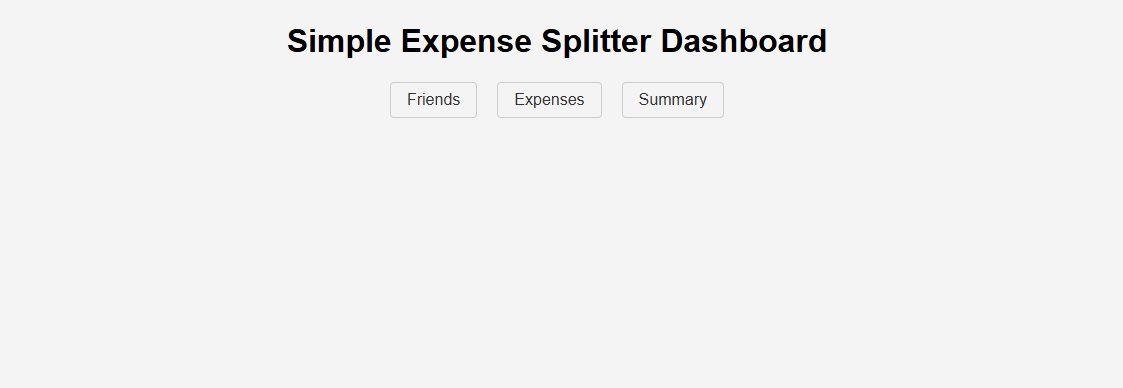
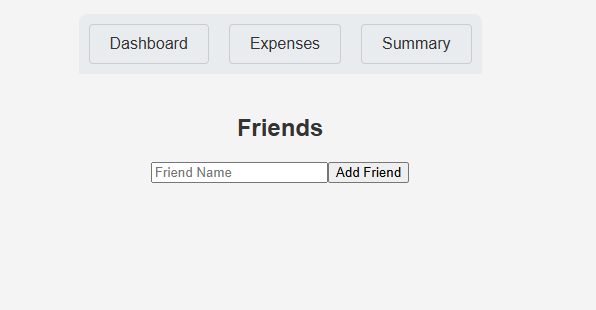
**Simple Expense Splitter - Milestone-2 Coding Challenge**

* **Palgun Jain V**

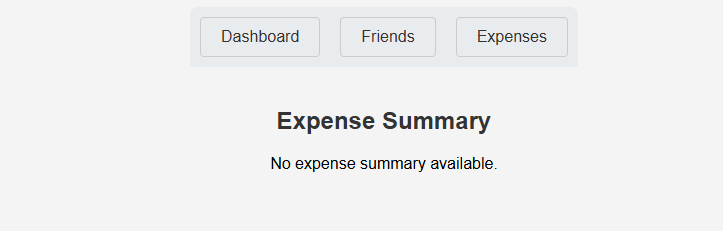
**Dashboard:**

****

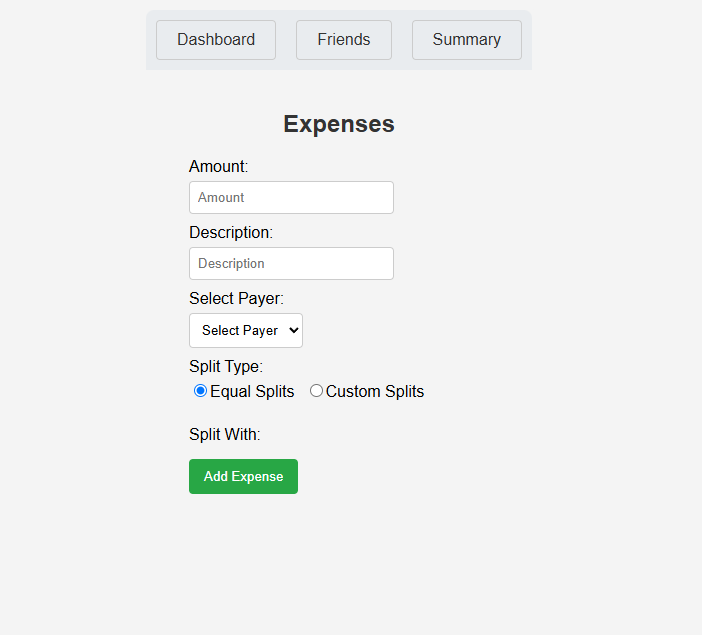
**Friends:**

****

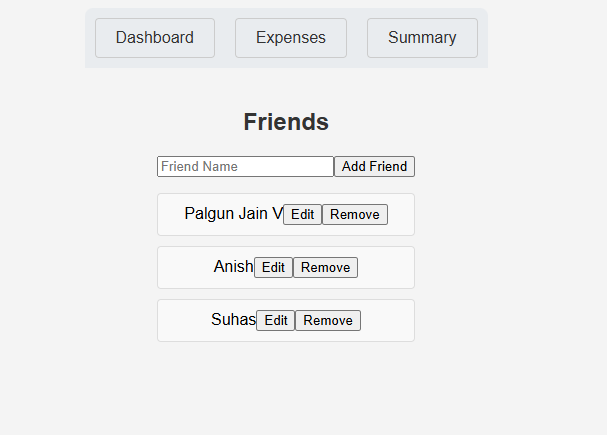
**Summary:**

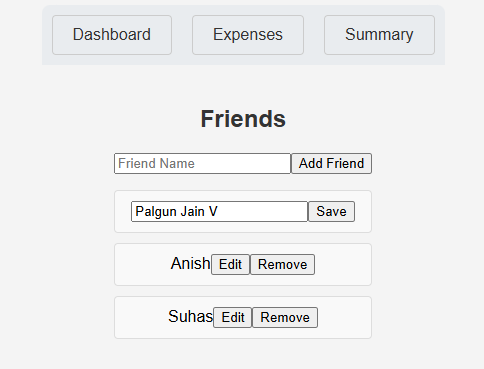
****

**Expenses:**

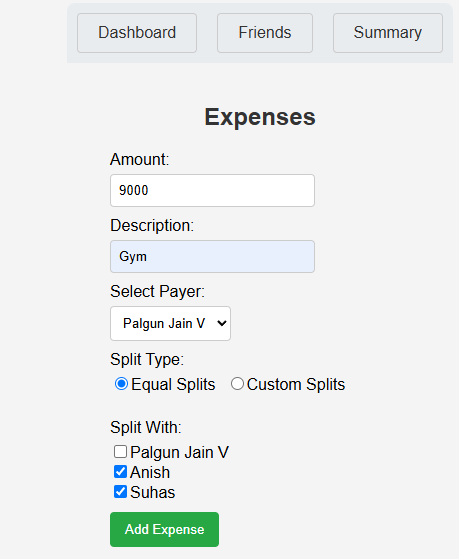
****

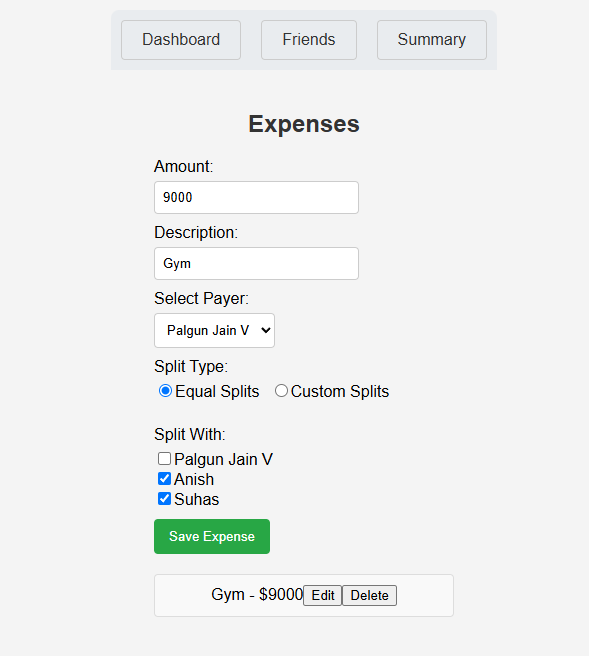
**Add and Manage Friends:**

****

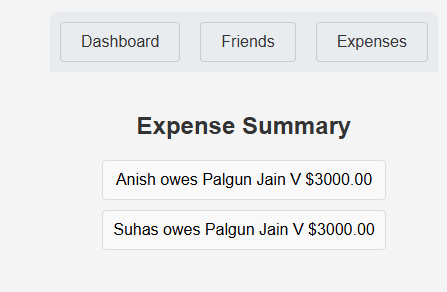
****

**Add and Manage Expenses:**

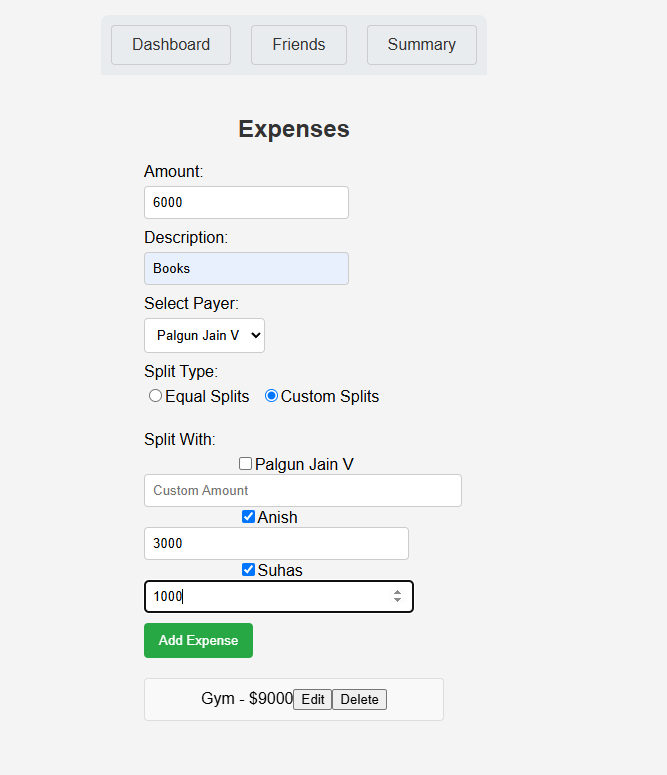
****

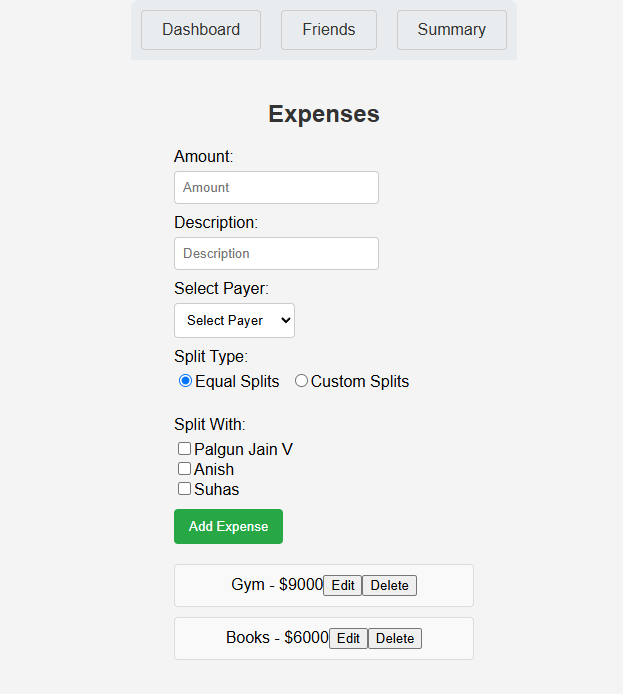


**Equal Split:**

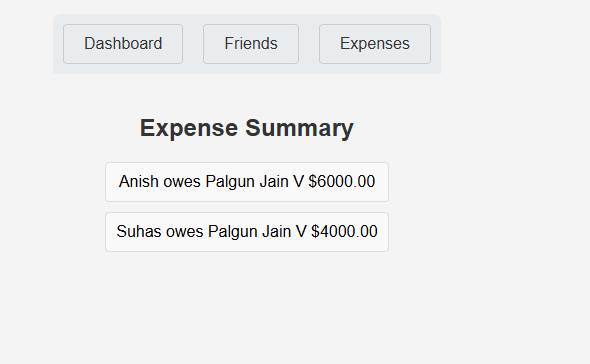


**Custom split:**

****

****

**Expense Summary:**

****

**Codes:**

**Dashboard.css:**

**.dashboard {**

**text-align: center;**

**padding: 20px;**

**}**

**.dashboard-nav {**

**margin-top: 20px;**

**display: flex;**

**justify-content: center;**

**}**

**.dashboard-nav a {**

**margin: 0 10px;**

**text-decoration: none;**

**padding: 8px 16px;**

**border: 1px solid #ccc;**

**border-radius: 4px;**

**color: #333;**

**}**

**.dashboard-nav a:hover {**

**background-color: #f0f0f0;**

**}**

**Dashboard.js:**

**import React from 'react';**

**import { Link } from 'react-router-dom';**

**import './Dashboard.css';**

**function Dashboard() {**

**return (**

**<div className="dashboard">**

**<h1>Simple Expense Splitter Dashboard</h1>**

**<nav className="dashboard-nav">**

**<Link to="/friends">Friends</Link>**

**<Link to="/expenses">Expenses</Link>**

**<Link to="/summary">Summary</Link>**

**</nav>**

**</div>**

**);**

**}**

**export default Dashboard;**

**FriendList.js:**

**import React, { useState, useEffect } from 'react';**

**import FriendService from '../services/FriendService';**

**import { Link } from 'react-router-dom';**

**import './FriendList.css';**

**function FriendList() {**

**const [friends, setFriends] = useState([]);**

**const [newFriendName, setNewFriendName] = useState('');**

**const [editingFriendId, setEditingFriendId] = useState(null);**

**const [editedFriendName, setEditedFriendName] = useState('');**

**useEffect(() => {**

**setFriends(FriendService.getFriends());**

**}, []);**

**const handleAddFriend = () => {**

**if (newFriendName.trim()) {**

**FriendService.addFriend(newFriendName);**

**setFriends(FriendService.getFriends());**

**setNewFriendName('');**

**}**

**};**

**const handleRemoveFriend = (id) => {**

**FriendService.removeFriend(id);**

**setFriends(FriendService.getFriends());**

**};**

**const handleEditFriend = (friend) => {**

**setEditingFriendId(friend.id);**

**setEditedFriendName(friend.name);**

**};**

**const handleSaveFriend = (id) => {**

**FriendService.updateFriend(id, editedFriendName);**

**setFriends(FriendService.getFriends());**

**setEditingFriendId(null);**

**setEditedFriendName('');**

**};**

**return (**

**<div className="friend-list">**

**<nav className="friend-list-nav">**

**<Link to="/">Dashboard</Link>**

**<Link to="/expenses">Expenses</Link>**

**<Link to="/summary">Summary</Link>**

**</nav>**

**<div className="friend-list-content">**

**<h2>Friends</h2>**

**<div>**

**<input**

**type="text"**

**value={newFriendName}**

**onChange={(e) => setNewFriendName(e.target.value)}**

**placeholder="Friend Name"**

**/>**

**<button onClick={handleAddFriend}>Add Friend</button>**

**</div>**

**<ul>**

**{friends.map((friend) => (**

**<li key={friend.id}>**

**{editingFriendId === friend.id ? (**

**<>**

**<input**

**type="text"**

**value={editedFriendName}**

**onChange={(e) => setEditedFriendName(e.target.value)}**

**/>**

**<button onClick={() => handleSaveFriend(friend.id)}>Save</button>**

**</>**

**) : (**

**<>**

**{friend.name}**

**<button onClick={() => handleEditFriend(friend)}>Edit</button>**

**<button onClick={() => handleRemoveFriend(friend.id)}>Remove</button>**

**</>**

**)}**

**</li>**

**))}**

**</ul>**

**</div>**

**</div>**

**);**

**}**

**export default FriendList;**

**FriendList.css:**

**.friend-list {**

**padding: 20px;**

**display: flex;**

**flex-direction: column;**

**align-items: center;**

**}**

**.friend-list-nav {**

**margin-bottom: 20px;**

**display: flex;**

**justify-content: center;**

**}**

**.friend-list-nav a {**

**margin: 0 10px;**

**padding: 8px 16px;**

**border: 1px solid #ccc;**

**border-radius: 4px;**

**text-decoration: none;**

**color: #333;**

**}**

**.friend-list-nav a:hover {**

**background-color: #f0f0f0;**

**}**

**.friend-list-content {**

**text-align: center;**

**}**

**.friend-list-content ul {**

**list-style: none;**

**padding: 0;**

**}**

**.friend-list-content li {**

**margin-bottom: 8px;**

**}**

**ExpenseList.js:**

**import React, { useState, useEffect } from 'react';**

**import ExpenseService from '../services/ExpenseService';**

**import FriendService from '../services/FriendService';**

**import { Link } from 'react-router-dom';**

**import './ExpenseList.css';**

**function ExpenseList() {**

**const [expenses, setExpenses] = useState([]);**

**const [friends, setFriends] = useState([]);**

**const [newExpense, setNewExpense] = useState({**

**amount: '',**

**description: '',**

**payer: '',**

**participants: [],**

**customSplits: {},**

**splitType: 'equal',**

**});**

**const [editingExpenseId, setEditingExpenseId] = useState(null);**

**const [editedExpense, setEditedExpense] = useState(null);**

**useEffect(() => {**

**setExpenses(ExpenseService.getExpenses());**

**setFriends(FriendService.getFriends());**

**}, []);**

**const handleAddExpense = () => {**

**if (newExpense.amount && newExpense.description && newExpense.payer) {**

**ExpenseService.addExpense(newExpense);**

**setExpenses(ExpenseService.getExpenses());**

**setNewExpense({**

**amount: '',**

**description: '',**

**payer: '',**

**participants: [],**

**customSplits: {},**

**splitType: 'equal',**

**});**

**}**

**};**

**const handleParticipantChange = (friendId) => {**

**if (newExpense.participants.includes(friendId)) {**

**setNewExpense({**

**...newExpense,**

**participants: newExpense.participants.filter((id) => id !== friendId),**

**});**

**} else {**

**setNewExpense({**

**...newExpense,**

**participants: [...newExpense.participants, friendId],**

**});**

**}**

**};**

**const handleCustomSplitChange = (friendId, amount) => {**

**setNewExpense({**

**...newExpense,**

**customSplits: {**

**...newExpense.customSplits,**

**[friendId]: amount,**

**},**

**});**

**};**

**const handleEditExpense = (expense) => {**

**setEditingExpenseId(expense.id);**

**setEditedExpense({**

**...expense,**

**customSplits: { ...expense.customSplits },**

**splitType: expense.splitType,**

**});**

**};**

**const handleSaveExpense = (id) => {**

**ExpenseService.updateExpense(id, editedExpense);**

**setExpenses(ExpenseService.getExpenses());**

**setEditingExpenseId(null);**

**setEditedExpense(null);**

**};**

**const handleDeleteExpense = (id) => {**

**ExpenseService.deleteExpense(id);**

**setExpenses(ExpenseService.getExpenses());**

**};**

**const handleSplitTypeChange = (type) => {**

**setNewExpense({ ...newExpense, splitType: type });**

**};**

**return (**

**<div className="expense-list">**

**<nav className="expense-list-nav">**

**<Link to="/">Dashboard</Link>**

**<Link to="/friends">Friends</Link>**

**<Link to="/summary">Summary</Link>**

**</nav>**

**<div className="expense-list-content">**

**<h2>Expenses</h2>**

**<div className="expense-form">**

**<div className="form-group">**

**<label htmlFor="amount">Amount:</label>**

**<input**

**type="number"**

**id="amount"**

**value={editingExpenseId === null ? newExpense.amount : editedExpense?.amount || ''}**

**onChange={(e) =>**

**editingExpenseId === null**

**? setNewExpense({ ...newExpense, amount: e.target.value })**

**: setEditedExpense({ ...editedExpense, amount: e.target.value })**

**}**

**placeholder="Amount"**

**/>**

**</div>**

**<div className="form-group">**

**<label htmlFor="description">Description:</label>**

**<input**

**type="text"**

**id="description"**

**value={editingExpenseId === null ? newExpense.description : editedExpense?.description || ''}**

**onChange={(e) =>**

**editingExpenseId === null**

**? setNewExpense({ ...newExpense, description: e.target.value })**

**: setEditedExpense({ ...editedExpense, description: e.target.value })**

**}**

**placeholder="Description"**

**/>**

**</div>**

**<div className="form-group">**

**<label htmlFor="payer">Select Payer:</label>**

**<select**

**id="payer"**

**value={editingExpenseId === null ? newExpense.payer : editedExpense?.payer || ''}**

**onChange={(e) =>**

**editingExpenseId === null**

**? setNewExpense({ ...newExpense, payer: e.target.value })**

**: setEditedExpense({ ...editedExpense, payer: e.target.value })**

**}**

**>**

**<option value="">Select Payer</option>**

**{friends.map((friend) => (**

**<option key={friend.id} value={friend.id}>**

**{friend.name}**

**</option>**

**))}**

**</select>**

**</div>**

**<div className="form-group">**

**<label>Split Type:</label>**

**<div className="split-options">**

**<label>**

**<input**

**type="radio"**

**value="equal"**

**checked={newExpense.splitType === 'equal'}**

**onChange={() => handleSplitTypeChange('equal')}**

**/>**

**Equal Splits**

**</label>**

**<label>**

**<input**

**type="radio"**

**value="custom"**

**checked={newExpense.splitType === 'custom'}**

**onChange={() => handleSplitTypeChange('custom')}**

**/>**

**Custom Splits**

**</label>**

**</div>**

**</div>**

**<div className="form-group">**

**<label>Split With:</label>**

**{friends.map((friend) => (**

**<div key={friend.id}>**

**<label>**

**<input**

**type="checkbox"**

**checked={**

**editingExpenseId === null**

**? newExpense.participants.includes(friend.id)**

**: editedExpense?.participants.includes(friend.id) || false**

**}**

**onChange={() =>**

**editingExpenseId === null**

**? handleParticipantChange(friend.id)**

**: setEditedExpense({**

**...editedExpense,**

**participants: editedExpense?.participants.includes(friend.id)**

**? editedExpense?.participants.filter((id) => id !== friend.id)**

**: [...(editedExpense?.participants || []), friend.id],**

**})**

**}**

**/>**

**{friend.name}**

**</label>**

**{newExpense.splitType === 'custom' && (**

**<input**

**type="number"**

**placeholder="Custom Amount"**

**value={newExpense.customSplits[friend.id] || ''}**

**onChange={(e) => handleCustomSplitChange(friend.id, e.target.value)}**

**/>**

**)}**

**</div>**

**))}**

**</div>**

**{editingExpenseId === null ? (**

**<button onClick={handleAddExpense}>Add Expense</button>**

**) : (**

**<button onClick={() => handleSaveExpense(editingExpenseId)}>Save Expense</button>**

**)}**

**</div>**

**<ul>**

**{expenses.map((expense) => (**

**<li key={expense.id}>**

**{expense.description} - ${expense.amount}**

**<button onClick={() => handleEditExpense(expense)}>Edit</button>**

**<button onClick={() => handleDeleteExpense(expense.id)}>Delete</button>**

**</li>**

**))}**

**</ul>**

**</div>**

**</div>**

**);**

**}**

**export default ExpenseList;**

**ExpenseList.css::**

**.expense-list {**

**padding: 20px;**

**display: flex;**

**flex-direction: column;**

**align-items: center;**

**}**

**.expense-list-nav {**

**margin-bottom: 20px;**

**display: flex;**

**justify-content: center;**

**}**

**.expense-list-nav a {**

**margin: 0 10px;**

**padding: 8px 16px;**

**border: 1px solid #ccc;**

**border-radius: 4px;**

**text-decoration: none;**

**color: #333;**

**}**

**.expense-list-nav a:hover {**

**background-color: #f0f0f0;**

**}**

**.expense-list-content {**

**text-align: center;**

**}**

**.expense-list-content ul {**

**list-style: none;**

**padding: 0;**

**}**

**.expense-list-content li {**

**margin-bottom: 8px;**

**}**

**.expense-form {**

**display: flex;**

**flex-direction: column;**

**align-items: flex-start;**

**width: 300px; /\* Adjust as needed \*/**

**margin-bottom: 20px;**

**}**

**.expense-form .form-group {**

**margin-bottom: 10px;**

**display: flex;**

**flex-direction: column;**

**align-items: flex-start;**

**}**

**.expense-form label {**

**margin-bottom: 5px;**

**}**

**.split-options {**

**display: flex;**

**gap: 10px; /\* Space between radio buttons \*/**

**}**

**ExpenseSummary.js:**

**import React, { useState, useEffect } from 'react';**

**import ExpenseService from '../services/ExpenseService';**

**import CalculationService from '../services/CalculationService';**

**import FriendService from '../services/FriendService';**

**import { Link } from 'react-router-dom';**

**import './ExpenseSummary.css';**

**function ExpenseSummary() {**

**const [expenses, setExpenses] = useState([]);**

**const [friends, setFriends] = useState([]);**

**const [summary, setSummary] = useState([]);**

**useEffect(() => {**

**const fetchExpensesAndFriends = () => {**

**const fetchedExpenses = ExpenseService.getExpenses();**

**const fetchedFriends = FriendService.getFriends();**

**setExpenses(fetchedExpenses);**

**setFriends(fetchedFriends);**

**};**

**fetchExpensesAndFriends();**

**}, []);**

**useEffect(() => {**

**console.log('Expenses updated:', expenses);**

**console.log('Friends updated:', friends);**

**if (expenses.length > 0 && friends.length > 0) {**

**const calculatedSummary = CalculationService.calculateSplits(expenses, friends);**

**setSummary(calculatedSummary);**

**console.log('Summary updated:', calculatedSummary);**

**} else {**

**setSummary([]);**

**}**

**}, [expenses, friends]);**

**return (**

**<div className="expense-summary">**

**<nav className="expense-summary-nav">**

**<Link to="/">Dashboard</Link>**

**<Link to="/friends">Friends</Link>**

**<Link to="/expenses">Expenses</Link>**

**</nav>**

**<div className="expense-summary-content">**

**<h2>Expense Summary</h2>**

**{summary && summary.length > 0 ? (**

**<ul>**

**{summary.map((item) => (**

**<li key={`${item.from}-${item.to}-${item.amount}`}>**

**{item.from} owes {item.to} ${item.amount}**

**</li>**

**))}**

**</ul>**

**) : (**

**<p>No expense summary available.</p>**

**)}**

**</div>**

**</div>**

**);**

**}**

**export default ExpenseSummary;**

**ExpenseSummary.css:**

**.expense-summary {**

**padding: 20px;**

**display: flex;**

**flex-direction: column;**

**align-items: center;**

**}**

**.expense-summary-nav {**

**margin-bottom: 20px;**

**display: flex;**

**justify-content: center;**

**}**

**.expense-summary-nav a {**

**margin: 0 10px;**

**padding: 8px 16px;**

**border: 1px solid #ccc;**

**border-radius: 4px;**

**text-decoration: none;**

**color: #333;**

**}**

**.expense-summary-nav a:hover {**

**background-color: #f0f0f0;**

**}**

**.expense-summary-content {**

**text-align: center;**

**}**

**.expense-summary-content ul {**

**list-style: none;**

**padding: 0;**

**}**

**.expense-summary-content li {**

**margin-bottom: 8px;**

**}**

**FriendService.js:**

**let friends = [];**

**const FriendService = {**

**addFriend: (name) => {**

**friends.push({ id: Date.now(), name });**

**},**

**removeFriend: (id) => {**

**friends = friends.filter((friend) => friend.id !== id);**

**},**

**updateFriend: (id, newName) => {**

**friends = friends.map((friend) =>**

**friend.id === id ? { ...friend, name: newName } : friend**

**);**

**},**

**getFriends: () => {**

**return friends;**

**},**

**};**

**export default FriendService;**

**ExpenseService.js**

**let expenses = [];**

**let nextExpenseId = 1;**

**const ExpenseService = {**

**getExpenses: () => {**

**return [...expenses];**

**},**

**addExpense: (expense) => {**

**const newExpense = { ...expense, id: nextExpenseId++ };**

**expenses.push(newExpense);**

**return newExpense;**

**},**

**updateExpense: (id, updatedExpense) => {**

**expenses = expenses.map((expense) =>**

**expense.id === id ? { ...updatedExpense, id } : expense**

**);**

**return expenses.find((expense) => expense.id === id);**

**},**

**deleteExpense: (id) => {**

**expenses = expenses.filter((expense) => expense.id !== id);**

**},**

**};**

**export default ExpenseService;**

**CalculationService.js**

**const CalculationService = {**

**calculateSplits: (expenses, friends) => {**

**const friendBalances = {};**

**friends.forEach((friend) => {**

**friendBalances[friend.id] = 0;**

**});**

**expenses.forEach((expense) => {**

**const payerId = parseInt(expense.payer);**

**const amount = parseFloat(expense.amount);**

**const participants = expense.participants.map(Number);**

**const splitType = expense.splitType;**

**const customSplits = expense.customSplits;**

**if (participants.length === 0) {**

**friendBalances[payerId] += amount;**

**} else {**

**if (splitType === 'equal') {**

**const splitAmount = amount / (participants.length + 1);**

**friendBalances[payerId] += amount - splitAmount;**

**participants.forEach((participantId) => {**

**friendBalances[participantId] -= splitAmount;**

**});**

**} else if (splitType === 'custom') {**

**let totalCustomSplit = 0;**

**participants.forEach((participantId) => {**

**if (customSplits[participantId]) {**

**totalCustomSplit += parseFloat(customSplits[participantId]);**

**}**

**});**

**const payerShare = amount - totalCustomSplit;**

**friendBalances[payerId] += payerShare;**

**participants.forEach((participantId) => {**

**if (customSplits[participantId]) {**

**friendBalances[participantId] -= parseFloat(customSplits[participantId]);**

**}**

**});**

**}**

**}**

**});**

**const detailedSummary = [];**

**Object.keys(friendBalances).forEach((friendId1) => {**

**Object.keys(friendBalances).forEach((friendId2) => {**

**if (friendId1 !== friendId2) {**

**const balance1 = parseFloat(friendBalances[friendId1]);**

**const balance2 = parseFloat(friendBalances[friendId2]);**

**if (balance1 > 0 && balance2 < 0) {**

**const amount = Math.min(balance1, Math.abs(balance2));**

**if (amount > 0) {**

**const friend1Name = friends.find((f) => f.id === parseInt(friendId1))?.name;**

**const friend2Name = friends.find((f) => f.id === parseInt(friendId2))?.name;**

**detailedSummary.push({**

**from: friend2Name,**

**to: friend1Name,**

**amount: amount.toFixed(2),**

**});**

**}**

**}**

**}**

**});**

**});**

**return detailedSummary;**

**},**

**};**

**export default CalculationService;**