

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:

1. 1
1. I want to create a personalized online learning platform. I want to start with the server side. Recommend a good design and architecture for the serv

Copied!

▼ Click here to view the sample response generated

You will need a robust and scalable server-side architecture to create a personalized online learning platform. One popular design is a microservices architecture. This architecture divides the application into smaller, independent services that you can develop, deploy, and scale individually.

Here are some key components and requirements for the server side of your learning platform:

1. **Authentication and authorization:** Implement a secure user authentication system to ensure that only authorized users can access the platform. Some examples of a secure user authentication system are OAuth or JSON Web Tokens (JWT).
2. **User management:** Develop features to manage user profiles, including registration, login, password reset, and profile customization.
3. **Content management:** Create a system to manage and organize learning content, such as courses, lessons, quizzes, and multimedia resources. You may consider using a content management system (CMS) or a database to store and retrieve this information.
4. **Learning management system (LMS):** Implement features to track and manage learners' progress, including course enrollment, completion tracking, and assessment management.
5. **Analytics and reporting:** Incorporate analytics tools to gather data on user activities, course performance, and engagement metrics. This data can help you improve the platform and personalize the learning experience.
6. **Payment integration:** To offer paid courses or subscriptions, integrate a secure payment gateway to handle transactions and manage user subscriptions.
7. **Scalability and performance:** Design your server-side architecture to handle concurrent users and ensure quick response times. Consider using cloud-based solutions like AWS or Azure to scale your infrastructure as needed.

8. **API development:** Create a well-documented API that allows integration with other systems, such as mobile apps or third-party services.

Remember to prioritize security, scalability, and performance while designing your server-side architecture. Additionally, keep an eye on the latest trends and technologies in the e-learning industry to stay up-to-date with the best practices.

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.

```
1. 1
2. 2
3. 3
4. 4
5. 5
6. 6
7. 7
```

```
1. I would like to use a microservices architecture for the server side. These are the services I want to be able to provide.
2.
3. Personalized learning recommendations,
4. Interactive coding exercises
5. Real-time feedback to help developers improve their skills and knowledge.
6.
7. What are the various components I should have?
```

Copied!

The response will comprise the recommended services.

► [Click here to view the sample response generated](#)

Create the user service

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

Type the following in the prompt:

```
1. 1
```

```
1. I would like to create the user service. I would like to use Node.js and MongoDB for this project. How do I create a project structure?
```

Copied!

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

▼ [Click here to view the sample response generated](#)

To create a project structure for your user management service using Node.js and MongoDB, you can follow these steps:

- Set up a new Node.js project:** Open your terminal or command prompt and navigate to the intended directory where you want to create your project. Run the command `mkdir user-management-service` to create a new directory for your project. Then, navigate into the project directory by running `cd user-management-service`.
- Initialize a new Node.js project:** Run the command `npm init` to initialize a new Node.js project. Follow the prompts to set up your project details, such as the project name, version, description, and so on. This initialization will create a `package.json` file in your project directory.
- Install required dependencies:** Run the command `npm install express mongoose bcrypt jsonwebtoken` to install the required dependencies for your user management service. Express is a popular web framework for Node.js. Mongoose is an object modeling tool for MongoDB. Bcrypt is a library for password hashing. The JSON web token is a library for handling JSON Web Tokens for authentication.
- Create the project structure:** In your project directory, create the following directories and files:
 - `src` directory will contain your source code.
 - `src/controllers` directory will contain your user-related controller logic.
 - `src/models` directory will contain your user-related database models.
 - `src/routes` directory will contain your user-related API routes.
 - `src/config` directory will contain your configuration files.
 - `src/index.js` file will serve as the entry point for your application.
- Implement the User Management Service:** Create the necessary files and folders within the `src` directory. For example, you can create `src/controllers/userController.js`, `src/models/userModel.js`, and `src/routes/userRoutes.js` to handle user-related operations.
 - In `userController.js`, define functions for user registration, login, profile management, and so on.
 - In `userModel.js`, define the user schema using Mongoose to define the user data structure.
 - In `userRoutes.js`, define the API routes for user registration, login, profile management, and so on.
- Configure the MongoDB connection:** In the `src/config` directory, create a file called `database.js` to configure the MongoDB connection. Use the Mongoose library to connect to your MongoDB database.

7. **Set up the Express server:** In `index.js`, import the necessary dependencies, configure the Express server, and set up the API routes.

8. **Test the User Management Service:** Run the command `node src/index.js` to start the Express server and test your User Management Service using tools like Postman or by making API requests from your frontend application.

Remember to follow best practices for code organization, error handling, and security when developing your User Management Service.

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:

1. 1
1. Please give me the code that is to be included in each of the files.

Copied!

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](#)

Note :

1. To obtain the secret key, execute the following command in the terminal.

```
node -e "console.log(require('crypto').randomBytes(32).toString('hex'))"
```

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, refer 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/index.js` to start your Node.js server.

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.

1. 1
1. Can you please provide the user data in JSON format?

Copied!

► [Click here to view the sample response generated](#)

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:

- User registration: Send a POST request to `http://localhost:3000/users/register` with the following request body:

```
1. 1
2. 2
3. 3
4. 4
1. {
2.   "username": "john",
3.   "password": "password123"
4. }
```

Copied!

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.

```
1. 1
1. curl -X POST -H "Content-Type: application/json" -d '{"username": "john", "password": "password123"}' http://localhost:3000/users/register
```

Copied!

- User login: Send a POST request to `http://localhost:3000/users/login` with the following request body:

```
1. 1
2. 2
3. 3
4. 4
1. {
2.   "username": "john",
3.   "password": "password123"
4. }
```

Copied!

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.

```
1. 1
1. curl -X POST -H "Content-Type: application/json" -d '{"username": "john", "password": "password123"}' http://localhost:3000/users/login
```

Copied!

- User profile update: Send a PUT request to `http://localhost:3000/users/john_smith` (Replace "john_doe" with the actual username) with the following request body:

```
1. 1
2. 2
3. 3
1. {
2.   "newUsername": "john_smith"
3. }
```

Copied!

You should receive a response with status code **200** and the message "User profile updated successfully".

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

```
1. 1
1. curl -X PUT -H "Content-Type: application/json" -d '{"newUsername": "john_smith"}' http://localhost:3000/users/john_smith
```

Copied!

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.

```
1. 1
1. Can you review the code below?
```

Copied!

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

Documentation

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

- 1
1. Can you provide documentation and comments for the code to make it readable?

Copied!

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

Code review

You have to iteratively give the code in each file you have created and get them reviewed.

- 1
- 2
- 3
- 4
- 5
- 6
- 7
- 8
- 9
1. Kindly review the userRoutes.js file code.
- 2.
3. const express = require('express');
4. const router = express.Router();
5. const userController = require('../controllers/userController');
6. router.post('/register', userController.registerUser);
7. router.post('/login', userController.loginUser);
8. router.put('/:username', userController.updateUserProfile);
9. module.exports = router;

Copied!

► [Click here to view the sample response generated](#)

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.

- 1
1. Can you provide the docker file to bundle the application and the MongoDB server in a container?

Copied!

► [Click here to view the sample response generated](#)

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

© IBM Corporation. All rights reserved.