***MINI PROJECT***

***TOPIC: EXPENSE TRACKER***

**Group members:**

**1) Vedantika Ranaware – B24CE1013**

**2) Rugved Deshwant – B24CE1014**

**3) Swarada Deshpande - B24CE1015**

**4) Janavhi Dhanapune - B24CE1016**

**RESEARCH**

**User Needs**

**During the research phase, we focused on understanding the following key needs of users:**

* **Tracking daily expenses: Most users wanted a straightforward way to log their daily expenses without complexity.**
* **Categorization: Users wanted the ability to categorize their expenses into predefined categories such as food, entertainment, transportation, etc., to better track where their money goes.**
* **Monthly Overview: Users desired the ability to track their monthly spending patterns and evaluate which categories were the most expensive.**

**Technology Stack**

* **HTML: We used HTML to structure the layout of the website, creating forms for input, tables for displaying the expense records, and dropdown menus for filtering data.**
* **CSS: CSS was used to style the webpage, making it visually appealing, responsive, and easy to navigate. We made sure the design would work well on both desktop and mobile devices.**
* **JavaScript: JavaScript powered the dynamic features of the website, such as adding expenses, updating the total amount, and filtering by category. It also handled the logic for the monthly summary.**

**Competitive Research**

**We reviewed a few online expense tracking tools to understand what features are commonly included. These tools provided inspiration for:**

* **Expense Categories: Most tools allow users to categorize expenses, and this feature was included in the project.**
* **Filtering and Sorting: Being able to filter by date, category, and amount was a common feature in expense tracker tools.**
* **Total Calculation: Automatic calculation of the total amount was also a common feature.**

**ANALYSE**

**User Experience (UX) Analysis**

**We ensured that the user interface (UI) was intuitive and straightforward. We focused on:**

* **Ease of Navigation: The user does not need to navigate through multiple pages to add or view expenses. Everything happens on the same page.**
* **Form Validation: Ensured that only valid data (e.g., a positive amount, a valid date) could be submitted in the form.**
* **Responsiveness: The layout was designed to look good on all screen sizes. Users should be able to interact with the website comfortably on mobile devices and desktops.**

**Data Handling**

* **In-memory Storage: Expenses are stored temporarily in the browser's memory using JavaScript arrays. While this is sufficient for the mini-project's scope, it would be more practical in a production environment to use a database to store user data.**
* **Handling Large Data: We analyzed potential issues when the number of expenses grows. For this project, performance issues are unlikely to arise because the data set is small and stored in memory.**

**Security Concerns**

* **Data Persistence: As the project does not involve any sensitive personal data (e.g., payment details), security concerns are minimal. However, if user data were stored permanently, we would need to ensure secure handling, possibly using server-side databases and HTTPS for data transmission.**

**IDEATE**

**Brainstorming Features**

During the ideation phase, we brainstormed several features that would make the expense tracker more useful for users:

* **Expense Tracking**: Basic tracking of individual expenses, including name, amount, category, and date.
* **Expense Editing**: Allow users to modify or delete previously added expenses.
* **Category Filtering**: Allow users to filter the list of expenses based on categories such as Food, Transport, etc.
* **Monthly Summary**: A section where users can view their total expenses by category for any given month.
* **Export Functionality**: We considered adding an option to export expenses to a CSV file or print them for future reference, though this feature is not implemented in the current version.

**BUILD**

**Frontend Development**

* **HTML**: We started by building the basic structure of the webpage. The main components include:
  + A form to input expense details (name, amount, category, and date).
  + A table to display the added expenses.
  + A filter to sort expenses by category.
  + A section for monthly summary results.
* **CSS**: The styling focuses on:
  + A clean, minimalistic design with adequate spacing and alignment.
  + Responsive elements to ensure the website adapts well on various screen sizes.
  + Hover effects for buttons to improve user interaction.
* **JavaScript**: JavaScript is used to:
  + **Handle form submissions**: When the user submits an expense, the data is processed, and the expense is added to an array.
  + **Update the total amount**: The total amount spent is updated dynamically every time an expense is added or removed.
  + **Filter expenses**: The user can filter the displayed expenses by category.
  + **Handle monthly summary**: The monthly summary feature updates when the user selects a month and shows totals for each category.

#### Code Example: Adding an Expense

expenseForm.addEventListener("submit", (e) => {

e.preventDefault(); // Prevent form from submitting and refreshing the page

const name = document.getElementById("expense-name").value;

const amount = parseFloat(document.getElementById("expense-amount").value);

const category = document.getElementById("expense-category").value;

const date = document.getElementById("expense-date").value;

const expense = {

id: Date.now(),

name,

amount,

category,

date: new Date(date)

};

expenses.push(expense); // Add the new expense to the array

displayExpenses(expenses); // Update the expense list displayed on the webpage

updateTotalAmount(); // Update the total displayed on the page

});

INDEX PAGE -

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

<meta name="viewport" content="width=device-width, initial-scale=1.0">

<title>BudgetBuddy - Expense Tracker</title>

<style>

/\* Basic styles \*/

body {

margin: 0;

padding: 0;

font-family: Arial, sans-serif;

background-color: #e8f5ff;

display: flex;

justify-content: center;

align-items: center;

height: 100vh;

overflow: auto;

}

.main-container {

width: 90%;

max-width: 800px;

background: white;

border-radius: 10px;

box-shadow: 0 4px 10px rgba(0, 0, 0, 0.2);

overflow: hidden;

display: flex;

flex-direction: column;

justify-content: center;

padding: 20px;

}

.header {

background-color: #4caf50;

color: white;

text-align: center;

padding: 20px;

border-radius: 10px 10px 0 0;

}

.header h1 {

margin: 0;

font-size: 2.5rem;

}

.header p {

margin: 5px 0 0;

font-size: 1rem;

}

.greeting {

text-align: center;

margin: 10px 0;

font-size: 1.5rem;

}

.content {

padding: 20px;

text-align: center;

}

h2 {

color: #4caf50;

margin-bottom: 15px;

}

.form-section,

.list-section,

.summary-section,

.category-limit-section {

margin-bottom: 20px;

}

label {

display: block;

margin-bottom: 8px;

font-weight: bold;

text-align: left;

}

input, select {

width: 100%;

padding: 10px;

margin-bottom: 15px;

border: 1px solid #ccc;

border-radius: 5px;

font-size: 1rem;

}

button {

background-color: #4caf50;

color: white;

border: none;

border-radius: 5px;

padding: 10px 15px;

cursor: pointer;

font-size: 1rem;

}

button:hover {

background-color: #45a049;

}

#expense-list {

list-style: none;

padding: 0;

}

#expense-list li {

background-color: #f9f9f9;

margin-bottom: 10px;

padding: 10px;

display: flex;

justify-content: space-between;

border: 1px solid #ddd;

border-radius: 5px;

}

#total-expense {

font-size: 2rem;

font-weight: bold;

color: #4caf50;

}

.notification {

color: red;

font-weight: bold;

margin: 15px 0;

}

</style>

</head>

<body>

<div class="main-container">

<header class="header">

<h1>BudgetBuddy</h1>

<p>Your simple expense tracker</p>

</header>

<div id="greeting" class="greeting"></div>

<main class="content">

<section class="form-section">

<h2>Add Expense</h2>

<form id="expense-form">

<div class="form-group">

<label for="expense-name">Expense Name:</label>

<input type="text" id="expense-name" placeholder="Enter expense name" required>

</div>

<div class="form-group">

<label for="expense-amount">Amount (₹):</label>

<input type="number" id="expense-amount" placeholder="Enter amount" required>

</div>

<div class="form-group">

<label for="expense-category">Category:</label>

<select id="expense-category" required>

<option value="" disabled selected>Select category</option>

<option value="Food & Dining">Food & Dining</option>

<option value="Fuel">Fuel</option>

<option value="Movies/Concerts/Events">Movies/Concerts/Events</option>

<option value="Shopping">Shopping</option>

<option value="Personal Care">Personal Care</option>

<option value="Socializing">Socializing</option>

<option value="Travel & Leisure">Travel & Leisure</option>

<option value="Health & Medical">Health & Medical</option>

<option value="Education & Learning">Education & Learning</option>

<option value="Miscellaneous">Miscellaneous</option>

</select>

</div>

<div class="form-group">

<label for="expense-date">Date:</label>

<input type="date" id="expense-date" required>

</div>

<button type="submit">Add Expense</button>

</form>

</section>

<section class="list-section">

<h2>Expense List</h2>

<ul id="expense-list"></ul>

<p id="notification" class="notification"></p>

</section>

<section class="summary-section">

<h2>Total Expense</h2>

<p id="total-expense">₹0.00</p>

</section>

<section class="category-limit-section">

<h2>Set Category Limits</h2>

<label for="category-select">Select Category:</label>

<select id="category-select">

<option value="" disabled selected>Select category</option>

<option value="Food & Dining">Food & Dining</option>

<option value="Fuel">Fuel</option>

<option value="Movies/Concerts/Events">Movies/Concerts/Events</option>

<option value="Shopping">Shopping</option>

<option value="Personal Care">Personal Care</option>

<option value="Socializing">Socializing</option>

<option value="Travel & Leisure">Travel & Leisure</option>

<option value="Health & Medical">Health & Medical</option>

<option value="Education & Learning">Education & Learning</option>

<option value="Miscellaneous">Miscellaneous</option>

</select>

<label for="category-limit-amount">Set Limit (₹):</label>

<input type="number" id="category-limit-amount" placeholder="Enter limit" min="0">

<button id="set-limit-button">Set Limit</button>

</section>

</main>

</div>

<script>

document.addEventListener("DOMContentLoaded", () => {

const expenseForm = document.getElementById("expense-form");

const expenseList = document.getElementById("expense-list");

const totalExpense = document.getElementById("total-expense");

const notification = document.getElementById("notification");

let expenses = JSON.parse(localStorage.getItem("expenses")) || [];

let categoryLimits = JSON.parse(localStorage.getItem("categoryLimits")) || {};

function greetUser() {

const hours = new Date().getHours();

let greeting = "Welcome";

if (hours < 12) {

greeting = "Good Morning";

} else if (hours < 18) {

greeting = "Good Afternoon";

} else {

greeting = "Good Evening";

}

notification.textContent = ${greeting}! Start managing your expenses.;

}

expenseForm.addEventListener("submit", (e) => {

e.preventDefault();

const name = document.getElementById("expense-name").value;

const amount = parseFloat(document.getElementById("expense-amount").value);

const category = document.getElementById("expense-category").value;

const date = document.getElementById("expense-date").value;

if (name && !isNaN(amount) && category && date) {

expenses.push({ name, amount, category, date });

localStorage.setItem("expenses", JSON.stringify(expenses));

displayExpenses();

calculateTotalExpense();

// Check if the added expense exceeds the category limit

const categoryTotal = expenses

.filter(exp => exp.category === category)

.reduce((sum, exp) => sum + exp.amount, 0);

if (categoryLimits[category] && categoryTotal > categoryLimits[category]) {

notification.textContent = Warning: ${category} exceeded the limit of ₹${categoryLimits[category]};

} else {

notification.textContent = "";

}

expenseForm.reset();

} else {

notification.textContent = "Please fill out all fields.";

}

});

document.getElementById("set-limit-button").addEventListener("click", () => {

const category = document.getElementById("category-select").value;

const limit = parseFloat(document.getElementById("category-limit-amount").value);

if (category && !isNaN(limit) && limit > 0) {

categoryLimits[category] = limit;

localStorage.setItem("categoryLimits", JSON.stringify(categoryLimits));

notification.textContent = Limit set for ${category}: ₹${limit};

}

});

function displayExpenses() {

expenseList.innerHTML = expenses

.map(

(exp, idx) =>

<li>${exp.name} - ₹${exp.amount} (${exp.category}) on ${exp.date} <button class="delete-btn" data-index="${idx}">Delete</button></li>

)

.join("");

const deleteButtons = document.querySelectorAll(".delete-btn");

deleteButtons.forEach(button => {

button.addEventListener("click", () => {

deleteExpense(parseInt(button.dataset.index, 10));

});

});

}

function calculateTotalExpense() {

const total = expenses.reduce((sum, exp) => sum + exp.amount, 0);

totalExpense.textContent = ₹${total.toFixed(2)};

}

function deleteExpense(index) {

const removedExpense = expenses[index];

expenses.splice(index, 1);

localStorage.setItem("expenses", JSON.stringify(expenses));

displayExpenses();

calculateTotalExpense();

// Check if the category total after deletion exceeds the limit

const categoryTotal = expenses

.filter(exp => exp.category === removedExpense.category)

.reduce((sum, exp) => sum + exp.amount, 0);

if (categoryLimits[removedExpense.category] && categoryTotal > categoryLimits[removedExpense.category]) {

notification.textContent = Warning: ${removedExpense.category} exceeded the limit of ₹${categoryLimits[removedExpense.category]};

} else {

notification.textContent = "";

}

}

displayExpenses();

calculateTotalExpense();

greetUser();

});

</script>

</body>

</html>

**TEST**

**Testing Process**

**We used a variety of tests to ensure the application works as expected:**

* **Unit Tests: Testing specific functions like updateTotalAmount() to ensure it correctly calculates the total amount of all expenses.**
* **Manual Testing: We manually tested the app in different browsers (Chrome, Firefox, Edge) to ensure compatibility and cross-browser functionality.**
* **Edge Case Handling: We checked how the app handled invalid inputs, such as leaving fields empty or entering negative amounts.**

**Common Bugs Found**

* **Form Submission: We had to implement proper form validation to ensure users don’t submit empty or invalid data (like negative values).**
* **Date Formatting: Handling the date input required converting user inputs into a consistent format for storage and display.**

**IMPLEMENT**

### Deployment

### As this is a simple web project, we used local deployment for testing. However, for a more advanced version, we could deploy it on platforms such as:

### GitHub Pages: Free hosting for static websites.

### Netlify: A popular platform for deploying web projects with continuous deployment support.

### Future Improvements

### User Authentication: Implementing user login functionality would allow users to have personalized expense records stored securely.

### Graphical Reports: We could add graphs to visually represent expenses by category or over time, using libraries like Chart.js.

### Cloud Storage: Moving from in-memory storage to a cloud-based database such as Firebase or MongoDB would make the tracker more robust.

### Mobile App: The expense tracker could be further developed as a mobile app using frameworks like React Native, making it more accessible.

### Conclusion

This Expense Tracker project provides users with a simple and intuitive way to track and manage their expenses. The project incorporates important features such as expense categorization, monthly summary, and total calculation. By following a structured approach of research, analysis, ideation, building, testing, and implementation, we have created a functional prototype that can be further expanded with more advanced features like user authentication, persistent storage, and reporting tools. The next steps could involve making the application more interactive and scalable.

Reference links: