1 Introduction

For this assignment, you will be tasked with developing an Expense Splitter web application and deploying it using Docker containers and Docker Compose for local deployment. Your objective is to document the deployment process thoroughly, prepare a detailed report outlining your steps and solutions, include relevant code submissions, and provide screenshots to showcase key aspects of the application's functionality.

2 Expense Splitter Web Application

2.1 Application Overview

The Expense Splitter web application is a collaborative tool designed to simplify the management and equitable distribution of shared expenses among multiple participants. It provides a user-friendly interface for organizing and tracking expenses incurred during group activities, such as trips, events, or shared living arrangements.

Similar to KittySplit, the Expense Splitter application allows users to:

- Create expense groups for different activities or events.
- Add and categorize expenses with details such as amount, date, and payer.
- Assign expenses to specific participants and calculate individual balances.

The application aims to streamline the process of splitting expenses, reducing the complexity of managing shared financial obligations within groups. By leveraging modern web technologies and microservices architecture, the Expense Splitter web application provides a scalable and efficient solution for collaborative expense management.

2.2 Features:

- **Homepage:** The landing page exclusively presents the **login and sign-up options** for accessing the application.
- **User Signup:** New users can create an account by providing necessary details such as name, email, and password.
- **User Login:** Existing users can securely log in using their credentials to access their account and manage expense groups.
- **Dashboard:** After logging in, users are directed to a personalized dashboard displaying a list of expense groups they have created.
- **Expense Group Management:** Users can create new expense groups using the expense group name and date of creation. **No delete or update is required** for expense groups once added.
- **Expense Group Details:** Clicking on an expense group within the dashboard reveals detailed information, including listed expenses, participants, total amount of expenses, and balance distributions among participants.
- **Expense Management:** Users can add expenses within their expense groups, triggering real-time updates to debt calculations and balance distributions. When adding expenses, users specify details such as the expense name, amount (in Turkish Lira), expense date and time, and the participant who paid for the expense. **No delete or update is required** for expenses once added.

2.3 Debt Calculation and Expense Management

The application dynamically calculates and displays who owes whom based on the assigned expenses within each expense group. Whenever a new expense is added, the system automatically updates the owed amounts among participants to reflect the latest expense distributions.

2.4 Development Details

You can use any programming language and framework/library for backend and frontend development according to your preferences and project requirements. Additionally, you are encouraged to **utilize code and concepts from Homework 1** to support your development process.

3 Deployment

3.1 Implementing Microservices Architecture with Docker Compose

We use Docker Compose to orchestrate multiple Docker containers, with **each container encapsulating specific functionalities of the Expense Splitter web application as microservices**. This approach allows us to assign distinct responsibilities to different containers, facilitating easier maintenance and scalability.

3.2 Containerization with Docker and Docker-Compose

Objective:

- Containerize the Expense Splitter web application using Docker and `docker-compose`.

Task Description:

1. Define services and dependencies for the Expense Splitter application in a `docker-compose.yml` file.
2. Configure environment variables, networks, and volumes as needed for each service.
3. Build Docker images for the application services using `docker-compose build`.
4. Start the containers and services using `docker-compose up`.
5. Verify that all containers are running and accessible within the Docker network.

3.3 Exposure

Objective:

- Expose the Expense Splitter application locally via ports defined in the `docker-compose.yml` file.

Task Description:

1. Identify the exposed ports for accessing the application services.
2. Access the application in a web browser using the specified port (e.g., `localhost:8080`).

3.4 Validation

Objective:

- Validate the deployment and functionality of the Expense Splitter application using `docker-compose`.

Task Description:

1. Access the Expense Splitter web application via a web browser on `localhost`.
2. Test the functionality of the application, including creating expense groups, adding expenses, and viewing balances.
3. Monitor logs and container status using `docker-compose logs` and `docker-compose ps` commands.

3.5 System Design Diagram

Objective:

- Visualize the architecture and container layout of the Expense Splitter web application.

Diagram Description:

- Create a system design diagram illustrating the Docker containers, services, and their interactions.
- Include labeled components such as frontend, backend, database, and any other services defined in `docker-compose`.

Note: Include the System Design Diagram in your report for a comprehensive overview of the application architecture.

4 Documentation and Submission

To prepare your submission for the Expense Splitter web application deployment assignment, follow these guidelines:

4.1 Code Submission

Include all relevant code files necessary for the deployment of the Expense Splitter application. This includes:

- Frontend code (e.g., HTML, CSS, JavaScript)
- Backend code (e.g., Python, Node.js, Java)
- Dockerfile for containerization
- Docker Compose file for defining services and dependencies

Organize your code files into a structured directory.

4.2 Report Preparation

Create a detailed report documenting every step of the deployment process. Include the following information:

- Decisions made during the deployment process and their justifications
- Challenges encountered and how they were resolved
- Detailed description of the deployment workflow using Docker and Docker Compose
- System architecture overview, including diagram.
- Screenshots illustrating key aspects of the deployment and application functionality

Format your report as a PDF file for easy readability and submission.

4.3 Screenshots of Localhost Deployment

Capture screenshots of the Expense Splitter web application running locally on localhost. Include screenshots that demonstrate:

- Accessing the application via a web browser
- Creating and managing expense groups
- Adding expenses and viewing balances
- Any other relevant functionality of the deployed application

Ensure the screenshots are clear and labeled appropriately.

Once you have compiled all the required materials (code files, report, and screenshots), archive your project into a .zip file named as follows: `firstname_lastname.zip`
Ensure that the `.zip` file includes all components of your project. Finally, submit the `.zip` file.

This is an individual assignment. You can discuss your ideas with your peers but the application code, deployment process, and the report should be your own work. The violators will get no grade from this assignment and will be punished according to the department regulations.