Steps of Workflow Implementation

The application is organized into two main parts:

- Frontend (Client): A React-based application located in the client folder. It handles user interaction, routing, and communicates with the backend API.
- Backend (Server): A Node.js/Express application located in the server folder. It manages authentication, connects to MongoDB, and handles business logic.

For containerization and deployment:

- Docker is used to build container images for both the frontend and backend.
- docker-compose is used for orchestrating local builds and can be adapted for production deployments.
- GitHub Actions automates testing (CI) and building/tagging/pushing Docker images (CD) to Docker Hub.

CI Pipeline Implementation

- File: .github/workflows/node.js.yml
- Trigger: Runs on any push or pull request to the main branch.
- Matrix Strategy: Tests run on Node.js versions 18.x, 20.x, and 22.x.
- Steps:
    - Checkout: Uses actions/checkout@v4 to clone the repository.
    - Setup Node.js: Uses actions/setup-node@v4 to install the correct Node.js version and enable npm caching.
    - Testing Frontend: Changes directory to client, installs dependencies, and runs tests.
    - Testing Backend: Installs dependencies at the project root, then moves into the server folder to install dependencies and run backend tests.

CD Pipeline Implementation

- File: .github/workflows/docker-deploy-workflow.yml
- This workflow is designed to be manually triggered (workflow_dispatch)
- Checkout: Clones the repository.
- Docker Compose Build: Uses Docker Compose (with the command docker compose build) to build both frontend and backend images based on the definitions in docker-compose.yml.

- Check Docker Compose Version: Verifies that Docker Compose V2 is available on the runner.
- Log in to Docker Hub: Uses docker/login-action@v2 to authenticate with Docker Hub using stored secrets.
- Tag and Push: Tags the built images with your Docker Hub repository names and pushes them.
- File: docker-compose.yml
    - This file defines how to build the Docker images locally (or in CI) using two services:
    - backend: Uses the Dockerfile in ./server and tags the image as recipeconnect-backend:latest.
    - frontend: Uses the Dockerfile in ./client and tags the image as recipeconnect-frontend:latest.
- File: aure_config.yml
    - This file is intended for production deployment.
    - It doesn't rebuild the images; instead, it uses images already pushed to Docker Hub.
    - It substitutes the Docker Hub username (using the environment variable DOCKER_USERNAME) into the image names.
- Dockerfiles (client/Dockerfile and server/Dockerfile)
    - These files build the frontend and backend container images.