

Final Project Documentation

Submitted By:

M. Abbas Hafeez 22F-3725

Marsad Saqib 22F-3733

BSE-5A

Submitted To: Mr. Jawad Khalid

# **Containerization and CI/CD pipelines of Test Automation**

**Note-Taking App**

**1. Overview**

The **Note-Taking App** is a simple web application that allows users to create, view, and delete notes. Each note consists of a title and content, and users can manage their notes through a user-friendly interface. The app is built using **Node.js** and **Express**, with testing and containerization enabled to ensure consistent deployment and functionality.

**2. Features**

* **Create Notes:** Users can create new notes with a title and content.
* **View Notes:** The app displays a list of all the saved notes, showing both the title and content.
* **Delete Notes:** Users can delete any note they no longer need.
* **Automated Testing:** The app includes unit and integration tests that verify the functionality of the routes and the service layer.
* **Containerization:** The application can be run in any environment using Docker, allowing for easy setup and deployment.

**3. Project Structure**

The project is organized into several components to promote clean architecture and separation of concerns. Here is a high-level overview of the structure:

* **/src Directory:** Contains the core application code, including services, routes, and the database simulation.
  + **mockDb.js:** Simulates a simple in-memory database for storing notes.
  + **noteService.js:** Contains the business logic for managing notes, including creating, deleting, and retrieving notes.
  + **routes.js:** Defines the API endpoints for interacting with the notes.
  + **index.js:** The entry point for the application, where the Express server is initialized and routes are configured.
* **/tests Directory:** Contains the test files that ensure the application behaves as expected.
  + **noteAppTests.test.js:** Defines unit and integration tests using **Jest** and **Supertest** to check the functionality of the app.
* **Dockerfile:** Defines the steps to containerize the app using **Docker**. It specifies the base image, the working directory, and commands to install dependencies and start the app.
* **package.json & jest.config.js:** Configuration files for the application dependencies and testing setup.

**4. Installation and Setup**

To set up the **Note-Taking App** on a local machine, the following steps are typically followed:

* Clone the repository to your local machine.
* Install the required dependencies using **npm** (Node Package Manager).
* Run the app locally using the npm start command.
* Tests can be run to ensure that the app is functioning correctly using npm test.

**5. Running the Application**

Once the app is installed, it can be run locally on a web browser. By default, the app will be hosted on localhost at port 3000. The app can be accessed using any browser to interact with the user interface for creating, viewing, and deleting notes.

**6. API Endpoints**

The app exposes the following API endpoints for interacting with the notes:

* **POST /notes:** Allows users to create a new note by sending the title and content.
* **GET /notes:** Retrieves all the notes stored in the system.
* **DELETE /notes/:id:** Deletes a specific note based on the provided note ID.

These endpoints are the backbone of the app’s functionality, enabling users to interact with their notes through HTTP requests.

**7. Test Automation**

Test automation ensures the reliability of the app by running automated tests to check for errors or unexpected behavior. The **Jest** testing framework is used for unit testing, while **Supertest** is used for testing the API routes to ensure they behave as expected.

Tests are typically divided into unit tests (which verify the functionality of individual methods or services) and integration tests (which verify that the entire system works as expected when components are integrated).

By running tests, developers can catch bugs early and ensure the app is functioning correctly after every change or update.

**8. Dockerization**

The application is containerized using **Docker**, which allows for a consistent and isolated environment for running the app. The app's dependencies, configurations, and setup are packaged into a **Docker image**, which can be deployed anywhere Docker is supported.

By using Docker, developers and teams can avoid inconsistencies between development and production environments. The Docker container ensures that the app behaves the same way regardless of where it’s run.

To containerize the app:

* A **Dockerfile** is used to define the environment and setup instructions, including installing the necessary dependencies and copying the application code into the container.
* Once the Docker image is built, the container can be started, and the app will run in an isolated environment.

**9. CI/CD Pipeline**

The **CI/CD (Continuous Integration/Continuous Deployment)** pipeline is a set of automated processes used to build, test, and deploy the application. The pipeline helps ensure that code is consistently integrated, tested, and deployed, reducing the likelihood of human error and speeding up the release process.

For this app, a **GitHub Actions** workflow is set up to automate the following processes:

* **Checkout the Repository:** The code is pulled from the GitHub repository.
* **Install Dependencies:** All necessary dependencies are installed.
* **Run Tests:** Automated tests are run to ensure that the app works as expected.
* **Build Docker Image:** A Docker image is built to containerize the app.
* **Deploy the App:** (Optional) The app can be deployed to a server or cloud platform as part of the pipeline.

This CI/CD pipeline helps automate the deployment process, ensuring that new changes are quickly tested and deployed to production.

**10. Conclusion**

The **Note-Taking App** provides a simple interface for managing notes, while also integrating modern software practices such as automated testing, containerization, and CI/CD pipelines. These features not only enhance the app’s functionality but also improve the reliability, scalability, and ease of deployment.

In summary, **containerization** and **CI/CD pipelines** have provided immense benefits to the **Note-Taking App** development process. Dockerized deployment ensures consistency across environments, while the CI/CD pipeline automates testing, integration, and deployment. These practices have resulted in faster development cycles, reduced manual errors, and a higher-quality application that is easier to maintain and deploy. The experiences from implementing these technologies have proven invaluable, particularly in enhancing collaboration, reducing deployment times, and improving the overall reliability of the app.