Project 2: Expense Tracker Web App - Step-by-Step Guide

# Project Overview

The objective of this project is to develop a full-stack web application that allows users to track personal expenses. Users should be able to add, edit, delete, and view their expenses in a categorized and user-friendly interface. The project showcases skills in web development, CRUD operations, database handling, and possibly authentication.

# Step-by-Step Process

## Step 1: Set Up the Environment (2–3 hours)

1. Install necessary software: Python, Node.js, and a database (e.g., SQLite or MySQL).  
2. Set up a virtual environment and install dependencies (Flask, SQLAlchemy, Flask-WTF, Bootstrap, etc.).  
3. Create a basic project structure with folders for templates, static files (CSS/JS), and routes.

## Step 2: Design the Application (1–2 hours)

1. Define the key features: expense entry, category tagging, editing, deletion, and summary view.  
2. Create wireframes or mockups for major pages (Home, Add Expense, Expense List).  
3. Design the database schema with an ERD (Entity Relationship Diagram).  
 - Example tables: users, expenses, categories.

## Step 3: Implement Core Backend Functionality (3–4 hours)

1. Set up Flask application and routing.  
2. Implement models for Expense, Category, and optionally User.  
3. Create forms for submitting expenses with Flask-WTF.  
4. Implement basic CRUD operations.

## Step 4: Build Frontend Interfaces (3–4 hours)

1. Design and style pages using HTML, CSS (Bootstrap), and Jinja templates.  
2. Display expense data in a table with options to edit/delete entries.  
3. Add client-side form validation.

## Step 5: Add Optional Features (2–3 hours)

1. User authentication with Flask-Login (if multi-user support is desired).  
2. Summary view (e.g., pie chart by category using Chart.js).  
3. Export expenses to CSV or PDF.

## Step 6: Testing and Debugging (2–3 hours)

1. Unit test models and forms.  
2. Manually test all user interactions and edge cases.  
3. Handle errors gracefully (e.g., 404 pages, invalid inputs).

## Step 7: Deployment and Finalization (2–3 hours)

1. Prepare the app for deployment (use gunicorn, Zappa, or Heroku).  
2. Test deployment in a cloud environment.  
3. Write README.md with setup instructions, features, and screenshots.  
4. Create documentation for key modules and functionality.  
5. Push final version to GitHub.