Documentation for Capstone Project:

Expense Tracker Application

Author: Champa Venkataramanaswamy

Project Overview

The Expense Tracker Application is a simple tool designed to manage daily expenses. It consists of a backend built with Java and Spring Boot, database integration using MySQL, and a frontend developed using React. The application provides REST APIs for managing expenses and features a responsive user interface.

Table of Contents

Introduction

The Expense Tracker Application helps users to:

- Add expenses
- View a list of expenses
- Delete specific expenses

This document outlines the step-by-step implementation of the project.

Technologies Used

- **Backend**: Java, Spring Boot
- **Frontend**: React
- **Database**: MySQL
- **Build Tools**: Maven, npm,yarn
- **Version Control**: Git
- **Testing Tools**: JUnit 5, Postman

Phase 1: Version Control with Git

Steps:

1. **Initialize a Git Repository:**

   git init

2. **Create Branches:**

   git checkout -b Backend

   git checkout -b frontend

   git checkout -b database

3. **Push Changes to GitHub:**

   git add .

   git commit -m "Initial commit"

   git push origin backend

4. **Pull Request and Merge:**

   Create a pull request on GitHub for each branch.

   Review and merge changes into the **main** branch.

   Code is available in

   https://github.com/champa1987/ExpenseTrackerApp.git

   champa1987/ExpenseTrackerApp at master

Phase 2: Backend Development

Steps:

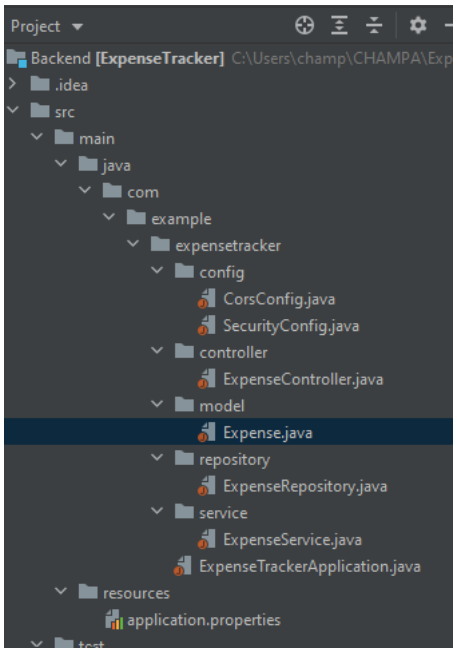
1. **Setup Spring Boot Application:**
   - Generate a Spring Boot project using Maven
   - Add dependencies for Spring Web and JPA.
2. **Create the Expense Model:**

   [ExpenseTrackerApp/Backend/src/main/java/com/example/expensetracker/model/Expense.java at main · champa1987/ExpenseTrackerApp]

3. **Create the controller, Service, Repository and Main class**

   [ExpenseTrackerApp/Backend/src/main/java/com/example/expensetracker at main · champa1987/ExpenseTrackerApp]

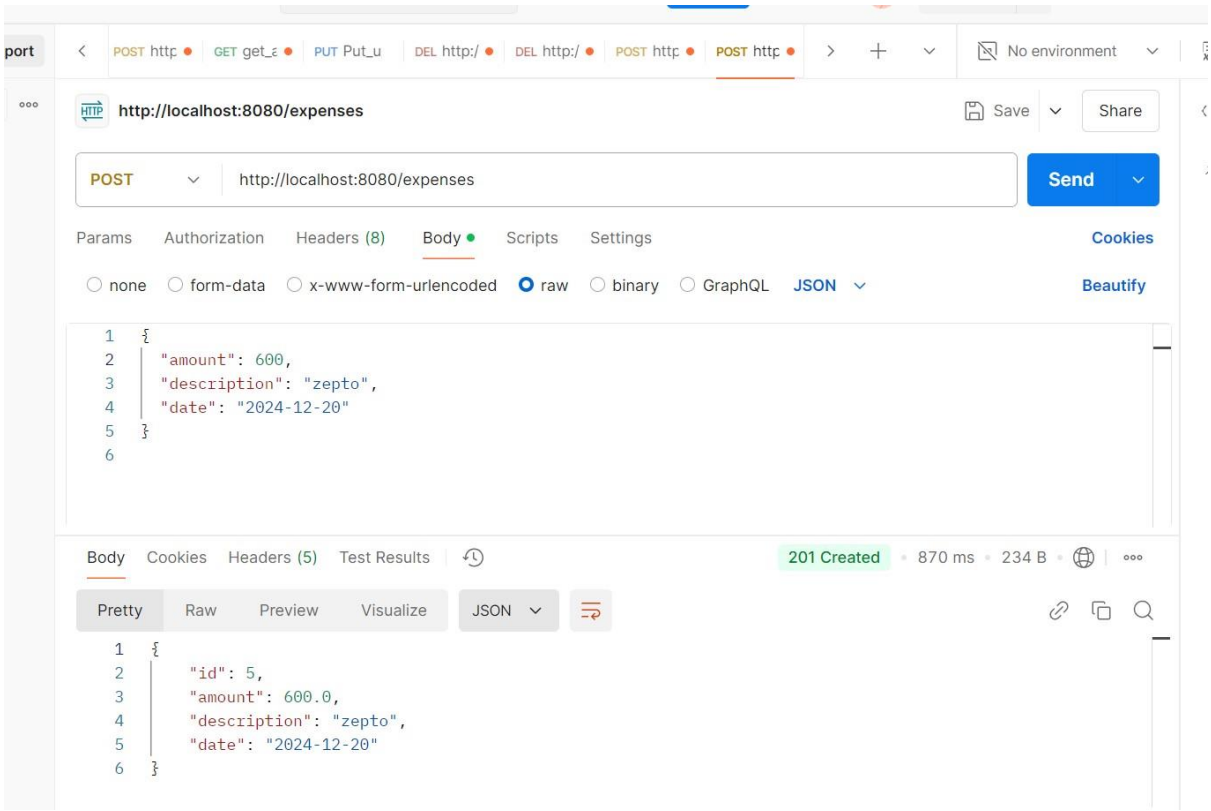
Folder structure created:



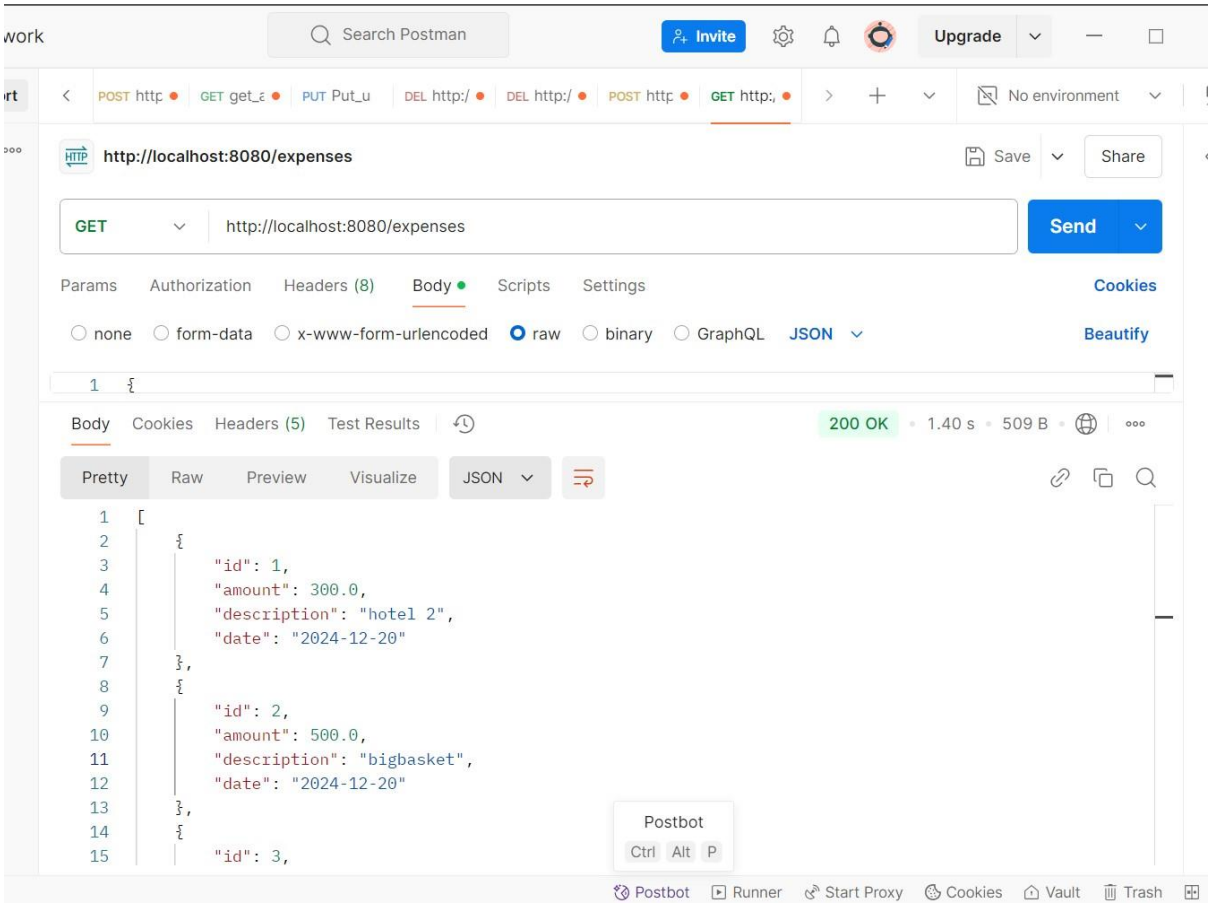
4. **Test Backend APIs:**

Used Postman to test APIs for creating, fetching, and deleting expenses.

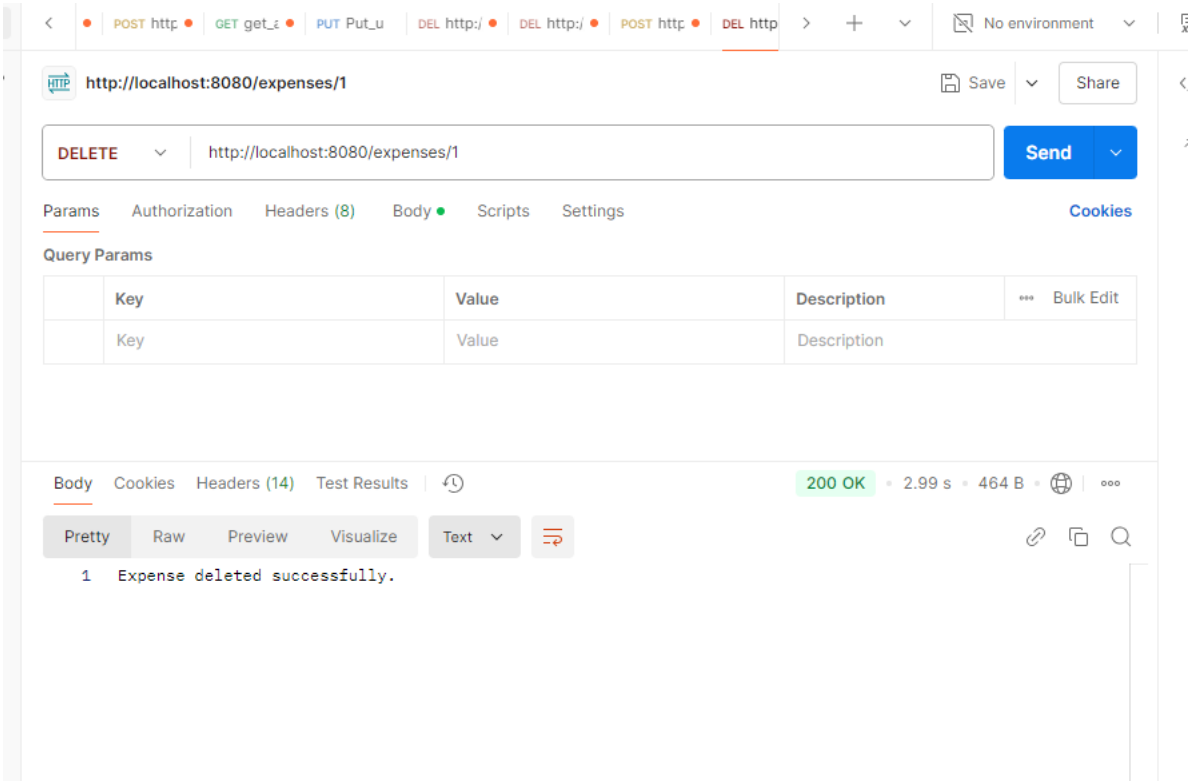
a. creating an expense



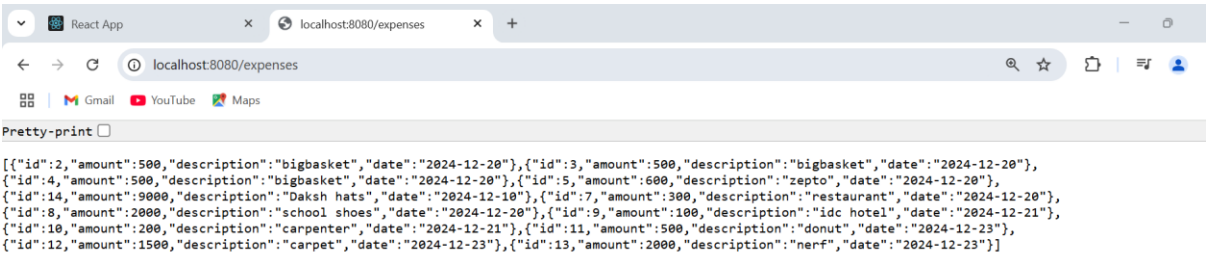
b. Fetching all expense



c. Deleting an expense



Output:



Phase 3: Database Integration

Steps:

1. **Setup MySQL Database:**

I have used AWS to create a database

jdbc:mysql://champadatabase.clc228kwmlzf.eu-north-1.rds.amazonaws.com:3306/champadatabase

endpoint : champadatabase.clc228kwmlzf.eu-north-1.rds.amazonaws.com

Created database and table.

Database Accessing:

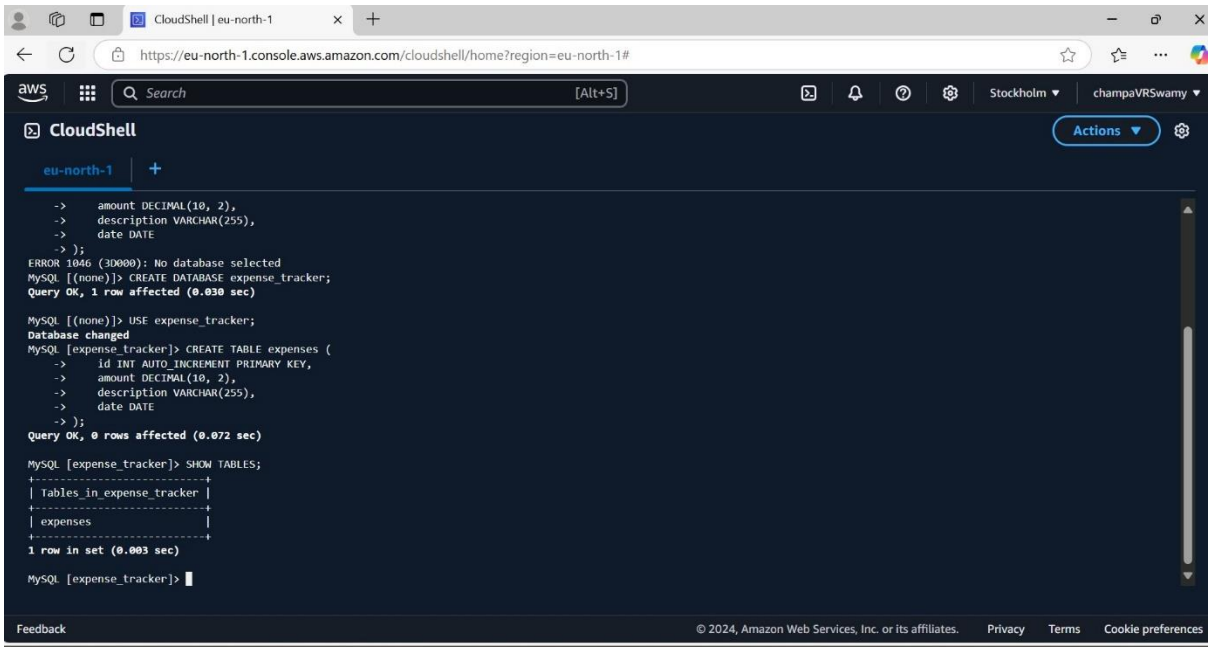
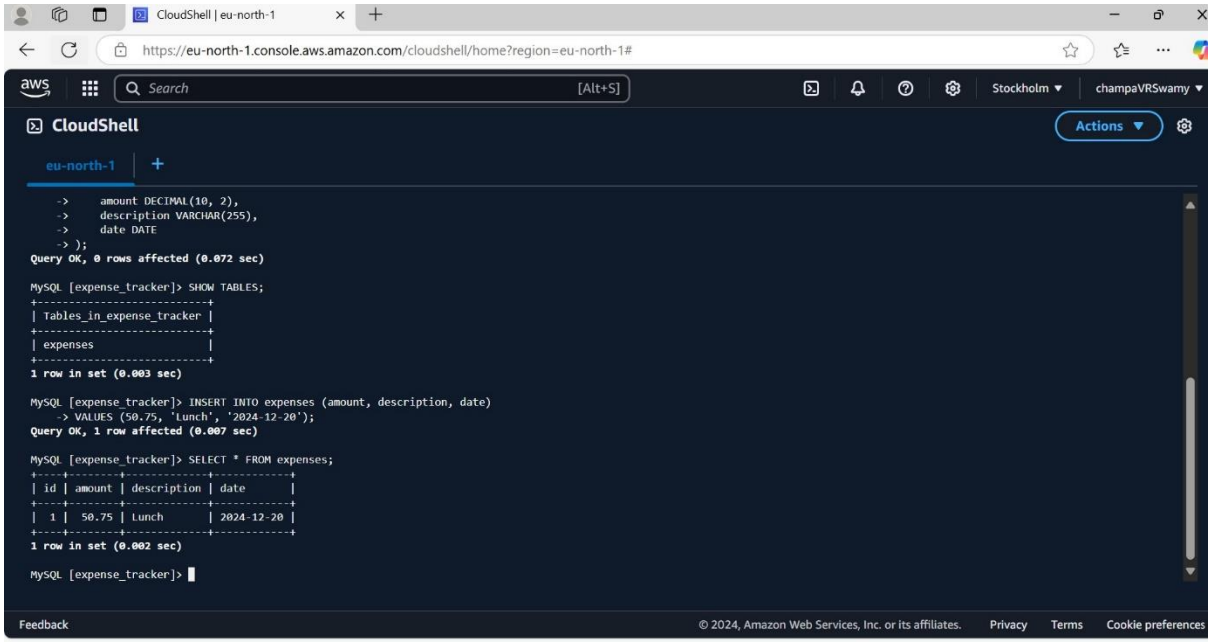


Table Creation:



Expense table data:

Configure Spring Boot Application: Add database configurations in application.properties:

```
# AWS RDS Database Connection
spring.datasource.url=jdbc:mysql://champadatabase.clc228kwmlzf.eu-north-1.rds.amazonaws.com:3306/champadatabase
spring.datasource.username=admin
spring.datasource.password=pass1234

# Hibernate Dialect for MySQL
spring.jpa.database-platform=org.hibernate.dialect.MySQL5Dialect

# Hibernate DDL auto settings
spring.jpa.hibernate.ddl-auto=update
spring.jpa.show-sql=true

# JPA repository settings
spring.datasource.driver-class-name=com.mysql.cj.jdbc.Driver

spring.main.allow-circular-references=true
```

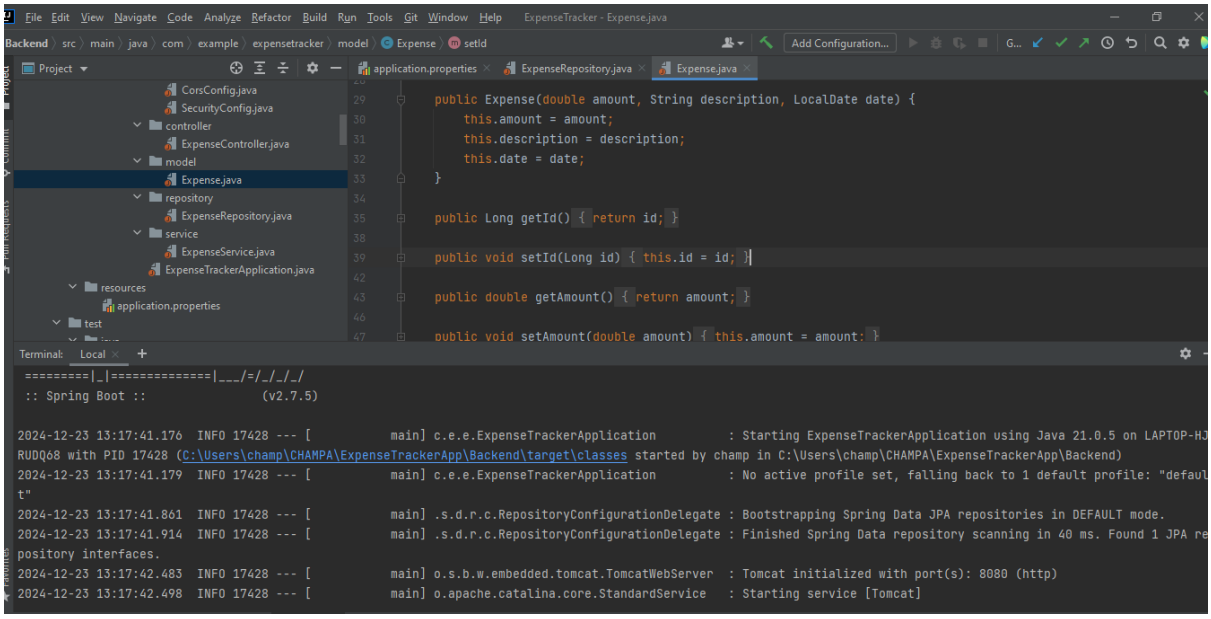
2. **Use JpaRepository:**

```
package com.example.expensetracker.repository;

import com.example.expensetracker.model.Expense;

import org.springframework.data.jpa.repository.JpaRepository;

public interface ExpenseRepository extends JpaRepository<Expense, Long>
{
    // Additional query methods can be added here if needed
}
```

3. **Added configuration for connecting to front end**

[ExpenseTrackerApp/Backend/src/main/java/com/example/expensetracker/config at main · champa1987/ExpenseTrackerApp]

Springboot project execution:



Phase 4: Frontend Development

Steps:

1. **Initialize React Project:**

   npx create-react-app expense-tracker-frontend

2. **Build UI Components:**

   - Create a form to add expenses.

     [ExpenseTrackerApp/frontend/expense-tracker/src/components/ExpenseForm.js at main · champa1987/ExpenseTrackerApp]

   - Create a table to display expenses.

   [ExpenseTrackerApp/frontend/expense-tracker/src/components/ExpenseTable.js at main · champa1987/ExpenseTrackerApp]

3. **Connect to Backend:**

   Use axios to call backend APIs.

   useEffect(() => {

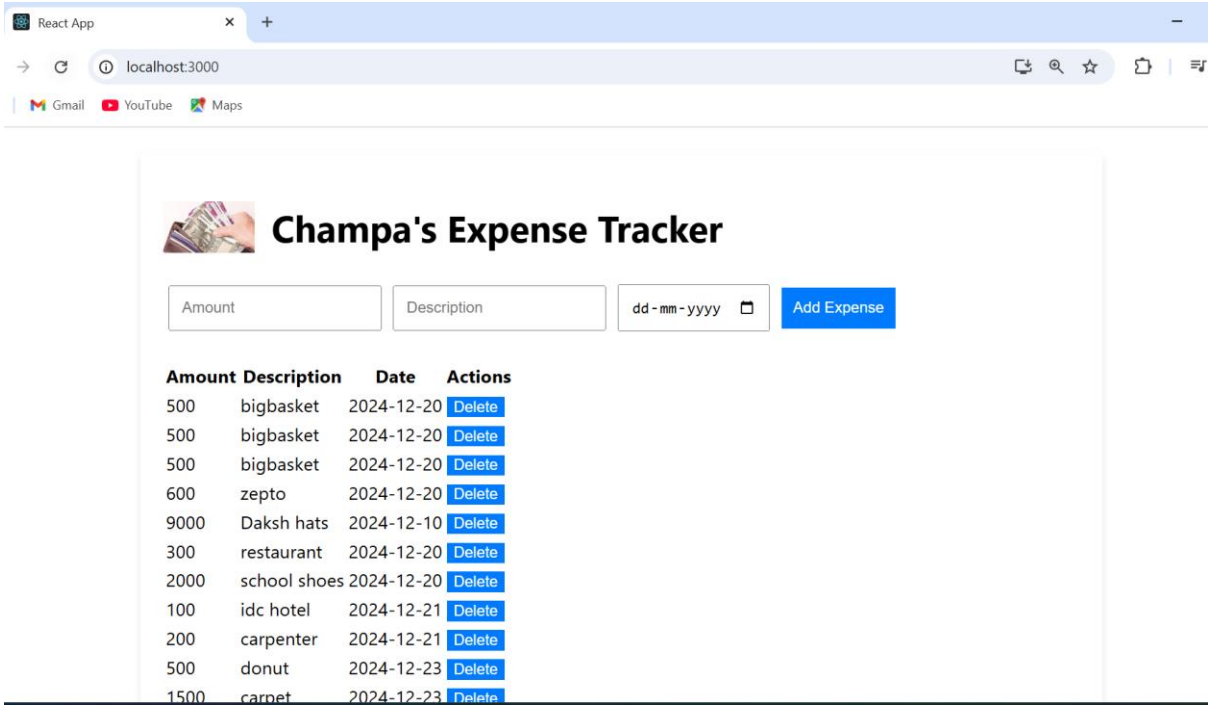
axios.get("http://localhost:8080/expenses"

then(response => setExpenses(response.data));

}, []);

4. **Run Frontend**

```
Microsoft Windows [Version 10.0.22631.4602]
(c) Microsoft Corporation. All rights reserved.

C:\Users\champ\CHAMPA\ExpenseTrackerApp\frontend\expense-tracker>yarn start
yarn run v1.22.22
warning ..\package.json: No license field
$ react-scripts start
(node:3924) [DEP_WEBPACK_DEV_SERVER_ON_AFTER_SETUP_MIDDLEWARE] DeprecationWarning: 'onAfterSetupMiddleware' option is deprecated. Please use the 'setupMiddlewares' option.
(Use `node --trace-deprecation ...` to show where the warning was created)
(node:3924) [DEP_WEBPACK_DEV_SERVER_ON_BEFORE_SETUP_MIDDLEWARE] DeprecationWarning: 'onBeforeSetupMiddleware' option is deprecated. Please use the 'setupMiddlewares' option.
Starting the development server...
Compiled successfully!

You can now view expense-tracker in the browser.

  Local:            http://localhost:3000
  On Your Network:  http://192.168.0.107:3000

Note that the development build is not optimized.
To create a production build, use yarn build.

webpack compiled successfully
```



Phase 5: Testing and Debugging

Steps:

1. **Test Backend APIs:**

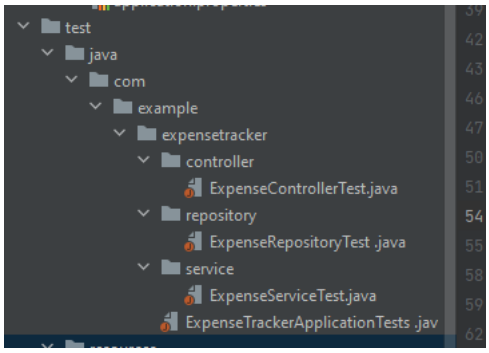
   Used Postman to ensure all APIs work as expected. – added all screenshots above

2. **Debug Frontend:**

   Checked console for errors and resolved data flow issues

3. **Write Unit Tests:**

Added test files as below structure:



Code is available here in below link :

[ExpenseTrackerApp/Backend/src/test/java/com/example/expensetracker at main · champa1987/ExpenseTrackerApp]

Running the Application:

1. **Start Backend:**

mvn spring-boot:run

2. **Start Frontend:**

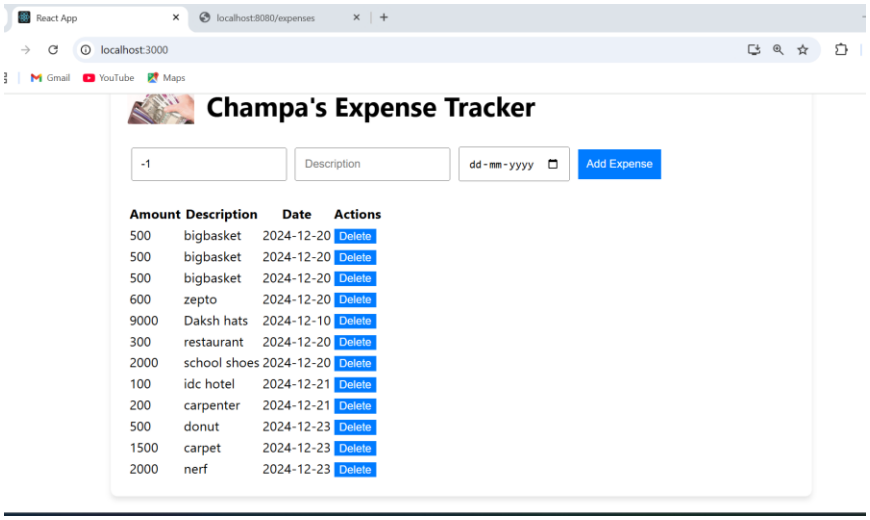
Yarn start

3. **Access the Application:**

Opened the browser and go to http://localhost:3000.

Final Features

1. **Add Expense:** Users can add a new expense via the frontend.
2. **View Expenses:** Users can see all expenses in a table.
3. **Delete Expense:** Users can delete an expense by clicking a delete button.



Conclusion

This project demonstrates the integration of Java, Spring Boot, MySQL, and React to build a fully functional Expense Tracker Application. It incorporates key concepts like REST API development, database interaction, and frontend-backend integration.