admin user
router.get(
  ”/users/adminuser”,
  validateToken,
  requireRoles([”admin”]),
  userController.adminUser
);

//admin user & Super admin user
router.get(
  ”/users/adminsuperadmin”,
  validateToken,
  requireRoles([”admin”, ”super admin”]),
  userController.admin_superadmin
);

//public user
router.get(
  ”/users/publicuser”,
  validateToken,
  requireRoles([”public user”]),
  userController.publicUser
);

module.exports = router;
```

db.js

```
const mongoose = require(”mongoose”);
mongoose.set(”strictQuery”, true);

mongoose.connect(”mongodb://127.0.0.1:27017/Week8HomeTask”, {
  useNewUrlParser: true,
  useUnifiedTopology: true,
});
```

```
const db = mongoose.connection;
db.on(”error”, (err) => {
  console.log(”Failed to connect with db”);
});
db.once(”open”, () => {
  console.log(”Connected with db”);
});
```