VIT®

Vellore Institute of Technology

(Deemed to be University under section 3 of UGC Act, 1956)

AI-Powered Expense Tracker

BCSE203E - WEB PROGRAMMING

PROJECT DOCUMENT

Winter Semester 2024 - 2025

23DS0338

Harsat Ponnusamy

Under The Guidance Of

Dr. KAUSER AHMED P

SCOPE

VIT®

Vellore Institute of Technology

(Deemed to be University under section 3 of UGC Act, 1956)

School of Computer Science and Engineering

DECLARATION

I/We hereby declare that the project entitled "***AI-Powered Expense Tracker***" submitted by me/us to the School of Computer Science and Engineering, VIT University, Vellore-14 in partial fulfillment of the requirements for the **Web Programming (BCSE203E)** course, is a record of bonafide work carried out by me under the supervision of **Dr. Kauser Ahmed, Assistant Professor Senior**. I further declare that the work reported in this project has not been submitted and will not be submitted, either in part or in full, for the award of any other course or degree or diploma of this institute or of any other institute or university.

Contents

List of Figures

List of Abbreviations

AI	Artificial Intelligence
LLM	Large Language Model
UI	User Interface
UX	User Experience
PWA	Progressive Web Application
SaaS	Software as a Service

Abstract

This project showcases an advanced AI-powered expense tracker application, developed using Next.js 14 and GPT-4o. The application is designed to transform the way individuals manage their finances by automating the expense tracking process and providing intelligent financial insights. With its innovative use of artificial intelligence, the platform categorizes expenses accurately and provides detailed, personalized analytics, allowing users to make informed financial decisions effortlessly.

By leveraging the capabilities of Next.js 14, the application ensures a fast, responsive, and smooth user experience across devices. It uses GPT-4o for intelligent expense categorization, where the AI analyzes spending patterns and assigns appropriate categories to each transaction, eliminating the need for manual data entry or sorting. This automation streamlines the often tedious process of tracking expenses, saving users valuable time.

In addition to expense categorization, the app provides insightful financial analytics, helping users gain a deeper understanding of their spending habits. Users can visualize their financial data through graphs and charts, making it easier to identify trends, track progress toward savings goals, and highlight areas where spending could be optimized. The application also offers personalized recommendations based on spending history, ensuring that users receive tailored advice on how to improve their financial health.

With the power of modern web technologies and AI, this solution presents a seamless and efficient alternative to traditional expense tracking methods, offering users a comprehensive, user-friendly platform that simplifies personal finance management. Whether you're looking to create a budget, save for a specific goal, or simply gain control over your spending, the application delivers the tools and insights necessary for smarter financial management.

1. Introduction

1.1 Aim

The aim of this project is to develop a comprehensive AI-powered expense tracking application that simplifies the process of recording, categorizing, and analyzing financial transactions. By integrating GPT-4o with Next.js, the application seeks to provide users with actionable insights into their spending patterns and help them make better financial decisions.

1.2 Objective

The objectives of this AI-powered expense tracker application are:

- To develop a responsive and user-friendly web application using Next.js 14 that allows users to track their income and expenses across multiple categories.
- To implement GPT-4o integration for generating personalized financial insights based on user's spending patterns.
- To create an intelligent categorization system that accurately classifies transactions based on merchant information and purchase descriptions.
- To design a comprehensive dashboard with visual representations of spending patterns, budget progress, and financial analytics.
- To develop a system for setting and monitoring budget limits with automated alerts when approaching thresholds.
- To create AI-generated financial insights and recommendations based on spending patterns and saving opportunities.
- To ensure cross-device compatibility through responsive design, allowing users to track expenses on both desktop and mobile devices.
- To build a scalable application architecture that can support future features and growing user bases.

2. System Requirements

- **Frontend:** Next.js 14, React, Tailwind CSS, Ant Design, Lucide-React
- **State Management:** Built-in React State Management
- **AI Integration:** GPT-4o API
- **Deployment:** Vercel

3. Recent Technology Adapted

3.1 Next.js 14

3.1.1 Technology-1 description

Next.js 14 is a React framework that enables server-side rendering, static site generation, and the development of full-stack web applications. For our expense tracker, we leverage Next.js 14's latest features including App Router and client-side components to create a performant and responsive user experience.

Key features utilized:

- App Router for optimized routing
- Client and Server Components for improved performance
- Built-in optimization for images and fonts
- Responsive design capabilities

3.1.2 Sample Code Showing layout.tsx

```
import './globals.css';
import type { Metadata } from 'next';
import { IBM_Plex_Sans } from 'next/font/google';

import Footer from '@/components/footer';

const ibmSans = IBM_Plex_Sans({weight:"400", preload:
   true, subsets: ["latin", "greek", "latin-ext"]});

export const metadata: Metadata = {
  title: 'EasyExpenses',
  description: 'An Expense Tracker app built with react
     ',
};

export default function RootLayout({
  children,
}: {
  children: React.ReactNode;
}) {
  return (
```

```
    <html lang="en">
      <body className={ibmSans.className}>{children}
        <Footer />
      </body>
    </html>
  );
}
}
```

3.2 GPT-4o Integration

3.2.1 Technology-2 description

GPT-4o is OpenAI's multimodal large language model. In our expense tracker application, we utilize GPT-4o for:

- Intelligent categorization of expenses based on transaction descriptions
- Generating financial insights and spending recommendations
- Providing personalized budgeting advice
- Identifying spending patterns and suggesting optimization strategies

The integration allows the application to offer intelligent financial assistance that goes beyond simple expense tracking, providing users with actionable insights to improve their financial health.

3.2.2 Sample Screenshot

```
export default function Home() {
    const [showChatbot, setShowChatbot] = useState(false
       );
    return(
        <button
                onClick={() => setShowChatbot(!
                   showChatbot)}
                className="fixed␣bottom-6␣right-6␣bg-
                   primary␣text-white␣p-4␣rounded-full␣
                   shadow-lg␣flex␣items-center␣space-x-2
                   "
              >
                <MessageCircle className="h-5␣w-5" />
                <span>{showChatbot ? "Close␣Chat" : "
                   Open␣Chat"}</span>
```

```
                </button>{showChatbot && (
                  <div␣className="fixed␣bottom-20␣right-10
                     ␣w-[450px]␣h-[600px]␣border␣border-
                     gray-300␣bg-white␣shadow-lg␣rounded-
                     lg␣overflow-hidden">
                    <iframe
                      src="https://webagent.ai/chatbot/
                         embed/2d33fe77-3dcf-4b24-a186-
                         ff3a0f231cf6/classic"
                      className="w-full␣h-full"
                    ></iframe>
                  </div>
                )}
    )
}
```

4. Solution for Existing problems

4.1 Manual Analysis Elimination

4.1.1 Problem description and its Solution

Problem: Traditional expense tracking applications require users to manually analyze their spending patterns, which is time-consuming and often leads to missed insights. Users struggle to identify spending trends and opportunities for saving money without dedicated financial expertise.

Solution: Our application leverages GPT-4o's processing capabilities to automatically analyze spending patterns and provide personalized financial insights. Users simply enter their expense data, and the AI generates meaningful insights on their spending patterns, identifies potential savings opportunities, and offers tailored financial advice, saving users valuable time and providing expert-level financial analysis.

4.1.2 Sample Screenshot

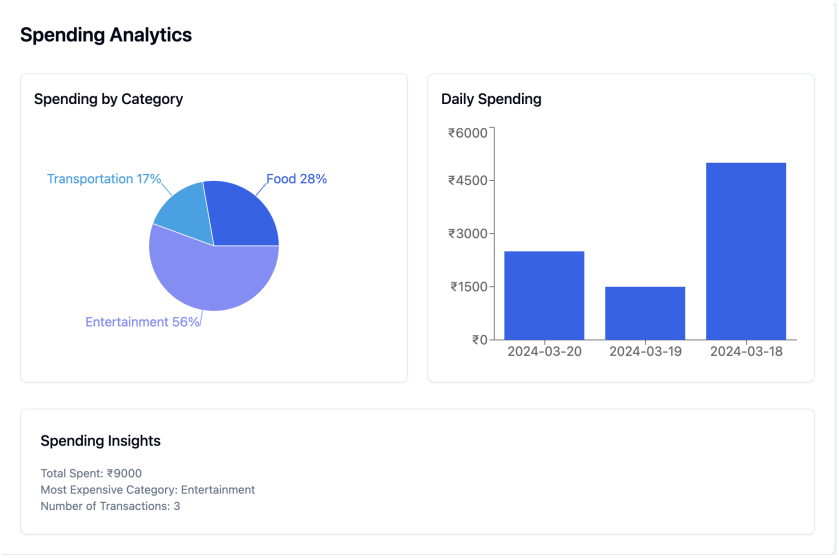

Figure 4.1: Visualizing Spending Patterns

4.2 Intelligent Expense Categorization

4.2.1 Problem description and its Solution

Problem: Manually categorizing expenses is time-consuming and often inconsistent. Users struggle to remember the correct categories for different types of expenses, leading to miscategorization and difficulties in analyzing spending patterns.

Solution: Our application uses GPT-4o to automatically categorize expenses based on transaction descriptions. The AI analyzes the expense details and assigns appropriate categories according to the user's personalized category system. This ensures consistent categorization and enables more accurate financial analysis, allowing users to better understand their spending habits across different categories.

4.2.2 Sample Screenshot

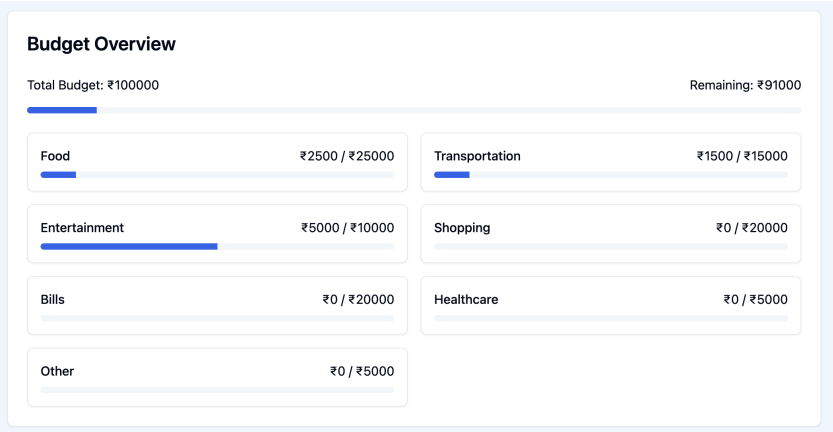

Figure 4.2: Budget Categorization

5. Screenshots

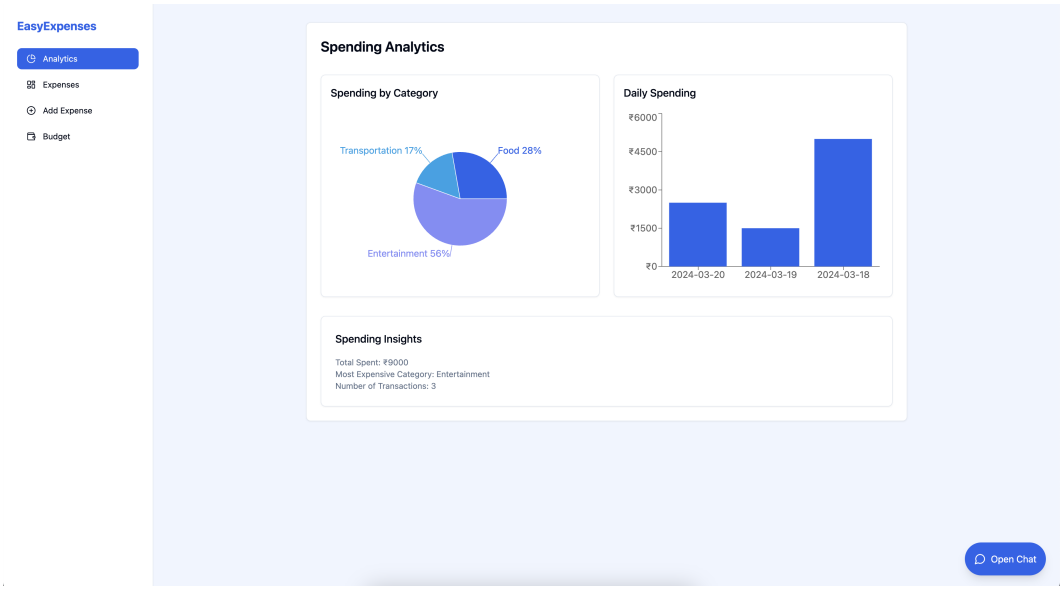

Figure 5.1: Analytics Dashboard - Visualizing Spending Patterns

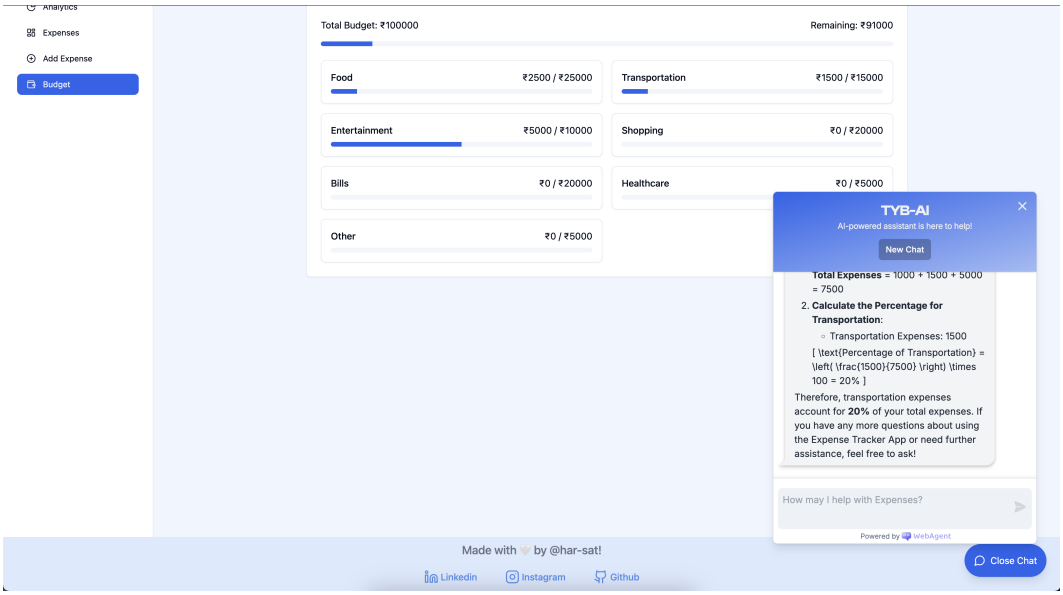

Figure 5.2: AI Chatbot Interface for Financial Insights

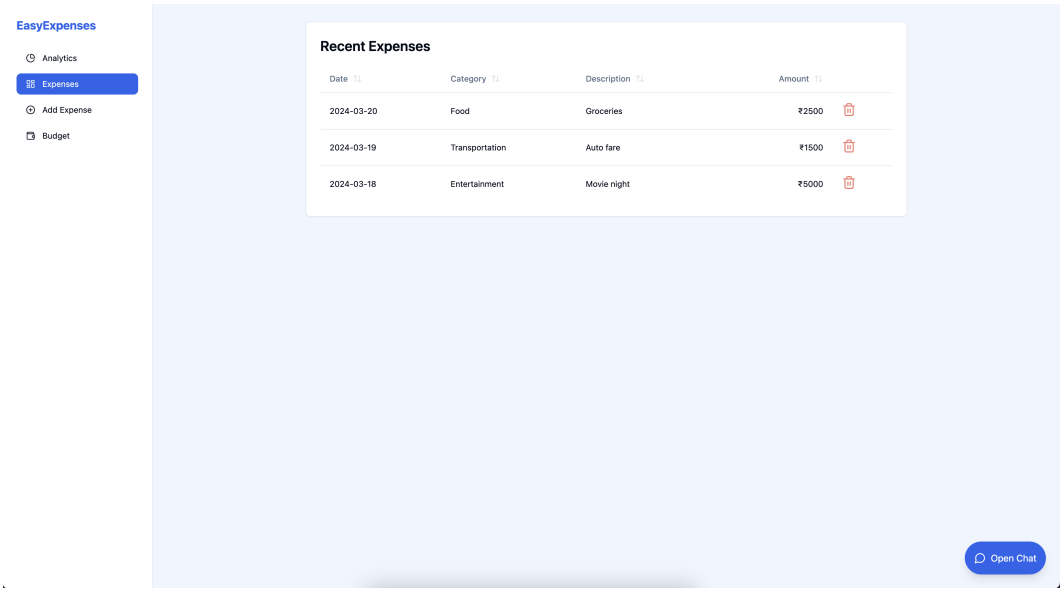

Figure 5.3: Expense Listing Interface

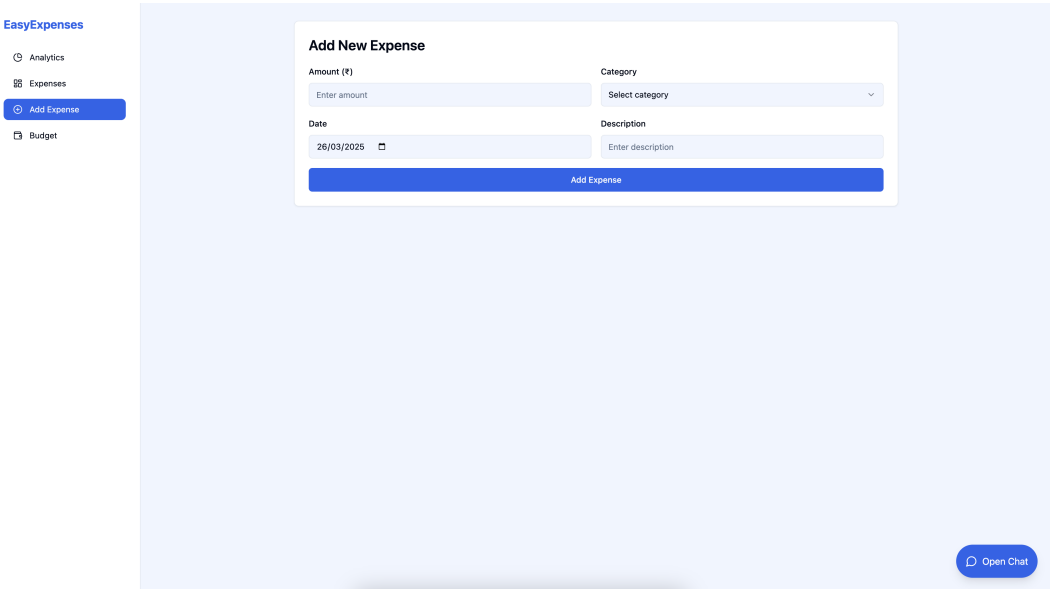

Figure 5.4: Add New Expense Form

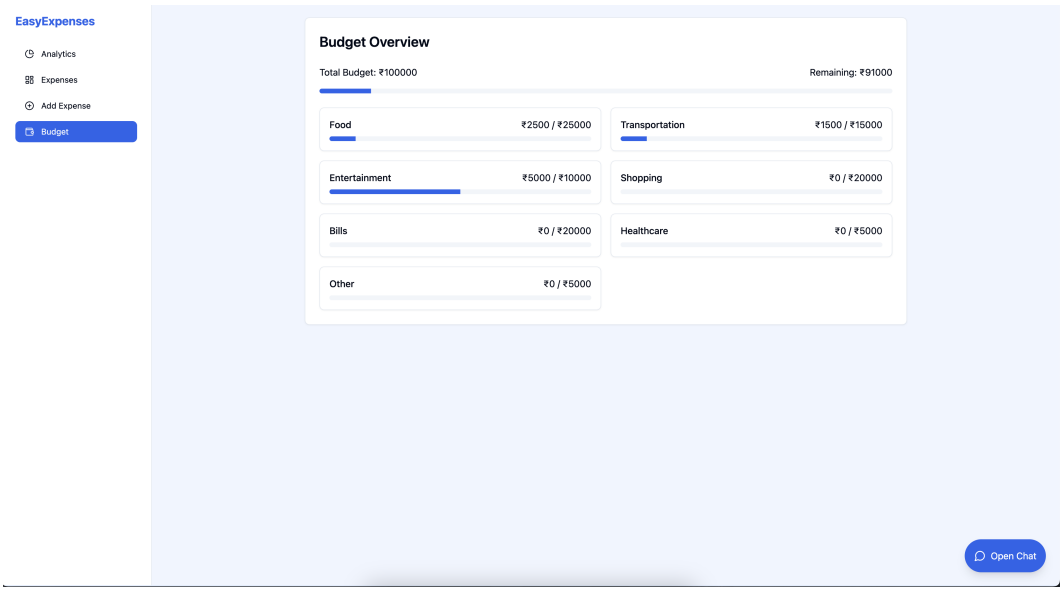

Figure 5.5: Budget Planning and Monitoring Interface

6. Implementation Code

6.1 Folder Structure

```
- app
      - globals.css
      - layout.tsx
      - page.tsx
- components
      - AddExpense.tsx
      - Analytics.tsx
      - BudgetOverview.tsx
      - ExpenseList.tsx
      - footer.tsx
      - ui(form lucide-react)
- components.json
- folder_structure.txt
- hooks
      - use-toast.ts
- lib
      - utils.ts
- next-env.d.ts
- next.config.js
- package-lock.json
- package.json
- postcss.config.js
- tailwind.config.ts
- tsconfig.json

6 directories, 66 files
```

6.2 page.tsx

```
"use client";

import { useState } from "react";
import { Card } from "@/components/ui/card";
import ExpenseList from "@/components/ExpenseList";
import AddExpense from "@/components/AddExpense";
import BudgetOverview from "@/components/BudgetOverview";
import Analytics from "@/components/Analytics";
import { LayoutDashboard, PlusCircle, PieChart, Wallet,
   MessageCircle } from "lucide-react";
import { cn } from "@/lib/utils";

export default function Home() {
  const [expenses, setExpenses] = useState([
    {
      id: 1,
      amount: 2500,
      category: "Food",
      date: "2024-03-20",
      description: "Groceries",
    },
    {
      id: 2,
      amount: 1500,
      category: "Transportation",
      date: "2024-03-19",
      description: "Auto fare",
    },
    {
      id: 3,
      amount: 5000,
      category: "Entertainment",
      date: "2024-03-18",
      description: "Movie night",
    },
  ]);

  const [currentPage, setCurrentPage] = useState("analytics
     ");
  const [showChatbot, setShowChatbot] = useState(false);

  const addExpense = (expense) => {
    setExpenses([...expenses, { ...expense, id: expenses.
       length + 1 }]);
```

```
};

const navigationItems = [
  { id: "analytics", label: "Analytics", icon: PieChart
    },
  { id: "expenses", label: "Expenses", icon:
    LayoutDashboard },
  { id: "add", label: "Add Expense", icon: PlusCircle },
  { id: "budget", label: "Budget", icon: Wallet },
];

const renderContent = () => {
  switch (currentPage) {
    case "analytics":
      return <Analytics expenses={expenses} />;
    case "expenses":
      return <ExpenseList expenses={expenses} />;
    case "add":
      return <AddExpense onAddExpense={addExpense} />;
    case "budget":
      return <BudgetOverview expenses={expenses} />;
    default:
      return <Analytics expenses={expenses} />;
  }
};

return (
  <div className="min-h-screen bg-background flex">
    {/* Sidebar */}
    <div className="w-64 bg-card border-r border-border">
      <div className="p-6">
        <h1 className="text-xl font-bold text-primary mb
          -6">
          EasyExpenses
        </h1>
        <nav className="space-y-2">
          {navigationItems.map((item) => (
            <button
              key={item.id}
              onClick={() => setCurrentPage(item.id)}
              className={cn(
                "flex items-center space-x-3 w-full px-4
                  py-2 rounded-lg text-sm transition-
                  colors",
                currentPage === item.id
                  ? "bg-primary text-primary-foreground"
```

```
                    : "hover:bg-accent hover:text-accent-
                      foreground"
                )}
              >
                <item.icon className="h-4 w-4" />
                <span>{item.label}</span>
              </button>
            ))}
          </nav>
        </div>
      </div>

      {/* Main Content */}
      <div className="flex-1 p-8">
        <div className="max-w-5xl mx-auto">
          <Card className="p-6">{renderContent()}</Card>
        </div>
      </div>

      {/* Chatbot Toggle Button */}
      <button
        onClick={() => setShowChatbot(!showChatbot)}
        className="fixed bottom-6 right-6 bg-primary text-
            white p-4 rounded-full shadow-lg flex items-
            center space-x-2"
      >
        <MessageCircle className="h-5 w-5" />
        <span>{showChatbot ? "Close Chat" : "Open Chat"}</
            span>
      </button>

      {/* Chatbot Frame (Increased Size) */}
      {showChatbot && (
        <div className="fixed bottom-20 right-10 w-[450px]
            h-[600px] border border-gray-300 bg-white shadow
            -lg rounded-lg overflow-hidden">
          <iframe
            src="https://webagent.ai/chatbot/embed/2d33fe77
                -3dcf-4b24-a186-ff3a0f231cf6/classic"
            className="w-full h-full"
          ></iframe>
        </div>
      )}
    </div>
  );
}
```

6.3 AddExpense.tsx

```
    "use client";

import { useState } from "react";
import { Button } from "@/components/ui/button";
import { Input } from "@/components/ui/input";
import { Select, SelectContent, SelectItem, SelectTrigger,
   SelectValue } from "@/components/ui/select";
import { Label } from "@/components/ui/label";

const categories = [
  "Food",
  "Transportation",
  "Entertainment",
  "Shopping",
  "Bills",
  "Healthcare",
  "Other"
];

export default function AddExpense({ onAddExpense }) {
  const [formData, setFormData] = useState({
    amount: "",
    category: "",
    date: new Date().toISOString().split("T")[0],
    description: ""
  });

  const handleSubmit = (e) => {
    e.preventDefault();
    onAddExpense(formData);
    setFormData({
      amount: "",
      category: "",
      date: new Date().toISOString().split("T")[0],
      description: ""
    });
  };

  return (
    <div>
      <h2 className="text-2xl font-semibold mb-4">Add New
          Expense</h2>
      <form onSubmit={handleSubmit} className="space-y-4">
        <div className="grid gap-4 md:grid-cols-2">
```

```
          <div className="space-y-2">
            <Label htmlFor="amount">Amount (    )</Label>
            <Input
              id="amount"
              type="number"
              placeholder="Enter amount"
              value={formData.amount}
              onChange={(e) => setFormData({ ...formData,
                 amount: e.target.value })}
              required
            />
          </div>

          <div className="space-y-2">
            <Label htmlFor="category">Category</Label>
            <Select
              value={formData.category}
              onValueChange={(value) => setFormData({ ...
                 formData, category: value })}
            >
              <SelectTrigger>
                <SelectValue placeholder="Select category"
                   />
              </SelectTrigger>
              <SelectContent>
                {categories.map((category) => (
                  <SelectItem key={category} value={
                     category}>
                    {category}
                  </SelectItem>
                ))}
              </SelectContent>
            </Select>
          </div>

          <div className="space-y-2">
            <Label htmlFor="date">Date</Label>
            <Input
              id="date"
              type="date"
              value={formData.date}
              onChange={(e) => setFormData({ ...formData,
                 date: e.target.value })}
              required
            />
          </div>
```

```
          <div className="space-y-2">
            <Label htmlFor="description">Description</Label
               >
            <Input
              id="description"
              placeholder="Enter description"
              value={formData.description}
              onChange={(e) => setFormData({ ...formData,
                 description: e.target.value })}
              required
            />
          </div>
        </div>

        <Button type="submit" className="w-full">Add
           Expense</Button>
      </form>
    </div>
  );
}
```

6.4 Analytics.tsx

```
"use client";

import { Card } from "@/components/ui/card";
import { PieChart, Pie, Cell, ResponsiveContainer, BarChart
   , Bar, XAxis, YAxis, Tooltip } from "recharts";

const COLORS = ['hsl(var(--chart-1))', 'hsl(var(--chart-2))
   ', 'hsl(var(--chart-3))', 'hsl(var(--chart-4))', 'hsl(
   var(--chart-5))'];

export default function Analytics({ expenses }) {
  // Process data for charts
  const categoryData = expenses.reduce((acc, expense) => {
    acc[expense.category] = (acc[expense.category] || 0) +
       Number(expense.amount);
    return acc;
  }, {});

  const pieData = Object.entries(categoryData).map(([name,
     value]) => ({
```

```
    name,
    value
  }));

  // Process daily spending data
  const dailySpending = expenses.reduce((acc, expense) => {
    acc[expense.date] = (acc[expense.date] || 0) + Number(
        expense.amount);
    return acc;
  }, {});

  const barData = Object.entries(dailySpending).map(([date,
      amount]) => ({
    date,
    amount
  }));

  const formatCurrency = (value) => `$ {value}`;

  return (
    <div className="space-y-8">
      <h2 className="text-2xl font-semibold mb-4">Spending
          Analytics</h2>

      <div className="grid gap-6 md:grid-cols-2">
        <Card className="p-4">
          <h3 className="text-lg font-medium mb-4">Spending
              by Category</h3>
          <div className="h-[300px]">
            <ResponsiveContainer width="100%" height="100%"
                >
              <PieChart>
                <Pie
                  data={pieData}
                  cx="50%"
                  cy="50%"
                  outerRadius={80}
                  fill="#8884d8"
                  dataKey="value"
                  label={({ name, percent }) => `${name} $
                      {(percent * 100).toFixed(0)}%`}
                >
                  {pieData.map((entry, index) => (
                    <Cell key={`cell-${index}`} fill={
                        COLORS[index % COLORS.length]} />
                  ))}
```

```
                </Pie>
                <Tooltip formatter={formatCurrency} />
              </PieChart>
            </ResponsiveContainer>
          </div>
        </Card>

        <Card className="p-4">
          <h3 className="text-lg font-medium mb-4">Daily
              Spending</h3>
          <div className="h-[300px]">
            <ResponsiveContainer width="100%" height="100%"
                >
              <BarChart data={barData}>
                <XAxis dataKey="date" />
                <YAxis tickFormatter={formatCurrency} />
                <Tooltip formatter={formatCurrency} />
                <Bar dataKey="amount" fill="hsl(var(--chart
                    -1))" />
              </BarChart>
            </ResponsiveContainer>
          </div>
        </Card>
      </div>

      <Card className="p-6">
        <h3 className="text-lg font-medium mb-4">Spending
            Insights</h3>
        <div className="space-y-4">
          <div>
            <p className="text-sm text-muted-foreground">
              Total Spent:    {expenses.reduce((sum, exp)
                 => sum + Number(exp.amount), 0)}
            </p>
            <p className="text-sm text-muted-foreground">
              Most Expensive Category: {
                Object.entries(categoryData).reduce((a, b)
                   => a[1] > b[1] ? a : b)[0]
              }
            </p>
            <p className="text-sm text-muted-foreground">
              Number of Transactions: {expenses.length}
            </p>
          </div>
        </div>
      </Card>
```

```
    </div>
  );
}
```

6.5 BudgetOverview.tsx

```
"use client";

import { Progress } from "@/components/ui/progress";
import { Card } from "@/components/ui/card";

export default function BudgetOverview({ expenses }) {
  const totalBudget = 100000;
  const totalExpenses = expenses.reduce((sum, expense) =>
     sum + Number(expense.amount), 0);
  const remainingBudget = totalBudget - totalExpenses;
  const progressPercentage = (totalExpenses / totalBudget)
     * 100;

  const categoryBudgets = {
    Food: 25000,
    Transportation: 15000,
    Entertainment: 10000,
    Shopping: 20000,
    Bills: 20000,
    Healthcare: 5000,
    Other: 5000
  };

  const categoryExpenses = expenses.reduce((acc, expense)
     => {
    acc[expense.category] = (acc[expense.category] || 0) +
       Number(expense.amount);
    return acc;
  }, {});

  return (
    <div className="space-y-6">
      <h2 className="text-2xl font-semibold mb-4">Budget
         Overview</h2>

      <div className="space-y-4">
        <div className="flex justify-between mb-2">
          <span>Total Budget:    {totalBudget}</span>
```

```
          <span>Remaining:     {remainingBudget}</span>
        </div>
        <Progress value={progressPercentage} className="h-2
          " />
      </div>

      <div className="grid gap-4 md:grid-cols-2">
        {Object.entries(categoryBudgets).map(([category,
          budget]) => {
          const spent = categoryExpenses[category] || 0;
          const progress = (spent / budget) * 100;

          return (
            <Card key={category} className="p-4">
              <div className="flex justify-between mb-2">
                <span className="font-medium">{category}</
                  span>
                <span>   {spent} /    {budget}</span>
              </div>
              <Progress value={progress} className="h-2" />
            </Card>
          );
        })}
      </div>
    </div>
  );
}
```

6.6 ExpenseList.tsx

```
    "use client";

import { useState } from "react";
import {
  Table,
  TableBody,
  TableCell,
  TableHead,
  TableHeader,
  TableRow,
} from "@/components/ui/table";
import { ArrowUpDown, Trash2 } from "lucide-react";
import { Button } from "@/components/ui/button";
```

```
export default function ExpenseList({ expenses }) {
  const [sortConfig, setSortConfig] = useState({
    key: null,
    direction: "ascending",
  });

  const sortedExpenses = [...expenses].sort((a, b) => {
    if (!sortConfig.key) return 0;

    const aValue = a[sortConfig.key];
    const bValue = b[sortConfig.key];

    if (sortConfig.key === "amount") {
      return sortConfig.direction === "ascending"
        ? Number(aValue) - Number(bValue)
        : Number(bValue) - Number(aValue);
    }

    if (sortConfig.key === "date") {
      return sortConfig.direction === "ascending"
        ? new Date(aValue).getTime() - new Date(bValue).
          getTime()
        : new Date(bValue).getTime() - new Date(aValue).
          getTime();
    }

    return sortConfig.direction === "ascending"
      ? aValue.localeCompare(bValue)
      : bValue.localeCompare(aValue);
  });

  const requestSort = (key) => {
    let direction = "ascending";
    if (sortConfig.key === key && sortConfig.direction ===
       "ascending") {
      direction = "descending";
    }
    setSortConfig({ key, direction });
  };

  const getSortIcon = (key) => {
    if (sortConfig.key === key) {
      return (
        <ArrowUpDown
          className={`ml-2 h-4 w-4 inline ${
```

```
            sortConfig.direction === "descending" ? "
               transform rotate-180" : ""
          }`}
        />
      );
    }
    return <ArrowUpDown className="ml-2 h-4 w-4 inline
       opacity-20" />;
  };

  return (
    <div>
      <h2 className="text-2xl font-semibold mb-4">Recent
         Expenses</h2>
      <Table>
        <TableHeader>
          <TableRow>
            <TableHead>
              <Button
                variant="ghost"
                onClick={() => requestSort("date")}
                className="hover:bg-transparent p-0 font-
                   medium"
              >
                Date {getSortIcon("date")}
              </Button>
            </TableHead>
            <TableHead>
              <Button
                variant="ghost"
                onClick={() => requestSort("category")}
                className="hover:bg-transparent p-0 font-
                   medium"
              >
                Category {getSortIcon("category")}
              </Button>
            </TableHead>
            <TableHead>
              <Button
                variant="ghost"
                onClick={() => requestSort("description")}
                className="hover:bg-transparent p-0 font-
                   medium"
              >
                Description {getSortIcon("description")}
              </Button>
```

```
            </TableHead>
            <TableHead className="text-right">
              <Button
                variant="ghost"
                onClick={() => requestSort("amount")}
                className="hover:bg-transparent p-0 font-
                    medium ml-auto"
              >
                Amount {getSortIcon("amount")}
              </Button>
            </TableHead>
          </TableRow>
        </TableHeader>
        <TableBody>
          {sortedExpenses.map((expense) => (
            <TableRow key={expense.id}>
              <TableCell>{expense.date}</TableCell>
              <TableCell>{expense.category}</TableCell>
              <TableCell>{expense.description}</TableCell>
              <TableCell className="text-right">  {expense
                  .amount}</TableCell>
              <TableCell>
                <button>
                  <Trash2 color="#f17e70" />
                </button>
              </TableCell>
            </TableRow>
          ))}
        </TableBody>
      </Table>
    </div>
  );
}
```

7. Conclusion

The AI-Powered Expense Tracker application successfully addresses the challenges associated with traditional expense management by leveraging cutting-edge technologies such as Next.js 14 and GPT-4o. By intelligently categorizing expenses and providing insightful financial analytics, the application significantly reduces the time and effort required for expense tracking and analysis.

The integration of GPT-4o enables a seamless user experience where financial insights are automatically generated, helping users make better financial decisions. The application's responsive design ensures accessibility across different devices, allowing users to manage their expenses anytime, anywhere.

Future enhancements could include integration with banking APIs for automatic transaction imports, advanced budget forecasting using predictive AI models, and expanded reporting capabilities for business users. As AI technology continues to evolve, the expense tracker can be further refined to provide even more personalized financial insights and recommendations.