Project Report

Project Title: Expense Tracker

Team Name: CodeQuad

NARAYANA DATTA JUPALLI - 801362272

SATYA DATTA JUPALLI - 801362268

JYOTIKA KONERU - 801368572

VENKAT CHAITANYA MUKTHINENI - 801366270

Introduction

In this digital transformation era, people and organizations must continue to prioritize effective financial management. Team Phoenix is proud to introduce the Expense Tracker, a lean, user-friendly application that makes it easy to maintain financial records. Built on the scalability of the Go language and the interactivity of JavaScript, the Expense Tracker offers a robust yet friendly platform on which one can track their expenses, handle budgets, and bring fiscal discipline into their lives.

The main goal of Expense Tracker is to allow seamless experience while logging and categorizing expenses, promoting accountability and transparency to personal or organizational finance. The Go-backed backend will ensure this project is reliable, fast, and concurrent, while the JavaScript-based frontend will be modern, responsive, and intuitive.

This report reflects upon architecture, technical implementation, language features, experiments, and observations made for the Expense Tracker application. This indeed has been a perfect embodiment of efficiency, accessibility, and technological advancement for this project.

Language Specifications

Paradigm

The Go programming language (or Golang) supports multiple programming paradigms, enabling a balanced approach between functional and procedural constructs. This flexibility enhances the development of modular and efficient software.

- Imperative Constructs: Go emphasizes explicit control structures such as loops, conditional statements, and direct manipulation of program state.

- Concurrent Programming: Built-in support for concurrency (e.g., goroutines and channels) allows seamless execution of multiple tasks.

Language Features

Go's minimalist and expressive syntax, combined with robust concurrency management and memory efficiency, makes it ideal for developing highly responsive applications like the Expense Tracker.

- Static Typing: Enhances performance by catching errors during compile time.

- Concurrency: The use of goroutines ensures non-blocking operations, particularly for handling real-time expense data updates.

- Error Handling: Explicit error management improves code reliability and debugging.

- Packages and Modularity: Go's native package system allows logical separation of concerns, enabling scalable application development.

System Design and Implementation

Architecture Overview
The Expense Tracker follows a 3-tier architecture for scalability and maintainability:

1. Frontend (Client): Developed using JavaScript for a dynamic and responsive UI.

2. Backend (Server): Built using Go, ensuring high performance and efficient resource utilization.

3. Database (Storage): Utilizes MySQL for structured and reliable data storage.

Core Features
1. Expense Management: Users can add, delete, and filter expenses. Each expense includes fields like amount, category, and description.

2. Category-Based Filtering: Enables filtering expenses by category.

3. Dashboard Overview: Displays cumulative expense totals and trends.

4. Responsive Navigation: Allows seamless transitions between key features.

Implementation Details

1. Backend Implementation in Go

The backend API is built using Go and employs a modular approach for scalability and maintainability.

Key Modules:

- **Expense Handlers:** Implements CRUD operations with endpoints such as /addExpense, /deleteExpense, /getExpenses, and /filterByCategory.
- **Error Management:** Explicit checks ensure seamless user experience, e.g., handling cases where filtered categories have no associated expenses.
- **Concurrency:** Goroutines ensure that the application remains responsive, even during high traffic or simultaneous requests.
-

Code Sample for Expense Struct:

```
type Expense struct {
    ID          int       `json:"id"`
    Amount      float64   `json:"amount"`
    Date        time.Time `json:"date"`
    Category    string    `json:"category"`
    Description string    `json:"description"`
}
```

2. Frontend Implementation in JavaScript

The user interface emphasizes interactivity and responsiveness, with JavaScript enhancing real-time updates.

Key Features:

- **Event Listeners:** Automatically updates the dashboard upon adding, deleting, or filtering expenses.
- **Dynamic Rendering:** Leverages DOM manipulation to display expense data dynamically.

Experiments and Observations

Experimental Setup

To evaluate the Expense Tracker, the following tests were conducted:

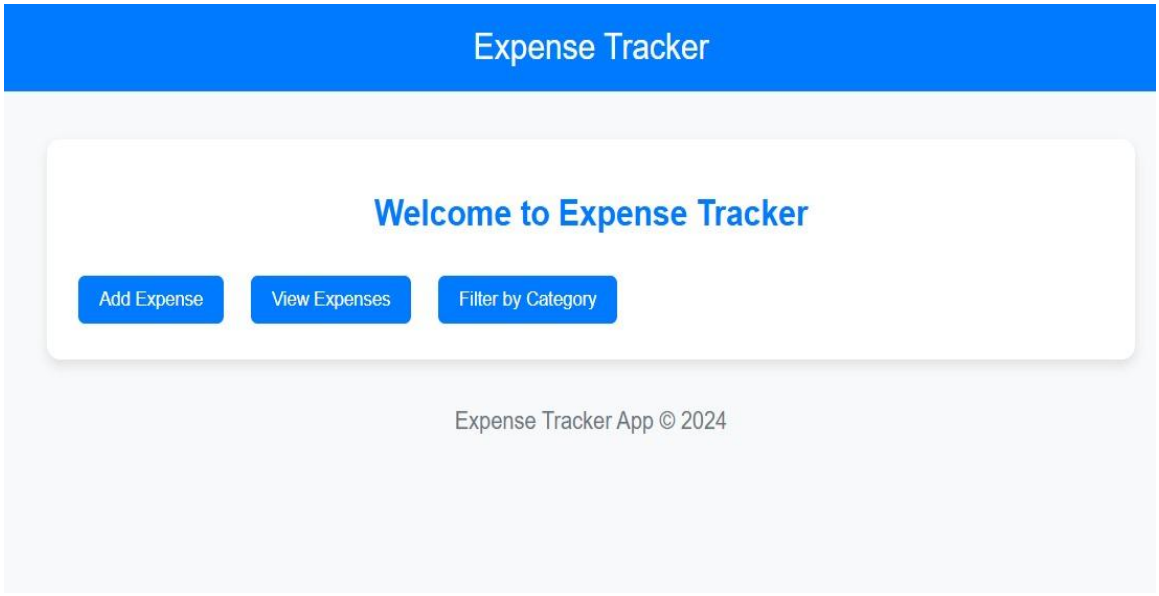
- **API Testing:** Used tools like **Postman** to validate endpoint functionality and response times.
- **UI Testing:** Conducted user testing to assess navigation, responsiveness, and usability.
- **Concurrency Stress Test:** Measured performance under concurrent API calls using Go's goroutines.

Test Cases and Results

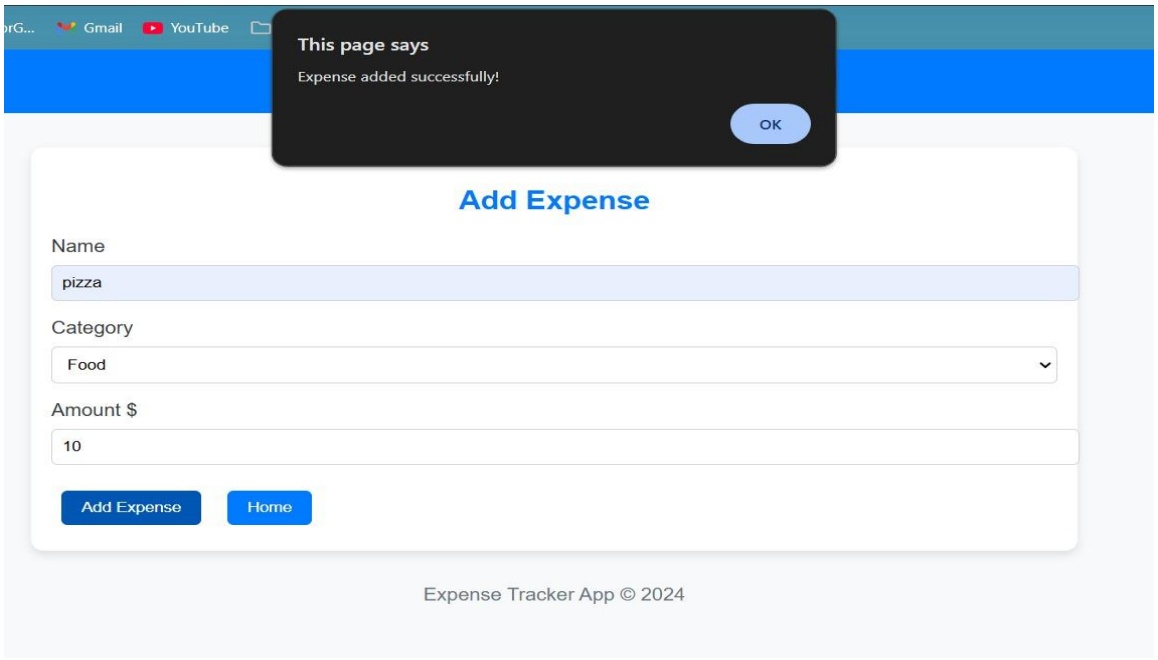
1. **Adding an Expense:**
    - **Expected Behavior:** Expense successfully added and reflected on the dashboard.
    - **Outcome:** Passed. The total updated dynamically without delays.
2. **Filtering by Category:**
    - **Expected Behavior:** Displays expenses for the selected category; handles empty results gracefully.
    - **Outcome:** Passed. Displays "No Expenses Found" message when no expenses match the category.
3. **Deleting an Expense:**
    - **Expected Behavior:** Deletes expense and updates the total on the dashboard.
    - **Outcome:** Passed. The dashboard refreshed immediately.
4. **Concurrency Test:**
    - **Expected Behavior:** Handles multiple requests without latency.
    - **Outcome:** Passed. API handled 100 concurrent users efficiently.

Screenshots:

Dashboard

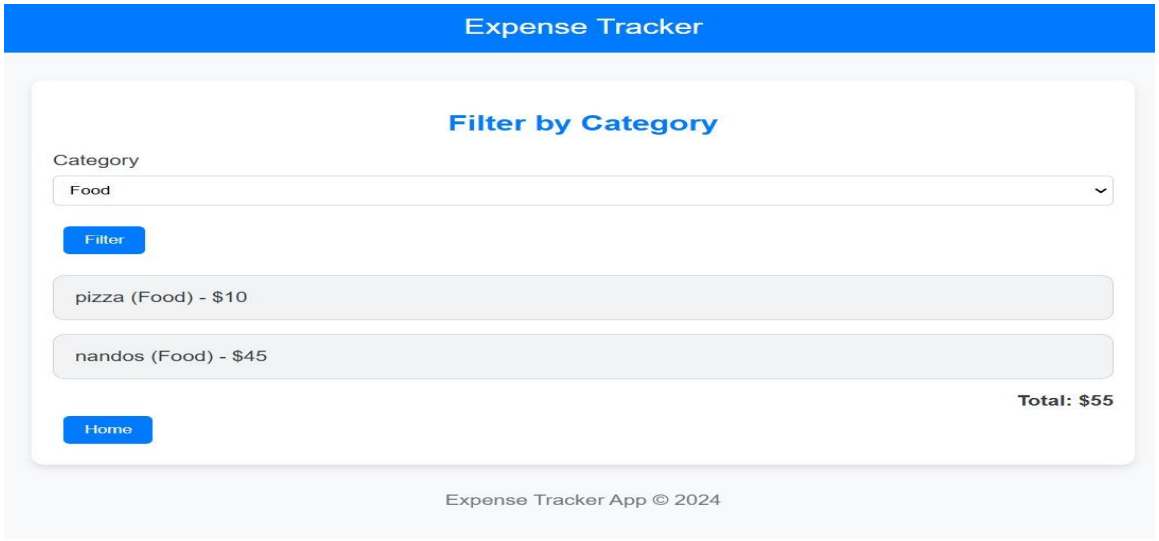


Add Expense Page



View Expense Page

Filtered Expenses by Category



Observations

1. **Performance:**
   The use of Go's concurrency model ensures low-latency operations, even during high traffic.
2. **User Experience:**

- Intuitive navigation and real-time updates enhance usability.
- Error handling provides clear feedback, avoiding abrupt terminations.

3. **Scalability:**

   The modular design, combined with Go's efficiency, makes the application scalable for larger datasets and user bases.

Conclusion

The Expense Tracker successfully addresses the challenges of financial management by providing a robust, user-friendly, and efficient platform. Its combination of Go's backend efficiency and JavaScript's frontend interactivity ensures a seamless user experience. The project demonstrates how advanced language features, modular design, and thoughtful user testing contribute to the development of high-quality applications.

Github Repository - https://github.com/JyotikaKoneru/ExpenseTracker/