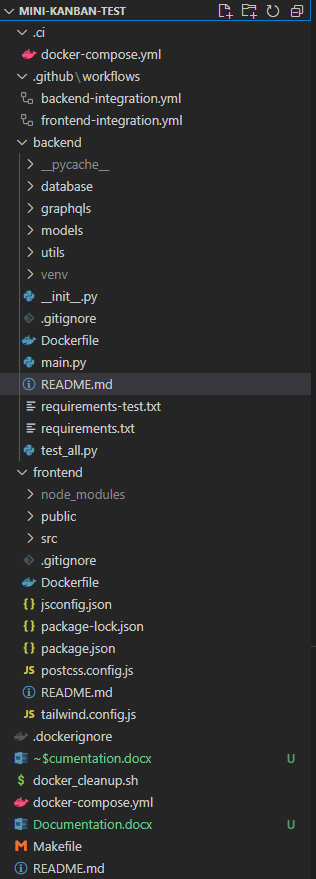
**Mini Kanban - Test Project**

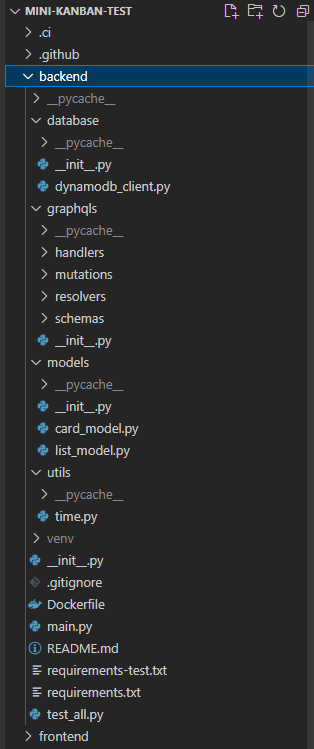
**Documentation**

## Project Structure

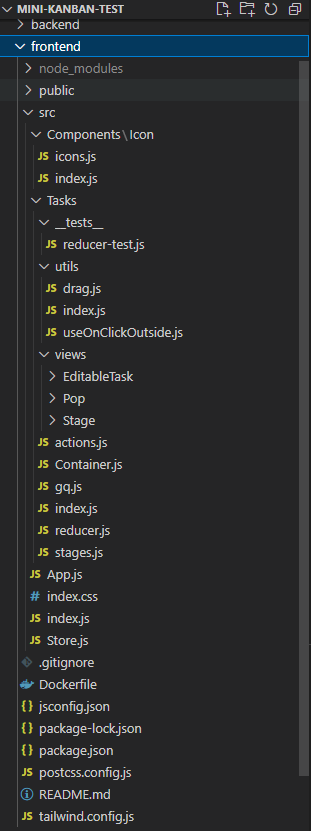
### Entire Project



### 1.2 Backend



### 1.3 Frontend



## Entire Structure

### 2.1. .ci

### 2.1.1 docker-compose.yml:

This file is used to define and run multi-container Docker applications. It contains the configuration for the services that make up your application, and allows you to define the relationships between them.

### 2.2. .github

#### 2.2.1 workflows:

This folder contains GitHub Actions workflows configuration files.

##### 2.2.1.1 backend-integration.yml:

This file contains the configuration for a GitHub Actions workflow named "backend-integration". It includes steps for continuous integration, such as running tests and checks on code changes of backend.

##### 2.2.1.2 frontend-integration.yml:

This file contains the configuration for a GitHub Actions workflow named "frontend-integration". It includes steps for continuous integration, such as running tests and checks on code changes of frontend.

### 2.3. backend

It contains API code for mini-kanban functionality developed using python-graphen, fastapi, and dynamodb.

### 2.3. frontend

It contains UI code for mini-kanban interface developed using React Hook, Tailwind CSS, useReducer, Apollo-Client.

### ### Files:

#### - .dockerignore:

This file specifies patterns to exclude from the Docker build context, similar to .gitignore for Git.

#### - docker-compose.yml:

This file is used to define the services, networks, and volumes for a Docker application. It specifies the infrastructure of the application.

#### - Makefile:

This file likely contains directives for building and compiling the project, enabling users to run build tasks with simple commands.

#### - Readme.md:

This file likely contains information about the project, including how to set it up, use it, and contribute to it.

## Backend Structure

### Database

#### 3.1.1 dynamodb\_client.py

This file contains the code for interacting with Amazon DynamoDB, such as setting up the client and connection to the DynamoDB service.

### Graphqls

#### Handlers

##### card\_handler.py

This file contains the business logic for handling GraphQL queries and requests related to cards on the Kanban board.

##### list\_handler.py

This file contains the business logic for handling GraphQL queries and requests related to lists on the Kanban board.

#### Mutations

##### card\_mutations.py

This file contains the code for defining GraphQL mutations related to cards, such as creating, updating, and deleting cards.

##### list\_mutations.py

This file contains the code for defining GraphQL mutations related to lists, such as creating, updating, and deleting lists.

##### mutations.py

This file contain shared code and utilities related to GraphQL mutations used across the application.

#### Resolvers

##### card\_resolvers.py

This file contains the resolver functions for resolving GraphQL queries related to cards, connecting the queries to the actual data and logic.

##### list\_resolvers.py

This file contains the resolver functions for resolving GraphQL queries related to lists, connecting the queries to the actual data and logic.

#### Schemas

##### card\_schema.py

This file contains the schema definition for cards, including GraphQL types and input objects.

##### list\_schema.py

This file contains the schema definition for lists, including GraphQL types and input objects.

##### schema.py

This file contain the shared schema definitions used across the GraphQL API.

### Models

#### card\_models.py

This file contains the Python classes and data structures that represent the data model for cards in the Kanban board.

#### list\_model.py

This file contains the Python classes and data structures that represent the data model for lists in the Kanban board.

### Utils

#### time.py

This file contains utility functions to convert to GraphQL Date format.

### ### Files:

#### - .gitignore

This file specifies intentionally untracked files to ignore when using Git for version control.

#### - Dockerfile

This file contains instructions for building a Docker image for the FastAPI application, specifying its environment and dependencies.

#### - main.py

This file serves as the entry point for the FastAPI application, containing the startup code, routes, and configuration.

#### - requirements-test.txt

This file contains the specific dependencies required for running tests and for the test environment.

#### - requirements.txt

This file contains the project's main dependencies.

#### - test\_all.py

This file contains a script to run all tests for the project.

## Frontend Structure

### Public

This directory contains the public assets, such as HTML and other assets, that are served by the development server.

### Src

#### Components

##### Icon

This directory contains components related to displaying icons used in the UI.

#### Tasks

##### \_\_test\_\_

This directory is for unit tests for the tasks-related components and utilities.

##### Utils

###### drag.js

This file contains utility functions related to drag and drop functionality in the Kanban board.

###### index.js

This file be the main file for the utility's exports in the "utils" directory.

###### useOnClickOutside.js

This file contain a custom hook or utility for handling click events outside of a specified element.

##### Views

###### EditableTask

This directory contains components related to editing tasks and cards on the Kanban board.

###### Pop

This directory contains components related to popovers in the UI.

###### Stage

This directory contains components related to displaying different cards on the Kanban board.

##### Actions.js

This file contains action creators or action constants used in the useReducer state management.

##### Container.js

This file contains a top-level container component for the entire application and a significant portion of it.

##### Gq.js

This file contain GraphQL query and mutation operations setup.

##### Index.js

This file is the entry point for the JavaScript bundling and rendering logic.

##### Reducer.js

This file contains the reducer function and initial state for the useReducer hook.

##### stages.js

This file contains the initial data for Kanban board test.

#### App.js

This file is the main component that represents the root of the application.

#### index.css

This file contains global CSS styles.

#### Store.js

This file is for configuring and setting up the application state management, such as useReducer state.

### ### Files:

#### - .gitignore

This file specifies intentionally untracked files to ignore when using Git for version control.

#### - Dockerfile

This file contains instructions for building a Docker image for the React application, specifying its environment and dependencies.

#### - jsconfig.json

This file contains configuration options for the JavaScript language service, such as project settings for Visual Studio Code.

#### - package-lock.json

This file represents the exact tree that was generated for the dependencies of the project and is used to lock the version of the dependencies.

#### - package.json

This file typically contains the metadata and dependencies for the project.

#### - postcss.config.js

This file contains configuration for PostCSS, a tool for transforming styles with JavaScript plugins.

#### - tailwind.config.js

This file contains configuration for Tailwind CSS, a popular utility-first CSS framework.

## Guide

**### Git clone**

``git clone https://github.com/apollo812/mini-kanban.git``

``cd mini-kanban``

**### Run the project**

``make up``

- Fronend: ``hosting localhost:3000``

- GraphQL: ``hosting localhost:5000/grapgql``

**### Build the project**

``make build``

**## 4. Docker hub link**

**### Backend**

https://hub.docker.com/repository/docker/gluck0101/mini-kanban-backend

**### Frontend**

https://hub.docker.com/repository/docker/gluck0101/mini-kanban-frontend