Inventory App

The inventory is like a warehouse where the items can be added, deleted, and updated. It supports rest APIs to perform the crud operations in the inventory. This is a full-stack application where the user performs the crud operations from the browser.

Frontend:

The front end is written in React.js, HTML, and CSS. The front end consumes the APIs exposed by the backend server to perform the operations.

Backend:

The backend is written in Golang, Gin, and MySQL. The drivers are used to make connections to the database from the backend server.

Running Application:

There are two ways to run the application.

1. Using docker-compose
2. Running front end and backend independently

1. Using Docker-Compose:

The application can be started using the below docker-compose command
docker-compose up - -build

2. Running front end and backend independently

Running Frontend:

Use the below steps to run the frontend server.

1. cd client (navigate to client directory from root folder)
2. npm install (install the dependencies)
3. npm start (starting frontend client)

After running the above commands we can see the frontend server running at http://localhost:3000

Running Backend:

In order to start the server you should first export the following variables:

export DB_USER=<your-db-user>
export DB_PASSWORD=<your-db-password>

When you are running the application for the first time you will have to create a database manually using the script given under `db/init.sql`. Head over to your shell/terminal and connect to mysql

$ mysql -u <user> -p <password>

mysql >

Once you enter the mysql execute the following commands:

```
CREATE DATABASE IF NOT EXISTS items_db;
USE items_db;
CREATE TABLE IF NOT EXISTS `items` (
   `id`           int(11)        NOT NULL AUTO_INCREMENT,
   `name`         varchar(255)   NOT NULL,
   `quantity`     int(11)        NOT NULL,
   `unit_price`   decimal(10,2)  NOT NULL,
   PRIMARY KEY (`id`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8;
```

Now once tables are created we are good to start the server

```
cd server
go get .
go run .
```

We can find server running at http://localhost:8080

APIs:

The following table highlights the RESTful endpoints

Endpoint	Description
`GET /api/health`	Check the health of the server
`POST /api/item`	Creates a new item `{id, name, quantity, unit_price}`
`GET /api/item`	Retrieves the list of items in form `{"items": [...]}`
`GET /api/item/:id`	Retrieves the details of the item with given id in form `{"item": {id: "<id>", "name": "<name>", ...}}`
`DELETE /api/item/:id`	Deletes the item with given id
`PATCH /api/item/:id`	Update the item with given id
`GET /api/item/csv`	Retrieve the list of items as csv file