A FIELD PROJECT REPORT ON

"Personal Expense Tracker and Advisory System"

Submitted

In partial fulfillment of the requirements for the award of the degree

BACHELOR OF TECHNOLOGY

In

COMPUTER SCIENCE and ENGINEERING

By

G.Harika	(231FA04C58)
K.Sai geethika	(231FA04C67)
B.Koteswararao	(231FA04C86)
S.Bhavya Harshitha	(231FA04F75)

Under the Guidance of

Dr. Nerella Sameera

Assistant Professor, CSE

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

SCHOOL OF COMPUTING AND INFORMATICS

VIGNAN'S FOUNDATION FOR SCIENCE, TECHNOLOGY & RESEARCH
(Deemed to be University)
Vadlamudi, Guntur -522213, INDIA.

April, 2025

VIGNAN'S
FOUNDATION FOR SCIENCE, TECHNOLOGY & RESEARCH
(Deemed to be University) - Estd. u/s 3 of UGC Act 1956

CERTIFICATE

This is to certify that the field project entitled "PERSONAL EXPENSE TRACKER AND ADVISORY SYSTEM" is being submitted by **G.HARIKA [231FA04C58]**, **K.Sai geethika [231FA04C67]**, **B.KOTESWARARAO [231FA04C86]**, and **S.BHAVYA HARSHITHA [231FA04F75]** in partial fulfilment of the requirements for the degree of **Bachelor of Technology (B.Tech.) in Computer Science and Engineering** at Vignan's Foundation for Science, Technology and Research (Deemed to be University), Vadlamudi, Guntur District, Andhra Pradesh, India.

This is a bonafide work carried out by the aforementioned students under my guidance and supervision.

Guide

Project Review Committee

HoD, CSE

HoD
Dept. of Computer Science & Engineering
VFSTR Deemed to be University
VADLAMUDI - 522 213
Guntur Dist., A.P., India.

VIGNAN'S
FOUNDATION FOR SCIENCE, TECHNOLOGY & RESEARCH
(Deemed to be University) - Estd. u/s 3 of UGC Act 1956

DECLARATION

Date: 26/04/25

We hereby declare that the work presented in the field project titled "PERSONAL EXPENSE TRACKER AND ADVISORY SYSTEM" is the result of our own efforts and investigations.

This project is being submitted under the supervision of **Dr. Nerella Sameera, Assistant Professor** in partial fulfillment of the requirements for the Bachelor of Technology (B.Tech.) degree in Computer Science and Engineering at Vignan's Foundation for Science, Technology and Research (Deemed to be University), Vadlamudi, Guntur, Andhra Pradesh, India.

G.Harika	(231FA04C58)	G. Harika.
K.Sai geethika	(231FA04C67)	K. Saigeethika
B.Koteswararao	(231FA04C86)	B. koteswaRaRao.
S.Bhavya Harshitha	(231FA04F75)	S. Bhavya

TABLE OF CONTENTS

CHAPTER-01
INTRODUCTION

1.INTRODUCTION

The **Personal Expenses Tracker & Advisory application** is an all-in-one, user-friendly platform that turns everyday money management into an engaging experience. It lets users log every rupee they earn or spend, automatically categorises those transactions, and visualises the data so spending patterns are instantly clear. Built-in budgeting tools make it simple to set limits for each category and receive timely alerts before overspending. Behind the scenes, smart analytics study individual habits and generate personalised recommendations that nudge users toward wiser choices—whether that means trimming discretionary costs, reallocating funds, or boosting savings. Goal-setting features help translate big ambitions—like building an emergency fund or paying off debt—into concrete milestones the app actively tracks and celebrates. By blending intuitive tracking, data-driven advice, and motivational feedback, the system empowers users to take control of their finances, sharpen their decision-making, and continually improve their overall financial health..

1.1 Problem Definition

Managing personal finances can be challenging without proper tracking and planning. Individuals often struggle to monitor their daily expenses, leading to overspending and poor savings. This project aims to develop a Personal Expenses Tracker and Advisory System that helps users record their income and expenses, analyze spending patterns, and receive suggestions to manage their budget more effectively.

1.2 Existing System

1.User Management

- Basic registration and login, storing credentials in local storage.
- No database integration, limiting security and scalability.

2. Income and Budget Setup

- Users input income and define budgets by category.
- Data saved locally for tracking against future expenses.

3. Expense Tracking

- Users log expenses with date, amount, category, and description.

• Alerts trigger when category budgets are exceeded.

4. Financial Summary and Visualization

• Displays total income, expenses, balance, and monthly overview.

• Uses Chart.js for visual insights into spending by category.

5. Limitations

• Data is device-specific and lacks security or cloud backup.

• No real-time sync, multi-user support, or cross-device access.

1.3 Propsed System

1. Cloud-Based User Management

• Secure user registration and login using a backend with authentication.

• User data stored in a cloud database, enabling session persistence and security.

2. Real-Time Income & Budget Setup

• Users can set income and dynamic budgets across categories.

• Cloud sync allows updates and access across multiple devices in real-time.

3. Advanced Expense Tracking

• Allows expense logging with filters, search, and recurring entries.

• Real-time validation and alerts when budgets are close to limits.

4. Smart Financial Summary & Insights

• Provides AI-based suggestions, monthly trends, and financial tips.

• Visual dashboards with interactive charts (line, pie, bar) for deep insights.

5. Enhanced Features & Scalability

• Multi-user support, data export (PDF/Excel), and cloud backup.

• Scalable architecture ready for mobile app integration and future upgrades.

1.4 Literature Survey

• Many existing budget trackers use local or cloud databases for storing user data securely. Studies show that visual aids like graphs improve financial awareness. However, most systems lack real-time syncing, personalized alerts, or offline-first design. There's a need for a lightweight, user-friendly tool tailored for individual budgeting and daily tracking.

CHAPTER-02
SYSTEM REQUIREMENTS

2. System Requirements

2.1 Hardware & Software Requirements

• **Hardware:** Basic PC/laptop with 2GB+ RAM, dual-core processor, keyboard, and modern browser. Also works on smartphones or tablets with internet for initial asset load.

• **Software:** HTML, CSS, JavaScript, Chart.js, browser (Chrome/Firefox), VS Code (optional), Live Server extension, and LocalStorage for data handling.

2.1 Requirement Analysis:

The application must let users securely sign in, record every income or expense, automatically sort those transactions into categories, and keep budgets updated in real time. It should present clear charts, warn when spending nears a limit, and offer tailored money-saving tips—all through a fast, mobile-friendly interface built with React on the front end, a Python API and MySQL/PostgreSQL on the back end, and protected by strong encryption and high uptime.

Hardware Requirements

Component	Specification
Processor	Intel Core i5 or equivalent and above
Memory	Minimum 8 GB
Storage	500 GB HDD or 256 GB SSD
Display	Minimum resolution 1366×768
Internet	Stable internet connection
General device	Standard desktop or laptop

Software Requirements

Software	Specification
Frontend Technologies	HTML,CSS,JavaScript
Backend Technologies	Server side languages,Databases,APIs
Authentication & Storage:	Basic simulation using JavaScript and localStorage
Visulisation	Chart.js
Browser (Chrome/Firefox)	Web browsers for testing and viewing web applications

2.2 Software Requirements Specifications(SRS)

- Frontend: HTML, CSS, JavaScript, Chart.js, Google Fonts
- Backend: Browser localStorage
- Visualization: Chart.js
- Authentication & Storage: Basic simulation using JavaScript and localStorage

CHAPTER-03
SYSTEM DESIGN

3. System Design

3.1 Modules of the System

• User Interface Module – Handles input forms and displays charts. • Data Storage Module – Manages storing/retrieving user data (via LocalStorage). • Analytics Module – Processes data to calculate performance and generate visuals. • Visualization Module – Uses Chart.js to render bar, line, and pie charts. • Validation Module – Ensures user inputs (marks, names) are valid and within range

3.2 UML Diagrams

you can include the following diagrams: 1. Use Case Diagram – Shows interactions between user and system (e.g., input marks, view chart). 2. Class Diagram – Represents key JS objects/functions like Student, ChartManager, etc. 3. Activity Diagram – Describes the flow from data input to visualization. 4. Sequence Diagram – Illustrates step-by-step execution: user inputs → validation → storage → chart generation.

1.User Interface Module

Input Forms: This module creates forms where users can enter their data, such as names and marks. It ensures that the forms are intuitive and easy to use.

Chart Display: After data is processed, this module is responsible for displaying the resulting charts, allowing users to visualize their performance

2.Data Storage Module –

Manages storing/retrieving user data (via LocalStorage).

3.Analytics Module

Data Processing: This module takes the raw data from the Data Storage Module and processes it to calculate various performance metrics, such as averages or trends over time.

Insights Generation: It may also generate insights or recommendations based on the analyzed data, helping users understand their performance better.

4.Visualization Module

Chart Creation: Using Chart.js, this module creates visual representations of the processed data. It can generate different types of charts (bar, line, pie) based on user preferences or data characteristics.

Dynamic Updates: The charts can be updated in real-time as users input new data, providing immediate visual feedback

5. Validation Module:

Input Validation: This module checks user inputs for validity, ensuring that names are not empty and marks fall within acceptable ranges (e.g., 0-100).

User Feedback: It provides immediate feedback to users if their inputs are invalid, helping them correct errors before submission

CHAPTER-04
IMPLEMENTATION

4.Implementation

4.1 Sample Code

```
let historyStack = ["home"];
let cartItems = [];
let selectedLocation = "";

ON_PAGE_LOAD:
CALL loadPage("home");

FUNCTION loadPage(page) {
  let content = GET_ELEMENT("content");
  let navbar = GET_ELEMENT("navbar");
  let backButton = GET_ELEMENT("backButton");

  IF page IN ["login", "register"]:
    HIDE(navbar);
  ELSE:
    SHOW(navbar);

  IF page NOT IN ["home", "login", "register"]:
    SHOW(backButton);
  ELSE:
    HIDE(backButton);

  PUSH(historyStack, page);

  SWITCH page {
    CASE "login":
```

```
      SET content TO login_form_html;
    CASE "register":
      SET content TO register_form_html;
    CASE "home":
      SET content TO home_html;
    CASE "income":
      SET content TO income_html;
    CASE "budget":
      SET content TO budget_html;
    CASE "expense":
      SET content TO expense_html;
    CASE "summary":
      SET content TO summary_html;
    CASE "savings":
      SET content TO savings_html;
    DEFAULT:
      SET content TO page_not_found_html;
  }

  SCROLL_TO_TOP();
}

FUNCTION goBack() {
  POP(historyStack);
  CALL loadPage(LAST(historyStack));
}

FUNCTION addToCart(name, price) {
  PUSH(cartItems, {name, price});
  CALL updateCart();
}
```

```
FUNCTION updateCart() {
  let cartItemsElement = GET_ELEMENT("cartItems");

  IF cartItems IS_EMPTY:
    SET cartItemsElement TO "Your cart is empty";
  ELSE:
    let total = SUM(item.price FOR item IN cartItems);
    SET cartItemsElement TO cart_items_html(cartItems, total);
}

FUNCTION clearCart() {
  CLEAR(cartItems);
  CALL updateCart();
}

FUNCTION filterByLocation() {
  selectedLocation = GET_VALUE("locationSelect");
  CALL loadPage(LAST(historyStack));
}

FUNCTION generateExpenseList(title, expenses) {
  let expensesHTML = "<h2>" + title + "</h2>";

  IF expenses IS_EMPTY:
    APPEND expensesHTML WITH "No expenses found";
  ELSE:
    FOR EACH expense IN expenses:
      APPEND expensesHTML WITH expense_card_html(expense);

  RETURN expensesHTML;
}
```

```
FUNCTION handleExpenseSubmission(event) {
  PREVENT_DEFAULT(event);
  let expenseName = GET_VALUE("expenseName");
  let expenseAmount = GET_VALUE("expenseAmount");

  // Logic to handle expense submission
  SHOW_ALERT("Expense added: " + expenseName + " for ₹" + expenseAmount);
  CALL loadPage("expense");
}

FUNCTION handleLogin(event) {
  PREVENT_DEFAULT(event);
  let username = GET_VALUE("username");
  let password = GET_VALUE("password");

  // Logic to handle login
  SHOW_ALERT("Login successful for " + username);
  CALL loadPage("income");
}

FUNCTION handleRegister(event) {
  PREVENT_DEFAULT(event);
  let username = GET_VALUE("registerUsername");
  let password = GET_VALUE("registerPassword");

  // Logic to handle registration
  SHOW_ALERT("Registration successful for " + username);
  CALL loadPage("login");
}
```

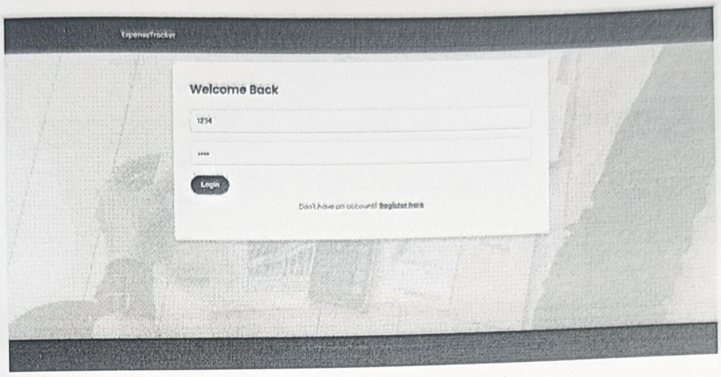
4.2 Test cases

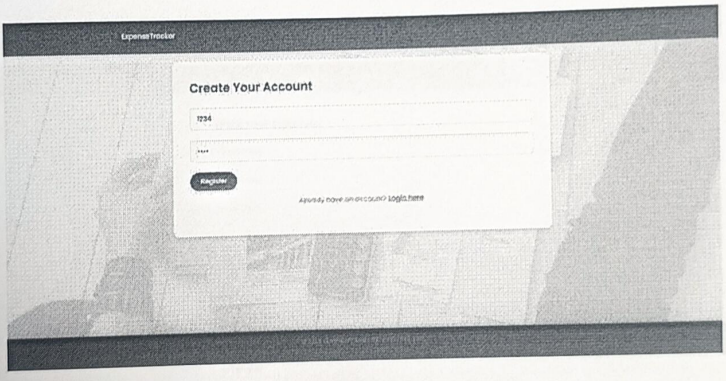
1. **User Registration**: Test valid registration, empty username, and empty password.
2. **User Login**: Test valid login and incorrect credentials.
3. **Income Management**: Test adding valid income and invalid negative amounts.
4. **Budget Management**: Test setting budgets and handling empty fields.
5. **Expense Tracking**: Test adding valid expenses and missing fields.
6. **Summary Calculation:** Verify total income, expenses, and remaining balance.
7. **Savings Goals:** Test adding valid goals and negative target amounts.
8. **Export Expenses**: Ensure expenses export correctly to CSV.
9. **Logout: Confirm** successful logout and redirection to the home page.

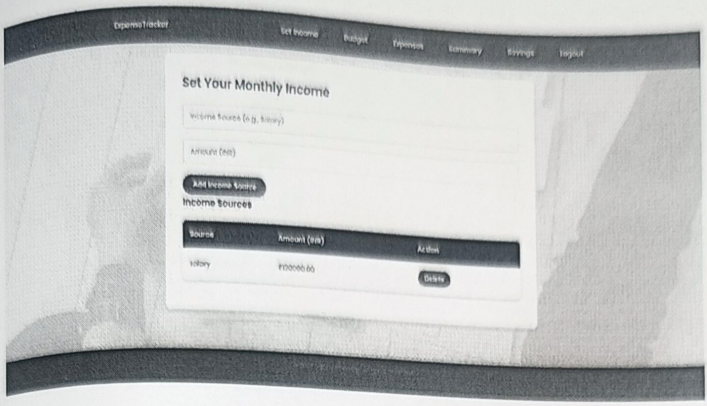
CHAPTER-05
RESULTS

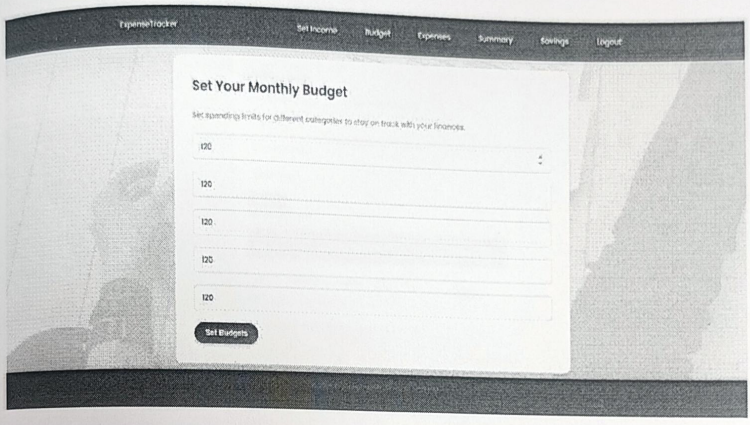
5. Results

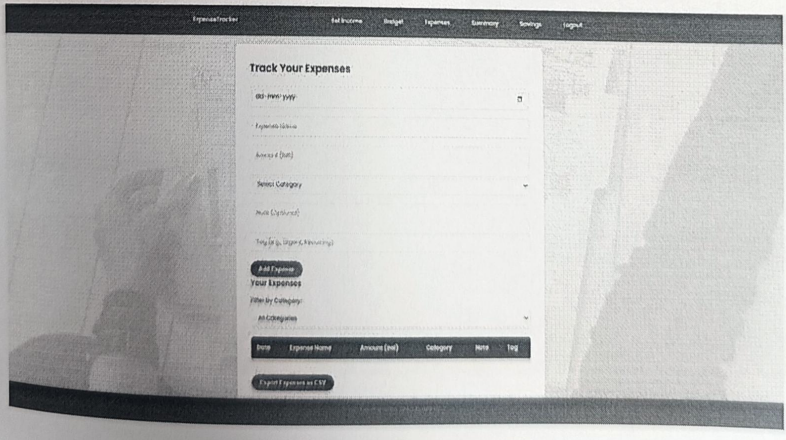

ExpenseTracker
Welcome to ExpenseTracker
Register
Login

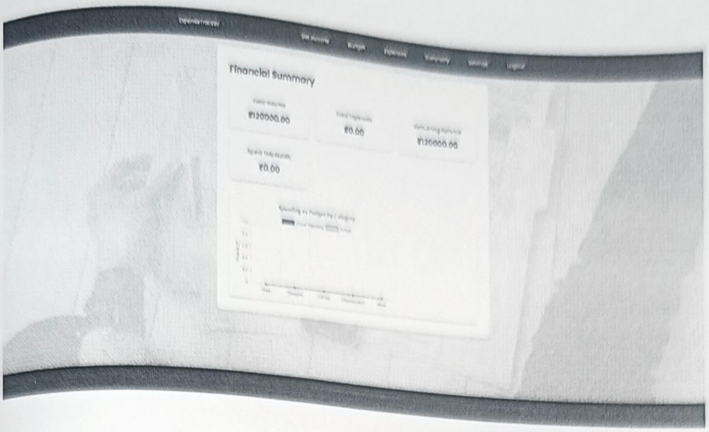

ExpenseTracker
Welcome Back
1234
Login
Don't have an account? Register here

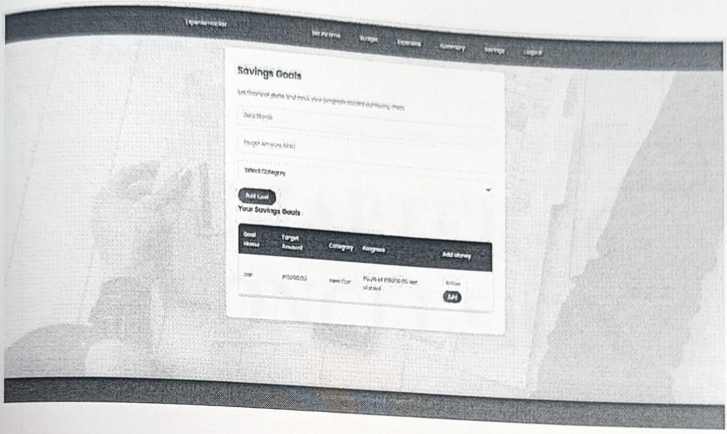

ExpenseTracker
Create Your Account
1234
Register
Already have an account? Login here

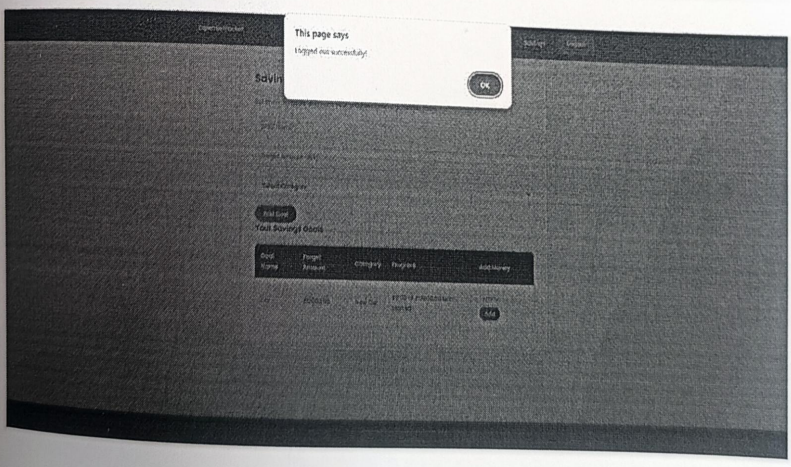

ExpenseTracker
Set Income
Budget
Expenses
Summary
Savings
Logout
Set Your Monthly Income
Income Source (e.g. Salary)
Amount (INR)
Add Income Source
Income Sources
Source
Amount (INR)
Action
Salary
100000.00
Delete


ExpenseTracker
Set Income
Budget
Expenses
Summary
Savings
Logout
Set Your Monthly Budget
Set spending limits for different categories to stay on track with your finances.
120
120
120
120
120
Set Budgets


ExpenseTracker
Set Income
Budget
Expenses
Summary
Savings
Logout
Track Your Expenses
dd-mm-yyyy
Expense Name
Amount (INR)
Select Category
Note (Optional)
Tag (e.g. Urgent, Recurring)
Add Expense
Your Expenses
Filter by Category:
All Categories
Date
Expense Name
Amount (INR)
Category
Note
Tag
Export Expenses as CSV


Financial Summary
₹120000.00
₹0.00
₹120000.00
₹0.00


Savings Goals
Add Goal
Your Savings Goals
Goal Name
Target Amount
Category
Progress
Add Money
Add


This page says
Logged out successfully!
OK

CHAPTER-06
CONCLUSION

1. Conclusion

In conclusion, the Personal Expenses Tracker and Advisory application is an essential tool for enhancing financial management. It empowers users to monitor their spending, create budgets, and set financial goals while providing personalized insights and learning opportunities. With a focus on fostering positive financial habits and ensuring data security, the application helps users achieve their financial aspirations, leading to improved financial health and greater confidence in managing their finances The Personal Expenses Tracker and Advisory app is far more than a digital ledger—it's a 24/7 financial coach in your pocket. By automatically categorizing transactions, flagging overspending patterns, and nudging users when they drift from budget, the app turns raw numbers into crystal-clear stories about spending behavior. Dynamic goal-setting modules help users plan for everything from an emergency fund to a down payment, while bite-sized lessons—tailored to each user's habits—fill critical knowledge gaps along the way. Robust encryption and optional biometric authentication keep every cent of data private, cultivating trust from day one. The result is a virtuous cycle: better insights inspire smarter decisions; smarter decisions accelerate progress toward goals; progress reinforces healthy money habits. Whether someone is paying off student loans or maximizing investment contributions, the app meets them where they are and grows with them. Ultimately, it empowers users to transform short-term budgeting into lifelong financial mastery—driving confidence, resilience, and lasting peace of mind.

References

The HTML document presents a comprehensive Personal Expense Tracker application. It includes functionalities for user registration, income and expense tracking, budget management, and savings goals. The application employs Chart.js for visual data representation and is styled with CSS to ensure a professional and user-friendly interface.

1. **Chart.js Documentation**: Chart.js
2. **CSS Flexbox Guide**: CSS-Tricks Flexbox Guide
3. **HTML5 and CSS3 Fundamentals**: W3Schools HTML/CSS
4. **JavaScript Local Storage**: MDN Web Docs on Local Storage
5. **Responsive Web Design**: Google Developers on Responsive Design

These resources provide additional insights into the technologies and techniques used in developing web applications like the Personal Expense Tracker.