

## MID EXAM

Name: Mico Martin

Student\_id: 001202300179

Class: IT Class 2

### 1. Application Description

Money Manager is a personal finance tracking web application that helps users manage their income, expenses. The application allows users to create a track for daily transactions, categorize them, and view financial summaries, empowering them to make better budgeting decisions.

#### Features:

##### Data List

- Overview of total income, expenses, and balance

##### Add Data

- Add new income or expense entries
- Assign category (Food, Transport, Category, etc)
- Set transaction date

##### Analytics

- Pie chart breakdown of spending by category

### 2. Frontend

Library used: React, Tailwind CSS

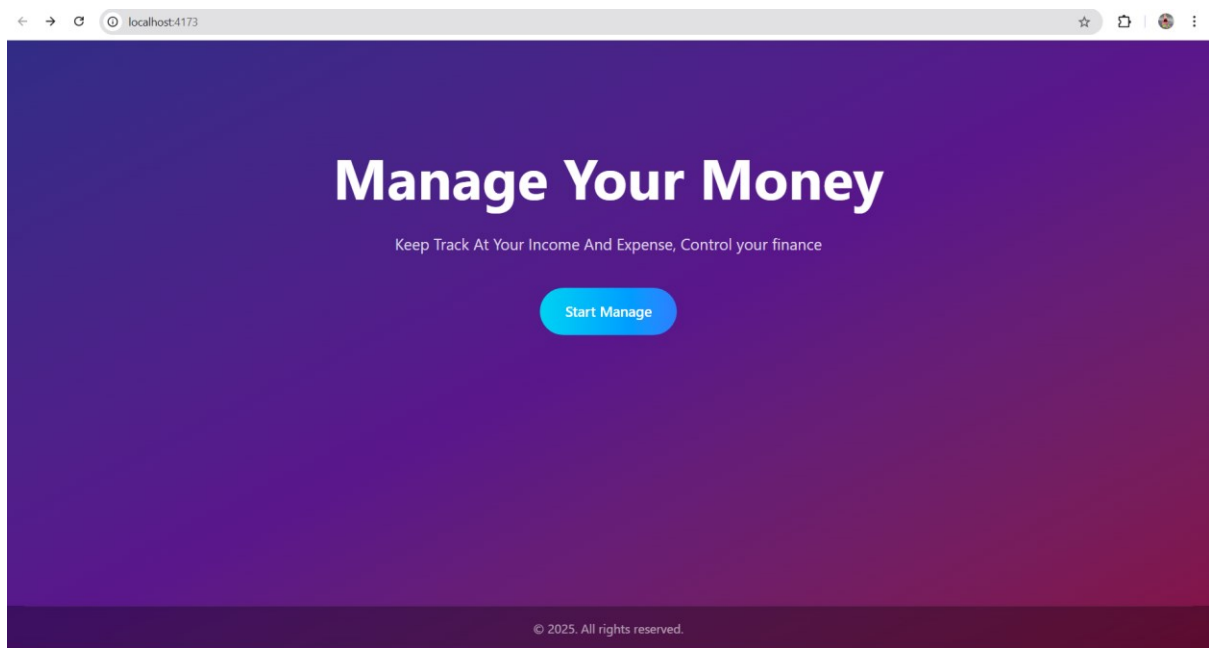
Main function:

UI Pages:

#### 1. Landing Page

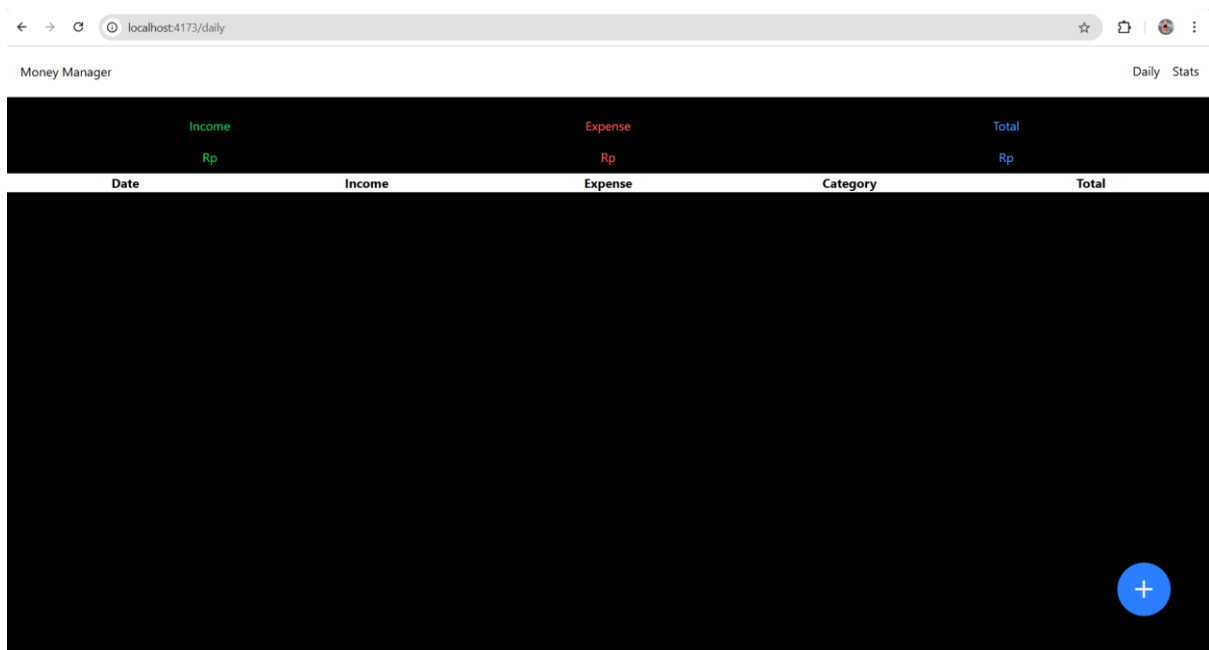
This is a landing page where the function is to serve as the entry point of the application, giving a simple interface with a simple description about the application and a button that

directs users to the dashboard.

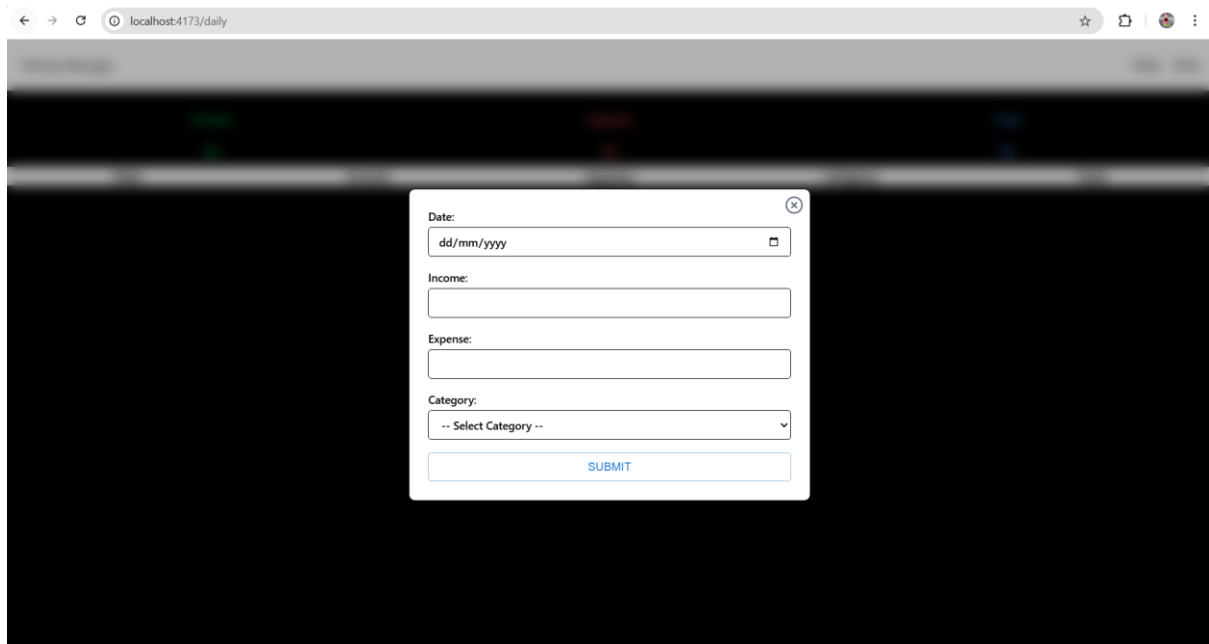


## 2. Main Page / DataList Page

In this page, user can add their data to track their income and expenses for a specific day. This page will serve the data based on the user input and calculate the total income, expense, and the total after subtraction.



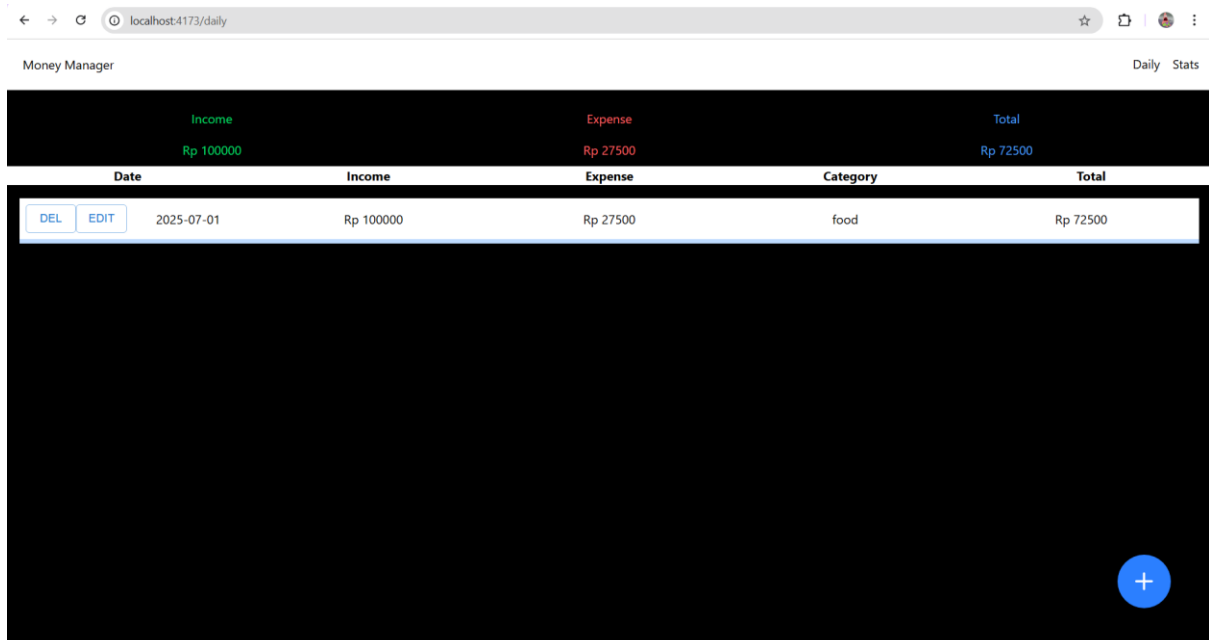
After clicking the plus button on the right below, an input form will appear on the screen, and the user can input their data.



The screenshot shows a web browser window with the address bar displaying 'localhost:4173/daily'. A modal form is centered on the screen, allowing the user to add a new transaction. The form includes the following fields:

- Date:** A text input field with a date picker icon, showing the placeholder 'dd/mm/yyyy'.
- Income:** A text input field.
- Expense:** A text input field.
- Category:** A dropdown menu with the placeholder '-- Select Category --'.
- SUBMIT:** A blue button at the bottom of the form.

After the user has input, the data will be displayed on the page. Where the list is provided with an edit and delete button. If the user wants to edit the data, they user can click the edit button. If the user wants to delete the data, they user can click the delete button.



The screenshot shows the 'Money Manager' application interface. At the top, there are tabs for 'Daily' and 'Stats'. Below the tabs, there is a summary section with three columns: 'Income' (Rp 100000), 'Expense' (Rp 27500), and 'Total' (Rp 72500). Below this, there is a table with the following columns: 'Date', 'Income', 'Expense', 'Category', and 'Total'.

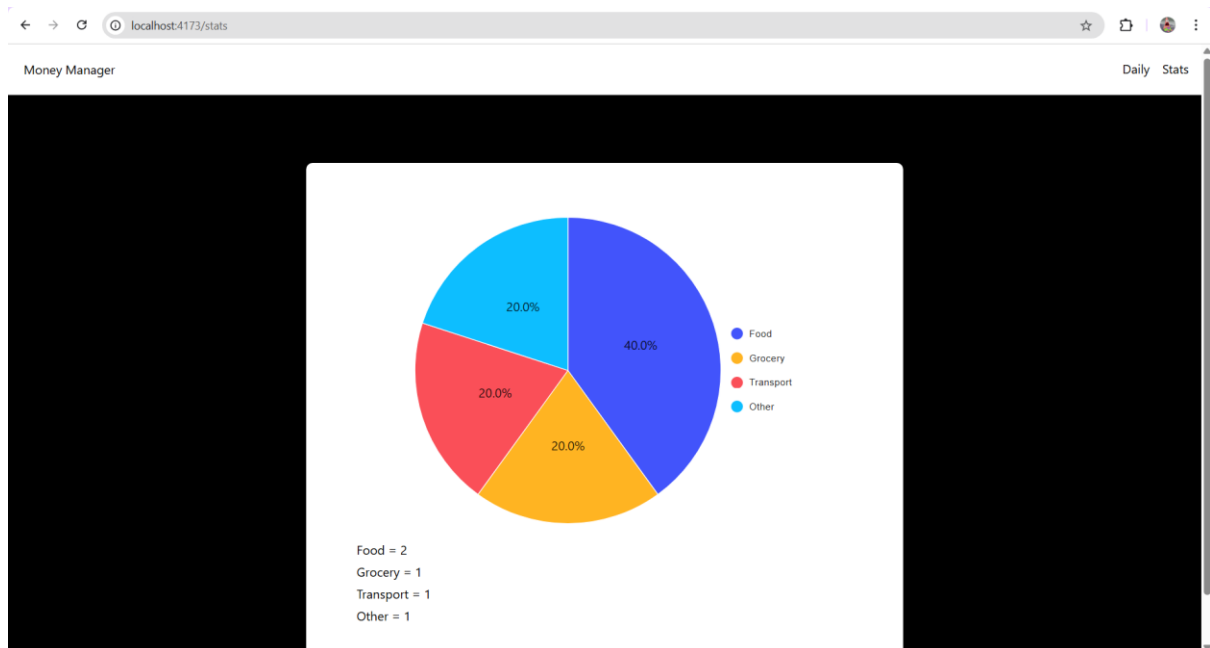
Date	Income	Expense	Category	Total
2025-07-01	Rp 100000	Rp 27500	food	Rp 72500

Each row in the table has two buttons: 'DEL' and 'EDIT'. A blue circular button with a white plus sign is located at the bottom right of the screen.



### 3. Stats Page

This page serves users with a pie chart, where the chart is based on the user category data. This chart shows to user what things they spend their money on what things.



### 3. Backend

language used: Go

API endpoint:

- /api/userData

The endpoint handles the data from the database, and sends the data to the frontend

- /api/getData

The endpoint handles data that is sent from the frontend, and inserts the data into the database

- /api/deleteData

Handle the delete data

- /api/editData

Handle data from the frontend, and use the data to update the database data

- /api/getData/total

Handle the total of income, expense, and total after subtraction

- /api/getCategory

Handle the pie chart data

Database Schema:

```
createTable := `
    CREATE TABLE IF NOT EXISTS daily_data(
    id SERIAL PRIMARY KEY ,
    date VARCHAR(100),
    income INT,
    expense INT,
    category VARCHAR(100)
    );`
```

#### 4. Deploying application

1. Create dockerfile in both frontend and backend folder

backend dockerfile

```
backend > dockerfile > ...
You, 3 days ago | 1 author (You)
1 FROM golang:1.23.3-alpine
2
3 WORKDIR /app
4
5 COPY go.mod go.sum ./
6 RUN go mod download
7
8 COPY . /app You, 5 days ago • init ..
9
10 RUN go build -o main
11
12 EXPOSE 8080
13
14 CMD ["/main"]
```

FROM: specify base images ( use golang since the backend using go)

WORKDIR: set the working directory inside the container which is /app

COPY go.mod go.sum: copy all the dependencies in go module to working directory in the container

RUN go mod download: downloads all the dependencies that listed in go modules



COPY ./app: copy all the file at the current directory to container working directory

RUN go build -o main: compiles go file

EXPOSE 8080: documents that the container listens on port 8080

CMD: start the backend server

frontend dockerfile

```
frontend >  dockerfile >  FROM
You, 3 minutes ago | 1 author (You)
1 FROM node:22.17.0-alpine You,
2
3 WORKDIR /app
4
5 COPY package.json .
6
7 RUN npm install
8
9 COPY ./app
10
11 RUN npm run build
12
13 EXPOSE 4173
14
15 CMD [ "npm", "run", "preview"]
```

FROM: specify base images (using node since react relies on node.js)

WORKDIR: set the working directory inside the container which is /app

COPY package.json: copy all the dependencies to working directory in the container

RUN npm install: downloads all the dependencies that listed in package.json

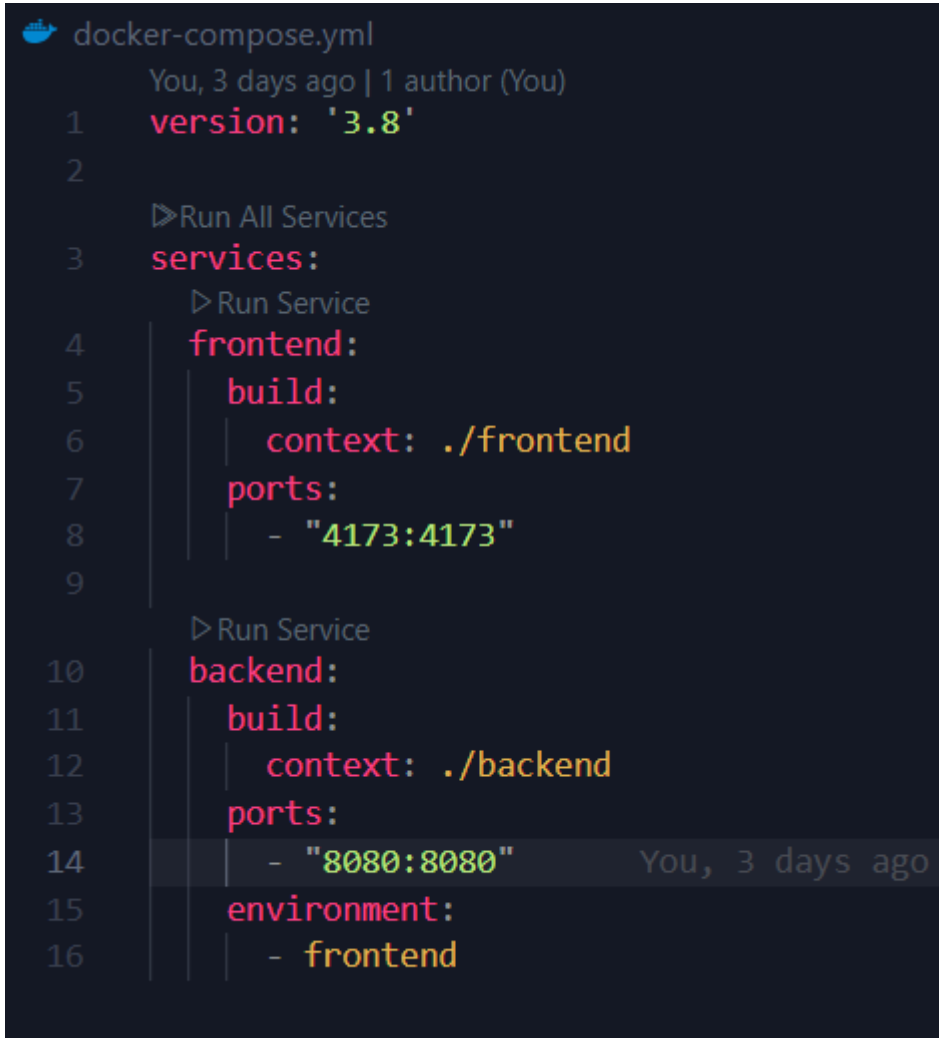
COPY ./app: copy all the file at the current directory to container working directory

RUN npm run build: compiles react file

EXPOSE 4173: documents that the container listens on port 4173

CMD: start the frontend server

2. Create docker-compose.yml in the root directory to manage the containers



```
1  version: '3.8'
2
3  services:
4    frontend:
5      build:
6        context: ./frontend
7      ports:
8        - "4173:4173"
9
10   backend:
11     build:
12       context: ./backend
13     ports:
14       - "8080:8080"
15     environment:
16       - frontend
```

Version: specify the docker version

Services: define all the application container

Context: ./frontend = tells docker to build images based on the frontend dockerfile

Port 4173: run the frontend server on port 4173

Context: ./backend = tells docker to build images based on the backend dockerfile

Port 8080: run the backend server on port 8080



3. Build the images using command “docker compose build” or directly run the container after build using command “docker compose up –build”

- Using docker compose build

```
PS D:\semester_6\distributed\mid_exam> docker compose build
time="2025-07-01T23:10:38+07:00" level=warning msg="D:\semester_6\distributed\mid_exam\docker-compose.yml: the attribute `version` is obsolete, it will be ignored, please remove it to avoid potential confusion"
[+] Building 12.0s (24/24) FINISHED
=> [backend internal] load build definition from dockerfile                                docker:desktop-linux 0.0s
=> => transferring dockerfile: 198B                                                    0.0s
=> [frontend internal] load build definition from dockerfile                            0.0s
=> => transferring dockerfile: 206B                                                    0.0s
=> [backend internal] load metadata for docker.io/library/golang:1.23.3-alpine          2.3s
=> [frontend internal] load metadata for docker.io/library/node:22.17.0-alpine         2.2s
=> [frontend internal] load .dockerignore                                              0.0s
=> => transferring context: 2B                                                         0.0s
=> [backend internal] load .dockerignore                                              0.0s
=> => transferring context: 2B                                                         0.0s
=> [frontend 1/6] FROM docker.io/library/node:22.17.0-alpine@sha256:5340cbfc2df14331ab021555fdd9f83f072ce811488e705b0e736b11adeec4bb 0.0s
=> [frontend internal] load build context                                            1.7s
=> => transferring context: 1.28MB                                                    1.7s
=> [backend 1/6] FROM docker.io/library/golang:1.23.3-alpine@sha256:c694a4d291a13a9f9d9493395673494fc2cc9d4777b85df3a7e70b3492d3574 0.0s
=> [backend internal] load build context                                            0.0s
=> => transferring context: 7.08kB                                                    0.0s
=> CACHED [backend 2/6] WORKDIR /app                                                 0.0s
=> CACHED [backend 3/6] COPY go.mod go.sum ./                                        0.0s
```

- using docker compose up –build

Local deployment using docker is successfull

```
=> => exporting layers                                                                0.0s
=> => writing image sha256:d1554538e9f1adea7fc6c5afd2eddef277b77082c7c13d485c7b315a34acdb47 0.0s
=> => naming to docker.io/library/mid_exam-frontend 0.0s
=> [frontend] resolving provenance for metadata file 0.0s
[+] Running 2/2
✓ Container mid_exam-frontend-1 Created 0.0s
✓ Container mid_exam-backend-1 Recreated 0.2s
Attaching to backend-1, frontend-1
backend-1 | 2025/07/01 16:12:05 <nil>
backend-1 | 2025/07/01 16:12:05 succesfully create table
backend-1 | 2025/07/01 16:12:05 port in 8888
frontend-1 |
frontend-1 | > frontend@0.0.0 preview
frontend-1 | > vite preview --host
frontend-1 |
frontend-1 | → Local: http://localhost:4173/
frontend-1 | → Network: http://172.19.0.2:4173/
View in Docker Desktop View Config Enable Watch
```