

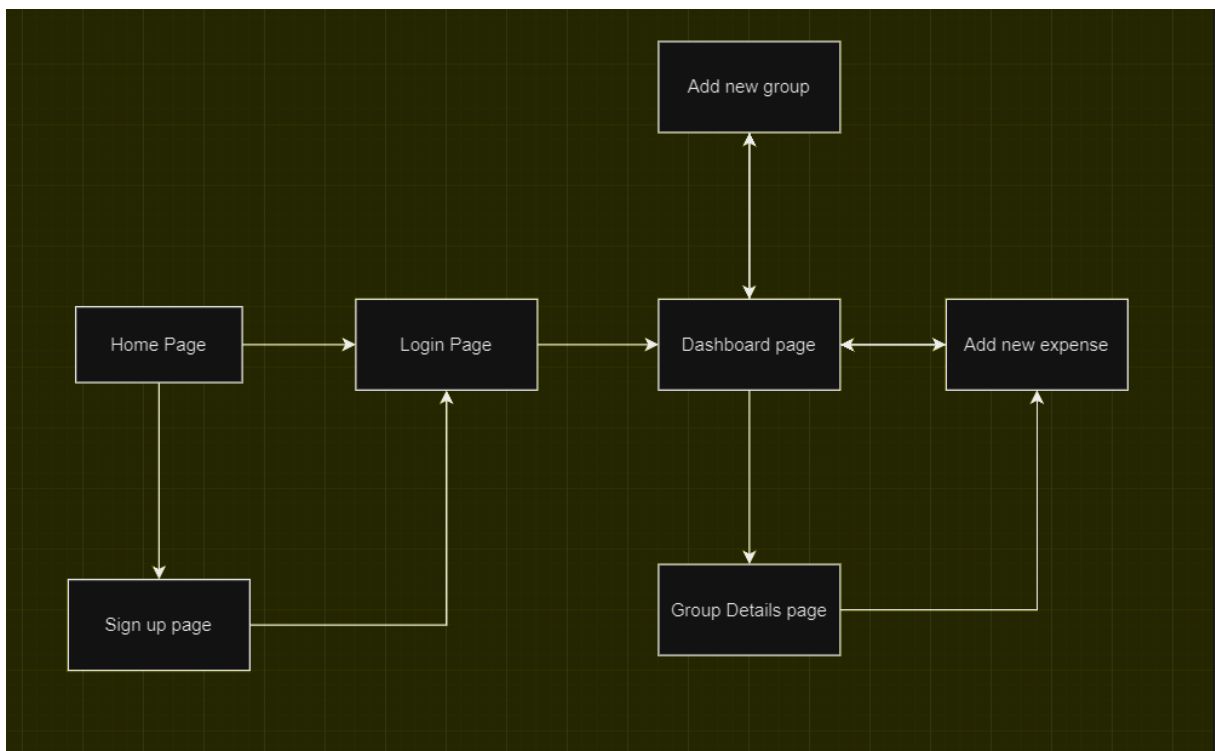
# CENG495 HW2 REPORT

**Emre KAPLAN 2380533**

In this assignment I implemented an Expense Splitter web application and deployed it using Docker containers and Docker Compose for local deployment. I used flask framework of Python for the Project. In addition, HTML and JS are used for Front-end. For the database issues, I used SQLAlchemy since it is easy to use for small projects.

You can find all the application related screenshots in the “Screenshots” folder.

Here is the system diagram of my application:

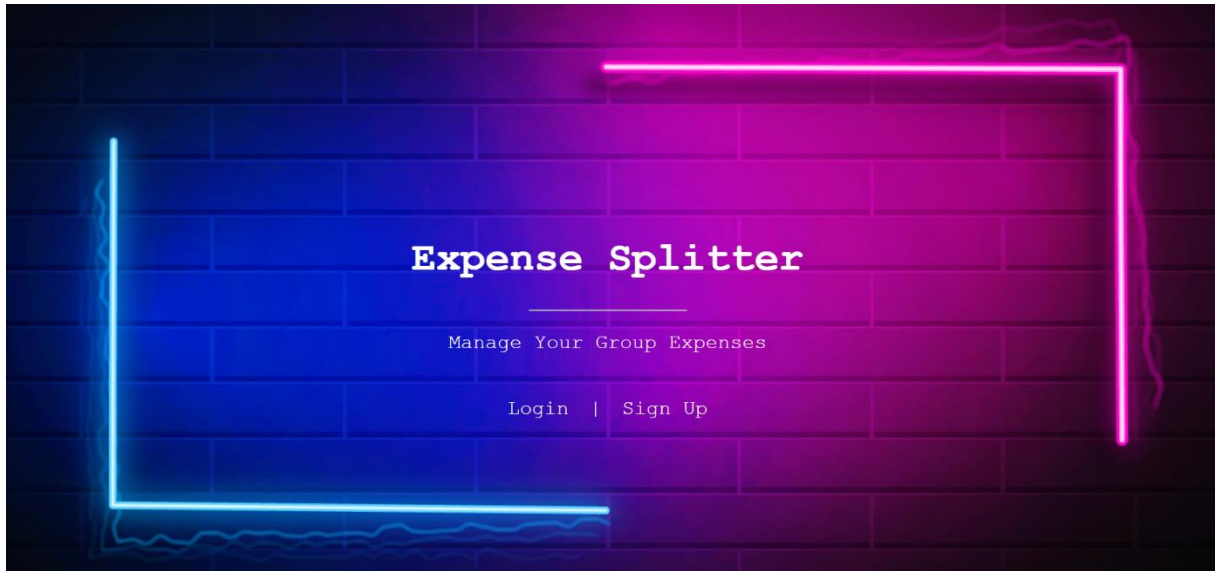


*System diagram*

Here is some related pages and their challenges for me while coding:

## **Homepage**

Homepage was easy to implement. Since it doesn't include any specific or complex algorithm.



*Homepage*

## **Sign up and login**

For sign up and login mechanisms, I used werkzeug.security's hashing mechanism. Users can sign up via their e-mail, username and password. They can login via e-mail and password.

A screenshot of a web form titled 'Register'. The form is enclosed in a light gray border. It contains three input fields: 'Full Name' with a placeholder 'Name', 'Email' with a placeholder 'Email', and 'Password' with a placeholder 'Password'. To the right of the password field is a blue button with a white eye icon. Below the password field is a blue 'Register' button.

*Sign up(Register)*

Login

Email

Email

Password

Password

Login

Don't have an account? [Signup](#)

Login

## Dashboard

In the dashboard page, users can create a new group, create a new expense for the existing groups and see existing group details.

Expense Splitter

🔍 mranewliz 🗑 Logout

Created Events

+ Create new Expense

Group1

Created on 1111-11-11

Add ExpenseShow Details

Group2

Created on 1111-11-11

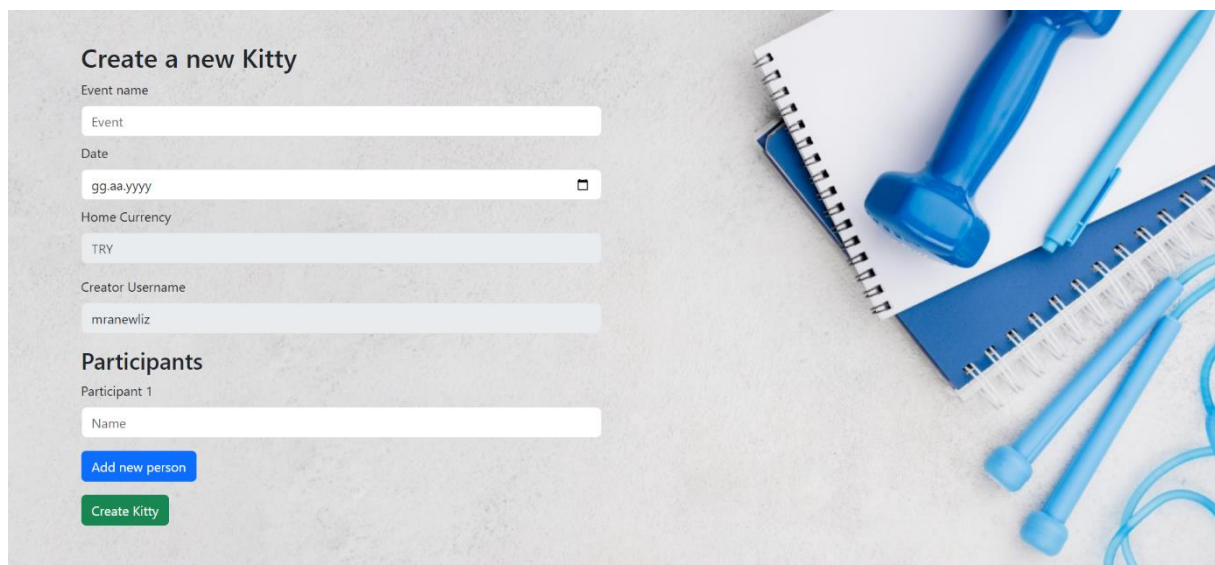
Add ExpenseShow Details

Dashboard

## Add new group

In this page, user can create a new group with event name, date, participants details. Currency is TRY as default. Creator of the group is the current user as default. Users need to include their own name if they are in the group that he/she is going to create. Thanks to this flexibility, users can create groups such that they are not the member of the group(creator is like a director or manager of group).

While coding this page, it was hard to design and determine the models' details since it includes lots of details while getting them in the expense group details page and add expense page.



**Create a new Kitty**

Event name

Date

Home Currency

Creator Username

**Participants**

Participant 1

[Add new person](#)

[Create Kitty](#)

*Add new group*

## Add new expense

In this page, I implemented that creating a new expense to the chosen group. Users can create the expense via the description, amount, date, payer and splitting details. User can choose the person who payed the expense from the group members. In addition, user can choose between “split equally” and “split differently” choices. When the user choose to split differently, he/she needs to indicate that who need to pay for the expense.

While coding this part, it was hard to implement selection parts and their database connections.

### Add Expense to Group2

Description

Enter description

₺ Amount

Enter amount in Turkish Lira

Date

gg.aa.yyyy

Payed by

P1

Split Equally: ☐ Split Differently: ☒

☐ P1

☐ P2

☐ P3

☐ P4

Add Expense

Add new expense

## Expense group details

In this page, users can see the details of the group such as total expense amount, name of group members, final payment details etc. While coding this part it was hard to implement the algorithm of final payments with the minimum transactions and its database connections.

### Expense Details: Group2

Total Expense

875.0

Group Members

- P1
- P2
- P3
- P4

Total results of the payments

- P2 need to pay 75.0 TL to P1
- P3 need to pay 50.0 TL to P1
- P3 need to pay 225.0 TL to P4

Expense Distribution Details

Add Expense

Description	Amount per person	Payed by	Group Members with Share	Creation Date
deneme1	250.0	P1	<ul style="list-style-type: none"><li>P1</li><li>P2</li><li>P3</li></ul>	1111-11-11

Group details

## Docker details

Docker and docker compose is implemented successfully.

It can be used via commands:

> docker build -t expense-splitter .

> docker-compose up

Some related screenshots are below:

The first screenshot shows a terminal window with the following output:

```
1. Sign in to your Docker account → docker login
2. View a summary of image vulnerabilities and recommendations → docker scout quickview
PS C:\Users\EMRE KAPLAN\Desktop\ceng495\hw2\expense_splitter> docker-compose up
time="2024-05-05T23:30:27+03:00" level=warning msg="C:\Users\EMRE KAPLAN\Desktop\ceng495\hw2\expense_splitter\docker-compose.yaml: `version` is obsolete"
[+] Running 0/1
- Container expense_splitter-web-1 Recreated
Attaching to web-1
web-1 | * Serving Flask app 'main.py'
web-1 | * Debug mode: off
web-1 | WARNING: This is a development server. Do not use it in a production deployment. Use a production WSGI server instead.
web-1 | * Running on all addresses (0.0.0.0)
web-1 | * Running on http://127.0.0.1:5000
web-1 | * Running on http://172.18.0.2:5000
web-1 | Press CTRL+C to quit
```

The second screenshot shows the Docker Desktop interface. The left sidebar contains the following menu items: Containers, Images, Volumes, Builds, Dev Environments (BETA), Docker Scout, Extensions, and Add Extensions. The main panel is titled 'Builds' and shows a table of build history for the 'expense\_splitter' image.

ID	Name	Builder	Status	Duration	Created	Author
5D996X	expense_splitter	default	Completed	10.5s	1 minute ago	N/A
SP3CV2	expense_splitter	default	Completed	12.7s	3 days ago	N/A
HYOK8E	expense_splitter	default	Completed	17.5s	5 days ago	N/A

