2468429
ALİ TUNAHAN GÜNER

CENG495 – HW2 REPORT

1. Deployment Workflow Using Docker and Docker Compose

   a) Building images
      - Each service has its own Dockerfile in its respective directory. Dockerfiles include instructions to build images using a Python 3.10-slim base image, set up the working directory, copy the source code and .env files, install dependencies, expose necessary ports, and definee the command to run the application.

   b) Configuration via docker-compose.yml
      - Version: Specifies the Docker Compose file format version
      - Networks: Two networks are defined, namely frontend and backend
      - Services: Defines five separate services corresponding to different comoonenets of the application.Each service configuration includes the following:
        1- Build: Path to Dockerfile of the service
        2- Ports: Port mapping between the host and the container
        3- Networks: Specifies which network the service is attached to
        4- Env_file: Specifies environment files that are to be used by the service

   c) Building and running the services
      - Running ***docker-compose build*** would build and ***docker-compose up*** would run the services as configured.
      - Each service then communicate with others on the same network and the defined portsa re exposed to the host, allowing external Access where nnecessary.

   d) Environment Variables
      - The .env file included in each service configuration allows the system to manage configuration and secrets

efficiently and securely without hardcoding them into the Dockerfile or application code.

e) Management

- Docker Compose also facilitates the management of the entire application lifecycle. Commands like ***docker-compose start***, ***docker-compose stop*** and ***docker-compose down*** allow for easy starting, stopping and removing all services defined in the compose file.

2. System Arcitecture Overview

   The system consists of several interacting services, each running in a Docker container.

   - Auth: Manages authentication processes.
   - Others: A generic service that could handle various tasks.
   - Add_expense: Manages operations related to adding expenses.
   - Add_expense_group: Handlles operations related to managing expense groups.
   - Database: Handles database interaction

3. Decisions made & challenges encountered during the deployment process and their justifications:
   - I have used Flask while developing backend and basic Javascript/HTML/CSS while developing frontend for their ease of use and debug.
   - Since I have familiarity with the mongoDB tech, I also used it in this homework. I wasn’t able to deploy a local database, and so instead I have used Atlas’s mongodb as a service.

4. Screenshots of the Docker and Docker Compose usage:

2468429
ALİ TUNAHAN GÜNER

-

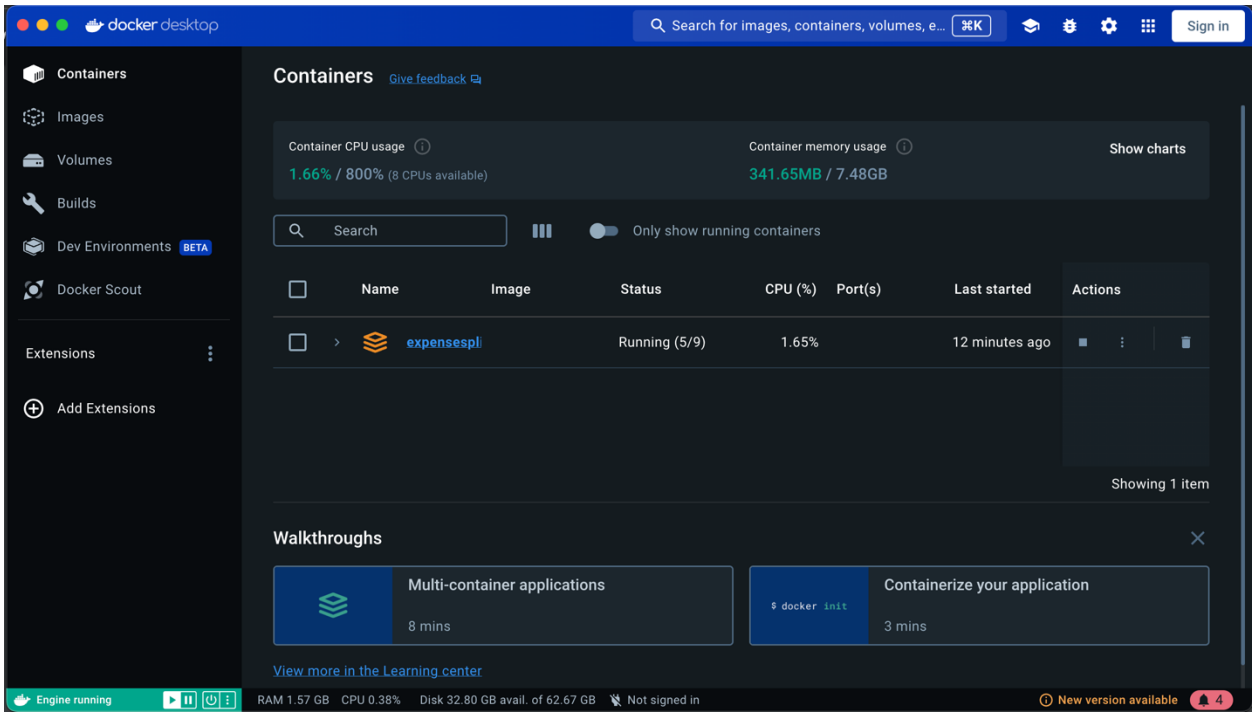
```
cw@192 expenseSplitter % docker-compose build
[+] Building 2.3s (42/42) FINISHED
 => [add_expense internal] load build definition from Dockerfile
 => => transferring dockerfile: 622B
 => [database internal] load build definition from Dockerfile
 => => transferring dockerfile: 613B
 => [add_expense internal] load metadata for docker.io/library/python:3.10-slim
 => [add_expense-group internal] load build definition from Dockerfile
 => => transferring dockerfile: 641B
 => [others internal] load build definition from Dockerfile
 => => transferring dockerfile: 606B
 => [auth internal] load build definition from Dockerfile
 => => transferring dockerfile: 601B
 => [others internal] load .dockerignore
 => => transferring context: 2B
 => [add_expense internal] load .dockerignore
 => => transferring context: 2B
 => [database internal] load .dockerignore
 => => transferring context: 2B
 => [add_expense-group internal] load .dockerignore
 => => transferring context: 2B
 => [auth internal] load .dockerignore
 => => transferring context: 2B
 => [others 1/5] FROM docker.io/library/python:3.10-slim@sha256:161d1e3deddba156d7834d15e6fe4d010b3b6f69f18ee1cfba9348e712f4a4ba
 => [add_expense internal] load build context
 => => transferring context: 567B
 => [auth internal] load build context
 => => transferring context: 408B
 => [add_expense-group internal] load build context
 => => transferring context: 573B
```

docker-compose build command

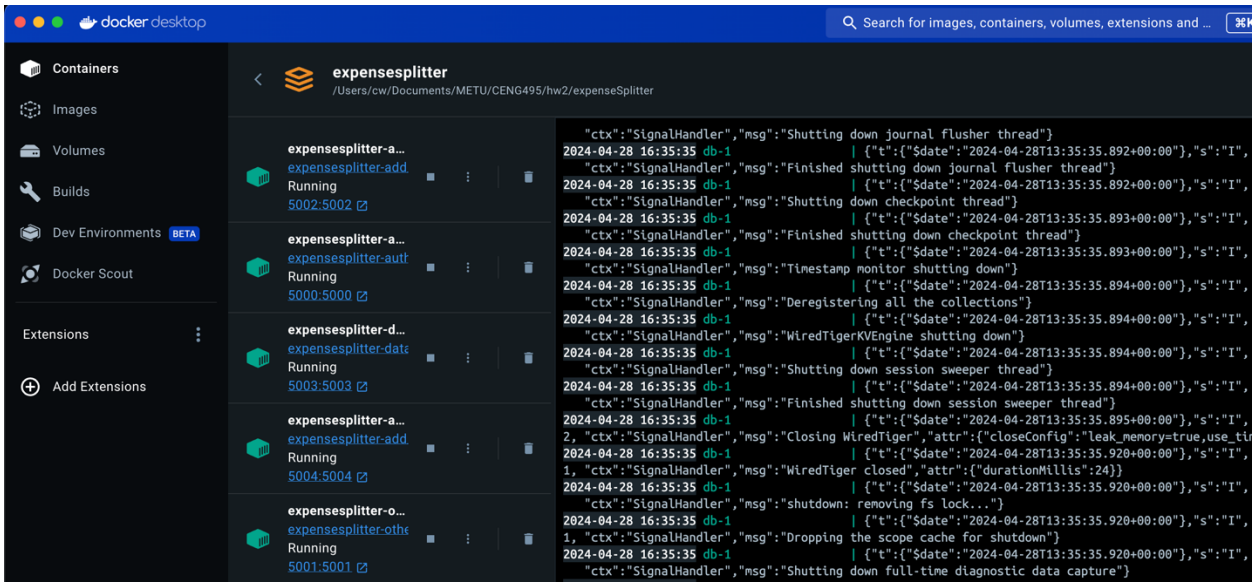
-

```
cw@192 expenseSplitter % docker-compose up
WARN[0000] Found orphan containers ([expensesplitter-dbdummy-1 expensesplitter-db-1 expensesplitter-dbdummy3-1 expensesplitter-dbdummy4-1]) for this pro
un this command with the --remove-orphans flag to clean it up.
[+] Running 5/0
 ✔ Container expensesplitter-auth-1              Created
 ✔ Container expensesplitter-database-1          Created
 ✔ Container expensesplitter-add_expense-group-1  Created
 ✔ Container expensesplitter-others-1            Created
 ✔ Container expensesplitter-add_expense-1       Created
Attaching to add_expense-1, add_expense-group-1, auth-1, database-1, others-1
add_expense-1          |  * Serving Flask app 'app'
add_expense-1          |  * Debug mode: on
add_expense-1          | WARNING: This is a development server. Do not use it in a production deployment. Use a production WSGI server instead.
add_expense-1          |  * Running on all addresses (0.0.0.0)
add_expense-1          |  * Running on http://127.0.0.1:5002
add_expense-1          |  * Running on http://172.28.0.4:5002
add_expense-1          | Press CTRL+C to quit
add_expense-1          |  * Restarting with stat
add_expense-group-1    |  * Serving Flask app 'app'
add_expense-group-1    |  * Debug mode: on
add_expense-group-1    | WARNING: This is a development server. Do not use it in a production deployment. Use a production WSGI server instead.
add_expense-group-1    |  * Running on all addresses (0.0.0.0)
add_expense-group-1    |  * Running on http://127.0.0.1:5004
add_expense-group-1    |  * Running on http://172.28.0.3:5004
add_expense-group-1    | Press CTRL+C to quit
add_expense-group-1    |  * Restarting with stat
auth-1                 |  * Serving Flask app 'app'
auth-1                 |  * Debug mode: on
others-1               |  * Serving Flask app 'app'
others-1               |  * Debug mode: on
auth-1                 | WARNING: This is a development server. Do not use it in a production deployment. Use a production WSGI server instead.
auth-1                 |  * Running on all addresses (0.0.0.0)
auth-1                 |  * Running on http://127.0.0.1:5000
auth-1                 |  * Running on http://172.28.0.6:5000
auth-1                 | Press CTRL+C to quit
auth-1                 |  * Restarting with stat
others-1               | WARNING: This is a development server. Do not use it in a production deployment. Use a production WSGI server instead.
others-1               |  * Running on all addresses (0.0.0.0)
others-1               |  * Running on http://127.0.0.1:5001
others-1               |  * Running on http://172.28.0.5:5001
others-1               | Press CTRL+C to quit
others-1               |  * Restarting with stat
add_expense-1          |  * Debugger is active!
add_expense-1          |  * Debugger PIN: 622-120-945
add_expense-group-1    |  * Debugger is active!
add_expense-group-1    |  * Debugger PIN: 356-025-869
auth-1                 |  * Debugger is active!
auth-1                 |  * Debugger PIN: 353-161-503
others-1               |  * Debugger is active!
others-1               |  * Debugger PIN: 252-026-028
database-1             | Pinged your deployment. You successfully connected to MongoDB!
database-1             |  * Serving Flask app 'app'
database-1             |  * Debug mode: on
database-1             | WARNING: This is a development server. Do not use it in a production deployment. Use a production WSGI server instead.
database-1             |  * Running on all addresses (0.0.0.0)
database-1             |  * Running on http://127.0.0.1:5003
database-1             |  * Running on http://172.28.0.2:5003
database-1             | Press CTRL+C to quit
database-1             |  * Restarting with stat
database-1             |  * Debugger is active!
database-1             |  * Debugger PIN: 212-230-481
```

docker-compose up command

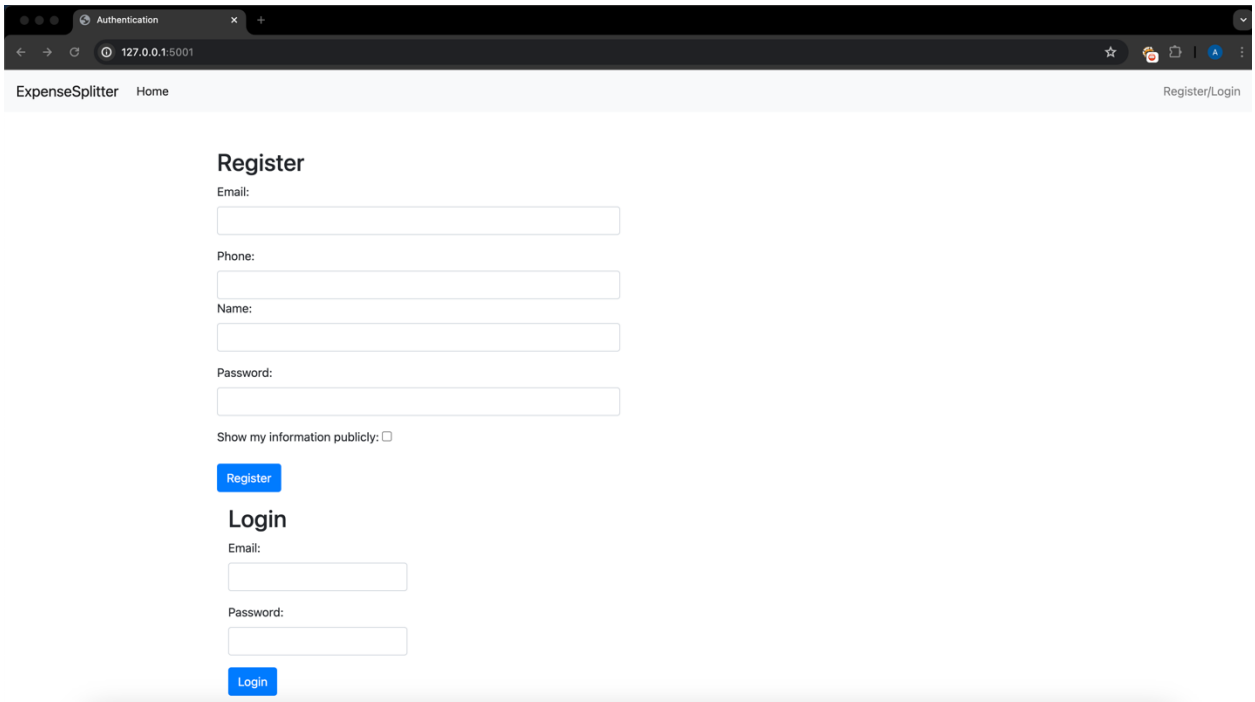
- 

Docker Desktop

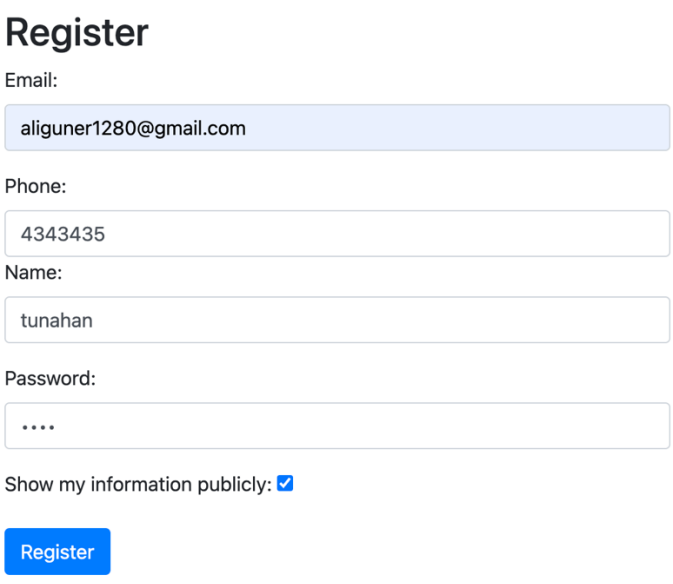
- 

Docker Desktop-2

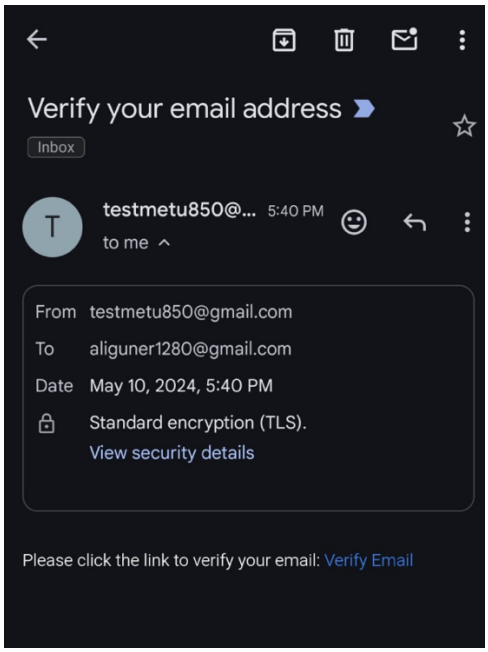
5. Screenshots of the Localhost deployment

- 

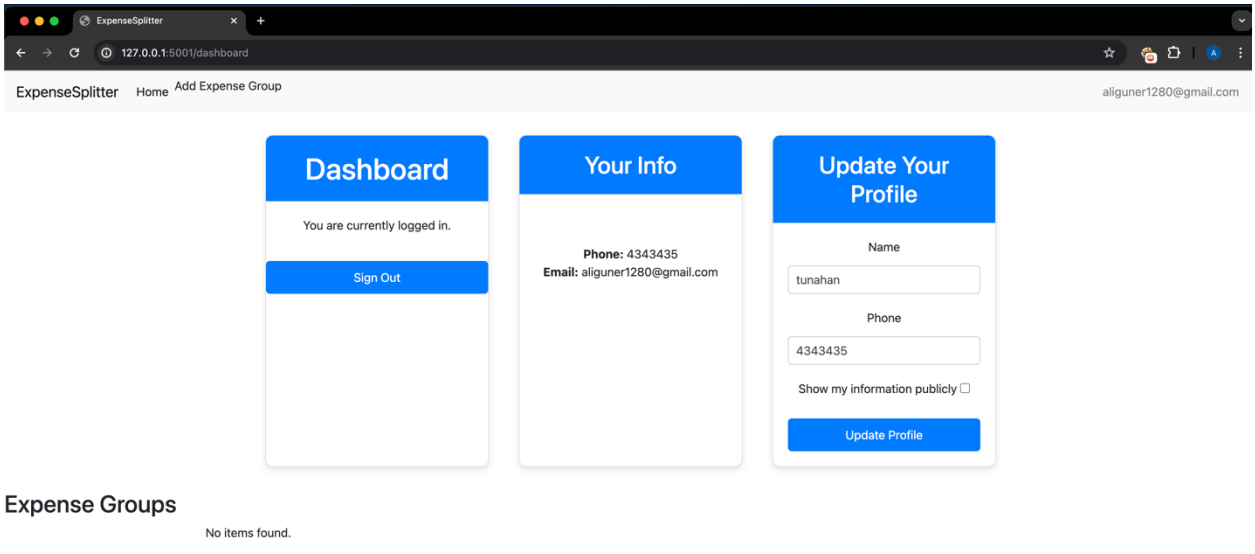
Main Page

- 

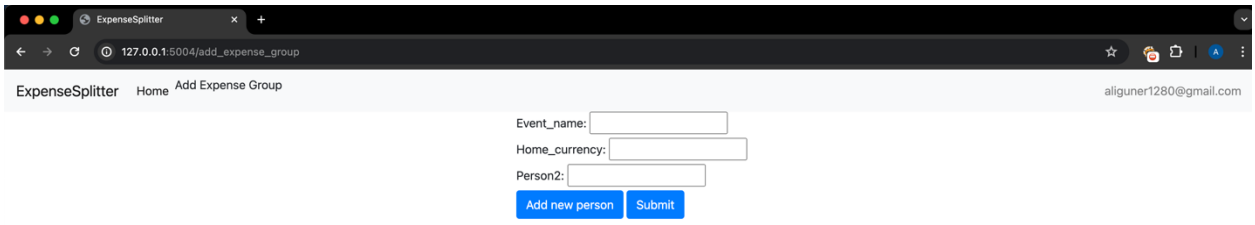
Register

- 

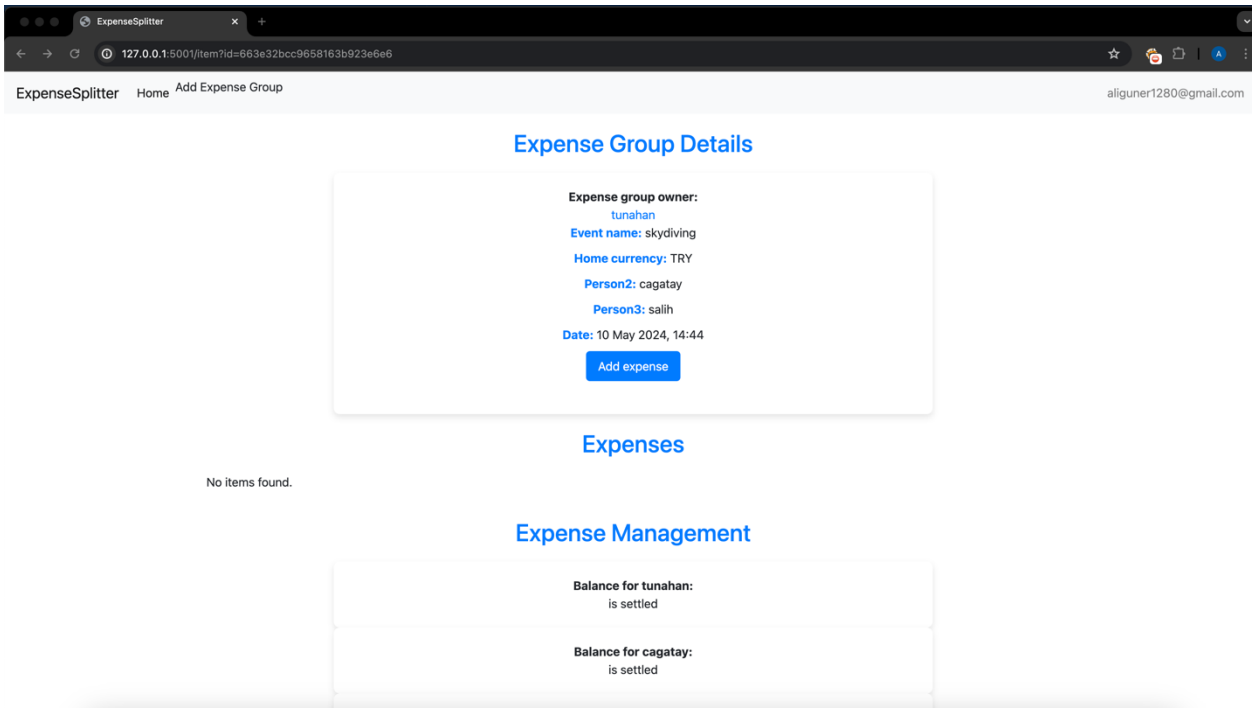
Confirmation Mail

- 

After Login

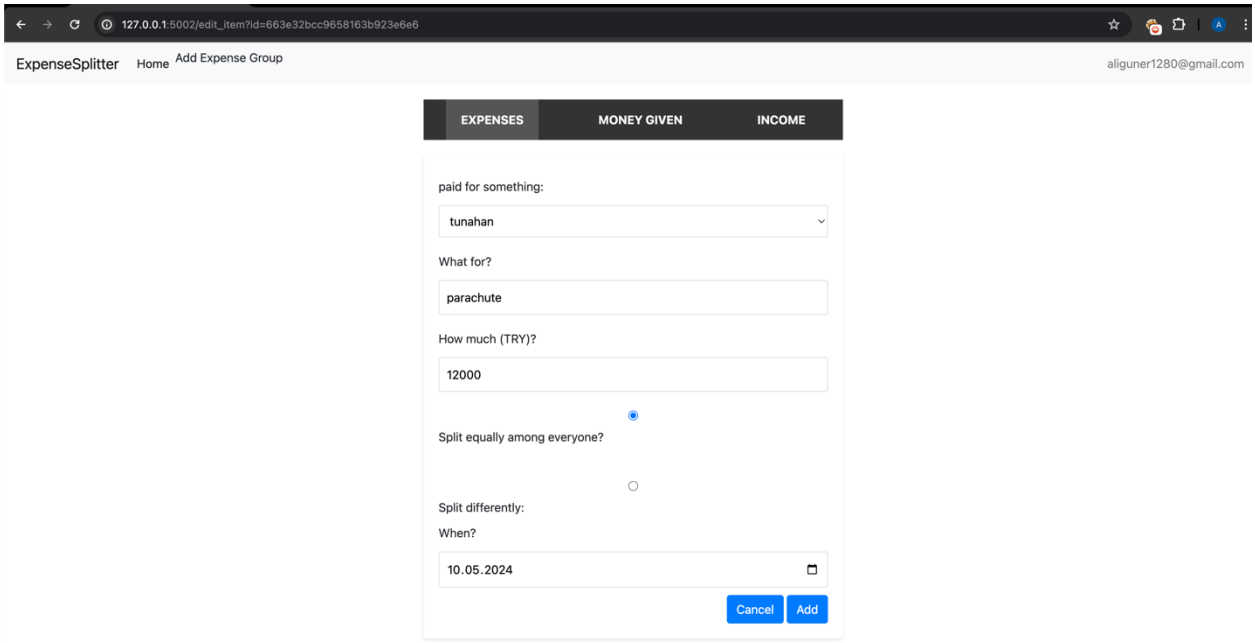
- 

Add Expense Group



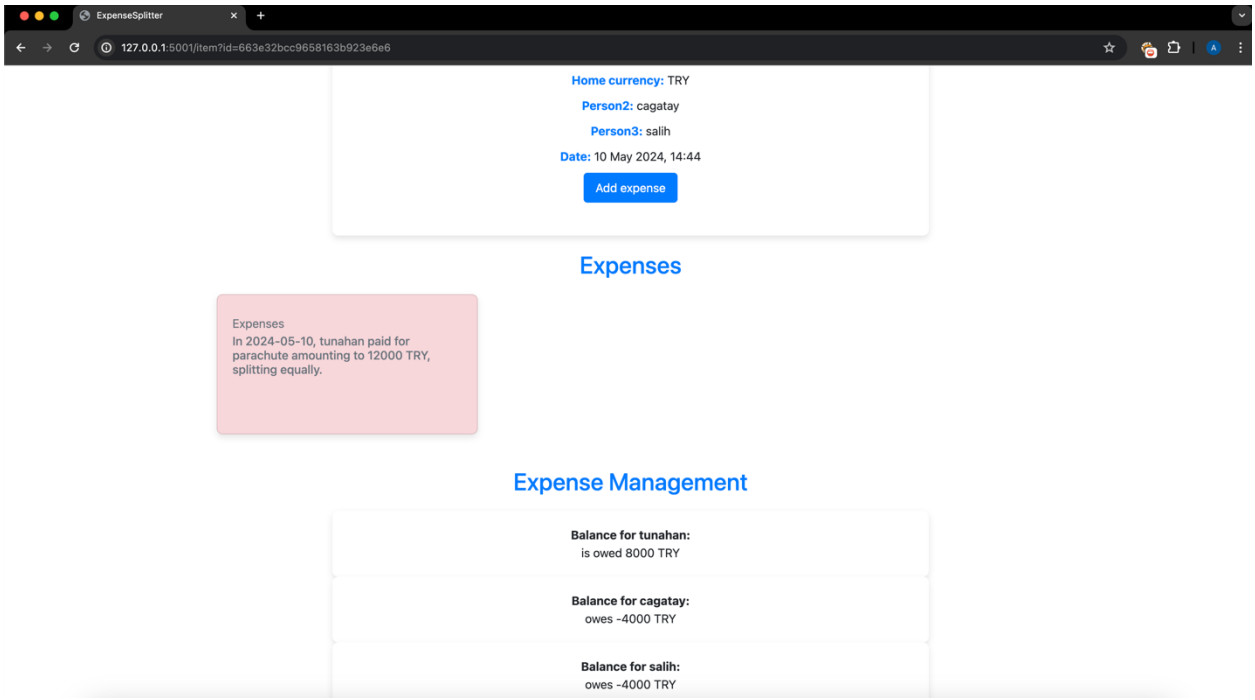
-

Viewing expense group



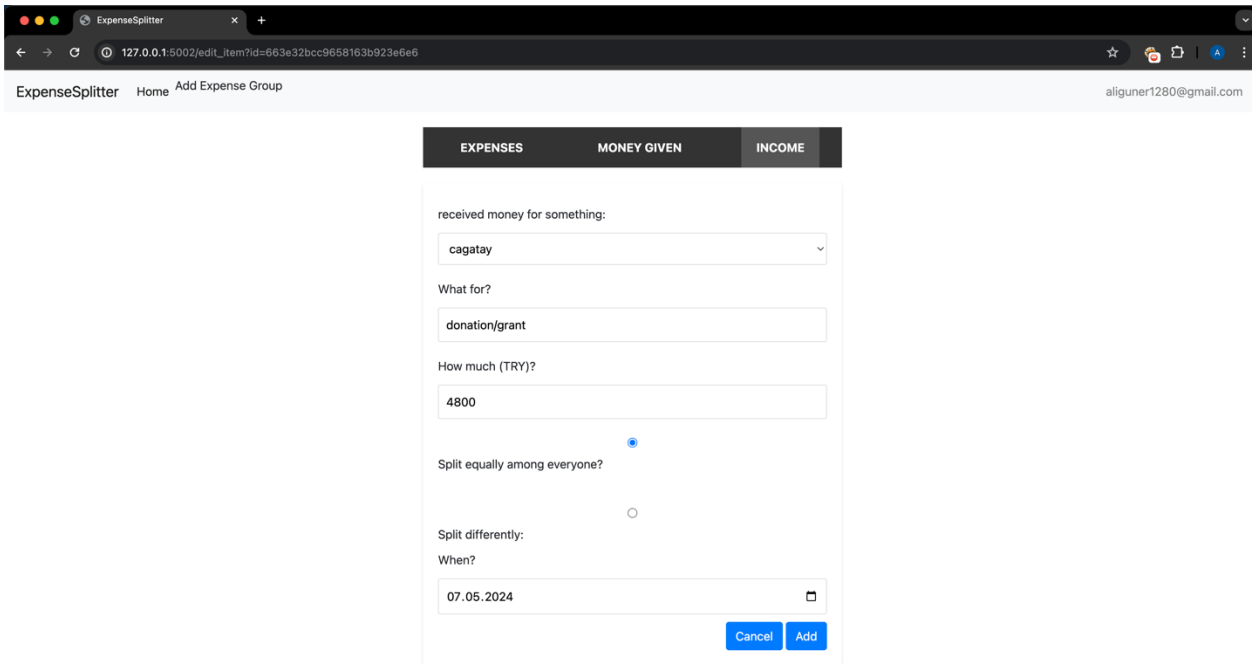
-

Add Expense



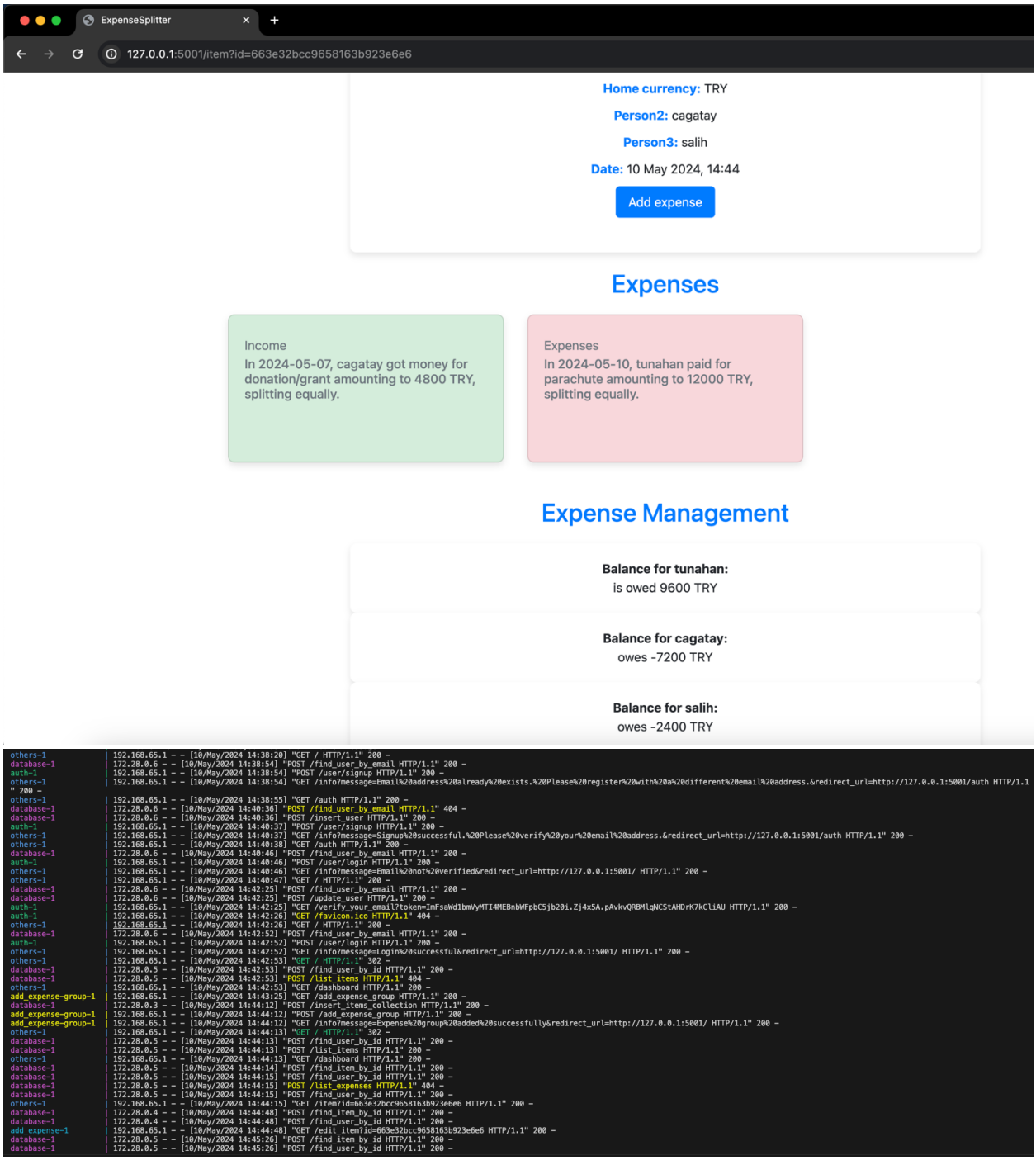
-

View expense group after adding an expense

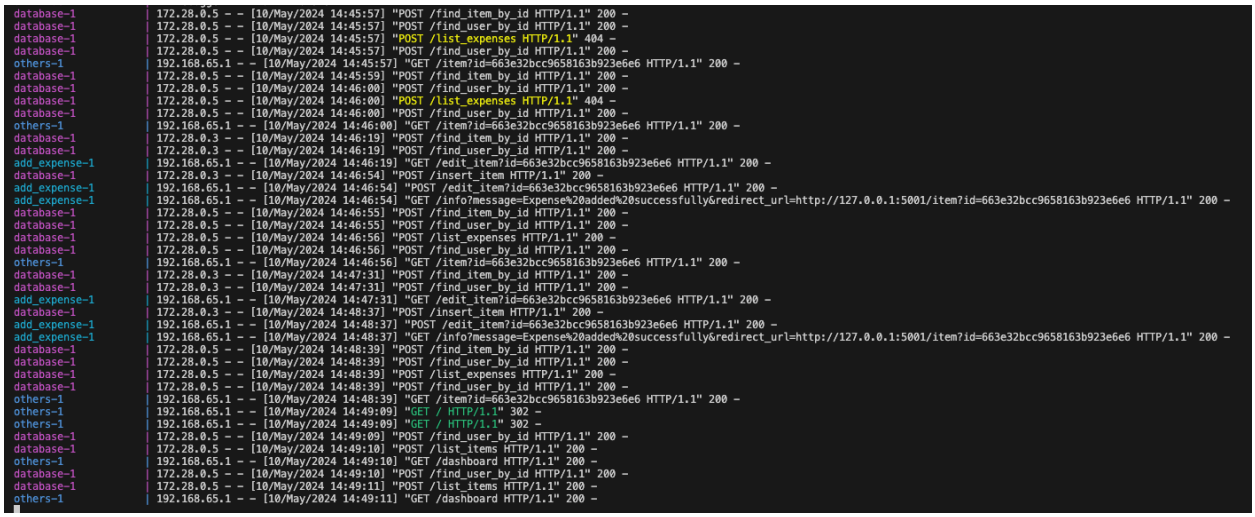


-

Add Income



Backend logs-1



Backend logs-2

6. System Architecture Diagram