Smart Expense Tracker

Developed by: MALGIREDDY YOGITHA REDDY

ID: 700773145

Date: April 27, 2025

Table of Contents

Introduction

This project is a Smart Expense Tracker web application designed to help users manage and track their daily expenses securely using AWS services.

Project Objective

To develop a secure, scalable, and user-friendly expense tracker using AWS Cognito for authentication, API Gateway and Lambda for backend services, and DynamoDB for database storage.

Technologies Used

- HTML/CSS/JavaScript

- AWS Cognito

- AWS API Gateway

- AWS Lambda

- AWS DynamoDB

- Amazon Cognito Identity SDK

- Chart.js

System Architecture

The system architecture consists of a frontend (HTML/CSS/JS) interacting with AWS services via API Gateway endpoints which trigger Lambda functions connected to a DynamoDB database.

AWS Services Setup

1. Cognito User Pool for authentication.

2. API Gateway to expose REST APIs.

3. Lambda functions for add, get, delete expenses.

4. DynamoDB to store expenses with userId as Partition Key.

Frontend Development

The frontend consists of three main pages:

- index.html: Login page

- signup.html: Registration page

- confirm.html: Email confirmation page

- dashboard.html: Dashboard to view/add/delete expenses

JavaScript Files:

- login.js, signup.js, confirm.js, dashboard.js

CSS Files:

- index.css, dashboard.css, styles.css

Backend Development

Three Lambda functions were developed:

- addExpense: Adds a new expense.

- getExpense: Fetches all expenses for a user.

- deleteExpense: Deletes a specific expense by timestamp.

Data Flow Explanation

1. User logs in through Cognito.

2. After authentication, userId (email) is stored in localStorage.

3. Dashboard uses userId to load/add/delete expenses uniquely.

4. All requests are routed via API Gateway to Lambda.

Error Handling

Implemented error checks for authentication failures, missing fields in forms, and HTTP request errors on frontend and backend.

Security Features

- AWS Cognito handles user passwords securely.

- Only authenticated users can add/delete/fetch their own expenses.

- CORS policies configured correctly.

- Minimal sensitive data is stored on client side (only userId).

Testing & Results

Tested all user flows:

- Signup, Email Confirmation, Login

- Add Expense, View Expense, Delete Expense

Results were successful and consistent across multiple users.

Challenges Faced

- Handling unconfirmed users during login.

- Separating expenses per user in DynamoDB.

- Correct CORS configuration for API Gateway.

Future Improvements

- Add Google/Facebook login.

- Add monthly report download.

- Add profile settings page for users.

Conclusion

Successfully built a cloud-based expense tracker app using AWS services ensuring scalability, security, and a smooth user experience.