Final Project: CodeCraftHub: Building Personalized Learning Platform for Developers

In this project, you'll leverage the power of Generative AI and a diverse array of technologies to transform your vision into a reality.

Learning objectives

After completing this lab, you will be able to perform the following tasks:

- Design and develop software applications using Generative AI
- Create documentation for the code with Generative AI
- Create test cases with Generative AI
- Deploy the deployable application designed and developed entirely with Generative AI

Prerequisites

1. You must be familiar with at least one programming language and know software architectures well.
2. You must have a GitHub account.
3. You must be comfortable using the IDE.
4. You must be familiar with using Postman.
5. You must be familiar with Docker applications and commands.

Setting up the AI classroom

In case you need help with the Interface/classroom, please click here

Gathering requirements for the development of the learning platform

Using GenAI, gather requirements for developing the server-side learning platform by asking the following questions:

- The effectiveness of the responses depends on the prompts provided.
- The prompts provided here are suggestions; you can use your discretion to change them.
- You should also use your subject matter expertise and judgment as a developer.
- It is your responsibility to check the correctness of the responses.

Type the following prompt to give the context and the objective:

> **Note:** Make sure you are using the ChatGPT Mini 40 model.

```
I want to create a personalized online learning platform, starting with the server side. Recommend an optimal design and architectur
```

▶ Click here to view the sample response generated

> Disclaimer: Your response might vary.

For the following exercise, microservices architecture is the recommended architecture.

Choosing the architecture and components

Type the following in the prompt to choose the microservices architecture and the appropriate server-side components.

```
I want to use a microservices architecture for the server side of a learning platform. The key services should include:
Personalized learning recommendations,
Interactive coding exercises
Real-time feedback to help developers improve their skills and knowledge.
Recommend the complete set of microservices and supporting server-side components I should include, along with their roles and how t
```

The response will comprise the recommended services.

▶ Click here to view the sample response generated

Disclaimer: Your response might vary.

Create the user service

User Management Service are a pivotal service. You will create that service using Node.js and MongoDB.

Type the following in the prompt:

```
I want to create a User Management Service for my learning platform. I will use Node.js and MongoDB. Please recommend an optimal prc
```

This prompt's response will be similar to the following description.

▶ Click here to view the sample response generated

Disclaimer: Your response might vary.

If the provided response doesn't align with the expected project structure, consider refining your prompt by incorporating more specific questions.

In the IDE, create the recommended directory structure and add the files as necessary.

Insert code into each file

Please include the following statement in the prompt:

```
I have set up the User Management Service project using Node.js and MongoDB based on the recommended folder structure. Please give m
```

The goal is to leverage Generative AI for generating the entire code. After manually setting up the files in the IDE based on the previous instructions, you can now include the provided code. Make sure to prompt it to provide the intended fields.

▶ Click here to view the sample response generated

Disclaimer: Your response might vary.

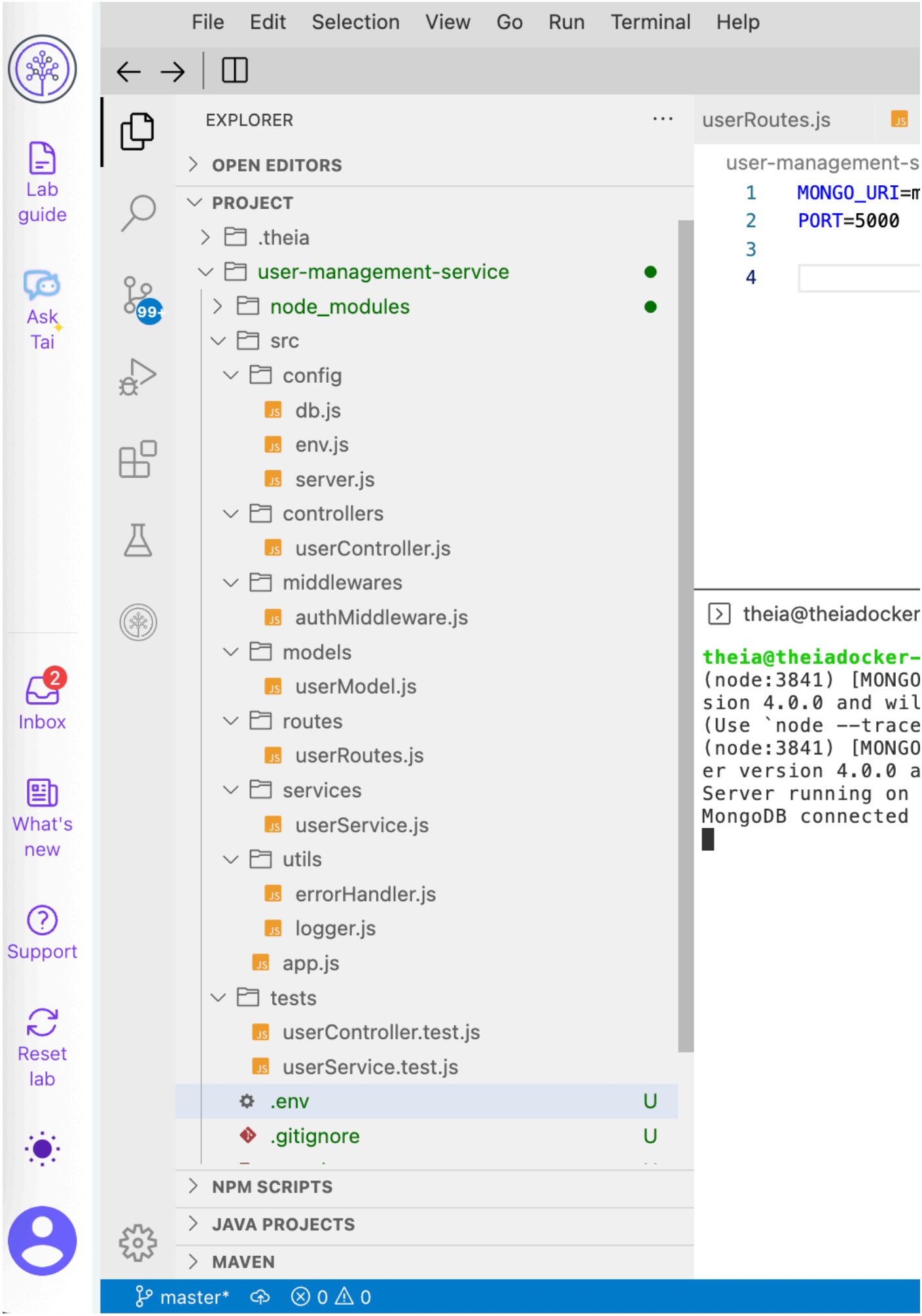
▶ Click here to view the steps to push your work to GitHub

Test the application

To run and test your user management service, you can follow these steps:

1. To start your MongoDB server, use to the pre-work lab for the final project.
2. Start your Node.js server: In your project directory, open a new terminal or command prompt window and run the command `node src/app.js` to start your Node.js server.

   Note: Before running the command `node src/app.js`, first install your packages by running `npm install` or `npm i` to start your Node.js server successfully.


Lab guide
Ask Tai
Inbox
What's new
Support
Reset lab
File Edit Selection View Go Run Terminal Help
EXPLORER
OPEN EDITORS
PROJECT
.theia
user-management-service
node_modules
src
config
db.js
env.js
server.js
controllers
userController.js
middlewares
authMiddleware.js
models
userModel.js
routes
userRoutes.js
services
userService.js
utils
errorHandler.js
logger.js
app.js
tests
userController.test.js
userService.test.js
.env U
.gitignore U
NPM SCRIPTS
JAVA PROJECTS
MAVEN
userRoutes.js
user-management-s
1 MONGO_URI=m
2 PORT=5000
3
4
theia@theiadocker
theia@theiadocker-
(node:3841) [MONGO
sion 4.0.0 and wil
(Use `node --trace
(node:3841) [MONGO
er version 4.0.0 a
Server running on
MongoDB connected
master*
0 0

Generating a database to test the application

You have the code now but you have not created the database yet. You will now use Generative AI to populate the database.

```
Can you please provide the user data in JSON format?
```

▶ Click here to view the sample response generated

Disclaimer: Your response might vary.

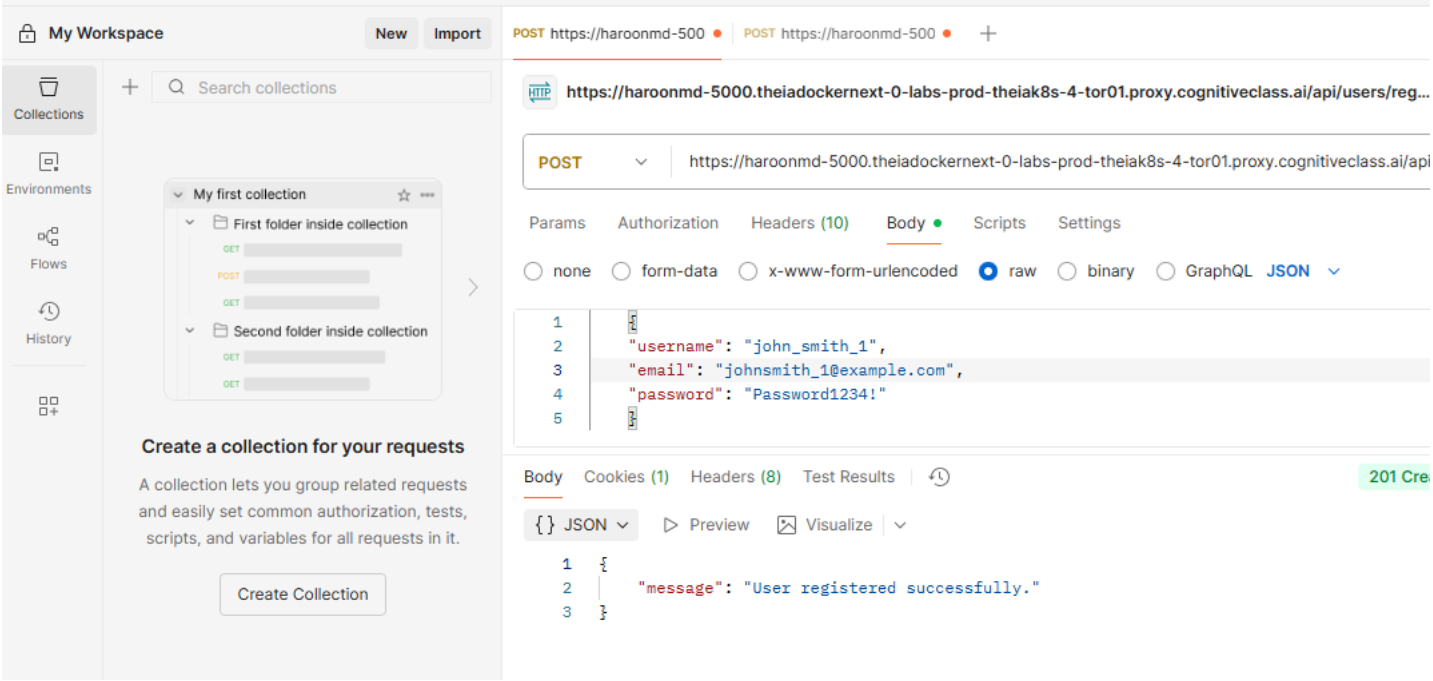
Test the API endpoints: You can use tools like Postman or make API requests from your front-end application to test the API endpoints. Here are some example requests:

Note: The testing instructions provided are based on the response codes received in our example. Your implementation may produce different codes depending on your logic. Please adjust your testing steps accordingly to match your code's output.

1. User Registration: Send a POST request to `http://localhost:5000/api/users/register` with the following request body:

```
{
"username": "john_smith_1",
"email": "johnsmith_1@example.com",
"password": "Password1234!"
}
```

You should receive a response with status code **201** and the message "User registered successfully".



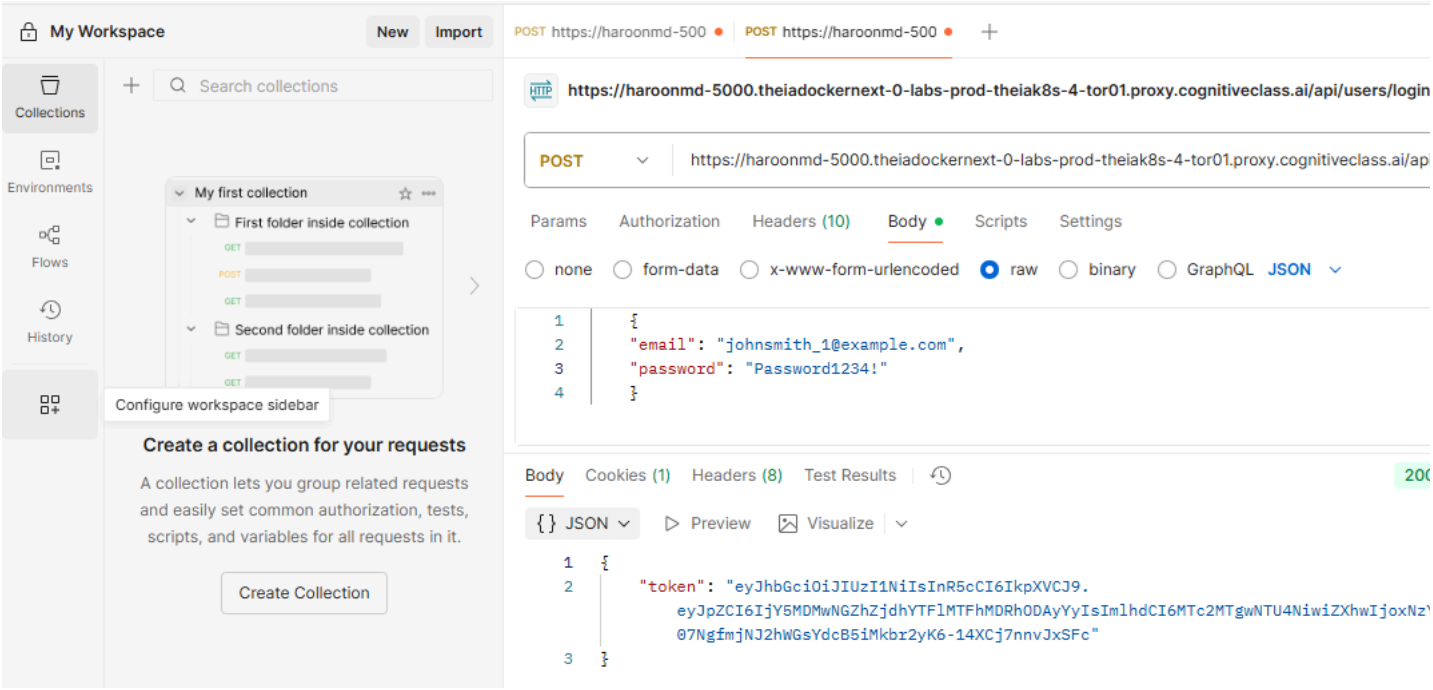
You can verify the endpoint using a curl command in the terminal window.

```
curl -X POST -H "Content-Type: application/json" -d '{"username": "john_smith_1", "email": "johnsmith_1@example.com", "password": "p
```

2. User Login: Send a POST request to `http://localhost:5000/api/users/login` with the following request body:

```
{
"email": "johnsmith_1@example.com",
"password": "Password1234!"
}
```

You should receive a response with status code **200** and a **JSON Web Token (JWT)** in the response body.



You can also verify the endpoint by using a curl command in the terminal window.

```
curl -X POST -H "Content-Type: application/json" -d '{"email": "johnsmith_1@example.com", "password": "Password1234!"}' http://local
```

Verify the data in the MongoDB database: You can use a MongoDB client or run MongoDB queries to verify the data was added to the database.

Remember to handle errors, implement additional API endpoints, and thoroughly test your User Management Service to ensure it meets your requirements.

Note: Ensure all your work pushes to your GitHub repository. Click here for detailed steps.

▶ Click here to view the steps to push your work to GitHub

Code review

You must provide the code in each file you created and get them reviewed.

```
Can you review the code below?
```

And then paste the code that you want to get reviewed.

▶ Click here to view the sample response generated

Disclaimer: Your response might vary.

Documentation

You need to provide documentation and comments for all the code written.

```
Can you provide documentation and comments for the code to make it readable?
```

You will use the prompt iteratively with the content of each file.

Dockerfile

You need to bundle the application in Docker. Type the following prompt to create a Dockerfile that bundles the application and MongoDB server in a container.

```
Can you provide the docker file to bundle the application and the MongoDB server in a container?
```

▶ Click here to view the sample response generated

Disclaimer: Your response might vary.

You may be prompted for a specific procedure if your response doesn't show how to deploy.

▶ Click here to view the steps to push your work to GitHub

Checklist

At this stage:

1. You now have a running application that offers CRUD microservices for the User Management Service.
2. The code has undergone a thorough review and is comprehensively documented.
3. The service has been successfully deployed within a Docker container.
4. Proceed to push all the code to your GitHub repository.

Summary

- You have successfully gathered requirements for a user management service of a programming-focused learning platform using Generative AI.
- You have explored vital aspects such as fundamental features, user-friendly design, interactive functionalities, and an efficient folder structure.

Subsequent actions involve:

- Employing MongoDB for user data
- Producing Node, Express, and Mongoose code
- Conducting a detailed code review with comprehensive documentation

Congratulations! You have successfully leveraged Generative AI to build a learning platform by choosing Microservices Architecture, Node.js, and MongoDB.

Author(s)

Sapthashree K S