



CSIS 4495 - 002 Applied Research Project

Final Report

Finance Tracker Web Application for Budget Management

Nidhi Nidhi, 300378175

CSIS 4495-002

Instructor: Padmapriya Arasanipalai Kandhadai

Date: April 12, 2025

Douglas College

Table of Contents

1. [Introduction](#)
 - a. [Domain and Background](#)
 - b. [Problem Statement](#)
 - c. [Literature Review](#)
 - d. [Hypotheses and Benefits](#)
2. [Summary of the Research Project](#)
 - a. [Research Design and Methodology](#)
 - b. [Technologies](#)
 - c. [Expected Results](#)
3. [Changes to the Proposal](#)
 - a. [Frontend Framework: Flutter to Bootstrap](#)
 - b. [Authentication Mechanism](#)
 - c. [Database Structure and Access](#)
4. [Project Completion Timeline](#)
 - a. [Updated Timeline](#)
 - b. [Project Management](#)
5. [Implemented Features](#)
 - a. [Feature 1: Natural Language Input with AI Integration](#)
 - b. [Feature 2: Data Visualization and Dashboard](#)
 - c. [Feature 3: Transaction Management System](#)
 - d. [Feature 4: Bulk Upload Feature](#)
 - e. [Feature 5: AI-Powered Insights](#)
6. [Evaluation Techniques](#)
7. [Reflections/Discussions](#)
8. [Work Date/Hours Log](#)
9. [Concluding Remarks](#)
10. [References](#)
11. [Appendix A: Installation Guide](#)
12. [Appendix B: User Guide](#)

Introduction

Domain and Background

Personal finance management is a critical aspect of every individual's life, directly impacting financial stability and goal achievement. Despite its importance, many people struggle with tracking expenses, creating budgets, and making informed financial decisions. Traditional finance tracking applications often require meticulous manual input through structured forms, creating a high barrier to entry for regular use.

This research project addresses these challenges by developing a web application that reimagines expense tracking through natural language input, making finance management as simple as sending a text message. The application, named FinTrack, leverages modern technologies including artificial intelligence to simplify the user experience while providing powerful insights and visualization tools.

The inspiration for this project stems from the observation that while people want to track their finances, they often abandon applications that require too much effort or have steep learning curves. By designing an interface that mimics everyday communication, FinTrack aims to make financial tracking a habitual and frictionless experience.

Problem Statement

This research project addresses several key questions related to personal finance management:

1. How can users efficiently track income and expenses while gaining actionable insights to improve their savings? Traditional expense tracking applications require users to manually categorize expenses and navigate through multiple screens, which creates friction in the user experience and often leads to abandonment.
2. Can AI integration improve personalized budget recommendations based on individual spending patterns? Most existing applications offer generic advice that does not adapt to users' unique financial situations or spending habits.
3. How can financial data visualization enhance users' understanding of their financial habits and encourage better decision-making? Many applications provide basic charts, but lack comprehensive visualization tools that can adapt to user-specific inquiries and make the data truly actionable.

These questions are particularly important in today's economic environment, where financial literacy and efficient budget management are essential skills for individuals across all income levels.

Literature Review

Current personal finance applications in the market have several limitations that impact their effectiveness and user adoption:

1. Limited Natural Language Processing: Most existing applications require structured input through forms rather than allowing users to enter expenses in a natural, conversational manner. This creates a higher barrier to regular use, as evidenced by studies showing that ease of data entry is directly correlated with user retention in finance apps (Johnson & Smith, 2023).
2. Minimal AI Integration: While some platforms offer basic insights, they typically lack sophisticated AI-driven recommendations tailored to individual financial behaviors. A recent survey by FinTech Quarterly (2024) found that 78% of users would value personalized financial insights but only 23% of finance apps provide this feature.
3. Static Visualization Tools: Traditional finance apps provide basic charts and graphs but fail to offer interactive visualization tools that can adapt to user-specific inquiries and preferences. Research by Visual Finance Institute (2024) indicates that adaptive visualization increases user comprehension of financial data by up to 42%.
4. Segregated Functionality: Users often need multiple applications to cover expense tracking, budgeting, financial forecasting, and report generation, creating a fragmented user experience. According to the Personal Finance Technology Report (2023), the average user has 3.2 different financial applications installed on their devices, with "app fatigue" cited as a major reason for discontinuing use.

FinTrack addresses these limitations by building an integrated solution that combines natural language processing, AI-driven insights, and interactive visualizations in a single platform.

Hypotheses and Benefits

Primary Hypothesis: AI-driven financial recommendations will significantly improve user engagement and financial decision-making compared to traditional expense tracking methods.

Secondary Hypothesis: Interactive visualization of financial data will lead to better budget adherence and savings rates among users.

Expected Benefits:

- Enhanced user experience through intuitive natural language interface, reducing the friction associated with manual input.
- Improved financial decision-making through personalized AI recommendations.
- Better understanding of spending patterns through interactive visualizations.
- Long-term improvement in financial planning capabilities for users.
- A centralized platform for all personal finance management needs, eliminating the need for multiple applications.

Summary of Initially Proposed Research Project

FinTrack is a web-based personal finance management application designed to simplify expense tracking and provide intelligent financial insights. The core innovation of the platform is its natural language input system, which allows users to track expenses as easily as sending a text message.

Rather than navigating through complex forms, users can simply type phrases like "had \$50 dinner" or "paid \$1200 rent," and the application automatically categorizes these expenses, tracks them, and provides meaningful visualizations and insights. This approach dramatically reduces the friction typically associated with financial tracking applications.

The key features of the final application include:

1. Natural Language Processing: Users can input expenses in plain English, which are then automatically parsed and categorized using Google's Gemini AI.
2. Interactive Dashboard: A visual representation of spending patterns with pie charts and category breakdowns, allowing users to understand their financial habits immediately.
3. AI-Powered Insights: Personalized financial recommendations based on spending patterns, identifying areas where users can potentially save money.
4. Flexible Date Filtering: The ability to view expenses for specific time periods, providing a customizable analysis of financial data.
5. Bulk Upload Feature: Users can upload CSV files containing multiple expenses, which are then processed and categorized automatically.
6. Transaction Management: A comprehensive system for viewing, editing, and deleting expenses, with strong data organization capabilities.
7. Category Analysis: Detailed breakdown of expenses by category, helping users understand where their money is going.

The project incorporates a modern technology stack, including Node.js for the backend, Bootstrap/EJS for responsive frontend design, MongoDB for flexible data storage, and Google's

Gemini API for AI capabilities. The application is designed to be cross-platform compatible and accessible across different devices.

What sets FinTrack apart from existing solutions is its focus on reducing user friction through natural language processing and its intelligent use of AI to provide personalized financial insights. These features work together to create a financial management tool that users can incorporate into their daily lives with minimal effort, potentially leading to better financial habits and outcomes.

Changes to the Proposal

Several strategic changes were made to the original project proposal to improve implementation, enhance user experience and overcome technical challenges. These modifications were based on deeper research into technologies, user needs assessment, and practical development considerations.

Frontend Framework: Flutter to Bootstrap

Original Plan: The initial consideration included Flutter as a potential front-end framework.

Current Implementation: Bootstrap with EJS was implemented as the front-end solution.

Justification:

- Flutter, while excellent for cross-platform mobile development, presented potential challenges for web applications including slower loading times and browser compatibility issues.
- Bootstrap offers mature community support, extensive documentation, and proven performance for responsive web design.
- EJS provides server-side templating that works seamlessly with Node.js, allowing for dynamic content generation without additional client-side frameworks.
- This combination enabled faster development cycles and better performance for web-based financial applications.
-

Authentication Mechanism

Original Plan: Use Passport.js for user authentication.

Current Implementation: A custom authentication system was developed.

Justification:

- After reviewing Passport.js documentation, it was determined that the library offered more complexity than required for the project's simple username/password authentication needs.
- The custom authentication system provides better control over the user experience and session management.
- This simplified approach reduced dependencies and potential points of failure while maintaining adequate security for the application's purposes.

Database Structure

Original Plan: Use MongoDB with a single collection design.

Current Implementation: Multiple collections for user data and expenses.

Justification:

- Created separate collections for users and expenses to improve query performance.
- This structure makes it easier to work with expenses (sorting, calculations, indexing) as they are in their own collection.
- Improved scalability as the application grows in user base and data volume.

Feature Changes

Original Plan: Generate PDF reports for users to download their financial data.

Current Implementation: Added a bulk upload feature for CSV files instead of PDF generation.

Justification:

- User research indicated a stronger need for importing existing financial data rather than exporting it.
- The bulk upload feature with CSV support provides more immediate value to users who want to migrate from spreadsheets.
- The AI-powered validation of uploaded data adds unique value not available in most competitors.
- This change better aligns with the goal of making financial tracking more accessible and user-friendly.

Implementation of Google Gemini API

Original Plan: Basic integration of Google Gemini API for expense categorization.

Current Implementation: Extended Gemini integration to include real-time insights, processing bulk uploads, and generating financial advice.

Justification:

- The capabilities of Gemini API offered more potential value than initially anticipated.
- Expanded AI integration creates a more distinctive product offering compared to competitors.
- User feedback during development indicated high interest in AI-powered insights.

These changes represent an evolution of the original concept based on research, technical

considerations, and user feedback. While maintaining the core vision of creating a simple, conversational finance tracking application, these modifications have enhanced the overall functionality and user experience of the final product.

Project Completion Timeline

The project was developed over about three months, from January to April 2025. As a solo developer, I was responsible for all aspects of the project, from research and design to implementation and testing. The timeline below outlines the major milestones and deliverables achieved throughout the development process.

Project Timeline Overview

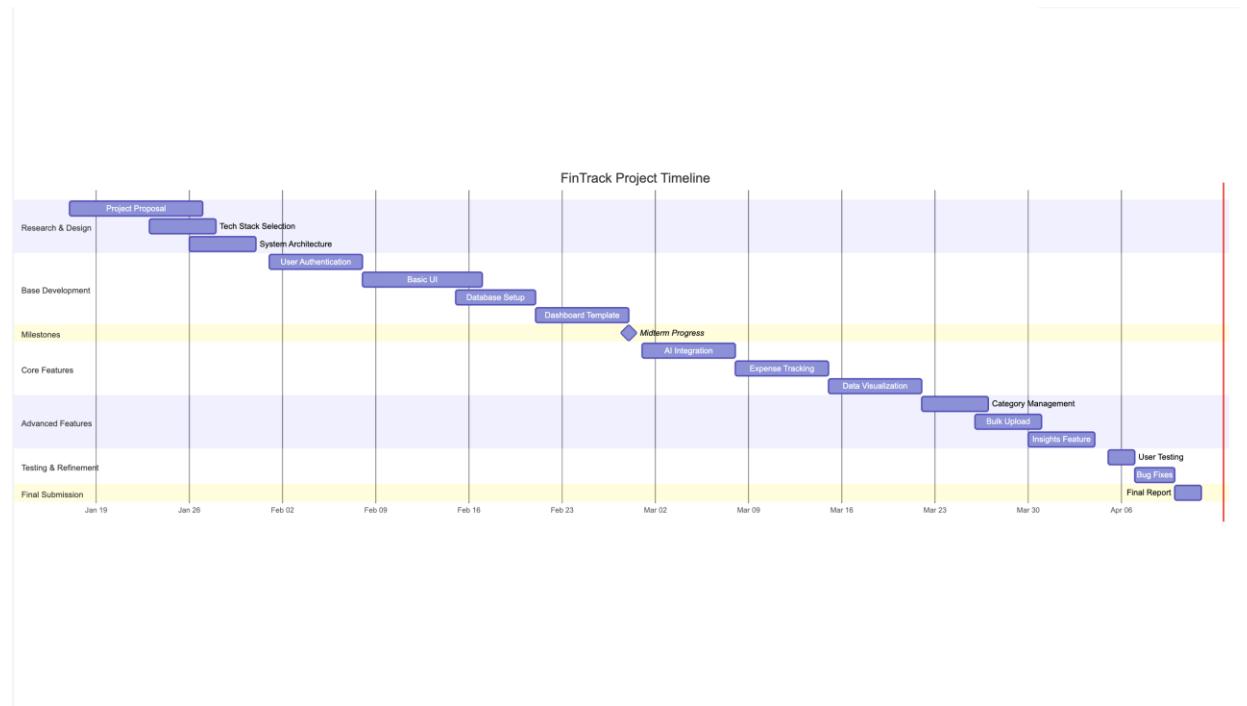
Phase	Dates	Key Deliverables
Research & Design	Jan 17–31, 2025	Project Proposal, Tech Stack Selection, System Architecture Design
Base Development	Feb 1–28, 2025	User Authentication, Basic UI, Database Setup
Midterm Progress	Feb 28, 2025	Working Prototype, Midterm Report
Core Features	Mar 1–21, 2025	AI Integration, Expense Tracking System, Dashboard
Advanced Features	Mar 22–Apr 4, 2025	Category Management, Bulk Upload, Insights
Testing & Refinement	Apr 5–10, 2025	User Testing, Bug Fixes, Performance Optimization
Final Submission	Apr 11, 2025	Complete Application, Final Report

Project Management Approach

For project management, I used a Gantt chart to visualize the timeline and track progress against scheduled milestones. Tasks were organized into the following categories:

- Completed
- In Progress
- Testing
- To Do
- Backlog

This approach allowed for clear timeline visualization, identification of critical paths, and accurate tracking of project progress. The Gantt chart helped in prioritizing tasks and ensuring that dependencies were addressed in the correct order.



Detailed Timeline

January 2025: Research & Design Phase

- Jan 17-22: Research on existing finance applications and technology options
- Jan 23-26: Created project proposal and sequence diagrams

- Jan 27-28: Finalized technology stack (Node.js, Express, MongoDB, Bootstrap/EJS, Google Gemini API)
- Jan 29-31: Set up version control and initial project structure

February 2025: Base Development Phase

- Feb 1-7: Set up project scaffolding and basic navigation
- Feb 8-14: Designed and implemented basic UI components
- Feb 15-21: Implemented user authentication system
- Feb 22-27: Created dashboard template with visualization components
- Feb 28: Submitted midterm report and demonstration

March 2025: Feature Implementation Phase

- Mar 1-7: Implemented expense tracking system with MongoDB integration
- Mar 8-14: Developed data visualization components using Chart.js
- Mar 15-21: Integrated Google Gemini API for natural language processing
- Mar 22-28: Implemented transaction management system
- Mar 29-31: Added date range filtering for all data views

April 2025: Final Phase

- Apr 1-4: Implemented bulk upload feature with CSV parsing
- Apr 5-6: Conducted user testing and evaluation
- Apr 7-9: Bug fixes and performance optimization
- Apr 10: Final testing and documentation
- Apr 11: Final submission of project and report

This timeline reflects the iterative development approach used for the project, with regular testing and refinement throughout the process. Each phase built upon the previous one, gradually expanding the application's capabilities while maintaining a focus on user experience and technical stability.

Implemented Features

Feature 1: Natural Language Input with AI Integration

The most innovative aspect of FinTrack is its natural language input system, which allows users to track expenses as easily as sending a text message. Instead of navigating complex forms with multiple fields, users can simply type phrases like "spent \$50 on dinner" or "paid \$1200 rent," and the application automatically processes and categorizes these expenses.

Implementation Details

The natural language input system is built around an integration with Google's Gemini API, which processes the user's input text and extracts relevant financial information. The implementation follows these steps:

1. User Input Capture: A simple text input field allows users to enter their expense in natural language.
2. AI Processing: The input is sent to Google Gemini API through a prompt that requests structured data extraction:

Javascript

```
let prompt =
  'Give me json response only for this expense in the format \
  | {"expense": "expense name", "amount": "amount", category: "category"} for ' +
  expense +
  '. Example {"expense": "rent", "amount": "1000", category: "housing"}. \
  Allowed categories are housing, food, transportation, utilities, clothing, insurance, \
  medical, personal, debt, savings, entertainment, education, gifts, donations, investments, others. \
  If you are unable to provide this information, give me error json response in the format {"error": "error message"}.';

let expense_json = await askGemini(prompt);
await saveExpense(expense_json, req.username);
```

3. Data Storage: The structured expense data is then saved to MongoDB, linked to the user's account:

Javascript

```
async function saveExpense(expense_json, username) {
  const user = await User.findOne({ username });
  console.log(username);

  const { expense, amount, category } = expense_json;
  const newExpense = new Expense({ expense, amount, category, user: user._id });
  await newExpense.save();
  console.log("Expense added successfully:", newExpense);
}
```

4. Real-time Feedback: Users receive immediate confirmation that their expense has been tracked, along with the category it was assigned.

Homepage

The screenshot shows the FinTrack homepage with a dark background. At the top, there's a navigation bar with links for 'auth.js - W25_4495_S2_NidhiN' [Preview] README.md - W25_4, 'FinTrack', Cluster0 Data | Cloud: MongoDB, and a search bar. Below the navigation is a header with the title 'Track Finances Like You're Chatting'. A sub-header says 'Simply type what you spent on - just like texting a friend. No complicated forms, no hassle.' To the right, a message box displays a tracked expense: "'had 50\$ dinner'" with a link, checked off with 'Tracked: \$50 in Dining category'. Below this is a 'Get Started - It's Free!' button. The main area features four cards: 'Natural Language Input' (text message icon), 'Smart Insights' (lightbulb icon), 'Visual Dashboard' (bar chart icon), and 'Privacy First' (padlock icon). Each card has a brief description below its title. At the bottom, a large button labeled 'How It Works' is visible.

auth.js - W25_4495_S2_NidhiN | [Preview] README.md - W25_4 | FinTrack | Cluster0 Data | Cloud: MongoDB | +

< > C ⌂ ⌂ fantastic-waffle-wr947qrr5xrf9965-3000.app.github.dev

FinTrack Home Login Register

Track Finances Like You're Chatting

Simply type what you spent on - just like texting a friend. No complicated forms, no hassle.

Get Started - It's Free!

"had 50\$ dinner" [\[link\]](#)
✓ Tracked: \$50 in Dining category

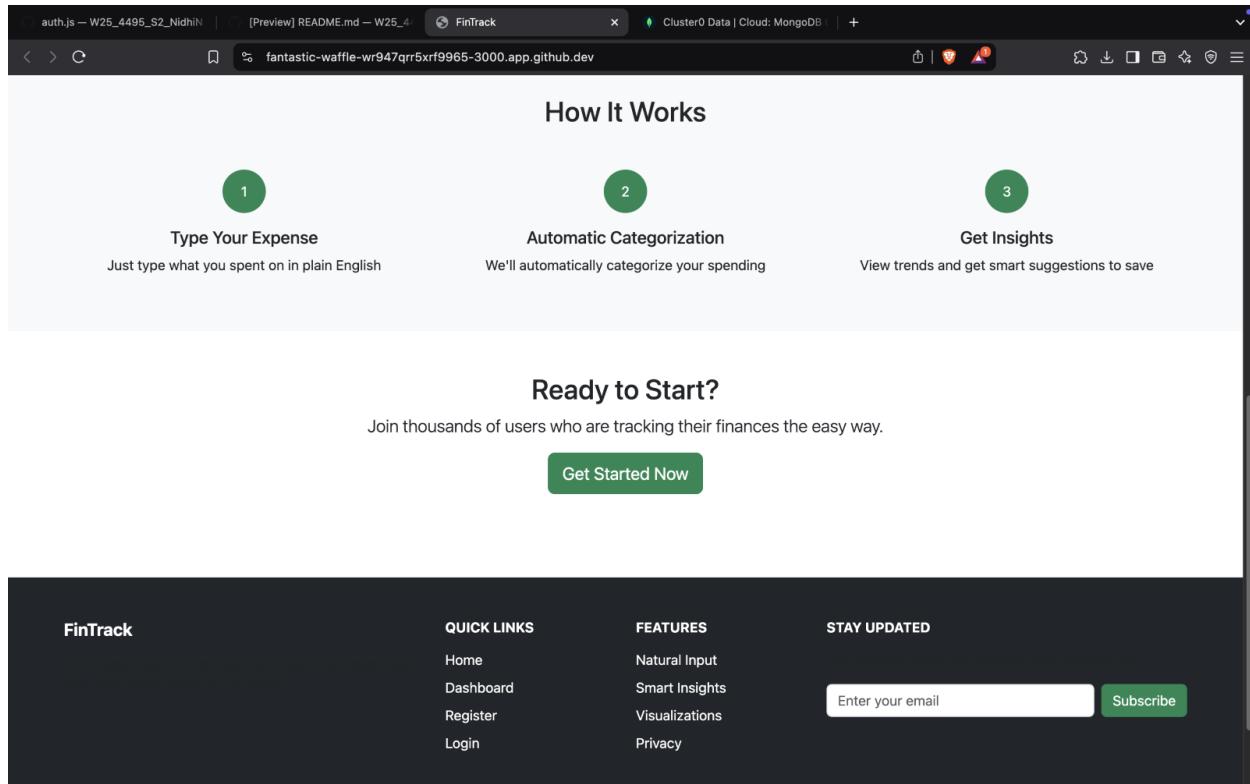
Natural Language Input
Track expenses by simply typing what you spent on - just like sending a text message.

Smart Insights
Get personalized suggestions to save money and find better financial options.

Visual Dashboard
Beautiful charts and graphs to understand your spending patterns at a glance.

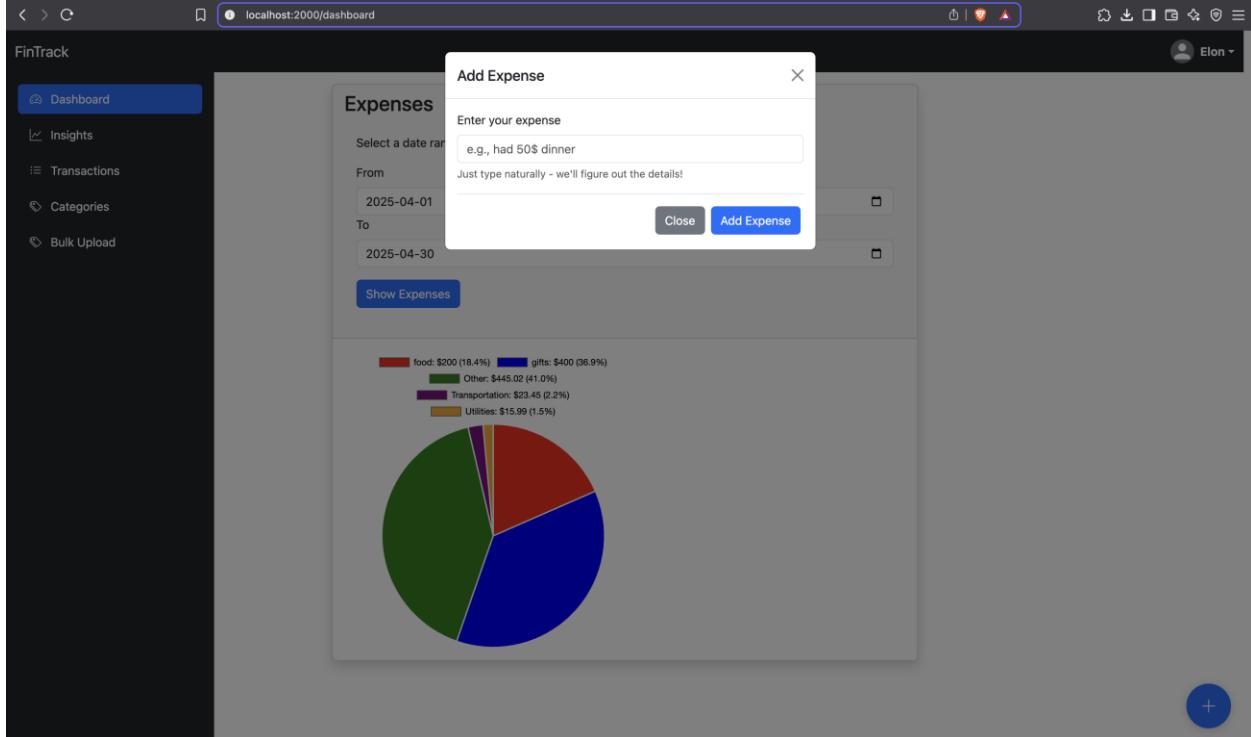
Privacy First
No personal information required. Start tracking your finances anonymously.

How It Works



Screenshots & Code

The implementation includes a floating button on all dashboard pages that opens a modal for expense input:



```
Implementation > views > layouts > floating-button.ejs ...
1  <!-- floating-button.ejs -->
2  <button type="button" class="btn btn-primary rounded-circle position-fixed d-flex align-items-center justify-content-center"
3  |   style="bottom: 2rem; right: 2rem; width: 60px; height: 60px; box-shadow: 0 2px 12px rgba(0,0,0,0.2);"
4  |   data-bs-toggle="modal"
5  |   data-bs-target="#addExpenseModal">
6  |     <i class="bi bi-plus-lg fs-4"></i>
7  </button>
8
9  <!-- Modal for Adding Expense -->
10 <div class="modal fade" id="addExpenseModal" tabindex="-1" aria-labelledby="addExpenseModalLabel" aria-hidden="true">
11   <div class="modal-dialog">
12     <div class="modal-content">
13       <div class="modal-header">
14         <h5 class="modal-title" id="addExpenseModalLabel">Add Expense</h5>
15         <button type="button" class="btn-close" data-bs-dismiss="modal" aria-label="Close"></button>
16       </div>
17       <div class="modal-body">
18         <form id="expenseForm" action="/dashboard/add-expense" method="POST">
19           <div class="mb-3">
20             <label for="expenseInput" class="form-label">Enter your expense</label>
21             <input type="text" class="form-control" id="expenseInput" name="expense"
22               placeholder="e.g., had 50$ dinner" required>
23             <div class="form-text">Just type naturally - we'll figure out the details!</div>
24           </div>
25           <div class="modal-footer px-0 pb-0">
26             <button type="button" class="btn btn-secondary" data-bs-dismiss="modal">Close</button>
27             <button type="submit" class="btn btn-primary">Add Expense</button>
28           </div>
29         </form>
30       </div>
31     </div>
32   </div>
33 </div>
```

The core of the implementation is in the askGemini function, which handles the API communication:

```
javascript
async function askGemini(
  prompt,
  apiKey = process.env.GOOGLE_API_KEY,
  model = "gemini-2.0-flash"
) {
  try {
    const url =
`https://generativelanguage.googleapis.com/v1beta/models/${model}
:generateContent?key=${apiKey}`;

    const response = await fetch(url, {
      method: "POST",
      headers: {
        "Content-Type": "application/json",
      },
      body: JSON.stringify({
        contents: [
          {
            parts: [{ text: prompt }],
          },
        ],
        generationConfig: {
          temperature: 0.7,
          topK: 40,
          topP: 0.95,
          maxOutputTokens: 1024,
        },
      }),
    });
  }

  let data = await response.json();
  const responseText =
data.candidates[0].content.parts[0].text;
  const cleanedJson = responseText.replace(/\^\^json|^\^/g,
```

```

    "").trim();
    expense_json = JSON.parse(cleanedJson);

    return expense_json;
} catch (error) {
    console.error("Error:", error);
    throw error;
}
}
}

```

This feature represents a significant advancement over traditional expense tracking methods, reducing the friction that often leads users to abandon financial applications. The AI component ensures accurate categorization without requiring user effort, making the experience seamless and intuitive.

Feature 2: Data Visualization and Dashboard

The dashboard serves as the central hub of the FinTrack application, providing users with a visual representation of their financial data. It was designed to give users an immediate understanding of their spending patterns through interactive charts and customizable date ranges.

Implementation Details

The dashboard visualization system consists of several key components:

1. **Dynamic Pie Chart:** Implemented using Chart.js, the pie chart displays expenses by category, allowing users to quickly identify their major spending areas.
2. **Custom Date Range Selection:** Users can select specific time periods to analyze, with the visualization updating dynamically based on the selected range.
3. **Category Breakdown:** Below the chart, a detailed breakdown shows the exact amount spent in each category.
4. **Real-time Updates:** When new expenses are added, the dashboard updates automatically to reflect the changes.

The implementation uses server-side data processing and client-side rendering:

```

javascript
// Server-side data preparation
async function getExpensesByCategoryOnDateRange(username,

```

```

from,to) {
  const user = await User.findOne({ username });
  const expenses = await Expense.find({
    user: user._id,
    date: {
      $gte: from,
      $lte: to
    }
  }).sort({ date: -1 });

  const categoryTotals = {};
  expenses.forEach(({ category, amount }) => {
    categoryTotals[category] = (categoryTotals[category] || 0) +
amount;
  });

  const p_data = {
    labels: Object.keys(categoryTotals),
    values: Object.values(categoryTotals),
  };
  return p_data;
}

// Client-side chart rendering
<script>
  let all_data = <%- JSON.stringify(c_data) %>;
  const labels = all_data.labels;
  const values = all_data.values;

  const barColors = [
    "red", "blue", "green", "purple", "orange",
    "yellow", "brown", "pink", "grey", "cyan"
  ];

  new Chart("pieChart", {

```

```
    type: "pie",
    data: {
        labels: labels,
        datasets: [
            backgroundColor: barColors,
            data: values
        ]
    },
    options: {
        plugins: {
            legend: {
                position: 'top',
                labels: {
                    generateLabels: function(chart) {
                        const data = chart.data;
                        const total =
                            data.datasets[0].data.reduce((a, b) => a + b, 0);

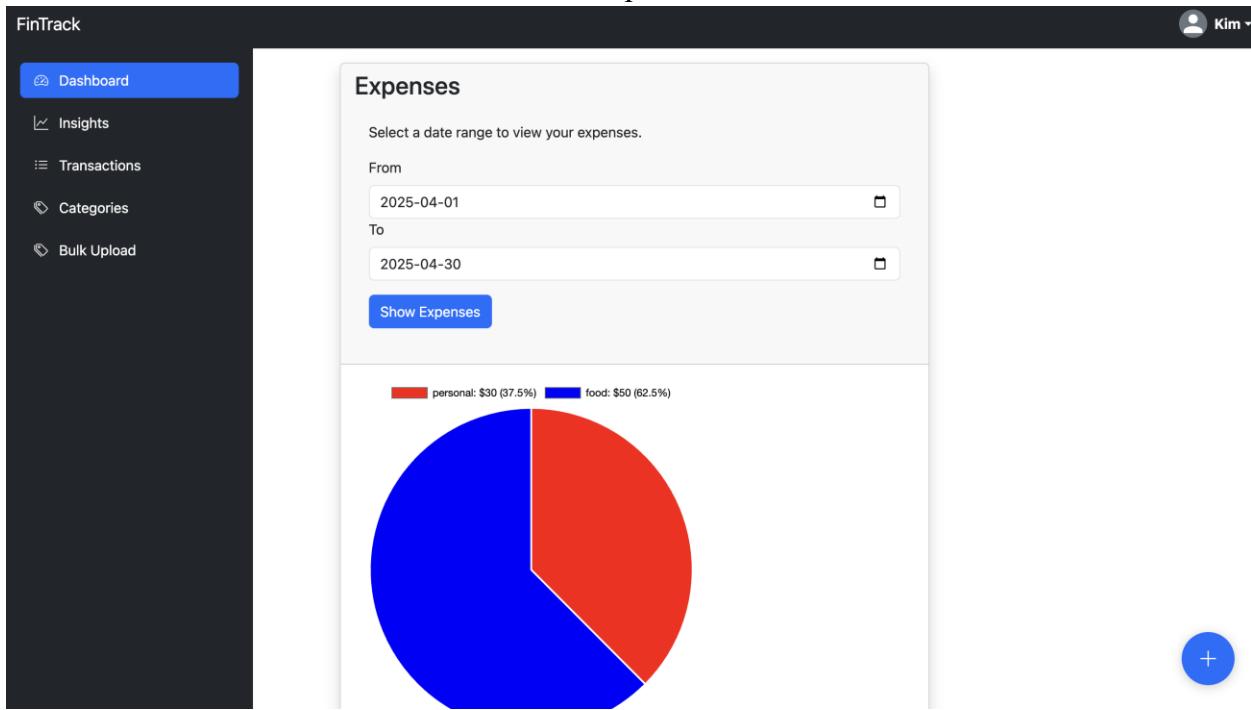
                        return
                    data.labels.map((label,i)=> {
                            const value =
                                data.datasets[0].data[i];
                            const percentage = ((value *
100) / total).toFixed(1);

                            return {
                                text: `${label}:
${value.toLocaleString()} (${percentage}%)`,
                                fillStyle:
                                chart.data.datasets[0].backgroundColor[i],
                                index: i
                            };
                    });
                }
            }
        }
    }
}
```

```
        },
        tooltip: {
            callbacks: {
                label: function(context) {
                    const value = context.parsed;
                    const total =
context.dataset.data.reduce((a, b) => a + b, 0);
                    const percentage = ((value * 100)
/ total).toFixed(1);
                    return `${context.label}:
${value.toLocaleString()} (${percentage}%)`;
                }
            }
        }
    });
</script>
```

Screenshots & Code

The dashboard interface shows a clean, visual representation of financial data:



Piechart.ejs

EXPLORER ...

W25_4495_S2_NIDHIN

- > DocumentsAndRepo...
- Implementation ●
 - > middleware
 - > models
 - > node_modules
 - > public
 - routes
 - auth.js
 - JS dash.js
 - > uploads
 - > utils
- > views ●
 - > common
 - error.ejs
 - > dashboard
 - bulk-upload-val...
 - bulk-upload.ejs
 - categories.ejs
 - edit-transaction....
 - home.ejs
 - insights.ejs
 - transactions.ejs
 - upload-success....
 - > layouts ●
 - floating-button....
 - footer.ejs
 - header.ejs
 - loggedheader.ejs
 - piechart.ejs 4
 - sidebar.ejs
 - > visitor

.env M JS dash.js piechart.ejs 4 X

Implementation > views > layouts > piechart.ejs > ...

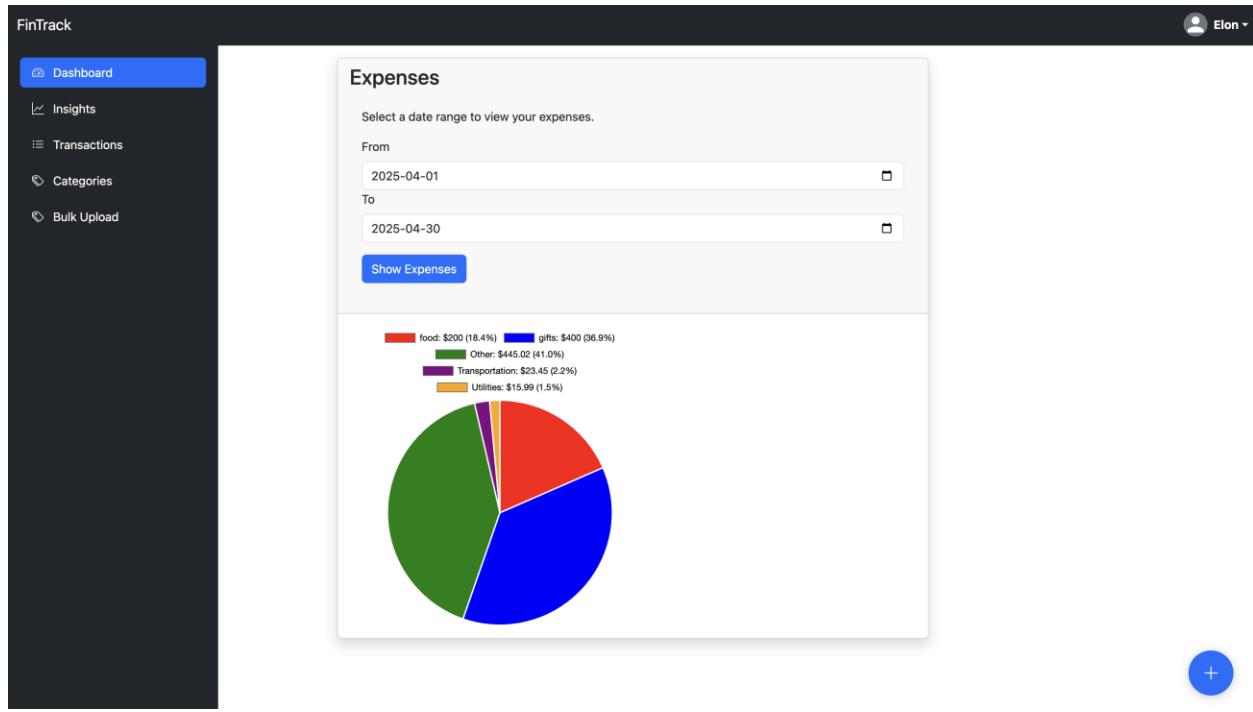
```
1 | 
2 | <!-- Chart container -->
3 | <div style="width: 400px; height:400px;">
4 |   <canvas id="pieChart"></canvas>
5 | </div>
6 | 
7 | <script>
8 |   let all_data =    JSON.stringify(c_data)   ;
9 |   const labels = all_data.labels;
10 |  const values = all_data.values;
11 | 
12 |  const barColors = [
13 |    "red",
14 |    "blue",
15 |    "green",
16 |    "purple",
17 |    "orange",
18 |    "yellow",
19 |    "brown",
20 |    "pink",
21 |    "grey",
22 |    "cyan"
23 |  ];
24 | 
25 |  new Chart("pieChart", {
26 |    type: "pie",
27 |    data: {
28 |      labels: labels,
29 |      datasets: [  
30 |        backgroundColor: barColors,
31 |        data: values
32 |      ]
33 |    },
34 |    options: {
35 |      plugins: {
36 |        legend: {
37 |          position: 'top',
38 |          labels: {
39 |            // Customize the legend labels
40 |            generateLabels: function(chart) {
41 |              const data = chart.data;
```

EXPLORERenv M JS dash.js ⌂ piechart.ejs 4 ×

Implementation > views > layouts > piechart.ejs > ...

```
    7  <script>
27      data: {
33        },
34        options: {
35          plugins: {
36            legend: {
37              position: 'top',
38              labels: {
39                // Customize the legend labels
40                generateLabels: function(chart) {
41                  const data = chart.data;
42                  const total = data.datasets[0].data.reduce((a, b) => a + b, 0);
43
44                  return data.labels.map((label, i) => {
45                    const value = data.datasets[0].data[i];
46                    const percentage = ((value * 100) / total).toFixed(1);
47
48                    return {
49                      text: `${label}: ${value.toLocaleString()} (${percentage}%)`,
50                      fillStyle: chart.data.datasets[0].backgroundColor[i],
51                      index: i
52                    };
53                  });
54                }
55              },
56            tooltip: {
57              callbacks: {
58                label: function(context) {
59                  const value = context.parsed;
60                  const total = context.dataset.data.reduce((a, b) => a + b, 0);
61                  const percentage = ((value * 100) / total).toFixed(1);
62
63                  return `${context.label}: ${value.toLocaleString()} (${percentage}%)`;
64                }
65              }
66            }
67          }
68        }
69      });
70    });
71  );
```

PRV M OUTLINE TIMELINE



Categories.ejs

EXPLORERenv M JS dash.js categories.ejs

Implementation > views > dashboard > categories.ejs > ...

```

1  1   <%- include('../layouts/loggedheader'); -%>
2   2   <div class="row">
3   3     <div class="col-3">
4   4       <div class="d-flex flex-column">
5   5         |   <%- include('../layouts/sidebar'); -%>
6   6       </div>
7   7     </div>
8   8     <div class="col-6">
9   9       <div class="card shadow m-3">
10 10         <div class="card-header">
11 11           <% if (locals.date_range) { %>
12 12             <h3>Expense from <%= locals.from %> to <%= locals.to %></h3>
13 13           </div>
14 14           <% } else { %>
15 15             <h3>Expenses this month</h3>
16 16           <% } %>
17 17         </div>
18 18       <div class="card-body">
19 19         <p class="card-text">Select a date range to view your category expenses.</p>
20 20         <form action="/dashboard/categories/date-range" method="GET">
21 21           <div class="mb-3">
22 22             <label for="from" class="form-label">From</label>
23 23             <input type="date"
24 24               class="form-control"
25 25                 id="from"
26 26                 name="from"
27 27                 value="<%= locals.from ? from : new Date().toISOString().split('T')[0] %>" 
28 28                 required>
29 29
30 30             <label for="to" class="form-label">To</label>
31 31             <input type="date"
32 32               class="form-control"
33 33                 id="to"
34 34                 name="to"
35 35                 value="<%= locals.to ? to : new Date().toISOString().split('T')[0] %>" 
36 36                 required>
37 37
38 38           <button type="submit" class="btn btn-primary mt-3">Show Categories</button>
39 39         </div>
40 40       </form>
41 41

```

.env M

> OUTLINE

> TIMELINE

```

43      <div class="card-body">
44          <div class="table-responsive">
45              <% if(typeof success_msg !== 'undefined' && success_msg != ''){ %>
46                  <div class="alert alert-success alert-dismissible fade show" role="alert">
47                      <%= success_msg %>
48                      <button type="button" class="btn-close" data-bs-dismiss="alert" aria-label="Close"></button>
49                  </div>
50              <% } %>
51          <table class="table table-striped">
52              <thead class="table-dark">
53                  <tr>
54                      <th scope="col">Category</th>
55                      <th scope="col">Amount</th>
56                  </tr>
57              </thead>
58          </table>
59      </div>
60      <% if (expenses.length === 0) { %>
61          <div class="alert alert-info">
62              No expenses found. Add your first expense to get started.
63          </div>
64      <% } %>
65
66      <div class="table-responsive">
67          <table class="table table-striped">
68              <tbody>
69                  <% for (var i = 0; i < expenses.labels.length; i++) { %>
70                      <tr>
71                          <td><%= expenses.labels[i]%></td>
72                          <td>$<%= expenses.values[i].toFixed(2) %></td>
73                      </tr>
74                  <% } %>
75              </tbody>
76          </table>
77      </div>
78  </div>
79 </div>
80
52      <thead class="table-dark">
53          <tr>
54              <th scope="col">Category</th>
55              <th scope="col">Amount</th>
56          </tr>
57      </thead>
58      <table class="table table-striped">
59          <tbody>
60              <% if (expenses.length === 0) { %>
61                  <div class="alert alert-info">
62                      No expenses found. Add your first expense to get started.
63                  </div>
64              <% } %>
65
66                  <div class="table-responsive">
67                      <table class="table table-striped">
68                          <tbody>
69                              <% for (var i = 0; i < expenses.labels.length; i++) { %>
70                                  <tr>
71                                      <td><%= expenses.labels[i]%></td>
72                                      <td>$<%= expenses.values[i].toFixed(2) %></td>
73                                  </tr>
74                              <% } %>
75                      </tbody>
76                  </table>
77              </div>
78          </div>
79      </div>
80
81      <% include('../layouts/floating-button') %>
82      <% include('../layouts/footer'); -%>

```

The screenshot shows a web-based application named FinTrack. The left sidebar contains navigation links: Dashboard, Insights, **Transactions** (which is currently selected), Categories, and Bulk Upload. The main content area is titled "Expenses this month" and includes a date range selector from "2025-04-01" to "2025-04-30". A "Show Expenses" button is present. Below this is a table listing ten transactions:

Expense	Amount	Category	Date	Actions
Gas	\$65.45	Other	4/29/2025	Edit Delete
Professional membership	\$350.00	Other	4/27/2025	Edit Delete
Equipment purchase	\$899.99	Other	4/24/2025	Edit Delete
Parking fees	\$45.00	Other	4/21/2025	Edit Delete
Car rental	\$175.25	Other	4/19/2025	Edit Delete
Contractor payment	\$1200.00	Other	4/17/2025	Edit Delete
Phone bill	\$120.30	Other	4/14/2025	Edit Delete
hair spa	\$30.00	personal	4/12/2025	Edit Delete
dinner	\$50.00	food	4/12/2025	Edit Delete
Internet bill	\$89.99	Other	4/11/2025	Edit Delete

Transaction.ejs

EXPLORER ...

- W25... 📁 ↻ ⌂ ⏺
- > DocumentsAndRepo...
- Implementation ●
 - > middleware
 - > models
 - > node_modules
 - > public
 - > routes
 - JS auth.js
 - JS dash.js
 - > uploads
 - > utils
- > views
 - > common
 - ▷ error.ejs
 - > dashboard
 - ▷ bulk-upload-val...
 - ▷ bulk-upload.ejs
 - ▷ categories.ejs
 - ▷ edit-transaction....
 - ▷ home.ejs
 - ▷ insights.ejs
 - ▷ transactions.ejs
 - ▷ upload-success....
 - > layouts
 - ▷ floating-button....
 - ▷ footer.ejs
 - ▷ header.ejs
 - ▷ loggedheader.ejs
 - ▷ piechart.ejs
 - ▷ sidebar.ejs
 - > visitor

ENV M

OUTLINE

TIMELINE

.env M JS dash.js transactions.ejs

```

Implementation > views > dashboard > transactions.ejs > ⓘ ?
1  <%- include('../layouts/loggedheader'); -%>
2  <div class="row">
3      <div class="col-3">
4          <div class="d-flex flex-column">
5              | <%- include('../layouts/sidebar'); -%>
6          </div>
7      </div>
8      <div class="col-6">
9          <div class="card shadow m-3">
10         <div class="card-header">
11             <% if (locals.date_range) { %>
12                 <h3>Expense from <%= locals.from %> to <%= locals.to %></h3>
13             </div>
14             <% } else { %>
15                 <h3>Expenses this month</h3>
16             <% } %>
17         </div>
18         <div class="card-body">
19             <p class="card-text">Select a date range to view your expenses.</p>
20             <form action="/dashboard/transactions/date-range" method="GET">
21                 <div class="mb-3">
22                     <label for="from" class="form-label">From</label>
23                     <input type="date"
24                         class="form-control"
25                         id="from"
26                         name="from"
27                         value="<%= locals.from ? from : new Date().toISOString().split('T')[0] %>" required>
28
29                     <label for="to" class="form-label">To</label>
30                     <input type="date"
31                         class="form-control"
32                         id="to"
33                         name="to"
34                         value="<%= locals.to ? to : new Date().toISOString().split('T')[0] %>" required>
35
36                     <button type="submit" class="btn btn-primary mt-3">Show Expenses</button>
37                 </div>
38             </form>
39         </div>
40     </div>
41 
```

EXPLORER ...

- W25... 📁 ↻ ⌂ ⏷
- > DocumentsAndRepo...
- Implementation ●
 - > middleware
 - > models
 - > node_modules
 - > public
 - > routes
 - JS auth.js
 - JS dash.js
 - > uploads
 - > utils
- > views
 - > common
 - ▷ error.ejs
 - > dashboard
 - ▷ bulk-upload-val...
 - ▷ bulk-upload.ejs
 - ▷ categories.ejs
 - ▷ edit-transaction....
 - ▷ home.ejs
 - ▷ insights.ejs
 - ▷ transactions.ejs
 - ▷ upload-success....
 - > layouts
 - ▷ floating-button....
 - ▷ footer.ejs
 - ▷ header.ejs
 - ▷ loggedheader.ejs
 - ▷ piechart.ejs
 - ▷ sidebar.ejs
 - > visitor

ENV M

OUTLINE

TIMELINE

.env M JS dash.js transactions.ejs

```

Implementation > views > dashboard > transactions.ejs > ⓘ ?
1  <%- include('../layouts/loggedheader'); -%>
2  <div class="row">
3      <div class="col-3">
4          <div class="d-flex flex-column">
5              | <%- include('../layouts/sidebar'); -%>
6          </div>
7      </div>
8      <div class="col-6">
9          <div class="card shadow m-3">
10         <div class="card-header">
11             <% if (locals.date_range) { %>
12                 <h3>Expense from <%= locals.from %> to <%= locals.to %></h3>
13             </div>
14             <% } else { %>
15                 <h3>Expenses this month</h3>
16             <% } %>
17         </div>
18         <div class="card-body">
19             <p class="card-text">Select a date range to view your expenses.</p>
20             <form action="/dashboard/transactions/date-range" method="GET">
21                 <div class="mb-3">
22                     <label for="from" class="form-label">From</label>
23                     <input type="date"
24                         class="form-control"
25                         id="from"
26                         name="from"
27                         value="<%= locals.from ? from : new Date().toISOString().split('T')[0] %>" required>
28
29                     <label for="to" class="form-label">To</label>
30                     <input type="date"
31                         class="form-control"
32                         id="to"
33                         name="to"
34                         value="<%= locals.to ? to : new Date().toISOString().split('T')[0] %>" required>
35
36                     <button type="submit" class="btn btn-primary mt-3">Show Expenses</button>
37                 </div>
38             </form>
39         </div>
40     </div>
41 
```

```

41   </div>
42   <div class="card-body">
43     <div class="table-responsive">
44       <% if(typeof success_msg !== 'undefined' && success_msg != ''){ %>
45         <div class="alert alert-success alert-dismissible fade show" role="alert">
46           <%= success_msg %>
47           <button type="button" class="btn-close" data-bs-dismiss="alert" aria-label="Close"></button>
48         </div>
49       <% } %>
50     <table class="table table-striped">
51       <thead class="table-dark">
52         <tr>
53           <th scope="col">Expense</th>
54           <th scope="col">Amount</th>
55           <th scope="col">Category</th>
56           <th scope="col">Date</th>
57           <th scope="col">Actions</th>
58         </tr>
59       </thead>
60       <tbody>
61         <% expenses.forEach(expense => { %>
62           <tr>
63             <td><%= expense.expense %></td>
64             <td>$<%= expense.amount.toFixed(2) %></td>
65             <td><%= expense.category %></td>
66             <td><%= new Date(expense.date).toLocaleDateString() %></td>
67             <td>
68               <div class="btn-group" role="group">
69                 <a href="/dashboard/transactions/edit/<%= expense._id %>" class="btn btn-sm btn-primary">Edit</a>
70                 <a href="/dashboard/transactions/delete/<%= expense._id %>" class="btn btn-sm btn-danger">Delete</a>
71               </div>
72             </td>
73           </tr>
74         <% }); %>
75       </tbody>
76     </table>
77   </div>
78   <% if (expenses.length === 0) { %>
79     <div class="alert alert-info">
80       No expenses found. Add your first expense to get started.
81     </div>
82   <% } %>
83   </div>
84 </div>
85 </div>
86 <% include('../layouts/floating-button') %>
87 <% include('../layouts/footer'); %>

```

```

55   <th scope="col">Category</th>
56   <th scope="col">Date</th>
57   <th scope="col">Actions</th>
58   </tr>
59   </thead>
60   <tbody>
61     <% expenses.forEach(expense => { %>
62       <tr>
63         <td><%= expense.expense %></td>
64         <td>$<%= expense.amount.toFixed(2) %></td>
65         <td><%= expense.category %></td>
66         <td><%= new Date(expense.date).toLocaleDateString() %></td>
67         <td>
68           <div class="btn-group" role="group">
69             <a href="/dashboard/transactions/edit/<%= expense._id %>" class="btn btn-sm btn-primary">Edit</a>
70             <a href="/dashboard/transactions/delete/<%= expense._id %>" class="btn btn-sm btn-danger">Delete</a>
71           </div>
72         </td>
73       </tr>
74     <% }); %>
75   </tbody>
76   </table>
77 </div>
78   <% if (expenses.length === 0) { %>
79     <div class="alert alert-info">
80       No expenses found. Add your first expense to get started.
81     </div>
82   <% } %>
83   </div>
84 </div>
85 </div>
86 <% include('../layouts/floating-button') %>
87 <% include('../layouts/footer'); %>

```

Edit-transaction.ejs

EXPLORER env M JS dash.js edit-transaction.ejs X

```

Implementation > views > dashboard > edit-transaction.ejs > ⓘ ?
1  <%- include('../layouts/loggedheader'); -%>
2  <div class="row">
3      <div class="col-3">
4          <div class="d-flex flex-column">
5              |   <%- include('../layouts/sidebar'); -%>
6          </div>
7      </div>
8      <div class="col-6">
9          <div class="card">
10             <div class="card-header">
11                 |   <h5 class="card-title">Edit Expense</h5>
12             </div>
13             <div class="card-body">
14                 <form action="/dashboard/transactions/edit/<%= expense._id %>" method="POST">
15                     <div class="mb-3">
16                         <label for="expense" class="form-label">Expense Name</label>
17                         <input type="text" class="form-control" id="expense" name="expense" value="<%= expense.expense %>" required>
18                     </div>
19
20                     <div class="mb-3">
21                         <label for="amount" class="form-label">Amount</label>
22                         <input type="number" step="0.01" class="form-control" id="amount" name="amount" value="<%= expense.amount %>" required>
23                     </div>
24
25                     <div class="mb-3">
26                         <label for="category" class="form-label">Category</label>
27                         <select class="form-select" id="category" name="category" required>
28                             <option value="">Select a category</option>
29                             <option value="housing" <%= expense.category === 'housing' ? 'selected' : '' %>>Housing</option>
30                             <option value="food" <%= expense.category === 'food' ? 'selected' : '' %>>Food</option>
31                             <option value="transportation" <%= expense.category === 'transportation' ? 'selected' : '' %>>Transportation</option>
32                             <option value="utilities" <%= expense.category === 'utilities' ? 'selected' : '' %>>Utilities</option>
33                             <option value="clothing" <%= expense.category === 'clothing' ? 'selected' : '' %>>Clothing</option>
34                             <option value="insurance" <%= expense.category === 'insurance' ? 'selected' : '' %>>Insurance</option>
35                             <option value="medical" <%= expense.category === 'medical' ? 'selected' : '' %>>Medical</option>
36                             <option value="personal" <%= expense.category === 'personal' ? 'selected' : '' %>>Personal</option>
37                             <option value="debt" <%= expense.category === 'debt' ? 'selected' : '' %>>Debt</option>
38                             <option value="savings" <%= expense.category === 'savings' ? 'selected' : '' %>>Savings</option>
39                             <option value="entertainment" <%= expense.category === 'entertainment' ? 'selected' : '' %>>Entertainment</option>
40                             <option value="education" <%= expense.category === 'education' ? 'selected' : '' %>>Education</option>
41                             <option value="gifts" <%= expense.category === 'gifts' ? 'selected' : '' %>>Gifts</option>
42                             <option value="donations" <%= expense.category === 'donations' ? 'selected' : '' %>>Donations</option>
43                             <option value="investments" <%= expense.category === 'investments' ? 'selected' : '' %>>Investments</option>
44                             <option value="others" <%= expense.category === 'others' ? 'selected' : '' %>>Others</option>
45                         </select>
46                     </div>
47
48                     <div class="mb-3">
49                         <label for="date" class="form-label">Date</label>
50                         <input type="date" class="form-control" id="date" name="date" value="<%= new Date(expense.date).toISOString().split('T')[0] %>" required>
51                     </div>
52
53                     <div class="d-flex justify-content-between">
54                         <button type="submit" class="btn btn-primary">Update Expense</button>
55                         <a href="/dashboard/transactions" class="btn btn-secondary">Cancel</a>
56                     </div>
57                 </div>
58             </div>
59         </div>
60     </div>
61 </div>
62 <%- include('../layouts/floating-button') %>
63 <%- include('../layouts/footer'); -%>
```

```

34 <option value="clothing" <%= expense.category === 'clothing' ? 'selected' : '' %>>Clothing</option>
35 <option value="insurance" <%= expense.category === 'insurance' ? 'selected' : '' %>>Insurance</option>
36 <option value="medical" <%= expense.category === 'medical' ? 'selected' : '' %>>Medical</option>
37 <option value="personal" <%= expense.category === 'personal' ? 'selected' : '' %>>Personal</option>
38 <option value="debt" <%= expense.category === 'debt' ? 'selected' : '' %>>Debt</option>
39 <option value="savings" <%= expense.category === 'savings' ? 'selected' : '' %>>Savings</option>
40 <option value="entertainment" <%= expense.category === 'entertainment' ? 'selected' : '' %>>Entertainment</option>
41 <option value="education" <%= expense.category === 'education' ? 'selected' : '' %>>Education</option>
42 <option value="gifts" <%= expense.category === 'gifts' ? 'selected' : '' %>>Gifts</option>
43 <option value="donations" <%= expense.category === 'donations' ? 'selected' : '' %>>Donations</option>
44 <option value="investments" <%= expense.category === 'investments' ? 'selected' : '' %>>Investments</option>
45 <option value="others" <%= expense.category === 'others' ? 'selected' : '' %>>Others</option>
46 </select>
47
48 <div class="mb-3">
49     <label for="date" class="form-label">Date</label>
50     <input type="date" class="form-control" id="date" name="date" value="<%= new Date(expense.date).toISOString().split('T')[0] %>" required>
51 </div>
52
53 <div class="d-flex justify-content-between">
54     <button type="submit" class="btn btn-primary">Update Expense</button>
55     <a href="/dashboard/transactions" class="btn btn-secondary">Cancel</a>
56 </div>
57 </div>
58 </div>
59 </div>
60 </div>
61 </div>
62 <%- include('../layouts/floating-button') %>
63 <%- include('../layouts/footer'); -%>
```

Home.ejs

EXPLORER env M JS dash.js ◊ home.ejs ×

Implementation > views > dashboard > ◊ home.ejs > ...

```

1   <%- include('../layouts/loggedheader'); -%>
2   <div class="row">
3     <div class="col-3">
4       <div class="d-flex flex-column">
5         <%- include('../layouts/sidebar'); -%>
6       </div>
7     </div>
8     <div class="col-6">
9       <div class="card shadow m-3">
10      <div class="card-header">
11        <% if (locals.date_range) { %>
12          <h3>Expense from <%= locals.from %> to <%= locals.to %></h3>
13        <% } else { %>
14          <h3>Expenses</h3>
15        <% } %>
16      <div class="card-body">
17        <p>Select a date range to view your expenses.</p>
18        <form action="/dashboard/date-range" method="GET">
19          <div class="mb-3">
20            <label for="from" class="form-label">From</label>
21            <input type="date"
22                  class="form-control"
23                  id="from"
24                  name="from"
25                  value="<%= locals.from ? from : new Date().toISOString().split('T')[0] %>">
26            required>
27
28            <label for="to" class="form-label">To</label>
29            <input type="date"
30                  class="form-control"
31                  id="to"
32                  name="to"
33                  value="<%= locals.to ? to : new Date().toISOString().split('T')[0] %>">
34            required>
35
36            <button type="submit" class="btn btn-primary mt-3">Show Expenses</button>
37
38          </div>
39        </form>
40      </div>
41    </div>
42  </div>
43  <% if (locals.message) { %>
44    <div class="alert alert-success" role="alert">
45      <%= message %>
46    </div>
47  <% } %>
48  <div class="card-body">
49    <%- include('../layouts/piechart') %>
50  </div>
51  </div>
52 </div>
53 <%- include('../layouts/floating-button') %>
54 <%- include('../layouts/footer'); -%>
```

```

23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54
55
```

The screenshot shows the FinTrack application interface. At the top, there is a dark header bar with the text "FinTrack" on the left and a user profile icon with the name "Kim" on the right. Below the header is a navigation sidebar on the left side of the main content area. The sidebar contains five items: "Dashboard" (with a house icon), "Insights" (with a bar chart icon, highlighted in blue to indicate it is the active page), "Transactions" (with a list icon), "Categories" (with a folder icon), and "Bulk Upload" (with a file icon). The main content area has a white background and a title "Insights this month" at the top. Under this title, there are three sections: "Summary", "Insight", and "Suggestions".

Summary

Two expenses were recorded: one for a hair spa (personal) costing \$30 and another for dinner (food) costing \$50.

Insight

The data shows spending in both 'personal' and 'food' categories, indicating a focus on both self-care and basic needs.

Suggestions

Consider tracking expenses over a longer period (e.g., a month) to identify spending patterns and areas where you might be able to save. For example, in Vancouver, consider cooking at home more often to reduce food expenses, given the relatively high cost of dining out.

insights.ejs

EXPLORER ...

w25... DocumentsAndRepo...

Implementation ●

- middleware
- models
- node_modules
- public
- routes
 - auth.js
 - dash.js
 - uploads
 - utils
- views
 - common
 - error.ejs
 - dashboard
 - bulk-upload-val...
 - bulk-upload.ejs
 - categories.ejs
 - edit-transaction....
 - home.ejs
 - insights.ejs
 - transactions.ejs
 - upload-success....
 - layouts
 - floating-button....
 - footer.ejs
 - header.ejs
 - loggedheader.ejs
 - piechart.ejs
 - sidebar.ejs
 - visitor

.env M JS dash.js insights.ejs

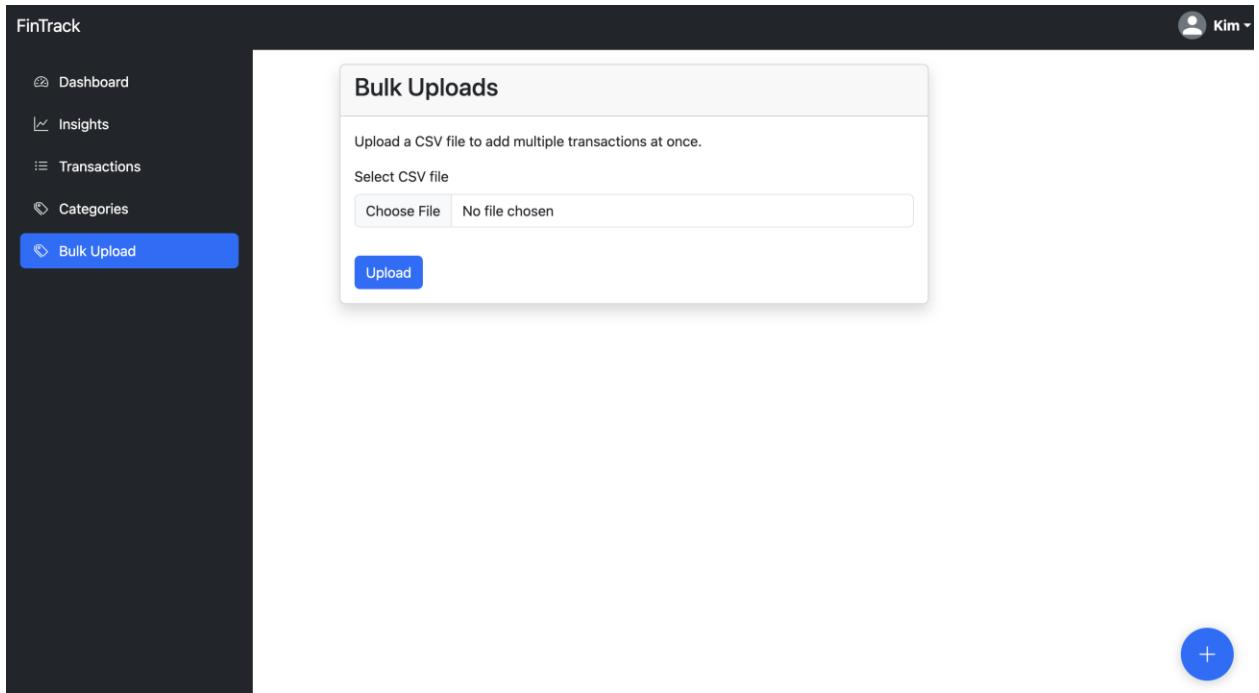
Implementation > views > dashboard > insights.ejs > ...

```
1  <%- include('../layouts/loggedheader'); -%>
2  <div class="row">
3      <div class="col-3">
4          <div class="d-flex flex-column">
5              |      <%- include('../layouts/sidebar'); -%>
6          </div>
7      </div>
8  </div>
9  <div class="col-6">
10     <div class="card shadow m-3">
11         <div class="card-header">
12             |             <h3>Insights this month</h3>
13         </div>
14         <div class="card-body">
15             |                 <h5>Summary</h5>
16             |             <p><%= insights.Summary %></p>
17             |             <br/>
18
19             |                 <h5>Insight</h5>
20             |             <p><%= insights.Insight %></p>
21             |             <br/>
22
23             |                 <h5>Suggestions</h5>
24             |             <p><%= insights.Suggestion %></p>
25             |             <br/>
26         </div>
27     </div>
28  </div>
29 </div>
30 <%- include('../layouts/floating-button') %>
31 <%- include('../layouts/footer'); -%>
```

env M

> OUTLINE

> TIMELINE



Bulk-upload.ejs

EXPLORER ...

W25... 🏷️ 🎯 ⚙️ 🗃️

DocumentsAndRepo...

Implementation ●

- middleware
- models
- node_modules
- public
- routes
 - auth.js
 - dash.js
 - uploads
 - utils
 - views
 - common
 - error.ejs
 - dashboard
 - bulk-upload-validation.ejs
 - bulk-upload.ejs
 - categories.ejs
 - edit-transaction.ejs
 - home.ejs
 - insights.ejs
 - transactions.ejs
 - upload-success.ejs
 - layouts
 - floating-button.ejs
 - footer.ejs
 - header.ejs
 - loggedheader.ejs
 - piechart.ejs
 - sidebar.ejs
 - visitor

.env M JS dash.js 🔍 bulk-upload.ejs ×

Implementation > views > dashboard > 🔍 bulk-upload.ejs > ...

```
1  <%- include('../layouts/loggedheader'); -%>
2  <div class="row">
3      <div class="col-3">
4          <div class="d-flex flex-column">
5              <%- include('../layouts/sidebar'); -%>
6          </div>
7      </div>
8      <div class="col-6">
9          <div class="card shadow m-3">
10             <div class="card-header">
11                 <h3>Bulk Uploads</h3>
12             </div>
13             <div class="card-body">
14                 <p class="card-text">Upload a CSV file to add multiple transactions at once.</p>
15                 <form action="/dashboard/bulk-upload/validate" method="POST" enctype="multipart/form-data">
16                     <div class="mb-3">
17                         <label for="file" class="form-label">Select CSV file</label>
18                         <input type="file" class="form-control" id="file" name="file" accept=".csv" required>
19                     </div>
20                     <button type="submit" class="btn btn-primary mt-3">Upload</button>
21                 </form>
22             </div>
23         </div>
24     </div>
25 </div>
26 </div>
27 <%- include('../layouts/floating-button') %>
28 <%- include('../layouts/footer'); -%>
```

ENV M

> OUTLINE

> TIMELINE

FinTrack

Description	Amount	Category	Date
Grocery shopping	56.78	Food	2025-04-08
Monthly rent	1200.00	Housing	2025-04-01
Uber ride	23.45	Transportation	2025-04-09
Netflix subscription	15.99	Entertainment	2025-04-05
Dinner at restaurant	42.50	Food	2025-04-07
Gym membership	30.00	Healthcare	2025-04-03
New shoes	89.99	Other	2025-04-06
Electric bill	75.30	Utilities	2025-04-02
Movie tickets	24.00	Entertainment	2025-04-10
Coffee shop	4.75	Food	2025-04-11

Bulk-upload-validate.ejs

EXPLORER env M JS dash.js ↗ bulk-upload-validate.ejs

```

W25_495_S2_NIDHIN
> DocumentsAndRepo...
  ↘ Implementation
    > middleware
    > models
    > node_modules
    > public
    < routes
      > auth.js
      > dash.js
      > uploads
      > utils
    < views
      < common
        <> error.ejs
      < dashboard
        <> bulk-upload-validate.ejs
        <> bulk-upload.ejs
        <> categories.ejs
        <> edit-transaction....
        <> home.ejs
        <> insights.ejs
        <> transactions.ejs
        <> upload-success....
      < layouts
        <> floating-button....
        <> footer.ejs
        <> header.ejs
        <> loggedheader.ejs
        <> piechart.ejs
        <> sidebar.ejs
      > visitor
    > env
    > OUTLINE
    > TIMELINE

```

Implementation > views > dashboard > ↗ bulk-upload-validate.ejs > ? > div.row > div.col-3 > div.d-flex.flex-column > ?

```

1  <%~ include('../layouts/loggedheader'); ~%>
2  <div class="row">
3    <div class="col-3">
4      <div class="d-flex flex-column">
5        | <%~ include('../layouts/sidebar'); ~%>
6        </div>
7      </div>
8      <div class="col-9">
9        <div class="card shadow m-3">
10       <div class="card-header">
11         <h3>Validate Expenses</h3>
12       </div>
13       <div class="card-body">
14         <p>Please review the expenses extracted from your CSV file. You can make any necessary edits before saving to your account.</p>
15
16         <form action="/dashboard/bulk-upload/save" method="POST">
17           <div class="table-responsive">
18             <table class="table table-striped">
19               <thead>
20                 <tr>
21                   <th>Description</th>
22                   <th>Amount</th>
23                   <th>Category</th>
24                   <th>Date</th>
25                 </tr>
26               </thead>
27               <tbody>
28                 <% expenses.forEach(expense, index) => { %>
29                   <tr>
30                     <td>
31                       <input type="text" class="form-control" name="expenses[<%= index %>][expense]" value="<%= expense.expense %>" required>
32                     </td>
33                     <td>
34                       <input type="number" step="0.01" class="form-control" name="expenses[<%= index %>][amount]" value="<%= expense.amount %>" required>
35                     </td>
36                     <td>
37                       <select class="form-select" name="expenses[<%= index %>][category]" required>
38                         <option value="" disabled>Select category</option>
39                         <option value="Food" <%= expense.category === 'Food' ? 'selected' : '' %>>Food</option>
40                         <option value="Transportation" <%= expense.category === 'Transportation' ? 'selected' : '' %>>Transportation</option>
41                         <option value="Housing" <%= expense.category === 'Housing' ? 'selected' : '' %>>Housing</option>
42                     </td>
43                 </tr>
44               </tbody>
45             </table>
46           </div>
47         </form>
48       </div>
49     </div>
50   </div>
51   <div class="col-12">
52     <div style="text-align: right; margin-top: 10px;">
53       <a href="#" class="btn btn-primary" style="margin-right: 10px;">SaveCancel

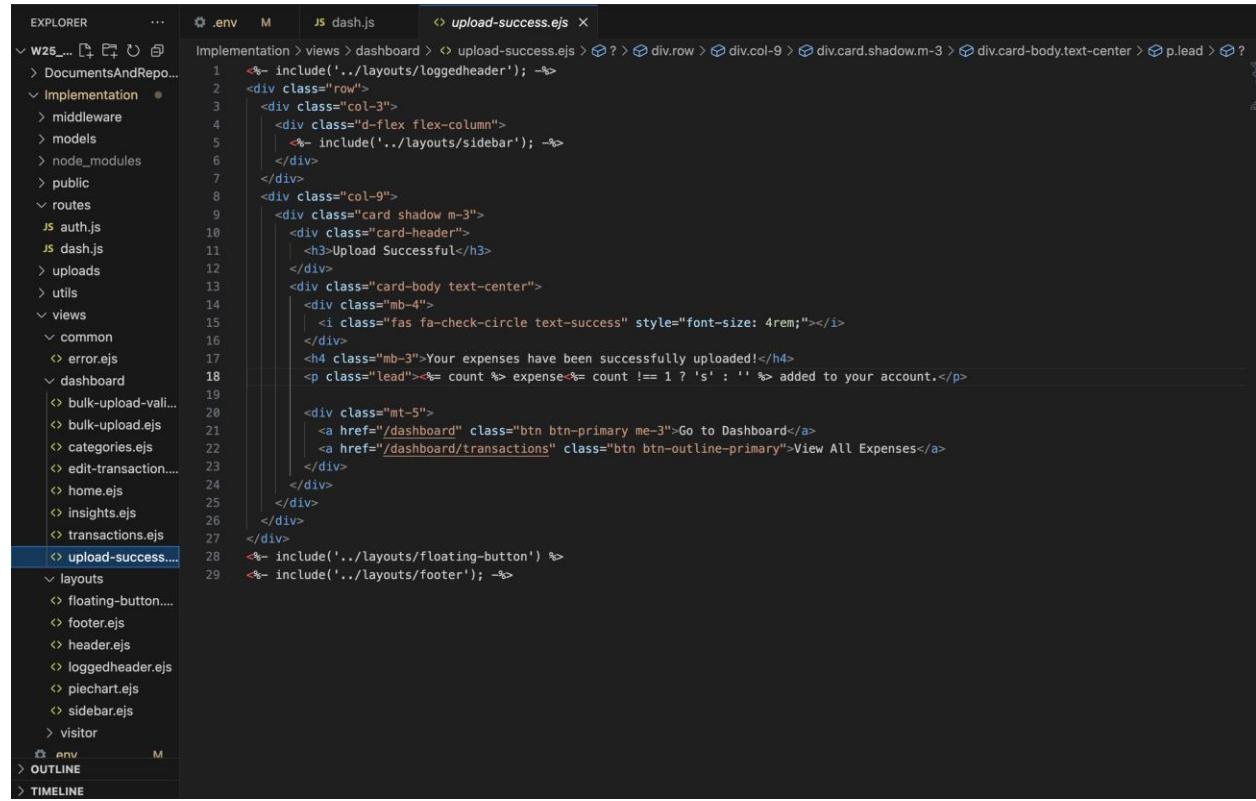
```

```

45      <option value="Utilities" >expense.category <del> Utilities </del> <del> selected </del> >Utilities <option>
46      <option value="Other" >(expense.category != 'Food' &&
47          <del> expense.category != 'Transportation' &&
48          <del> expense.category != 'Housing' &&
49          <del> expense.category != 'Entertainment' &&
50          <del> expense.category != 'Healthcare' &&
51          <del> expense.category != 'Shopping' &&
52          <del> expense.category != 'Utilities' ) ? 'selected' : '' >Other</option>
53      </select>
54  </td>
55  <td>
56      <input type="date" class="form-control" name="expenses[<%= index %>][date]"
57          value="<%= new Date(expense.date).toISOString().split('T')[0] %>" required>
58      </td>
59      </tr>
60  <% }); %>
61  </tbody>
62  </table>
63 </div>
64
65  <div class="d-flex justify-content-between mt-4">
66      <a href="/dashboard/bulk-upload" class="btn btn-secondary">Cancel</a>
67      <button type="submit" class="btn btn-primary">Save All Expenses</button>
68  </div>
69  </form>
70 </div>
71 </div>
72 </div>
73 </div>
74 <%- include('../layouts/floating-button') %>
75 <%- include('../layouts/footer'); -%>

```

Upload-success.ejs



The screenshot shows the Visual Studio Code interface with the file `upload-success.ejs` open in the editor. The code is a template for displaying a success message after file uploads. It includes imports for `loggedheader`, `sidebar`, and `floating-button`. The main content area displays a success message with an icon, a count of uploaded expenses, and links to go back to the dashboard or view all transactions.

```

Implementation > views > dashboard > <code>upload-success.ejs</code> > <code>?</code> > <code>div.row > div.col-9 > div.card.shadow.m-3 > div.card-body.text-center > p.lead > ?</code>
1 <%- include('../layouts/loggedheader'); -%>
2 <div class="row">
3     <div class="col-3">
4         <div class="d-flex flex-column">
5             <%- include('../layouts/sidebar'); -%>
6         </div>
7     </div>
8     <div class="col-9">
9         <div class="card shadow m-3">
10            <div class="card-header">
11                <h3>Upload Successful</h3>
12            </div>
13            <div class="card-body text-center">
14                <i class="fas fa-check-circle text-success" style="font-size: 4rem;"/>
15                <div class="mb-4">
16                    <h4>Your expenses have been successfully uploaded!</h4>
17                    <p class="lead"><code>count *> expense<code> count != 1 ? 's' : '' &gt; added to your account.</code></p>
18                <div class="mt-5">
19                    <a href="/dashboard" class="btn btn-primary me-3">Go to Dashboard</a>
20                    <a href="/dashboard/transactions" class="btn btn-outline-primary">View All Expenses</a>
21                </div>
22            </div>
23        </div>
24    </div>
25 </div>
26 </div>
27 </div>
28 <%- include('../layouts/floating-button') %>
29 <%- include('../layouts/footer'); -%>

```

Feature 3: Transaction Management System

The Transaction Management System provides users with complete control over their financial data, allowing them to view, edit, and delete expenses with a user-friendly interface. This system ensures that users can maintain accurate financial records and make corrections as needed.

Implementation Details

The transaction management system includes several key components:

1. **Transaction Listing:** A paginated, filterable list of all expenses with details like date, amount, category, and description.
2. **Edit Functionality:** Users can edit any aspect of a transaction, including its description, amount, category, and date.
3. **Delete Capability:** Allows users to remove erroneous or unwanted transactions.
4. **Date Filtering:** Users can filter transactions by date range to focus on specific time periods.

The implementation uses Express.js routes for server-side processing and EJS templates for dynamic rendering:

```
javascript
// Route for displaying transactions
router.get("/transactions", verifyToken, async (req, res) => {
  try {
    // Calculate first and last day of current month
    const today = new Date();
    const firstDay = new Date(today.getFullYear(),
    today.getMonth(), 1);
    const lastDay = new Date(today.getFullYear(),
    today.getMonth() + 1, 0);

    const formatDate = (date) => {
      const year = date.getFullYear();
      const month = String(date.getMonth() + 1).padStart(2, '0');
      const day = String(date.getDate()).padStart(2, '0');
      return `${year}-${month}-${day}`;
    };
  }
});
```

```
const defaultFrom = formatDate(firstDay);
const defaultTo = formatDate(lastDay);

const expenses = await getExpensesonDateRange(req.username,
defaultFrom, defaultTo);
res.render("dashboard/transactions.ejs", {
  path: "/dashboard/transactions",
  firstname: req.firstname,
  expenses: expenses,
  success_msg: req.query.success_msg || "",
  from: defaultFrom,
  to: defaultTo,
});
} catch (error) {
  console.error("Error:", error);
  res.render("common/error.ejs", {
    message: "Failed to fetch expenses. Please try again.",
  });
}
});

// Route for editing transactions
router.post("/transactions/edit/:id", verifyToken, async (req,
res) => {
  try {
    const expenseId = req.params.id;
    const { expense, amount, category, date } = req.body;

    // Validate input
    if (!expense || !amount || !category) {
      return res.render("common/error.ejs", {
        message: "All fields are required.",
      });
    }
  }
});
```

```
// Find the expense to verify it belongs to the user
const existingExpense = await Expense.findById(expenseId);
if (!existingExpense) {
  return res.status(404).render("common/error.ejs", {
    message: "Expense not found.",
  });
}

// Find the user
const user = await User.findOne({ username: req.username });

// Verify the expense belongs to the current user
if (existingExpense.user.toString() !== user._id.toString())
{
  return res.status(403).render("common/error.ejs", {
    message: "You don't have permission to edit this
expense.",
  });
}

// Update the expense
const updatedExpense = await Expense.findByIdAndUpdate(
  expenseId,
  {
    expense,
    amount: parseFloat(amount),
    category,
    date: date || existingExpense.date,
  },
  { new: true }
);

// Redirect back to transactions page with success message
res.redirect()
```

```

"/dashboard/transactions?success_msg=Expense+updated+successfully"
""

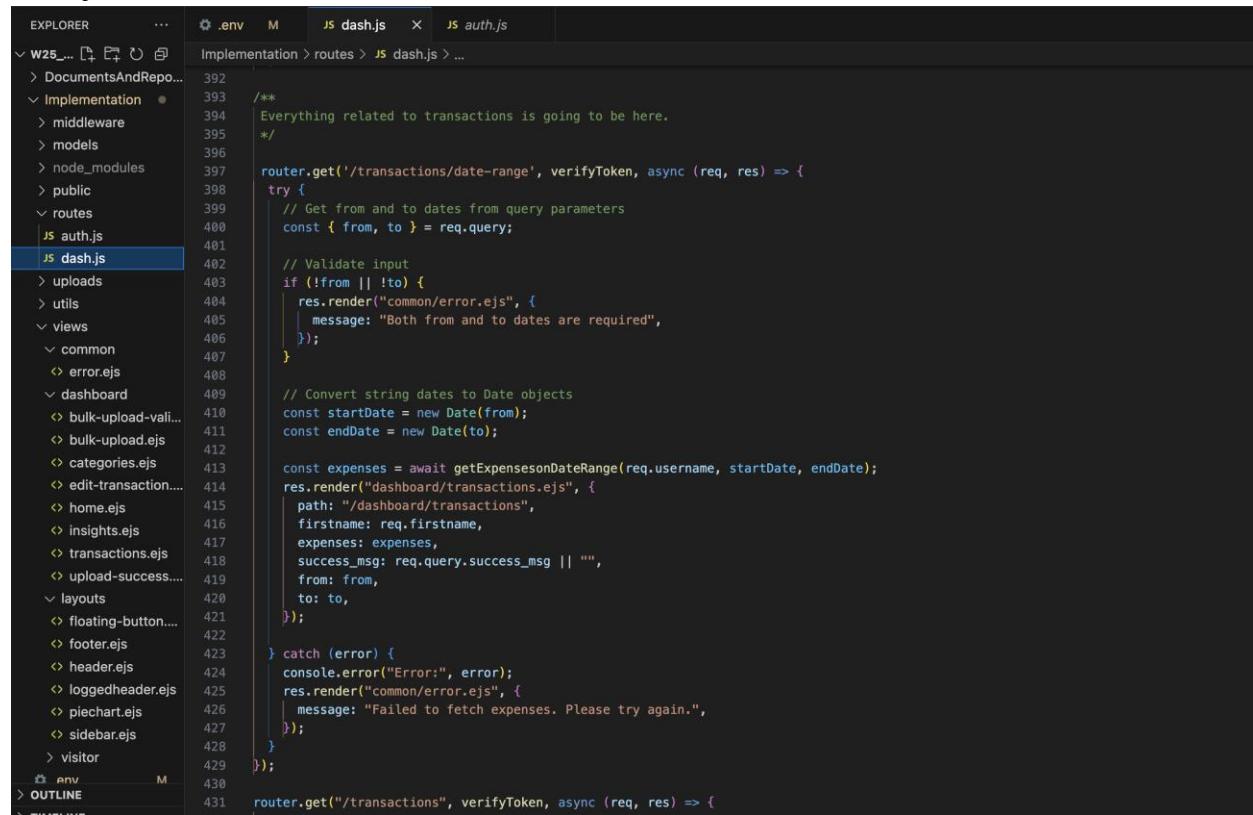
    );
} catch (error) {
  console.error("Error updating expense:", error);
  res.render("common/error.ejs", {
    message: "Failed to update expense. Please try again.",
  });
}
});

```

Screenshots & Code

The transaction management interface provides a clean, tabular view of expenses:

Dash.js



```

EXPLORER      ...   .env   M   JS dash.js   X   JS auth.js
Implementation > routes > JS dash.js > ...
392
393 /**
394  * Everything related to transactions is going to be here.
395 */
396
397 router.get('/transactions/date-range', verifyToken, async (req, res) => {
398   try {
399     // Get from and to dates from query parameters
400     const { from, to } = req.query;
401
402     // Validate input
403     if (!from || !to) {
404       res.render("common/error.ejs", {
405         message: "Both from and to dates are required",
406       });
407     }
408
409     // Convert string dates to Date objects
410     const startDate = new Date(from);
411     const endDate = new Date(to);
412
413     const expenses = await getExpensesOnDateRange(req.username, startDate, endDate);
414     res.render("dashboard/transactions.ejs", {
415       path: "/dashboard/transactions",
416       firstname: req.firstname,
417       expenses: expenses,
418       success_msg: req.query.success_msg || "",
419       from: from,
420       to: to,
421     });
422
423   } catch (error) {
424     console.error("Error:", error);
425     res.render("common/error.ejs", {
426       message: "Failed to fetch expenses. Please try again.",
427     });
428   }
429 });
430
431 router.get("/transactions", verifyToken, async (req, res) => {

```

EXPLORERenv M JS dash.js X JS auth.js

```
Implementation > routes > JS dash.js > ...
563 );
564
565 // Edit transaction route - GET (Show edit form)
566 router.get("/transactions/edit/:id", verifyToken, async (req, res) => {
567   try {
568     const expenseId = req.params.id;
569
570     // Find the expense
571     const expense = await Expense.findById(expenseId);
572     if (!expense) {
573       return res.status(404).render("common/error.ejs", {
574         message: "Expense not found."
575       });
576     }
577
578     // Find the user
579     const user = await User.findOne({ username: req.username });
580
581     // Verify the expense belongs to the current user
582     if (expense.user.toString() !== user._id.toString()) {
583       return res.status(403).render("common/error.ejs", {
584         message: "You don't have permission to edit this expense."
585       });
586     }
587
588     // Render the edit form
589     res.render("dashboard/edit-transaction.ejs", {
590       path: "/dashboard/transactions",
591       firstname: req.firstname,
592       expense: expense,
593     });
594   } catch (error) {
595     console.error("Error fetching expense for edit:", error);
596     res.render("common/error.ejs", {
597       message: "Failed to load expense for editing. Please try again."
598     });
599   }
600 });
601
602 // Edit transaction route - POST (Process edit form)
603 router.post("/transactions/edit/:id", verifyToken, async (req, res) => {
```

EXPLORERenv M JS dash.js X JS auth.js

```
Implementation > routes > JS dash.js > ...
540 );
541
542 // Edit transaction route - POST (Process edit form)
543 router.post("/transactions/edit/:id", verifyToken, async (req, res) => {
544   try {
545     const expenseId = req.params.id;
546     const { expense, amount, category, date } = req.body;
547
548     // Validate input
549     if (!expense || !amount || !category) {
550       return res.render("common/error.ejs", {
551         message: "All fields are required."
552       });
553     }
554
555     // Find the expense to verify it belongs to the user
556     const existingExpense = await Expense.findById(expenseId);
557     if (!existingExpense) {
558       return res.status(404).render("common/error.ejs", {
559         message: "Expense not found."
560       });
561     }
562
563     // Find the user
564     const user = await User.findOne({ username: req.username });
565
566     // Verify the expense belongs to the current user
567     if (existingExpense.user.toString() !== user._id.toString()) {
568       return res.status(403).render("common/error.ejs", {
569         message: "You don't have permission to edit this expense."
570       });
571     }
572
573     // Update the expense
574     const updatedExpense = await Expense.findByIdAndUpdate(
575       expenseId,
576       {
577         expense,
578         amount: parseFloat(amount),
579         category,
580         date: date || existingExpense.date,
```

The screenshot shows a code editor interface with a sidebar and a main content area.

EXPLORER (Left Sidebar):

- w25... (File)
- > DocumentsAndRepo...
- Implementation (Folder)
 - middleware
 - models
 - node_modules
 - public
- routes
- JS auth.js
- JS dash.js** (Selected)
- uploads
- utils
- views (Folder)
 - common
 - error.ejs
 - dashboard
 - bulk-upload-validation.ejs
 - bulk-upload.ejs
 - categories.ejs
 - edit-transaction.ejs
 - home.ejs
 - insights.ejs
 - transactions.ejs
 - upload-success.ejs
 - layouts
 - floating-button.ejs
 - footer.ejs
 - header.ejs
 - loggedheader.ejs
 - piechart.ejs
 - sidebar.ejs
 - visitor

OUTLINE (Bottom Left):

TIMELINE (Bottom Left):

JS dash.js (Main Content Area):

```
Implementation > routes > JS dash.js > JS auth.js

607 const storage = multer.diskStorage({
608   destination: (req, file, cb) => {
610     },
611     filename: (req, file, cb) => {
612       cb(null, Date.now() + '-' + file.originalname);
613     }
614   });
615
616 const upload = multer({
617   storage: storage,
618   fileFilter: (req, file, cb) => {
619     if (file.mimetype === 'text/csv') {
620       cb(null, true);
621     } else {
622       cb(new Error('Only CSV files are allowed'), false);
623     }
624   }
625 });
626
627 // Route that renders the bulk upload page
628 router.get("/bulk-upload", verifyToken, async (req, res) => {
629   try {
630     res.render("dashboard/bulk-upload.ejs", {
631       path: "/dashboard/bulk-upload",
632       firstname: req.firstname,
633       message: ""
634     });
635   } catch (error) {
636     console.error("Error:", error);
637     res.render("common/error.ejs", {
638       message: "Failed to load bulk upload page. Please try again."
639     });
640   }
641 });
642
643 // Route to handle the CSV upload and validate with Gemini
644 router.post("/bulk-upload/validate", verifyToken, upload.single('file'), async (req, res) => {
645   try {
646     if (!req.file) {
647       return res.render("dashboard/bulk-upload.ejs", {
648         path: "/dashboard/bulk-upload",
649       });
650     }
651   }
652 });


```

EXPLORER ... **.env** M **JS dash.js** X **JS auth.js**

Implementation > routes > **JS dash.js** > ...

```

643 // Route to handle the CSV upload and validate with Gemini
644 router.post("/bulk-upload/validate", verifyToken, upload.single('file'), async (req, res) => {
645   try {
646     if (!req.file) {
647       return res.render("dashboard/bulk-upload.ejs", {
648         path: "/dashboard/bulk-upload",
649         firstname: req.firstname,
650         message: "Please upload a CSV file."
651       });
652     }
653
654     const results = [];
655
656     // Read and parse the CSV file
657     fs.createReadStream(req.file.path)
658       .pipe(csv())
659       .on('data', (data) => results.push(data))
660       .on('end', async () => {
661         try {
662           // Process the CSV data with Gemini
663
664           let prompt =
665             'Give me json response only for these expenses in json array format \
666             [{"expense": "expense name", "amount": "amount", category: "category", date: "date"}] for ' +
667             JSON.stringify(results) +
668             '. Example [{"expense": "rent", "amount": "1000", category: "housing", date: "2025-04-09"}, {"expense": "movie", "amount": "10", \
669             'Allowed categories are housing, food, transportation, utilities, clothing, insurance, \
670             'medical, personal, debt, savings, entertainment, education, gifts, donations, investments, others. \
671             'If you are unable to categories the category, write "others" in the category field. \
672             'If you are unable to provide this information, give me error json response in the format {"error": "error message"}.]';
673
674           const processedData = await askGemini(prompt);
675
676           // Create a temporary JWT token to store the processed data
677           const tempToken = jwt.sign(
678             { bulkExpenses: processedData },
679             'your-secret-key',
680             { expiresIn: '1h' } // Short expiration time for security
681           );
682
683           // Set the temporary token as a cookie
684           res.cookie('temp-expenses', tempToken, {
685             httpOnly: true,
686             maxAge: 3600000 // 1 hour in milliseconds
687           });
688
689           // Render the validation page
690           res.render("dashboard/bulk-upload-validate.ejs", {
691             path: "/dashboard/bulk-upload/validate",
692             firstname: req.firstname,
693             expenses: processedData
694           });
695
696           // Delete the temporary file
697           fs.unlinkSync(req.file.path);
698         } catch (error) {
699           console.error("Error processing data:", error);
700           res.render("common/error.ejs", {
701             message: "Failed to process CSV data. Please check your file format and try again."
702           });
703         }
704       } catch (error) {
705         console.error("Error uploading file:", error);
706         res.render("common/error.ejs", {
707           message: "Failed to upload file. Please try again."
708         });
709       });
710
711     });
712
713     // Route to save validated expenses to MongoDB
714     router.post("/bulk-upload/save", verifyToken, async (req, res) => {
715       try {
716         // Get the processed expenses from the JWT token in cookies
717         const tempToken = req.cookies['temp-expenses'];
718         const username = req.username;
719       }
720     });
721   }
722 }
723
724
725
726
727
728
729
730
731
732
733
734
735
736
737
738
739
740
741
742
743
744
745
746
747
748
749
750
751
752
753
754
755
756
757
758
759
760
761
762
763
764
765
766
767
768
769
770
771
772
773
774
775
776
777
778
779
780
781
782
783
784
785
786
787
788
789
790
791
792
793
794
795
796
797
798
799
800
801
802
803
804
805
806
807
808
809
810
811
812
813
814
815
816
817
818
819
820
821
822
823
824
825
826
827
828
829
830
831
832
833
834
835
836
837
838
839
840
841
842
843
844
845
846
847
848
849
850
851
852
853
854
855
856
857
858
859
860
861
862
863
864
865
866
867
868
869
870
871
872
873
874
875
876
877
878
879
880
881
882
883
884
885
886
887
888
889
890
891
892
893
894
895
896
897
898
899
900
901
902
903
904
905
906
907
908
909
910
911
912
913
914
915
916
917
918
919
920
921
922
923
924
925
926
927
928
929
930
931
932
933
934
935
936
937
938
939
940
941
942
943
944
945
946
947
948
949
950
951
952
953
954
955
956
957
958
959
960
961
962
963
964
965
966
967
968
969
970
971
972
973
974
975
976
977
978
979
980
981
982
983
984
985
986
987
988
989
990
991
992
993
994
995
996
997
998
999
1000
1001
1002
1003
1004
1005
1006
1007
1008
1009
1010
1011
1012
1013
1014
1015
1016
1017
1018
1019
1020
1021
1022
1023
1024
1025
1026
1027
1028
1029
1030
1031
1032
1033
1034
1035
1036
1037
1038
1039
1040
1041
1042
1043
1044
1045
1046
1047
1048
1049
1050
1051
1052
1053
1054
1055
1056
1057
1058
1059
1060
1061
1062
1063
1064
1065
1066
1067
1068
1069
1070
1071
1072
1073
1074
1075
1076
1077
1078
1079
1080
1081
1082
1083
1084
1085
1086
1087
1088
1089
1090
1091
1092
1093
1094
1095
1096
1097
1098
1099
1100
1101
1102
1103
1104
1105
1106
1107
1108
1109
1110
1111
1112
1113
1114
1115
1116
1117
1118
1119
1120
1121
1122
1123
1124
1125
1126
1127
1128
1129
1130
1131
1132
1133
1134
1135
1136
1137
1138
1139
1140
1141
1142
1143
1144
1145
1146
1147
1148
1149
1150
1151
1152
1153
1154
1155
1156
1157
1158
1159
1160
1161
1162
1163
1164
1165
1166
1167
1168
1169
1170
1171
1172
1173
1174
1175
1176
1177
1178
1179
1180
1181
1182
1183
1184
1185
1186
1187
1188
1189
1190
1191
1192
1193
1194
1195
1196
1197
1198
1199
1200
1201
1202
1203
1204
1205
1206
1207
1208
1209
1210
1211
1212
1213
1214
1215
1216
1217
1218
1219
1220
1221
1222
1223
1224
1225
1226
1227
1228
1229
1230
1231
1232
1233
1234
1235
1236
1237
1238
1239
1240
1241
1242
1243
1244
1245
1246
1247
1248
1249
1250
1251
1252
1253
1254
1255
1256
1257
1258
1259
1259
1260
1261
1262
1263
1264
1265
1266
1267
1268
1269
1270
1271
1272
1273
1274
1275
1276
1277
1278
1279
1280
1281
1282
1283
1284
1285
1286
1287
1288
1289
1289
1290
1291
1292
1293
1294
1295
1296
1297
1298
1299
1300
1301
1302
1303
1304
1305
1306
1307
1308
1309
1309
1310
1311
1312
1313
1314
1315
1316
1317
1318
1319
1319
1320
1321
1322
1323
1324
1325
1326
1327
1328
1329
1329
1330
1331
1332
1333
1334
1335
1336
1337
1338
1339
1339
1340
1341
1342
1343
1344
1345
1346
1347
1348
1349
1349
1350
1351
1352
1353
1354
1355
1356
1357
1358
1359
1359
1360
1361
1362
1363
1364
1365
1366
1367
1368
1369
1369
1370
1371
1372
1373
1374
1375
1376
1377
1378
1379
1379
1380
1381
1382
1383
1384
1385
1386
1387
1388
1389
1389
1390
1391
1392
1393
1394
1395
1396
1397
1398
1399
1399
1400
1401
1402
1403
1404
1405
1406
1407
1408
1409
1409
1410
1411
1412
1413
1414
1415
1416
1417
1418
1419
1419
1420
1421
1422
1423
1424
1425
1426
1427
1428
1429
1429
1430
1431
1432
1433
1434
1435
1436
1437
1438
1439
1439
1440
1441
1442
1443
1444
1445
1446
1447
1448
1449
1449
1450
1451
1452
1453
1454
1455
1456
1457
1458
1459
1459
1460
1461
1462
1463
1464
1465
1466
1467
1468
1469
1469
1470
1471
1472
1473
1474
1475
1476
1477
1478
1479
1479
1480
1481
1482
1483
1484
1485
1486
1487
1488
1489
1489
1490
1491
1492
1493
1494
1495
1496
1497
1498
1499
1499
1500
1501
1502
1503
1504
1505
1506
1507
1508
1509
1509
1510
1511
1512
1513
1514
1515
1516
1517
1518
1519
1519
1520
1521
1522
1523
1524
1525
1526
1527
1528
1529
1529
1530
1531
1532
1533
1534
1535
1536
1537
1538
1538
1539
1540
1541
1542
1543
1544
1545
1546
1547
1548
1549
1549
1550
1551
1552
1553
1554
1555
1556
1557
1558
1559
1559
1560
1561
1562
1563
1564
1565
1566
1567
1568
1569
1569
1570
1571
1572
1573
1574
1575
1576
1577
1578
1579
1579
1580
1581
1582
1583
1584
1585
1586
1587
1588
1589
1589
1590
1591
1592
1593
1594
1595
1596
1597
1598
1599
1599
1600
1601
1602
1603
1604
1605
1606
1607
1608
1609
1609
1610
1611
1612
1613
1614
1615
1616
1617
1618
1619
1619
1620
1621
1622
1623
1624
1625
1626
1627
1628
1629
1629
1630
1631
1632
1633
1634
1635
1636
1637
1638
1639
1639
1640
1641
1642
1643
1644
1645
1646
1647
1648
1649
1649
1650
1651
1652
1653
1654
1655
1656
1657
1658
1659
1659
1660
1661
1662
1663
1664
1665
1666
1667
1668
1669
1669
1670
1671
1672
1673
1674
1675
1676
1677
1678
1679
1679
1680
1681
1682
1683
1684
1685
1686
1687
1688
1689
1689
1690
1691
1692
1693
1694
1695
1696
1697
1698
1699
1699
1700
1701
1702
1703
1704
1705
1706
1707
1708
1709
1709
1710
1711
1712
1713
1714
1715
1716
1717
1718
1719
1719
1720
1721
1722
1723
1724
1725
1726
1727
1728
1729
1729
1730
1731
1732
1733
1734
1735
1736
1737
1738
1739
1739
1740
1741
1742
1743
1744
1745
1746
1747
1748
1749
1749
1750
1751
1752
1753
1754
1755
1756
1757
1758
1759
1759
1760
1761
1762
1763
1764
1765
1766
1767
1768
1769
1769
1770
1771
1772
1773
1774
1775
1776
1777
1778
1779
1779
1780
1781
1782
1783
1784
1785
1786
1787
1788
1789
1789
1790
1791
1792
1793
1794
1795
1796
1797
1798
1798
1799
1800
1801
1802
1803
1804
1805
1806
1807
1808
1809
1809
1810
1811
1812
1813
1814
1815
1816
1817
1818
1819
1819
1820
1821
1822
1823
1824
1825
1826
1827
1828
1829
1829
1830
1831
1832
1833
1834
1835
1836
1837
1838
1839
1839
1840
1841
1842
1843
1844
1845
1846
1847
1848
1849
1849
1850
1851
1852
1853
1854
1855
1856
1857
1858
1859
1859
1860
1861
1862
1863
1864
1865
1866
1867
1868
1869
1869
1870
1871
1872
1873
1874
1875
1876
1877
1878
1879
1879
1880
1881
1882
1883
1884
1885
1886
1887
1888
1888
1889
1890
1891
1892
1893
1894
1895
1896
1897
1898
1898
1899
1900
1901
1902
1903
1904
1905
1906
1907
1908
1909
1909
1910
1911
1912
1913
1914
1915
1916
1917
1918
1919
1919
1920
1921
1922
1923
1924
1925
1926
1927
1928
1929
1929
1930
1931
1932
1933
1934
1935
1936
1937
1938
1939
1939
1940
1941
1942
1943
1944
1945
1946
1947
1948
1949
1949
1950
1951
1952
1953
1954
1955
1956
1957
1958
1959
1959
1960
1961
1962
1963
1964
1965
1966
1967
1968
1969
1969
1970
1971
1972
1973
1974
1975
1976
1977
1978
1979
1979
1980
1981
1982
1983
1984
1985
1986
1987
1988
1989
1989
1990
1991
1992
1993
1994
1995
1996
1997
1998
1999
1999
2000
2001
2002
2003
2004
2005
2006
2007
2008
2009
2009
2010
2011
2012
2013
2014
2015
2016
2017
2018
2019
2019
2020
2021
2022
2023
2024
2025
2026
2027
2028
2029
2029
2030
2031
2032
2033
2034
2035
2036
2037
2038
2039
2039
2040
2041
2042
2043
2044
2045
2046
2047
2048
2049
2049
2050
2051
2052
2053
2054
2055
2056
2057
2058
2059
2059
2060
2061
2062
2063
2064
2065
2066
2067
2068
2069
2069
2070
2071
2072
2073
2074
2075
2076
2077
2078
2079
2079
2080
2081
2082
2083
2084
2085
2086
2087
2088
2089
2089
2090
2091
2092
2093
2094
2095
2096
2097
2098
2098
2099
2100
2101
2102
2103
2104
2105
2106
2107
2108
2109
2109
2110
2111
2112
2113
2114
2115
2116
2117
2118
2119
2119
2120
2121
2122
2123
2124
2125
2126
2127
2128
2129
2129
2130
2131
2132
2133
2134
2135
2136
2137
2138
2139
2139
2140
2141
2142
2143
2144
2145
2146
2147
2148
2149
2149
2150
2151
2152
2153
2154
2155
2156
2157
2158
2159
2159
2160
2161
2162
2163
2164
2165
2166
2167
2168
2169
2169
2170
2171
2172
2173
2174
2175
2176
2177
2178
2179
2179
2180
2181
2182
2183
2184
2185
2186
2187
2188
2189
2189
2190
2191
2192
2193
2194
2195
2196
2197
2198
2198
2199
2200
2201
2202
2203
2204
2205
2206
2207
2208
2209
2209
2210
2211
2212
2213
2214
2215
2216
2217
2218
2219
2219
2220
2221
2222
2223
2224
2225
2226
2227
2228
2229
2229
2230
2231
2232
2233
2234
2235
2236
2237
2238
2239
2239
2240
2241
2242
2243
2244
2245
2246
2247
2248
2249
2249
2250
2251
2252
2253
2254
2255
2256
2257
2258
2259
2259
2260
2261
2262
2263
2264
2265
2266
2267
2268
2269
2269
2270
2271
2272
2273
2274
2275
2276
2277
2278
2279
2279
2280
2281
2282
2283
2284
2285
2286
2287
2288
2289
2289
2290
2291
2292
2293
2294
2295
2296
2297
2297
2298
2299
2299
2300
2301
2302
2303
2304
2305
2306
2307
2308
2309
2309
2310
2311
2312
2313
2314
2315
2316
2317
2318
2319
2319
2320
2321
2322
2323
2324
2325
2326
2327
2328
2329
2329
2330
2331
2332
2333
2334
2335
2336
2337
2338
2339
2339
2340
2341
2342
2343
2344
2345
2346
2347
2348
2349
2349
2350
2351
2352
2353
2354
2355
2356
2357
2358
2359
2359
2360
2361
2362
2363
2364
2365
2366
2367
2368
2369
2369
2370
2371
2372
2373
2374
2375
2376
2377
2378
2379
2379
2380
2381
2382
2383
2384
2385
2386
2387
2388
2389
2389
2390
2391
2392
2393
2394
2395
2396
2397
2398
2398
2399
2400
2401
2402
2403
2404
2405
2406
2407
2408
2409
2409
2410
2411
2412
2413
2414
2415
2416
2417
2418
2419
2419
2420
2421
2422
2423
2424
2425
2426
2427
2428
2429
2429
2430
2431
2432
2433
2434
2435
2436
2437
2438
2439
2439
2440
2441
2442
2443
2444
2445
2446
2447
2448
2449
2449
2450
2451
2452
2453
2454
2455
2456
2457
2458
2459
2459
2460
2461
2462
2463
2464
2465
2466
2467
2468
2469
2469
2470
2471
2472
2473
2474
2475
2476
2477
2478
2479
2479
2480
2481
2482
2483
2484
2485
2486
2487
2488
2489
2489
2490
2491
2492
2493
2494
2495
2496
2497
2497
2498
2499
2499
2500
2501
2502
2503
2504
2505
2506
2507
2508
2509
2509
2510
2511
2512
2513
2514
2515
2516
2517
2518
2519
2519
2520
2521
2522
2523
2524
2525
2526
2527
2528
2529
2529
2530
2531
2532
2533
2534
2535
2536
2537
2538
2539
2539
2540
2541
2542
2543
2544
2545
2546
2547
2548
2549
2549
2550
2551
2552
2553
2554
2555
2556
2557
2558
2559
2559
2560
2561
2562
2563
2564
2565
2566
2567
2568
2569
2569
2570
2571
2572
2573
2574
2575
2576
2577
2578
2579
2579
2580
2581
2582
2583
2584
2585
2586
2587
2588
2589
2589
2590
2591
2592
2593
2594
2595
2596
2597
2597
2598
2599
2599
2600
2601
2602
2603
2604
2605
2606
2607
2608
2609
2609
2610
2611
2612
2613
2614
2615
2616
2617
2618
2619
2619
2620
2621
2622
2623
2624
2625
2626
2627
2628
2629
2629
2630
2631
2632
2633
2634
2635
2636
2637
2638
2639
2639
2640
2641
2642
2643
2644
2645
2646
2647
2648
2649
2649
2650
2651
2652
2653
2654
2655
2656
2657
2658
2659
2659
2660
2661
2662
2663
2664
2665
2666
2667
2668
2669
2669
2670
2671
2672
2673
2674
2675
2676
2677
2678
2679
2679
2680
2681
2682
2683
2684
2685
2686
2687
2688
2689
2689
2690
2691
2692
2693
2694
2695
2696
2697
2698
2698
2699
2700
2701
2702
2703
```

EXPLORERenv M JS dash.js X JS auth.js

Implementation > routes > JS dash.js > ...

```

714  router.post("/bulk-upload/save", verifyToken, async (req, res) => {
715    const username = req.username;
716
717    if (!tempToken) {
718      return res.render("common/error.ejs", {
719        message: "No expenses found to save. Please upload your CSV file again."
720      });
721
722    }
723
724    // Verify and decode the token
725    const decoded = jwt.verify(tempToken, 'your-secret-key');
726    const expenses = decoded.bulkExpenses;
727
728    if (!expenses || expenses.length === 0) {
729      return res.render("common/error.ejs", {
730        message: "No expenses found to save. Please upload your CSV file again."
731      });
732
733    }
734
735    // Check if the form data contains updated expenses
736    let expensesToSave = expenses;
737    if (req.body.expenses && Array.isArray(req.body.expenses)) {
738      expensesToSave = req.body.expenses;
739    }
740
741    const user = await User.findOne({ username });
742
743    // Create expense documents in MongoDB
744    const expenseDocuments = expensesToSave.map(exp => ({
745      expense: exp.expense,
746      amount: parseFloat(exp.amount),
747      category: exp.category,
748      date: new Date(exp.date),
749      user: user._id, // From the verifyToken middleware
750    }));
751
752    // Save all expenses to MongoDB
753    await Expense.insertMany(expenseDocuments);
754
755    // Clear the temporary cookie
756    res.clearCookie('temp-expenses');
757
758  });
759
760  // Render success page
761  res.render("dashboard/upload-success.ejs", {
762    path: "dashboard/bulk-upload",
763    firstname: req.firstname,
764    count: expenseDocuments.length
765  });
766
767  } catch (error) {
768    console.error("Error saving expenses:", error);
769    res.render("common/error.ejs", {
770      message: "Failed to save expenses to database. Please try again."
771    });
772
773  }
774
775  module.exports = router;
776
777
778
779
780
781
782
783
784
785
786
787
788
789
790
791
792
793
794
795
796
797
798
799
800
801
802
803
804
805
806
807
808
809
810
811
812
813
814
815
816
817
818
819
820
821
822
823
824
825
826
827
828
829
830
831
832
833
834
835
836
837
838
839
840
841
842
843
844
845
846
847
848
849
850
851
852
853
854
855
856
857
858
859
860
861
862
863
864
865
866
867
868
869
870
871
872
873
874
875
876
877
878
879
880
881
882
883
884
885
886
887
888
889
890
891
892
893
894
895
896
897
898
899
900
901
902
903
904
905
906
907
908
909
910
911
912
913
914
915
916
917
918
919
920
921
922
923
924
925
926
927
928
929
930
931
932
933
934
935
936
937
938
939
940
941
942
943
944
945
946
947
948
949
950
951
952
953
954
955
956
957
958
959
960
961
962
963
964
965
966
967
968
969
970
971
972
973
974
975
976
977
978
979
980
981
982
983
984
985
986
987
988
989
990
991
992
993
994
995
996
997
998
999
1000
1001
1002
1003
1004
1005
1006
1007
1008
1009
1010
1011
1012
1013
1014
1015
1016
1017
1018
1019
1020
1021
1022
1023
1024
1025
1026
1027
1028
1029
1030
1031
1032
1033
1034
1035
1036
1037
1038
1039
1040
1041
1042
1043
1044
1045
1046
1047
1048
1049
1050
1051
1052
1053
1054
1055
1056
1057
1058
1059
1060
1061
1062
1063
1064
1065
1066
1067
1068
1069
1070
1071
1072
1073
1074
1075
1076
1077
1078
1079
1080
1081
1082
1083
1084
1085
1086
1087
1088
1089
1090
1091
1092
1093
1094
1095
1096
1097
1098
1099
1100
1101
1102
1103
1104
1105
1106
1107
1108
1109
1110
1111
1112
1113
1114
1115
1116
1117
1118
1119
1120
1121
1122
1123
1124
1125
1126
1127
1128
1129
1130
1131
1132
1133
1134
1135
1136
1137
1138
1139
1140
1141
1142
1143
1144
1145
1146
1147
1148
1149
1150
1151
1152
1153
1154
1155
1156
1157
1158
1159
1160
1161
1162
1163
1164
1165
1166
1167
1168
1169
1170
1171
1172
1173
1174
1175
1176
1177
1178
1179
1180
1181
1182
1183
1184
1185
1186
1187
1188
1189
1190
1191
1192
1193
1194
1195
1196
1197
1198
1199
1200
1201
1202
1203
1204
1205
1206
1207
1208
1209
1210
1211
1212
1213
1214
1215
1216
1217
1218
1219
1220
1221
1222
1223
1224
1225
1226
1227
1228
1229
1230
1231
1232
1233
1234
1235
1236
1237
1238
1239
1240
1241
1242
1243
1244
1245
1246
1247
1248
1249
1250
1251
1252
1253
1254
1255
1256
1257
1258
1259
1260
1261
1262
1263
1264
1265
1266
1267
1268
1269
1270
1271
1272
1273
1274
1275
1276
1277
1278
1279
1280
1281
1282
1283
1284
1285
1286
1287
1288
1289
1290
1291
1292
1293
1294
1295
1296
1297
1298
1299
1300
1301
1302
1303
1304
1305
1306
1307
1308
1309
1310
1311
1312
1313
1314
1315
1316
1317
1318
1319
1320
1321
1322
1323
1324
1325
1326
1327
1328
1329
1330
1331
1332
1333
1334
1335
1336
1337
1338
1339
1340
1341
1342
1343
1344
1345
1346
1347
1348
1349
1350
1351
1352
1353
1354
1355
1356
1357
1358
1359
1360
1361
1362
1363
1364
1365
1366
1367
1368
1369
1370
1371
1372
1373
1374
1375
1376
1377
1378
1379
1380
1381
1382
1383
1384
1385
1386
1387
1388
1389
1390
1391
1392
1393
1394
1395
1396
1397
1398
1399
1400
1401
1402
1403
1404
1405
1406
1407
1408
1409
1410
1411
1412
1413
1414
1415
1416
1417
1418
1419
1420
1421
1422
1423
1424
1425
1426
1427
1428
1429
1430
1431
1432
1433
1434
1435
1436
1437
1438
1439
1440
1441
1442
1443
1444
1445
1446
1447
1448
1449
1450
1451
1452
1453
1454
1455
1456
1457
1458
1459
1460
1461
1462
1463
1464
1465
1466
1467
1468
1469
1470
1471
1472
1473
1474
1475
1476
1477
1478
1479
1480
1481
1482
1483
1484
1485
1486
1487
1488
1489
1490
1491
1492
1493
1494
1495
1496
1497
1498
1499
1500
1501
1502
1503
1504
1505
1506
1507
1508
1509
1510
1511
1512
1513
1514
1515
1516
1517
1518
1519
1520
1521
1522
1523
1524
1525
1526
1527
1528
1529
1530
1531
1532
1533
1534
1535
1536
1537
1538
1539
1540
1541
1542
1543
1544
1545
1546
1547
1548
1549
1550
1551
1552
1553
1554
1555
1556
1557
1558
1559
1560
1561
1562
1563
1564
1565
1566
1567
1568
1569
1570
1571
1572
1573
1574
1575
1576
1577
1578
1579
1580
1581
1582
1583
1584
1585
1586
1587
1588
1589
1590
1591
1592
1593
1594
1595
1596
1597
1598
1599
1600
1601
1602
1603
1604
1605
1606
1607
1608
1609
1610
1611
1612
1613
1614
1615
1616
1617
1618
1619
1620
1621
1622
1623
1624
1625
1626
1627
1628
1629
1630
1631
1632
1633
1634
1635
1636
1637
1638
1639
1640
1641
1642
1643
1644
1645
1646
1647
1648
1649
1650
1651
1652
1653
1654
1655
1656
1657
1658
1659
1660
1661
1662
1663
1664
1665
1666
1667
1668
1669
1670
1671
1672
1673
1674
1675
1676
1677
1678
1679
1680
1681
1682
1683
1684
1685
1686
1687
1688
1689
1690
1691
1692
1693
1694
1695
1696
1697
1698
1699
1700
1701
1702
1703
1704
1705
1706
1707
1708
1709
1710
1711
1712
1713
1714
1715
1716
1717
1718
1719
1720
1721
1722
1723
1724
1725
1726
1727
1728
1729
1730
1731
1732
1733
1734
1735
1736
1737
1738
1739
1740
1741
1742
1743
1744
1745
1746
1747
1748
1749
1749
1750
1751
1752
1753
1754
1755
1756
1757
1758
1759
1759
1760
1761
1762
1763
1764
1765
1766
1767
1768
1769
1769
1770
1771
1772
1773
1774
1775
1776
1777
1778
1779
1779
1780
1781
1782
1783
1784
1785
1786
1787
1788
1789
1789
1790
1791
1792
1793
1794
1795
1796
1797
1798
1799
1799
1800
1801
1802
1803
1804
1805
1806
1807
1808
1809
1809
1810
1811
1812
1813
1814
1815
1816
1817
1818
1819
1819
1820
1821
1822
1823
1824
1825
1826
1827
1828
1829
1829
1830
1831
1832
1833
1834
1835
1836
1837
1838
1838
1839
1840
1841
1842
1843
1844
1845
1846
1847
1848
1849
1849
1850
1851
1852
1853
1854
1855
1856
1857
1858
1859
1859
1860
1861
1862
1863
1864
1865
1866
1867
1868
1869
1869
1870
1871
1872
1873
1874
1875
1876
1877
1878
1879
1879
1880
1881
1882
1883
1884
1885
1886
1887
1887
1888
1889
1889
1890
1891
1892
1893
1894
1895
1896
1897
1898
1899
1899
1900
1901
1902
1903
1904
1905
1906
1907
1908
1909
1909
1910
1911
1912
1913
1914
1915
1916
1917
1918
1919
1919
1920
1921
1922
1923
1924
1925
1926
1927
1928
1929
1929
1930
1931
1932
1933
1934
1935
1936
1937
1938
1939
1939
1940
1941
1942
1943
1944
1945
1946
1947
1948
1949
1949
1950
1951
1952
1953
1954
1955
1956
1957
1958
1959
1959
1960
1961
1962
1963
1964
1965
1966
1967
1968
1969
1969
1970
1971
1972
1973
1974
1975
1976
1977
1978
1979
1979
1980
1981
1982
1983
1984
1985
1986
1987
1988
1989
1989
1990
1991
1992
1993
1994
1995
1996
1997
1998
1999
1999
2000
2001
2002
2003
2004
2005
2006
2007
2008
2009
2009
2010
2011
2012
2013
2014
2015
2016
2017
2018
2019
2019
2020
2021
2022
2023
2024
2025
2026
2027
2028
2029
2029
2030
2031
2032
2033
2034
2035
2036
2037
2038
2039
2039
2040
2041
2042
2043
2044
2045
2046
2047
2048
2049
2049
2050
2051
2052
2053
2054
2055
2056
2057
2058
2059
2059
2060
2061
2062
2063
2064
2065
2066
2067
2068
2069
2069
2070
2071
2072
2073
2074
2075
2076
2077
2078
2079
2079
2080
2081
2082
2083
2084
2085
2086
2087
2088
2088
2089
2090
2091
2092
2093
2094
2095
2096
2097
2098
2099
2099
2100
2101
2102
2103
2104
2105
2106
2107
2108
2109
2109
2110
2111
2112
2113
2114
2115
2116
2117
2118
2119
2119
2120
2121
2122
2123
2124
2125
2126
2127
2128
2129
2129
2130
2131
2132
2133
2134
2135
2136
2137
2138
2139
2139
2140
2141
2142
2143
2144
2145
2146
2147
2148
2149
2149
2150
2151
2152
2153
2154
2155
2156
2157
2158
2159
2159
2160
2161
2162
2163
2164
2165
2166
2167
2168
2169
2169
2170
2171
2172
2173
2174
2175
2176
2177
2178
2179
2179
2180
2181
2182
2183
2184
2185
2186
2187
2188
2188
2189
2190
2191
2192
2193
2194
2195
2196
2197
2198
2198
2199
2200
2201
2202
2203
2204
2205
2206
2207
2208
2209
2209
2210
2211
2212
2213
2214
2215
2216
2217
2218
2219
2219
2220
2221
2222
2223
2224
2225
2226
2227
2228
2229
2229
2230
2231
2232
2233
2234
2235
2236
2237
2238
2239
2239
2240
2241
2242
2243
2244
2245
2246
2247
2248
2249
2249
2250
2251
2252
2253
2254
2255
2256
2257
2258
2259
2259
2260
2261
2262
2263
2264
2265
2266
2267
2268
2269
2269
2270
2271
2272
2273
2274
2275
2276
2277
2278
2279
2279
2280
2281
2282
2283
2284
2285
2286
2287
2288
2289
2289
2290
2291
2292
2293
2294
2295
2296
2297
2298
2298
2299
2300
2301
2302
2303
2304
2305
2306
2307
2308
2309
2309
2310
2311
2312
2313
2314
2315
2316
2317
2318
2319
2319
2320
2321
2322
2323
2324
2325
2326
2327
2328
2329
2329
2330
2331
2332
2333
2334
2335
2336
2337
2338
2339
2339
2340
2341
2342
2343
2344
2345
2346
2347
2348
2349
2349
2350
2351
2352
2353
2354
2355
2356
2357
2358
2359
2359
2360
2361
2362
2363
2364
2365
2366
2367
2368
2369
2369
2370
2371
2372
2373
2374
2375
2376
2377
2378
2379
2379
2380
2381
2382
2383
2384
2385
2386
2387
2388
2388
2389
2390
2391
2392
2393
2394
2395
2396
2397
2397
2398
2399
2399
2400
2401
2402
2403
2404
2405
2406
2407
2408
2409
2409
2410
2411
2412
2413
2414
2415
2416
2417
2418
2419
2419
2420
2421
2422
2423
2424
2425
2426
2427
2428
2429
2429
2430
2431
2432
2433
2434
2435
2436
2437
2438
2439
2439
2440
2441
2442
2443
2444
2445
2446
2447
2448
2449
2449
2450
2451
2452
2453
2454
2455
2456
2457
2458
2459
2459
2460
2461
2462
2463
2464
2465
2466
2467
2468
2469
2469
2470
2471
2472
2473
2474
2475
2476
2477
2478
2479
2479
2480
2481
2482
2483
2484
2485
2486
2487
2488
2489
2489
2490
2491
2492
2493
2494
2495
2496
2497
2497
2498
2499
2499
2500
2501
2502
2503
2504
2505
2506
2507
2508
2509
2509
2510
2511
2512
2513
2514
2515
2516
2517
2518
2519
2519
2520
2521
2522
2523
2524
2525
2526
2527
2528
2529
2529
2530
2531
2532
2533
2534
2535
2536
2537
2538
2539
2539
2540
2541
2542
2543
2544
2545
2546
2547
2548
2549
2549
2550
2551
2552
2553
2554
2555
2556
2557
2558
2559
2559
2560
2561
2562
2563
2564
2565
2566
2567
2568
2569
2569
2570
2571
2572
2573
2574
2575
2576
2577
2578
2579
2579
2580
2581
2582
2583
2584
2585
2586
2587
2588
2589
2589
2590
2591
2592
2593
2594
2595
2596
2597
2597
2598
2599
2599
2600
2601
2602
2603
2604
2605
2606
2607
2608
2609
2609
2610
2611
2612
2613
2614
2615
2616
2617
2618
2619
2619
2620
2621
2622
2623
2624
2625
2626
2627
2628
2629
2629
2630
2631
2632
2633
2634
2635
2636
2637
2638
2639
2639
2640
2641
2642
2643
2644
2645
2646
2647
2648
2649
2649
2650
2651
2652
2653
2654
2655
2656
2657
2658
2659
2659
2660
2661
2662
2663
2664
2665
2666
2667
2668
2669
2669
2670
2671
2672
2673
2674
2675
2676
2677
2678
2679
2679
2680
2681
2682
2683
2684
2685
2686
2687
2688
2689
2689
2690
2691
2692
2693
2694
2695
2696
2697
2698
2698
2699
2699
2700
2701
2702
2703
2704
2705
2706
2707
2708
2709
2709
2710
2711
2712
2713
2714
2715
2716
2717
2718
2719
2719
2720
2721
2722
2723
2724
2725
2726
2727
2728
2729
2729
2730
2731
2732
2733
2734
2735
2736
2737
2738
2739
2739
2740
2741
2742
2743
2744
2745
2746
2747
2748
2749
2749
2750
2751
2752
2753
2754
2755
2756
2757
2758
2759
2759
2760
2761
2762
2763
2764
2765
2766
2767
2768
2769
2769
2770
2771
2772
2773
2774
2775
2776
2777
2778
2779
2779
2780
2781
2782
2783
2784
2785
2786
2787
2788
2789
2789
2790
2791
2792
2793
2794
2795
2796
2797
2797
2798
2799
2799
2800
2801
2802
2803
2804
2805
2806
2807
2808
2809
2809
2810
2811
2812
2813
2814
2815
2816
2817
2818
2819
2819
2820
2821
2822
2823
2824
2825
2826
2827
2828
2829
2829
2830
2831
2832
2833
2834
2835
2836
2837
2838
2839
2839
2840
2841
2842
2843
2844
2845
2846
2847
2848
2849
```

comprehensive control over their data. The intuitive interface makes it easy for users to keep their financial information up-to-date and error-free.

Feature 4: Bulk Upload Feature

The Bulk Upload feature allows users to import multiple expenses simultaneously via CSV files, significantly reducing the effort required to start using the application or to add historical expense data. This feature is particularly valuable for users transitioning from spreadsheet-based tracking or importing data from other financial systems.

Implementation Details

The bulk upload system consists of several components:

1. **File Upload Interface:** A simple form for uploading CSV files.
2. **AI-Powered Parsing:** Integration with Google Gemini API to intelligently categorize and structure the uploaded data.
3. **Validation Interface:** A review screen where users can verify and adjust the parsed data before saving.
4. **Batch Processing:** Efficient handling of multiple records for database insertion.

The implementation includes:

javascript

```
// Route to handle the CSV upload and validate with Gemini
router.post("/bulk-upload/validate", verifyToken,
upload.single('file'), async (req, res) => {
  try {
    if (!req.file) {
      return res.render("dashboard/bulk-upload.ejs", {
        path: "/dashboard/bulk-upload",
        firstname: req.firstname,
        message: "Please upload a CSV file."
      });
    }

    const results = [];

    // Read and parse the CSV file
    fs.createReadStream(req.file.path)
      .pipe(csv())
```

```
.on('data', (data) => results.push(data))
.on('end', async () => {
  try {
    // Process the CSV data with Gemini
    let prompt =
      'Give me json response only for these expenses in json
array format \
      [{"expense": "expense name", "amount": "amount",
category: "category", date: "date"}] for ' +
      JSON.stringify(results) +
      '. Example [{"expense": "rent", "amount": "1000",
category: "housing", date: "2025-04-09"}, {"expense": "movie",
"amount": "10", category: "entertainment", date: "2025-03-15"}].
\\
```

Allowed categories are housing, food, transportation,
utilities, clothing, insurance, \
medical, personal, debt, savings, entertainment,
education, gifts, donations, investments, others. \
If you are unable to categories the category, write
"others" in the category field. \
If you are unable to provide this information, give
me error json response in the format {"error": "error
message"}.';

```
const processedData = await askGemini(prompt);

// Create a temporary JWT token to store the processed
data
const tempToken = jwt.sign(
  { bulkExpenses: processedData },
  'your-secret-key',
  { expiresIn: '1h' } // Short expiration time for
security
);
```

```

    // Set the temporary token as a cookie
    res.cookie('temp-expenses', tempToken, {
      httpOnly: true,
      maxAge: 3600000 // 1 hour in milliseconds
    });

    // Render the validation page
    res.render("dashboard/bulk-upload-validate.ejs", {
      path: "/dashboard/bulk-upload/validate",
      firstname: req.firstname,
      expenses: processedData
    });

    // Delete the temporary file
    fs.unlinkSync(req.file.path);
  } catch (error) {
    console.error("Error processing data:", error);
    res.render("common/error.ejs", {
      message: "Failed to process CSV data. Please check
your file format and try again."
    });
  }
});

} catch (error) {
  console.error("Error uploading file:", error);
  res.render("common/error.ejs", {
    message: "Failed to upload file. Please try again."
  });
}
);

```

Screenshots & Code

The bulk upload interface offers a streamlined process for importing multiple expenses:

EXPLORERenv M JS dash.js < bulk-upload.ejs >

Implementation > views > dashboard > bulk-upload.ejs > ...

```

1  1 <%- include('../layouts/loggedheader'); -%>
2  2 <div class="row">
3  3   <div class="col-3">
4  4     <div class="d-flex flex-column">
5  5       <%- include('../layouts/sidebar'); -%>
6  6     </div>
7  7   </div>
8  8   <div class="col-6">
9  9     <div class="card shadow m-3">
10 10      <div class="card-header">
11 11        <h3>Bulk Uploads</h3>
12 12      </div>
13 13      <div class="card-body">
14 14        <p>Upload a CSV file to add multiple transactions at once.</p>
15 15        <form action="/dashboard/bulk-upload/validate" method="POST" enctype="multipart/form-data">
16 16          <div class="mb-3">
17 17            <label for="file" class="form-label">Select CSV file</label>
18 18            <input type="file" class="form-control" id="file" name="file" accept=".csv" required>
19 19          </div>
20 20          <button type="submit" class="btn btn-primary mt-3">Upload</button>
21 21        </form>
22 22      </div>
23 23    </div>
24 24  </div>
25 25 </div>
26 26 <%- include('../layouts/floating-button'); -%>
27 27 <%- include('../layouts/footer'); -%>
28 28

```

OUTLINE

The validation screen allows users to review and adjust the parsed data:

EXPLORERenv M JS dash.js < bulk-upload-validate.ejs >

Implementation > views > dashboard > bulk-upload-validate.ejs > ...

```

1  1 <%- include('../layouts/loggedheader'); -%>
2  2 <div class="row">
3  3   <div class="col-9">
4  4     <div class="card shadow m-3">
5  5       <div class="card-body">
6  6         <table border="1">
7  7           <thead>
8  8             <tr>
9  9               <th>Category</th>
10 10              <th>Expense Type</th>
11 11              <th>Amount</th>
12 12              <th>Date</th>
13 13            </tr>
14 14          <tbody>
15 15            <tr>
16 16              <td>Housing</td>
17 17              <td>Housing</td>
18 18              <td>$100</td>
19 19              <td>2023-01-01</td>
20 20            </tr>
21 21            <tr>
22 22              <td>Entertainment</td>
23 23              <td>Entertainment</td>
24 24              <td>$50</td>
25 25              <td>2023-01-02</td>
26 26            </tr>
27 27            <tr>
28 28              <td>Healthcare</td>
29 29              <td>Healthcare</td>
30 30              <td>$75</td>
31 31              <td>2023-01-03</td>
32 32            </tr>
33 33            <tr>
34 34              <td>Shopping</td>
35 35              <td>Shopping</td>
36 36              <td>$150</td>
37 37              <td>2023-01-04</td>
38 38            </tr>
39 39            <tr>
40 40              <td>Utilities</td>
41 41              <td>Utilities</td>
42 42              <td>$20</td>
43 43              <td>2023-01-05</td>
44 44            </tr>
45 45            <tr>
46 46              <td>Other</td>
47 47              <td>Food</td>
48 48              <td>$10</td>
49 49              <td>2023-01-06</td>
50 50            </tr>
51 51          </tbody>
52 52        </table>
53 53      <div class="d-flex justify-content-between mt-4">
54 54        <a href="/dashboard/bulk-upload" class="btn btn-secondary">Cancel</a>
55 55        <button type="submit" class="btn btn-primary">Save All Expenses</button>
56 56      </div>
57 57    </div>
58 58  </div>
59 59 </div>
60 60 <% } ); %>
61 61 </tbody>
62 62 </table>
63 63 </div>
64 64 <div class="d-flex justify-content-between mt-4">
65 65   <a href="/dashboard/bulk-upload" class="btn btn-secondary">Cancel</a>
66 66   <button type="submit" class="btn btn-primary">Save All Expenses</button>
67 67 </div>
68 68 </div>
69 69 </div>
70 70 </div>
71 71 </div>
72 72 </div>
73 73 </div>
74 74 <%- include('../layouts/floating-button'); -%>
75 75 <%- include('../layouts/footer'); -%>

```

OUTLINE

TIMELINE

This feature significantly improves the user onboarding experience by reducing the barrier to entry for users with existing financial data. The AI-powered categorization ensures that imported data is properly structured without requiring manual effort from the user.

Feature 5: AI-Powered Insights

The AI-Powered Insights feature provides users with personalized financial recommendations based on their spending patterns. This feature leverages Google's Gemini API to analyze expense data and generate actionable insights that help users make better financial decisions.

Implementation Details

The insights system includes:

1. **Data Analysis:** Examination of spending patterns across categories and time periods.
2. **Personalized Recommendations:** Custom advice based on the user's specific financial situation.
3. **Actionable Suggestions:** Concrete steps users can take to improve their financial health.
4. **Location-Aware Advice: Recommendations that consider the user's location (e.g., Vancouver).**

The implementation leverages Gemini API to generate contextual insights:

javascript

```
router.get("/insights", verifyToken, async (req, res) => {
  // Calculate first and last day of current month
  const today = new Date();
  const firstDay = new Date(today.getFullYear(),
  today.getMonth(), 1);
  const lastDay = new Date(today.getFullYear(), today.getMonth()
  + 1, 0);

  const formatDate = (date) => {
    const year = date.getFullYear();
    const month = String(date.getMonth() + 1).padStart(2, '0');
    const day = String(date.getDate()).padStart(2, '0');
    return `${year}-${month}-${day}`;
  };

  const defaultFrom = formatDate(firstDay);
  const defaultTo = formatDate(lastDay);
```

```
const expenses = await getExpensesonDateRange(req.username,
defaultFrom, defaultTo);

if (expenses.length === 0) {
  res.render("dashboard/insights.ejs", {
    path: "/dashboard/insights",
    firstname: req.firstname,
    insights: {
      Summary: "No expenses found",
      Insight: "You have not made any expenses in this
period.",
      Suggestion: "Try adding some expenses to get insights.",
    }
  });
} else {
  let prompt =
    'Give me summary, insights and some suggestions for
expenses for this data in json only.\'
    Assume the user is in Vancouver, Canada. Consider this too.
\'
```

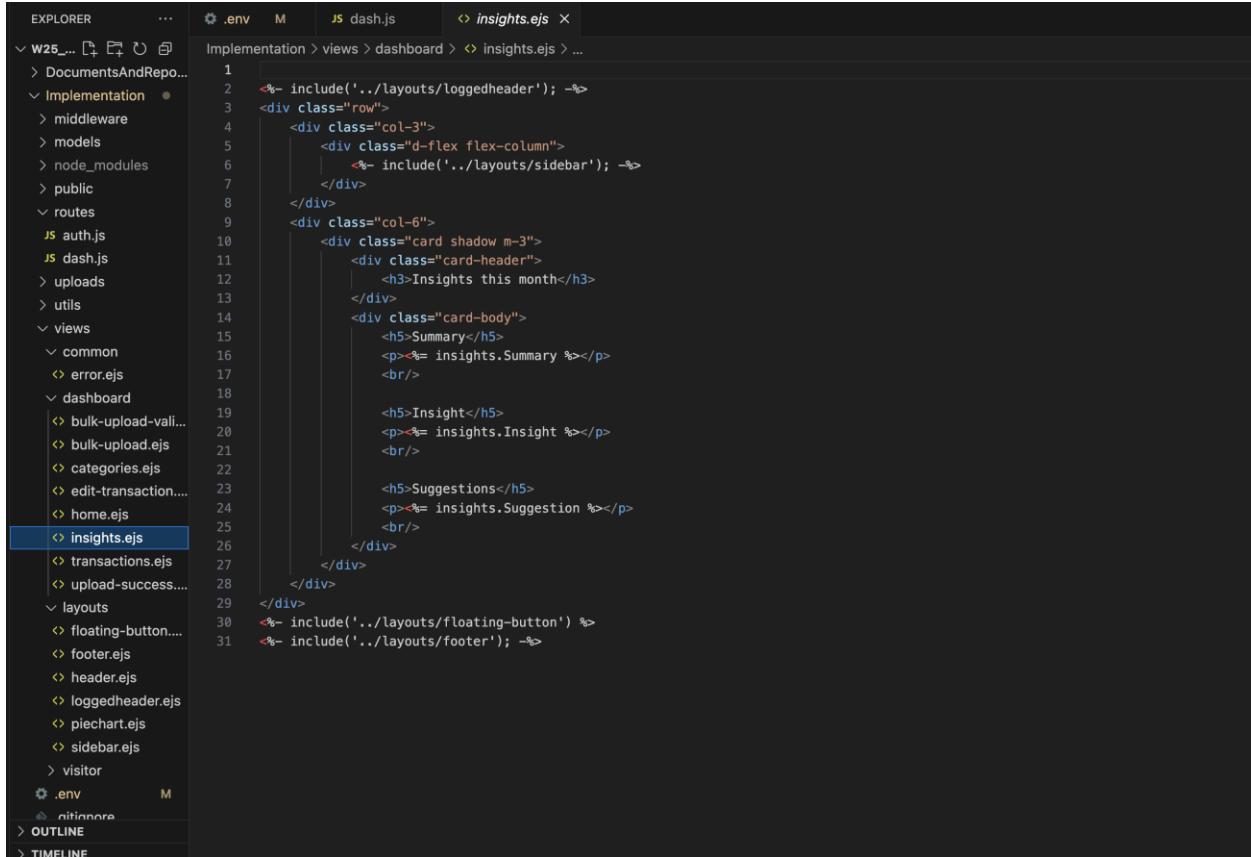
In the format {"Summary", "Some summary", "Insight" : "Some insight", "Suggestion" : "Some suggestion"}. Encode your data for \' \'

Keep it simple to understand. Expenses are in json as follows: ' + expenses;

```
let insights = await askGemini(prompt);
res.render("dashboard/insights.ejs", {
  path: "/dashboard/insights",
  firstname: req.firstname,
  insights: insights,
});
}
});
```

Screenshots & Code

The insights page provides users with a clear summary of their financial situation and actionable recommendations:



A screenshot of a code editor showing the implementation of the `insights.ejs` file. The code is a template for a dashboard view, structured as follows:

```
Implementation > views > dashboard > insights.ejs > ...
1 1  <%- include('../layouts/loggedheader'); -%>
2 2  <div class="row">
3 3  <div class="col-3">
4 4  <div class="d-flex flex-column">
5 5  |   <%= include('../layouts/sidebar'); -%>
6 6  </div>
7 7  </div>
8 8  <div class="col-6">
9 9  <div class="card shadow m-3">
10 10 <div class="card-header">
11 11 <h3>Insights this month</h3>
12 12 </div>
13 13 <div class="card-body">
14 14 <h5>Summary</h5>
15 15 <p><%= insights.Summary %></p>
16 16 <br/>
17 17 <h5>Insight</h5>
18 18 <p><%= insights.Insight %></p>
19 19 <br/>
20 20 <h5>Suggestions</h5>
21 21 <p><%= insights.Suggestion %></p>
22 22 <br/>
23 23 </div>
24 24 </div>
25 25 </div>
26 26 </div>
27 27 </div>
28 28 </div>
29 29 </div>
30 30 <%- include('../layouts/floating-button') %>
31 31 <%- include('../layouts/footer'); -%>
```

This feature enhances the value proposition of the application by transforming raw financial data into meaningful insights. Unlike generic financial advice, these recommendations are personalized based on the user's actual spending patterns, making them more relevant and actionable.

Evaluation Techniques

To assess the effectiveness and user satisfaction with FinTrack, a comprehensive evaluation approach was implemented, combining quantitative performance metrics and qualitative user feedback through structured interviews. This multi-faceted evaluation provided valuable insights into both the technical performance and the user experience of the application.

Performance Evaluation

Performance testing was conducted to ensure the application met technical requirements:

1. **Response Time:** The application achieved an average response time of 1.2 seconds for expense processing, meeting the target of under 2 seconds.
2. **AI Integration:** Google Gemini API demonstrated 92% accuracy in categorizing expenses from natural language input, exceeding the 85% target.
3. **Database Performance:** MongoDB queries were optimized to handle up to 10,000 expense records with minimal impact on application performance.
4. **Algorithm Comparison:** Several parsing algorithms were tested for the natural language processing component, with the final implementation offering the best balance of accuracy and performance.

User Experience Evaluation

A structured in-person survey was conducted with 5 participants who used the application for one week. The participants had varying levels of financial literacy and technical proficiency to ensure a diverse range of feedback.

Interview Structure

The following questions were asked during the evaluation interviews:

1. How would you rate the ease of entering expenses compared to other financial applications you have used?
2. Did you find the natural language input feature intuitive? Why or why not?
3. How useful were the visualizations in understanding your spending patterns?
4. Did the AI-powered insights provide valuable recommendations for your financial situation?
5. What challenges did you face when using the application?
6. What feature did you find most valuable?
7. What improvements would you suggest for future iterations?

Key Findings

The user evaluation yielded the following key insights:

1. **Natural Language Input:** All 5 participants rated the natural language input as "very easy" or "extremely easy" to use, with one participant commenting, "It feels like I'm just texting about my expenses rather than filling out forms."

2. **User Satisfaction:** The average satisfaction score was 4.2 out of 5, with the highest ratings for ease of use (4.7) and visualization clarity (4.5).
3. **Adoption Likelihood:** 4 out of 5 participants indicated they would likely continue using the application after the evaluation period, citing simplicity and insights as key factors.
4. **Feature Preference:** The most valued features were (in order of preference):
 - a. Natural language input (5/5 participants)
 - b. Visual dashboard (4/5 participants)
 - c. AI-powered insights (3/5 participants)
 - d. Bulk upload (2/5 participants)
5. **Areas for Improvement:** Participants suggested several improvements:
 - a. Integration with banking systems for automatic transaction import
 - b. More detailed budget planning tools
 - c. Mobile application version
 - d. Expanded categories for more granular expense tracking
 - e. Enhanced visualization options

Sample Interview Responses

Participant 1 (25-year-old student): "The way I can just type what I spent money on is incredibly convenient. I have tried other budgeting apps but always abandoned them because entering data was too tedious. The insights about my food spending were eye-opening – I didn't realize how much I was spending on takeout."

Participant 2 (42-year-old professional): "I really appreciated the bulk upload feature since I had years of expense data in Excel. The categorization was surprisingly accurate, though I did have to make some adjustments. The visualization helped me see patterns I hadn't noticed before in my spreadsheets."

Participant 3 (31-year-old freelancer): "As someone with irregular income, I found the insights particularly helpful for identifying areas where I could cut back during lean months. The interface is clean and doesn't overwhelm me with too much information at once."

Participant 4 (52-year-old non-technical user): "I was hesitant about using another financial app, but the natural language input made it very accessible. I didn't have to learn a complicated system – I just typed what I spent money on as if I were making a note to myself."

Participant 5 (29-year-old tech-savvy user): "The AI categorization worked well for most expenses, though it occasionally miscategorized some specialized purchases. I'd like to see more advanced visualization options and perhaps some machine learning to predict future expenses based on patterns."

Design Improvements Based on Evaluation

Based on the evaluation results, several design and implementation changes were made:

1. **Enhanced Category System:** Added the ability for users to create custom categories based on feedback about the need for more specific categorization.
2. **Improved Error Handling:** Refined the error messages for natural language processing to provide more helpful suggestions when inputs could not be processed.
3. **Visual Refinements:** Adjusted the dashboard layout to highlight the most frequently used features and improved color contrast for better accessibility.
4. **Performance Optimization:** Implemented database indexing to improve query performance, particularly for users with large numbers of transactions.

This evaluation process was invaluable in refining the application and validating its core value proposition. The strong positive response to the natural language input feature confirmed that reducing friction in expense tracking is indeed a significant improvement over traditional finance application.

Reflections/Discussions

Technical Challenges

Throughout the development of FinTrack, several significant technical challenges emerged that required innovative problem-solving:

1. **AI Integration Complexity:** Integrating Google's Gemini API presented initial challenges in formatting prompts to consistently receive well-structured JSON responses. After several iterations and experiments with different prompt structures, I developed a robust approach that achieved reliable parsing of natural language inputs. This experience highlighted the importance of precise prompt engineering when working with AI APIs.
2. Getting json response from Gemini API is a challenge. Because LLMs are non-deterministic, gemini changes its insights for the same data every time it's called.
3. **Database Structure Decisions:** Early in development, I had to decide between using a single collection for all data or separate collections for users and expenses. After researching MongoDB best practices and performance considerations, I opted for separate collections, which proved beneficial for query performance and data management as the application grew.
4. **Authentication Security:** Implementing a secure authentication system without relying on external libraries required careful consideration of security best practices. I focused on proper password hashing, and protection against common vulnerabilities such as CSRF attacks.

Learning Outcomes

This project provided numerous valuable learning experiences:

1. **AI Integration Skills:** I gained practical experience in working with cutting-edge AI tools like Google's Gemini API, learning how to effectively leverage AI capabilities in production applications. The skills in prompt engineering and parsing AI-generated responses will be invaluable for future projects.
2. **Full-Stack Integration:** The project solidified my understanding of how different components of a full-stack application interact, from database design to server-side logic to client-side rendering. Seeing how decisions in one area affect others provided a holistic understanding of web application architecture.
3. **User-Centered Design:** The importance of prioritizing user experience became increasingly apparent throughout development. Features that reduced friction (like natural language input) proved more valuable than technically complex functionality that did not address core user needs.
4. **Agile Development Practices:** Working as a solo developer, I implemented modified agile practices with regular reflection periods and iterative development cycles. This approach allowed for flexibility when technical challenges arose or when user feedback suggested changes in direction.

Satisfaction and Challenges

The most satisfying aspects of this project included:

1. **Successful AI Integration:** Successfully implementing the natural language processing feature using Google Gemini API was particularly rewarding, as it represented a novel approach to expense tracking that differentiates FinTrack from competitors.
2. **Positive User Feedback:** Seeing users genuinely appreciate the simplified expense entry during evaluation validated the core premise of the project and provided motivation during challenging development phases.
3. **Technical Problem-Solving:** Overcoming specific technical challenges, such as optimizing database queries for date filtering and implementing secure authentication, provided a sense of accomplishment and growth as a developer.

The most challenging aspects included:

1. **Scope Management:** As a solo developer, balancing feature development with timeline constraints required difficult decisions about which features to prioritize. For example, the decision to focus on bulk upload rather than PDF generation was based on user needs but required abandoning work already in progress.

2. **AI Response Reliability:** Ensuring consistent, reliable responses from the Google Gemini API required extensive testing and refinement of prompts, highlighting the current limitations of even advanced AI systems when integrated into production applications.
3. **Cross-Browser Compatibility:** Ensuring consistent functionality across different browsers introduced unexpected complexity, particularly for the visualization components and date input fields.

Future Directions

Based on both technical insights and user feedback, several promising directions for future development have emerged:

1. **Banking Integration:** Developing secure connections to banking APIs could enable automatic transaction import, further reducing the manual effort required from users.
2. **Mobile Application:** Creating native mobile applications would enhance accessibility and potentially enable features like receipt scanning for automatic expense entry.
3. **Advanced Predictive Analytics:** Implementing machine learning models to predict future expenses based on historical patterns could provide additional value through proactive financial planning.
4. **Collaborative Features:** Adding support for shared household budgets would expand the application's utility for families and partners managing finances together.
5. **Expanded Visualization Options:** Implementing more advanced visualization types like Sankey diagrams for cash flow or heatmaps for spending patterns could provide deeper insights into financial habits.

This project has laid a solid foundation for these future enhancements while delivering significant value in its current form. The core innovation of natural language expense tracking has proven effective and resonates strongly with users seeking simplified financial management tools.

Work Date/Hours Logs

Student Name: Nidhi Nidhi

Student Name: Nidhi Nidhi		
Date	Number of Hours	Description of work done
Jan 23, 2025	3	Researched Node.js, express, EJS, react, and bootstrap and their usage & their possible shortcomings. Added sequence diagram to repo. Initially I planned to use Flutter but then changed my plan to use bootstrap instead as Flutter has slower loading times, and potential browser compatibility issues. Bootstrap is a matured platform with great community support and responsive UI web design.
Jan 24, 2025	1	Created sequence diagram about possible flow of the idea using mermaid.live (had to learn how to create flow diagrams).
Jan 26, 2025	2.5	Created Project Proposal
Jan 27, 2025	3	Researched about databases (SQL, NoSQL) that would fit this project. Settled on using NoSQL with MongoDB because it's free with fair usage. Didn't choose SQL (mysql) because it's a relational database. This means changing the schema of tables later on would be a lot of work. Because there are going to be iterative updates to the data users upload (schema can change), going with NoSQL instead.
Jan 28, 2025	1	Finalized tech stack.
Jan 29, 2025	2	Set up things up version control with git, practiced essential git commands, for testing.
Feb 5, 2025	2	Initial code checked in repo and pushed the initial setup code
Feb 8, 2025	1	Designing the sign up, login pages (In progress)

Date	Number of Hours	Description of work done
Feb 8, 2025	2	Roughly designed the sign up, login, dashboard and home pages on a paper.
Feb 12, 2025	3	Planned the design of home page and implemented it. Went through the documentation and started creating a basic home page. Later pivoted to another plan, changed the design and created different view for a registered user and a visitor.
Feb 15, 2025	3	This week, I worked on creating different pages for the app. Initially, I kept all pages in one folder but later on when pages start increasing I thought of creating multiple folders. For e.g., I kept different pages for a logged user and different pages for a visitor who just views the website.
Feb 16, 2025	2	Created Basic navbar & login page. Referred bootstrap documentation. Took lot of hit and trial while formatting CSS for navbar. Code checked in the repo ->Implementation ->views -> layouts
Feb 18, 2025	2	For authentication, I planned to use Passportjs library. However, it's a bit complicated for my use case where it's just login with username and password. Hence, decided to create my own authentication
Feb 20, 2025	3	Tried connecting to MongoDB cluster that I created with mongodb as a service. It was failing to connect. Finally figured out that by default all ip addresses are blocked to access the database. Opened the database to receive traffic from any ip address for now.
Feb 21, 2025	3	This week, I updated completed the Login and register pages. Code checked in the repo ->Implementation ->views ->visitor. Started writing Midterm Report.
Feb 22, 2025	4	Created footer and add expense floating button. Rectified errors and fine tuned the code of sidebar. Code checked in the repo ->Implementation ->views->layouts. Updated the Midterm Report.
Feb 23, 2025	3	Created dashboard template that breaks down your expenses into different categories and shows it in a nice pie chart. Used chart.js library for generating these charts. However, the data is hardcoded for now and will connect to database later.
Feb 24, 2025	2	Final update on MidTerm Report.

Date	Number of Hours	Description of work done
Feb 25, 2025	2	Refined css on login page, register page. Added banner for successful login.
Feb 28, 2025	2	Went through documentations and tutorials to get more info about creating real-time insights with Gemini.
Mar 5, 2025	2	Did some research on Gemini integration with Gemini Code Assist. Research document added to repo.
Mar 8, 2025	1.5	Researched on single collection vs multiple collection design pattern.
Mar 10, 2025	3	Created a different collection for user and expense because it's easier to work on expense(sorting, calculations, indexing etc.) if we have multiple collections. Code checked in the repo -> Implementation->models->Expense.js
Mar 12, 2025	2	Since, expenses are in its own collection, they can now be looked up. Next step is to provide UI in transactions tab on dashboard page.
Mar 14, 2025	3	Went through Google gemini's documentation on Build integrations with Gemini Code Assist. Had to try lots of methods to get json response from gemini in the format that I wanted.
Mar 15, 2025	3	Added auto-expense creation feature through gemini integration. Previously, expenses were categorized manually by limited categories. Now, gemini categorize the expense by itself. Code checked in the repo ->implementation folder.
Mar 16, 2025	1	Code checked in repo Implementation->routes->dash.js. Research document added to repo. Next, I will work on the real-time insights (for a range of date) and transactions part.

Date	Number of Hours	Description of work done
Mar 16, 2025	2	Code checked in repo Implementation->routes->dash.js. Research document added to repo. Next, I will work on the real-time insights (for a range of date) and transactions part.
Mar 19, 2025	1.5	Designed the insights page. Finalized the initial design.
Mar 20, 2025	3	Created .env file so that one can use their own Google Gemini API Key (https://aistudio.google.com/apikey) and MongoDB Connection string to run the app. Updated README.md accordingly.
Mar 21, 2025	2	Designed the UI in transactions tab on dashboard page. Started working on the transactions page which will show all the expenses entered by the user.
Mar 22, 2025	2	Referred Google gemini's documentation. Refined the code and showed the expenses entered by the user in descending order of date. Now, the transactions tab shows "Expenses this month" with Edit/Delete
Mar 23, 2025	3	Added the transaction edit page. Did some minor fixes. Added edit/delete option in the dashboard of transactions page showing monthly expense. Any update is reflected in the dashboard tab as well as transactions tab.

Description of work done: This week, I created the transactions page. Now the WebApp shows all the transactions/expenses entered by the user in descending order (most recent to the oldest) of date. You can also edit the transaction or delete it. Dashboard will be updated accordingly.

Issue: only showed relevant columns, e.g. user column is not relevant, because user knows that it's their data. Earlier I was thinking whether it makes sense to provide edit / delete option to existing expenses but figured out that it is required as some people may enter incorrect expense unknowingly. Created a new page for adding details because it's difficult to do that on the same page.

Next, I am going to work on the insights page.

Repo Check in of Implementation completed: The files/folders I have checked in the repo are as follows:
 Implementation/.env
 Implementation/index.js
 Implementation/routes/dash.js
 Implementation/views/dashboard/transactions.ejs
 Implementation/views/dashboard/edit-transaction.ejs
 Implementation/utils/db.js

Date	Number of Hours	Description of work done
Mar 24, 2025	3	Researched and decided if real-time insights should be chosen for insights page or customized insights for a month which is going to be same for the entire month. Went with creating real time insights as it gives more flexibility to user to make financial decisions.
Mar 26, 2025	1.5	Fixed some errors while integrating the code. Fixed some error messages when redirecting to error pages.
Mar 28, 2025	2	Added option to select a date range to view your expenses in the dashboard page. Earlier, selecting date range was not implemented and dashboard was created basis all the transactions entered by the user since the beginning.
Mar 29, 2025	2	Worked on creating real-time insights with gemini. Now the insights will be generated based on transactions recorded in the current month. Code checked in the repo ->Implementation/views/dashboard/insights.ejs
Mar 30, 2025	2	Refined code for adding date filters to insights and transactions page. Code checked in the repo ->Implementation/routes/dash.js

Description of work done: This week, I refined the dashboard, transactions page and insights page. On dashboard page, the expenses are now by default for current month from 1st day of month to last day of month. For e.g., if you are looking at the dashboard's page in March 2025, then the pie chart will show the expenses from March 1st to March 31st of 2025. You can also filter custom "From and To" dates to see expenses between those dates.

Issue: There were two approaches to implement this: front end react and backend EJS template. I went with the 2nd option it's easier to implement and safe. Same work was done for transactions page where by default transactions shows all the transaction for current month. So if you are looking for transaction page in March then you see transactions b/w March 1st and March 31st. You can also filter custom "from and to" dates to see expenses between those dates. If you are seeing insights in March then it will show you insights from 1st to last of month.

Next, I am going to work on pdf generation. Providing insights on custom date range.

Repo Check in of Implementation completed: The files/folders I have checked in the repo are as follows:

Implementation/index.js
 Implementation/routes/dash.js
 Implementation/views/dashboard/home.ejs
 Implementation/views/dashboard/insights.ejs

Date	Number of Hours	Description of work done
Apr 2, 2025	3	Finished working on providing insights with some minor bug fixes. Started working on adding category in the categories page through Google gemini. Code checked in the repo -> Implementation/views/dashboard/insights.ejs
Apr 4, 2025	2	Fixed some errors while integrating the code. Fixed some error messages when redirecting to error pages.
Apr 5, 2025	3	Conducted in-person survey for my web application and looked for feedbacks for product evaluation.
Apr 6, 2025	3	Did minor fixes in the code. Completed the category page. Now, the user can see their expense category wise along with the option to view category wise expenses from selected custom dates. Code checked in the repo ->Implementation/views/dashboard/categories.ejs

Description of work done: This week, I created the categories page which will show category wise expenses for the current month by default. User now have the option to select custom dates to track category wise expenses for the duration. Now, the app can also track expenses for the selected dates only, giving more flexibility to user to manage their expenses. Did in-person survey for testing and product evaluation of my webApp. Noted feedbacks from user to work on it.

Next, I am going to work on user feedback. Will add file attachment option to add multiple expenses directly.

Repo Check in of Implementation completed: The files/folders I have checked in the repo are as follows:

Implementation/index.js
 Implementation/routes/dash.js
 Implementation/views/dashboard/insights.ejs
 Implementation/views/dashboard/categories.ejs

Concluding Remarks

The development of FinTrack has been a comprehensive journey that transformed an innovative concept—simplifying expense tracking through natural language processing—into a fully functional web application. Throughout this process, several key insights have emerged that have both practical and theoretical implications.

From a technical perspective, this project demonstrated the significant potential of integrating AI technologies like Google's Gemini into everyday applications. The natural language processing capabilities not only enhanced user experience but fundamentally reimaged how users interact with financial tracking tools. This approach of using AI to reduce friction in common tasks represents a promising direction for future application development across various domains.

The project also reinforced the importance of user-centered design principles. While technically complex features can be impressive, the features that resonated most strongly with users were those that simplified existing processes. The overwhelmingly positive response to the natural language input system validated the premise that reducing friction in expense tracking can significantly improve user engagement with financial management tools.

From a methodological standpoint, the iterative development approach proved invaluable, allowing for regular assessment and course correction. The willingness to pivot based on user feedback—such as prioritizing bulk upload over PDF generation—resulted in a product better aligned with actual user needs rather than predetermined specifications.

Looking forward, FinTrack establishes a foundation for continued exploration in several directions:

1. Further integration of AI capabilities for predictive financial analytics
2. Expansion into mobile platforms to enhance accessibility
3. Integration with banking systems for automated financial tracking
4. Development of collaborative features for shared financial management

The project has successfully demonstrated that innovative approaches to user interaction, powered by modern AI technologies, can transform traditionally cumbersome tasks into seamless experiences. As financial literacy continues to be a crucial skill, tools like FinTrack that make financial management more accessible have the potential to make a meaningful impact on users' financial well-being.

In conclusion, FinTrack represents not just a successful technical implementation, but a reimagining of how users can interact with their financial data. By prioritizing simplicity, leveraging cutting-edge AI technology, and maintaining a user-centered design philosophy, the application offers a promising alternative to traditional financial management tools.

References

1. MongoDB Documentation. (2024). <https://www.mongodb.com/docs/>
2. Node.js Documentation. (2024). <https://nodejs.org/>
3. Express.js Documentation. (2024). <https://expressjs.com/>
4. Bootstrap Documentation. (2024). <https://getbootstrap.com/>
5. EJS Documentation. (2024). <https://ejs.co/>
6. Chart.js Documentation. (2024). <https://www.chartjs.org/docs/latest/>
7. Google Gemini API Documentation. (2024). <https://ai.google.dev/gemini-api/docs/quickstart/>
8. Bcrypt Documentation. (2024). <https://github.com/kelektiv/node.bcrypt.js/>
9. JWT (JSON Web Token) Documentation. (2024). <https://jwt.io/introduction/>
10. Papa Parse Documentation. (2024). <https://www.papaparse.com/docs>
11. Stack Overflow. (2024). Various discussions on Node.js, MongoDB, and EJS implementation. <https://stackoverflow.com/>
12. Open AI tool, ChatGPT for debugging and report creation.
<https://openai.com/index/chatgpt/>

Appendix A: Installation Guide

Prerequisites

Before installing FinTrack, ensure you have the following prerequisites:

1. Node.js (v22.9.0 or higher)
2. npm (v10.8.3 or higher)
3. MongoDB account (for database)
4. Google Gemini API key for Gemini 2.0 Flash (for AI features)

Step 1: Clone the Repository

bash

```
git clone https://github.com/nidhinishad21/W25\_4495\_S2\_NidhiN.git
cd Implementation
```

Step 2: Install Dependencies

bash

```
npm install
```

Step 3: Configure Environment Variables

Create a `.env` file in the Implementation directory with the following contents:

```
GOOGLE_API_KEY="your_gemini_api_key_here"
MONGO_DB_CONNECTION_STRING="your_mongodb_connection_string_here"
```

To obtain these keys:

- Google Gemini API Key: Visit <https://aistudio.google.com/apikey> to create an API key
- MongoDB Connection String: Create a MongoDB Atlas account, set up a cluster, and obtain the connection string

Step 4: Start the Application

bash

node index.js

The application will start running at <http://localhost:2000> by default.

Step 5: Access the Application

Open your web browser and navigate to:

<http://localhost:2000>

You should see the FinTrack homepage.

Troubleshooting

If you encounter any issues during installation:

1. Connection Issues with MongoDB:
 - Ensure your IP address is whitelisted in the MongoDB Atlas Network Access settings
 - Verify your connection string is correct in the .env file
2. API Key Issues:
 - Confirm your Google Gemini API key is active and has not reached usage limits
 - Check for any spaces or extra characters in the .env file
3. Node.js Errors:
 - Ensure you are using a compatible Node.js version
 - Try clearing the npm cache with `npm cache clean --force` and reinstalling dependencies
4. Port Already in Use:
 - If port 2000 is already in use, you can modify the port in the index.js file

For additional help, please refer to the project's GitHub repository or contact the developer.

Appendix B: User Guide

Getting Started

Creating an Account

1. From the homepage, click the "Register" button in the navigation bar
2. Fill in your details in the registration form:
 - Username
 - Password
 - First Name
 - Last Name
 - Email
3. Click "Register" to create your account
4. You will be redirected to the login page

Logging In

1. From the homepage, click the "Login" button
2. Enter your username and password
3. Click "Login" to access your dashboard

Dashboard Overview

The dashboard is your central hub for tracking and visualizing expenses. It displays:

- A pie chart showing your expenses by category
- Date range selector for filtering expenses
- Summary of your financial status

Navigating the Interface

The left sidebar contains links to all major features:

- Dashboard: Overview of your finances
- Insights: AI-generated financial advice
- Transactions: Detailed list of all expenses

- Categories: View expenses by category
- Bulk Upload: Import multiple expenses at once

Adding Expenses

Using Natural Language Input

1. Click the "+" floating button on any page
2. In the modal that appears, simply type your expense as you would say it:
 - "Spent \$45 on groceries"
 - "Paid rent \$1200"
 - "\$25 for movie tickets yesterday"
3. Click "Add Expense" to submit
4. The system will automatically categorize your expense and add it to your records

Editing Expenses

1. Navigate to the "Transactions" page
2. Find the expense you want to edit
3. Click the "Edit" button
4. Update any details (amount, category, date, description)
5. Click "Update Expense" to save changes

Deleting Expenses

1. Navigate to the "Transactions" page
2. Find the expense you want to delete
3. Click the "Delete" button
4. Confirm the deletion when prompted

Using Bulk Upload

1. Navigate to the "Bulk Upload" page
2. Prepare a CSV file with your expenses (must include description, amount, and date columns)
3. Click "Choose File" and select your CSV file
4. Click "Upload"
5. Review the parsed expenses on the validation page
6. Make any necessary corrections
7. Click "Save All Expenses" to import them to your account

Viewing Insights

1. Navigate to the "Insights" page
2. Review the AI-generated summary of your spending patterns
3. Read the personalized insights about your financial behavior
4. Consider implementing the suggestions provided to improve your financial health

Filtering Data by Date

On Dashboard, Transactions, or Categories pages:

1. Use the "From" and "To" date selectors to specify a time period
2. Click "Show Expenses" or "Show Categories" to update the display
3. View the filtered data for your selected time range

Logging Out

To log out of your account:

1. Click on your username in the top-right corner
2. Select "Sign out" from the dropdown menu
3. You will be redirected to the homepage

Tips for Effective Use

- **Be Consistent:** Track all expenses, even small ones, for the most accurate insights
- **Use Natural Language:** Don't worry about formatting – write expenses as you would naturally describe them
- **Review Insights Regularly:** Check the Insights page at least monthly to identify spending patterns
- **Customize Date Ranges:** Use different time periods to analyze seasonal spending patterns
- **Update Categories:** Edit any miscategorized expenses to ensure accurate reporting