1. TITLE: TRAVEL TOES – A TRAVEL EXPENSE MANAGER

2. ABSTRACT:

"TRAVEL TOES" is a travel expense management application that helps users plan and track their travel budgets with ease. The app features a user-friendly interface with login and signup options, followed by a home page where users can input travel details like destination, total budget, and duration to generate an expense planner. The expense tracker allows users to log daily expenses, categorize them, and visualize their spending through charts. A budget management page helps users monitor their remaining budget, can set limits and track spending progress with a visual progress bar. Additionally, users receive helpful tips for better expense and budget management throughout the application.

3. MODULES WITH EXPLANATION:

1. User Authentication Module:

- **Login:** Allows existing users to log in using their email and password.
- **Signup:** Enables new users to create accounts with their name, email, and password.

2. Expense Tracking Module:

- **Add Expenses:** Allows users to record expenses with date, category, and amount.
- **Edit Expenses:** Enables users to modify existing expenses.
- **Delete Expenses:** Allows users to remove expenses from the record.
- **Total Expenses Calculation:** Displays the total amount spent, aggregated from all expenses.

3. Budget Management Module:

- **Set Budget:** Allows users to create a budget for specific categories with a designated amount.
- **View Budget Breakdown:** Displays a breakdown of the set budget, amount spent, and remaining balance per category.
- **Delete Budget:** Enables users to remove a budget for a particular category.
- **Total Budget Calculation:** Displays the total budget set for all categories.

4. Expense Planner Module:

- **Create Expense Planner:** Enables users to input travel information (country, state, city, places, total budget, and days) and generate an expense planner.
- **Calculate Per-Day Budget:** Automatically calculates the per-day budget based on total budget and duration.
- **Distribute Budget:** Distributes the budget across categories (transport, food, and activities).

5. Downloadable Reports Module:

- Generates downloadable expense reports in PDF form with detailed expense breakdowns for each city and place, including individual budget amounts for transportation, food, and activities.

6. Visualization Module:

- **Line Chart:** Displays daily spending over time to visualize spending patterns.
- **Pie Chart:** Breaks down expenses by category, showing the proportion of spending in each area.

7. Tips Module:

- Offers tips for better expense and budget management based on user behavior and financial status throughout the app.

8. Budget Alert Module:

- Turns the budget amount red as users get close to their limit, giving a clear, easy reminder to help manage spending.

4. SOFTWARES USED:

1. Frontend:

- **React.js**: Used for building a dynamic and efficient user interface, enabling responsive

and interactive components.

- **HTML:** Provides the foundational structure of web pages and content.
- **CSS:** Styles the application to ensure a visually appealing, consistent, and responsive design.
- **React Router**: Manages in-app routing, allowing for seamless navigation between different pages and components within the application.

2. Backend:

- **Node.js**: A runtime environment for executing JavaScript on the server, enabling server-side operations and communication with the frontend.
- **Express.js:** A lightweight, modular framework that simplifies backend development by handling routing, middleware, and HTTP request handling.
- **JSON Web Tokens (JWT)**: Used for securely managing user authentication and session management, ensuring that data remains private and protected.

3. Database:

- **MongoDB:** A NoSQL database that stores user and trip data in a flexible, document-based format, providing scalability and ease of management for dynamic data.
- **Mongoose**: An ODM (Object Data Modeling) library that simplifies MongoDB interactions by providing a structured schema for handling data in Node.js.

4. Charting Libraries:

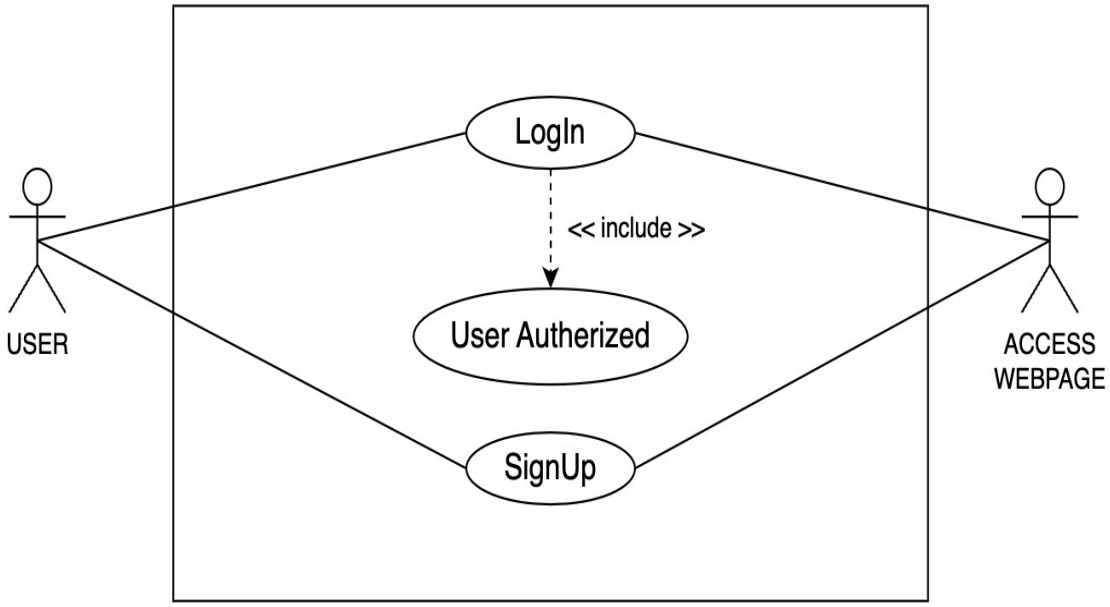
- **Chart.js:** Utilized for creating data visualizations like line charts and pie charts, presenting spending trends and category breakdowns in an easily interpretable format.

5. Others:

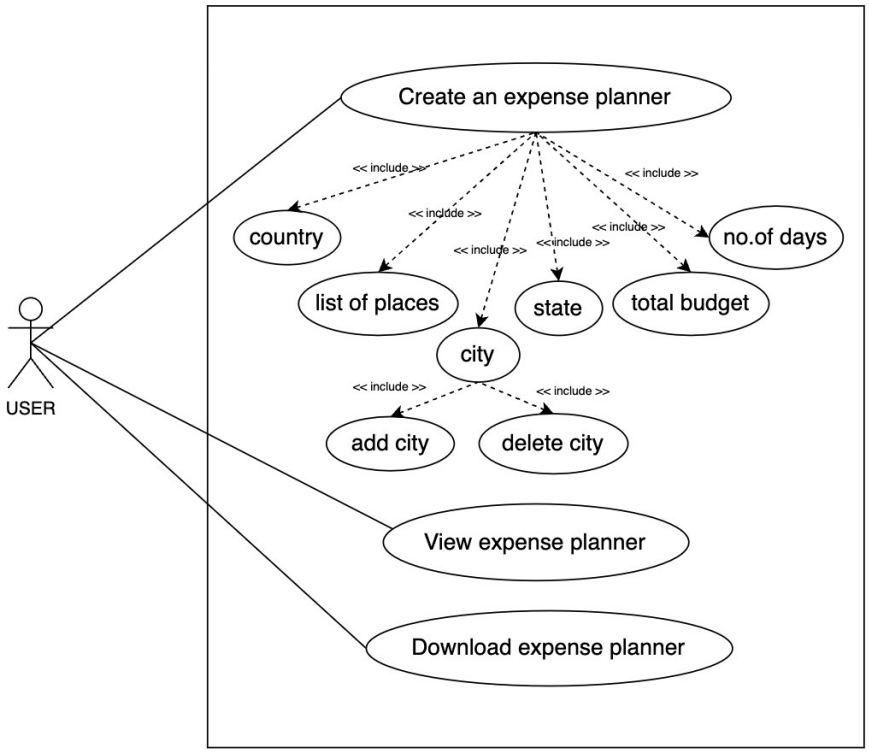
- **Git:** A version control system for managing and tracking code changes, enhancing collaboration and development history.
- **jsPDF:** A JavaScript library used for generating PDF reports, enabling users to export spending summaries and other data as PDF documents for easy sharing and record-keeping.

5. USE CASE DIAGRAM:

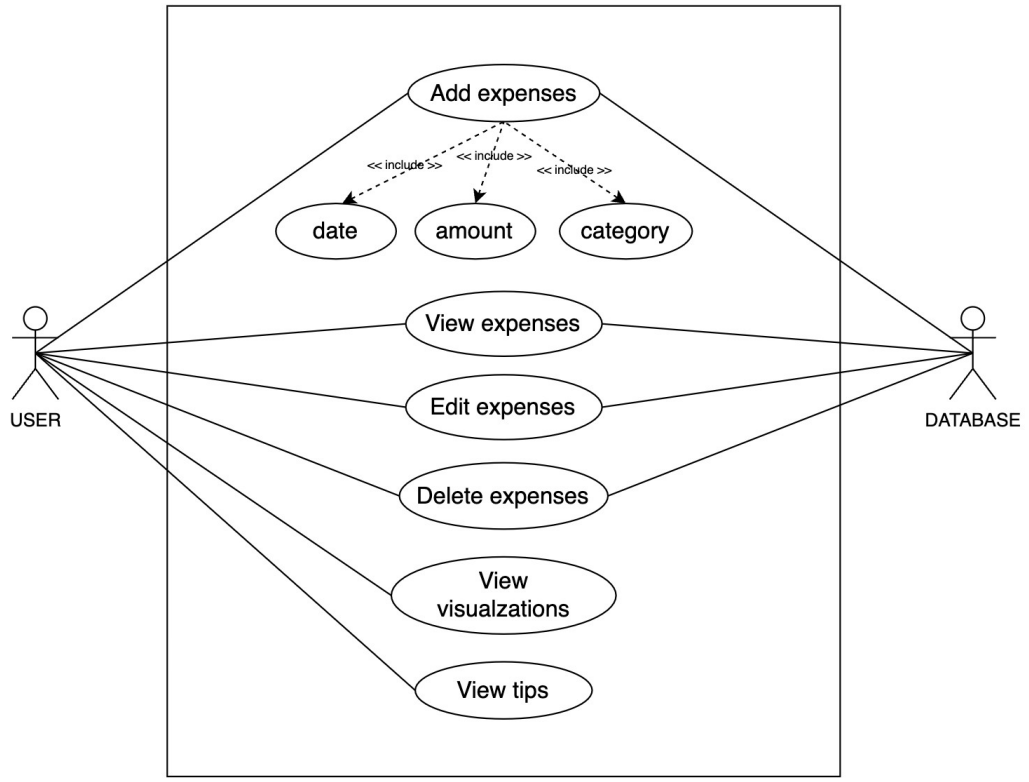
User Authentication:



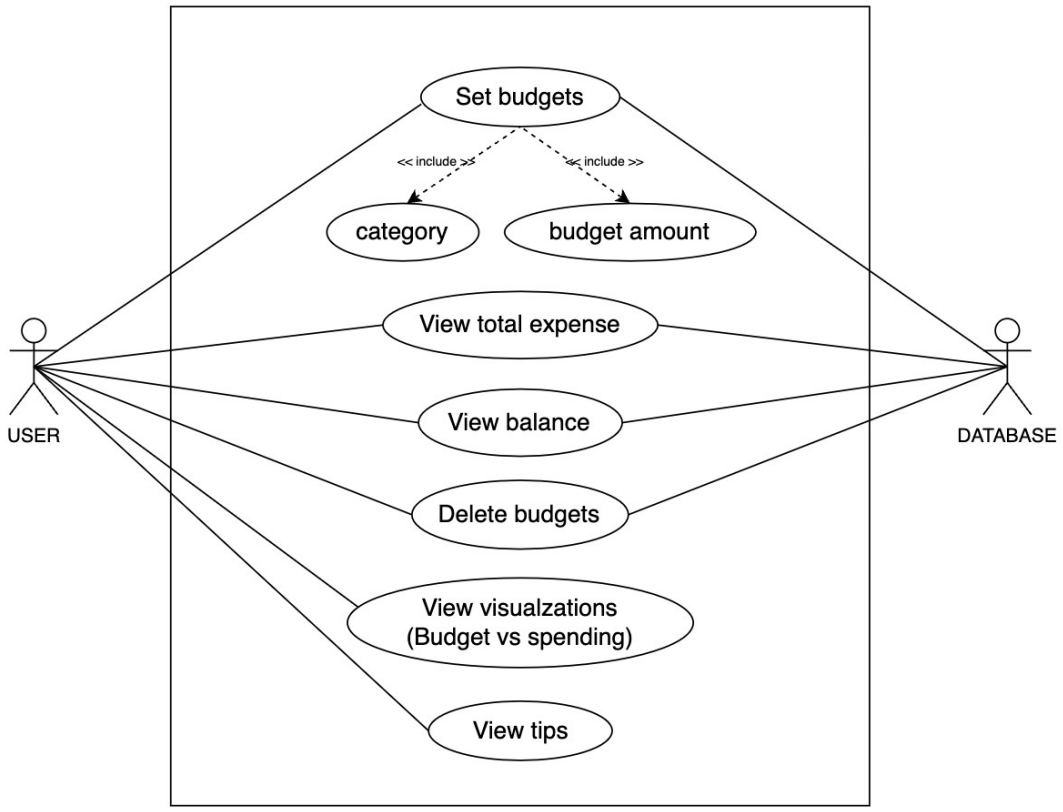
Expense Planner:



Expense Tracker:



Budget Management:



6. SCHEMA:

User.js:

```
const mongoose = require('mongoose');
const userSchema = new mongoose.Schema({
  fullName: { type: String, required: true,},
  email: { type: String,  required: true, unique: true, },
  password: { type: String, required: true, },
});
module.exports = mongoose.model('User', userSchema);
```

Field	Type	Description
`_id`	ObjectId	Unique identifier for each user document.
`fullName`	String	Full name of the user.
`email`	String	Email address of the user.
`password`	String (Hashed)	Hashed password of the user.
`__v`	Number (default 0)	Version key used by MongoDB for internal purposes.

Expense.js:

```
const mongoose = require('mongoose');
const expenseSchema = new mongoose.Schema({
  userId: {  type: mongoose.Schema.Types.ObjectId,  ref: 'User',  required: true },
  date: {  type: String,  required: true },
  category: {  type: String,  required: true },
  amount: { type: Number, required: true }
});
module.exports = mongoose.model('Expense', expenseSchema);
```

Field	Type	Description
`_id`	ObjectId	Unique identifier for each expense document.
`userId`	ObjectId	Reference to the user who added the expense.
`date`	Date	Date when the expense was made.
`category`	String	Category of the expense (e.g., "Entertainment", "Shopping").
`amount`	Number	Amount of the expense.
`__v`	Number (default 0)	Version key used by MongoDB for internal purposes.

Budget.js:

```
const mongoose = require('mongoose');
const budgetSchema = new mongoose.Schema({
  userId: { type: mongoose.Schema.Types.ObjectId, ref: 'User', required: true },
  category: { type: String, required: true },
  amount: { type: Number, required: true  }
});
budgetSchema.index({ userId: 1, category: 1 }, { unique: true });
module.exports = mongoose.model('Budget', budgetSchema);
```

Field	Type	Description
`_id`	ObjectId	Unique identifier for each budget document.
`userId`	ObjectId	Reference to the user who created the budget.
`category`	String	Category for which the budget is allocated (e.g., "Food", "Shopping").
`amount`	Number	Budgeted amount for the respective category.
`__v`	Number (default 0)	Version key used by MongoDB for internal purposes.

7. SOURCE CODE:

backend/.env:

```
MONGODB_URI=mongodb://localhost:27017/travel-toes
JWT_SECRET=your_jwt_secret
```

backend/middleware/authMiddleware.js:

```
const jwt = require('jsonwebtoken');

const authMiddleware = (req, res, next) => {
  console.log('Auth middleware processing request:', {
    path: req.path,
    method: req.method,
    hasAuthHeader: !!req.headers.authorization
  });

  const token = req.headers.authorization && req.headers.authorization.split(' ')[1];
```

```
  if (!token) {
    console.log('No token provided');
    return res.status(401).json({ message: 'No token, authorization denied' });
  }

  try {
    const decoded = jwt.verify(token, process.env.JWT_SECRET);
    console.log('Token decoded successfully:', {
      userId: decoded.id,
      exp: new Date(decoded.exp * 1000)
    });

    req.userId = decoded.id;
    next();
  } catch (error) {
    console.error('Token verification failed:', error);
    res.status(401).json({ message: 'Token is not valid', error: error.message });
  }
};

module.exports = authMiddleware;
```

backend/server.js:

```
const express = require('express');
const cors = require('cors');
const bodyParser = require('body-parser');
const mongoose = require('mongoose');
const authRoutes = require('./routes/auth');
const expenseRoutes = require('./routes/expenses');
const budgetRoutes = require('./routes/budgets');
require('dotenv').config(); // Load environment variables

const app = express();

// Middleware
app.use(cors()); // Enable CORS for all routes
app.use(bodyParser.json()); // Parse JSON request bodies
```

```
// Database connection
mongoose.connect(process.env.MONGODB_URI)
  .then(() => {
    console.log('Connected to MongoDB');
  })
  .catch((err) => {
    console.error('Error connecting to MongoDB:', err);
  });

// Routes
app.use('/api/auth', authRoutes);
app.use('/api/expenses', expenseRoutes);
app.use('/api/budgets', budgetRoutes);

// Server
const PORT = process.env.PORT || 5000;
app.listen(PORT, () => {
  console.log(`Server is running on http://localhost:${PORT}`);
});
```

frontend/src/App.js:

```
import React from 'react';
import { BrowserRouter as Router, Route, Routes, Navigate } from 'react-router-dom';
import Header from './components/Header';
import Footer from './components/Footer';
import Login from './pages/Login';
import Signup from './pages/Signup';
import Home from './pages/Home';
import ExpenseTracker from './pages/ExpenseTracker'; // Import the ExpenseTracker page
import BudgetManagement from './pages/BudgetManagement';
import './styles.css';

function App() {
  // Function to check if the user is authenticated
  const isAuthenticated = () => {
    return !!localStorage.getItem('token'); // If token exists, return true
  };

  // Private route component to protect certain routes
```

```
  const PrivateRoute = ({ element: Component }) => {
    return isAuthenticated() ? <Component /> : <Navigate to="/" />;
  };

  return (
    <Router>
      <Header />
      <div className="container mt-4">
        <Routes>
          {/* Public routes */}
          <Route path="/" element={<Login />} />
          <Route path="/signup" element={<Signup />} />
          {/* Protected routes */}
          <Route path="/home" element={<PrivateRoute element={Home} />} />
          <Route path="/expense-tracker" element={<PrivateRoute element={ExpenseTracker}
/>} />  {/* Add ExpenseTracker */}
          <Route path="/budget-management" element={<PrivateRoute
element={BudgetManagement} />} />
        </Routes>
      </div>
      <Footer />
    </Router>
  );
}

export default App;
```

frontend/src/pages/Home.js:

```
import React, { useState } from 'react';
import { Link } from 'react-router-dom';
import jsPDF from 'jspdf';
import 'jspdf-autotable';

const Home = () => {
  const [country, setCountry] = useState('');
  const [state, setState] = useState('');
  const [cities, setCities] = useState([{ city: '', places: '', totalBudget: '', days: '' }]);
  const [expensePlan, setExpensePlan] = useState(null);
  const budgetPercentages = {  transport: 0.1, food: 0.3, activities: 0.6, };
```

```
  const handleAddCity = () =>{setCities([...cities, { city: '', places: '', totalBudget: '', days: '' }]);};
  const handleCityChange = (index, event) => {
    const newCities = [...cities];   newCities[index][event.target.name] = event.target.value;
    setCities(newCities);
  };
  const handleDeleteCity = (index) => {
    const newCities = cities.filter((_, i) => i !== index);  setCities(newCities);
  };
  const handleGeneratePlanner = () => {
    const plan = cities.map((cityObj) => {
      const totalBudget = parseFloat(cityObj.totalBudget || 0);
      const numberOfDays = parseInt(cityObj.days || 1, 10);
      const perDayBudget = (totalBudget / numberOfDays).toFixed(2);
      const transportBudget = (perDayBudget * budgetPercentages.transport).toFixed(2);
      const foodBudget = (perDayBudget * budgetPercentages.food).toFixed(2);
      const activitiesBudget = (perDayBudget * budgetPercentages.activities).toFixed(2);
      const placesArray = cityObj.places.split(',').map((place) => place.trim());
      const perPlaceBudget = (totalBudget / placesArray.length).toFixed(2);
      const placesBudgetDetails = placesArray.map((place) => ({
        name: place, totalPlaceBudget: perPlaceBudget,
        transportBudget: (perPlaceBudget * budgetPercentages.transport).toFixed(2),
        foodBudget: (perPlaceBudget * budgetPercentages.food).toFixed(2),
        activitiesBudget: (perPlaceBudget * budgetPercentages.activities).toFixed(2),
      }));

      return {
        city: cityObj.city, totalBudget: totalBudget.toFixed(2),  days: numberOfDays,
        transportBudget, foodBudget,  activitiesBudget, places: placesBudgetDetails,  };
    });  setExpensePlan(plan);
  };
  const handleDownloadPDF = () => {
    const doc = new jsPDF(); doc.setFont('helvetica', 'normal'); doc.setFontSize(14);
    doc.text('Expense Planner', 14, 20);  let yPosition = 30;
    if (expensePlan) {
      expensePlan.forEach((cityObj) => {
        doc.setFontSize(12);  doc.text(`${cityObj.city} - Total Budget: ₹${cityObj.totalBudget} for
${cityObj.days} Days`, 14, yPosition);  yPosition += 10;
        const tableData = cityObj.places.map((place) => [
          place.name, `₹${place.totalPlaceBudget}`,  `₹${place.transportBudget}`,
          `₹${place.foodBudget}`, `₹${place.activitiesBudget}`,   ]);
        doc.autoTable({
```

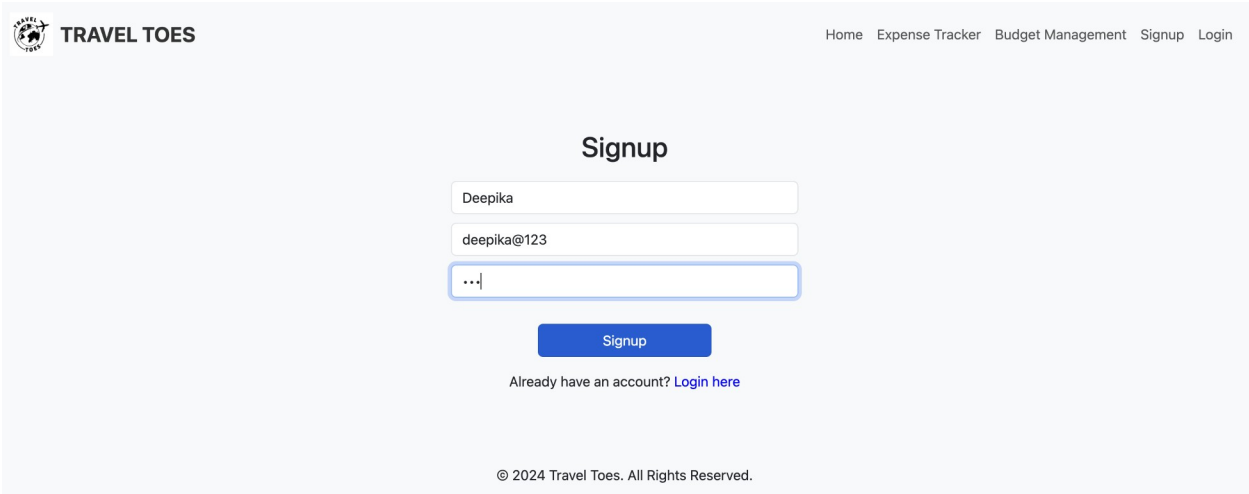
```
        head: [['Place Name', 'Total Budget', 'Transport', 'Food', 'Activities']], body: tableData,
        startY: yPosition, theme: 'grid', styles: { ... },   headStyles: { … },  columnStyles: { … },
      });
      yPosition = doc.lastAutoTable.finalY + 10;   });  }  doc.save('expense_planner.pdf');
  };

  return (
    <div className="container-fluid p-4">
      <h1 className="text-center mb-4">Welcome to Travel Toes</h1>
      <p className="text-center mb-5">Your ultimate expense tracker and budget mgnt tool.</p>
      <div className="row justify-content-center">
        <div className="col-md-8"> <div className="card shadow-sm">
          <div className="card-body">  <div className="input-form">
            <div className="row"><div className="col-md-6 mb-3">
              <input type="text" placeholder="Country" value={country}
                onChange={(e) => setCountry(e.target.value)} className="form-control" />
             </div>
             <div className="col-md-6 mb-3">
              <input type="text" placeholder="State" value={state}
                onChange={(e) => setState(e.target.value)} className="form-control"  /> </div>
            </div>
            {cities.map((cityObj, index) => (
             <div key={index} className="city-input mb-3">
              <div className="row"> <div className="col-md-6 mb-2">
                <input  type="text" name="city"  placeholder="City" value={cityObj.city}
                  onChange={(e) => handleCityChange(index, e)}className="form-control" />
               </div> <div className="col-md-6 mb-2">
                <input type="text" name="places"
                  placeholder="List of Places (comma-separated)" value={cityObj.places}
                  onChange={(e) => handleCityChange(index, e)} className="form-control" />
               </div> <div className="col-md-6 mb-2">
                <input type="number" name="totalBudget" placeholder="Total Budget"
                  value={cityObj.totalBudget} onChange={(e) => handleCityChange(index, e)}
                  className="form-control" />
               </div> <div className="col-md-6 mb-2">
                <input type="number" name="days" placeholder="Number of Days"
                  value={cityObj.days} onChange={(e) => handleCityChange(index, e)}
                  className="form-control"   />
               </div> {index > 0 && (  <div className="col-md-12 mb-2">
                  <button className="btn btn-danger" onClick={() =>
handleDeleteCity(index)}>  Delete  </button>  </div>
```

```
                )} </div> </div>
            ))}
            <div className="d-flex justify-content-between mt-3">
              <button className="btn btn-outline-primary" onClick={handleAddCity}>
                Add Another City </button>
              <button className="btn btn-primary" onClick={handleGeneratePlanner}>
                Generate Expense Planner </button>
            </div></div>  </div> </div> </div>
    </div>
    {expensePlan && (  <div className="row justify-content-center mt-5">
       <div className="col-md-8">  <div className="card shadow-sm">
          <div className="card-body">
            <h3 className="card-title text-center mb-4">Expense Planner</h3>
            {expensePlan.map((cityObj, index) => (
              <div key={index} className="city-expense mb-4 text-center">
                <h4>{cityObj.city} - Total Budget: ₹{cityObj.totalBudget} for {cityObj.days}
Days</h4>
                {cityObj.places.map((place, i) => (
                  <div key={i} className="place-expense mb-3">
                    <h5>{place.name} - Total Budget for Place: ₹{place.totalPlaceBudget}</h5><p>
                      Transport: ₹{place.transportBudget} | Food: ₹{place.foodBudget} | Activities: ₹
{place.activitiesBudget}
                    </p>  </div>    ))}   </div> ))}
            <div className="text-center mt-4">
              <button className="btn btn-success" onClick={handleDownloadPDF}>
                Download PDF </button> </div> </div>  </div> </div>
      </div>
    )}
    <div className="mt-5">
      <div className="card shadow-sm"> <div className="card-body">
         <h5 className="text-center mb-3">To explore more, click on the below buttons:</h5>
         <div className="d-flex justify-content-center">
           <Link to="/expense-tracker" className="btn btn-primary me-3">
             Go to Expense Tracker </Link>
           <Link to="/budget-management" className="btn btn-secondary">
             Go to Budget Management </Link> </div> </div> </div>  </div>
   </div>
  );
};
export default Home;
```

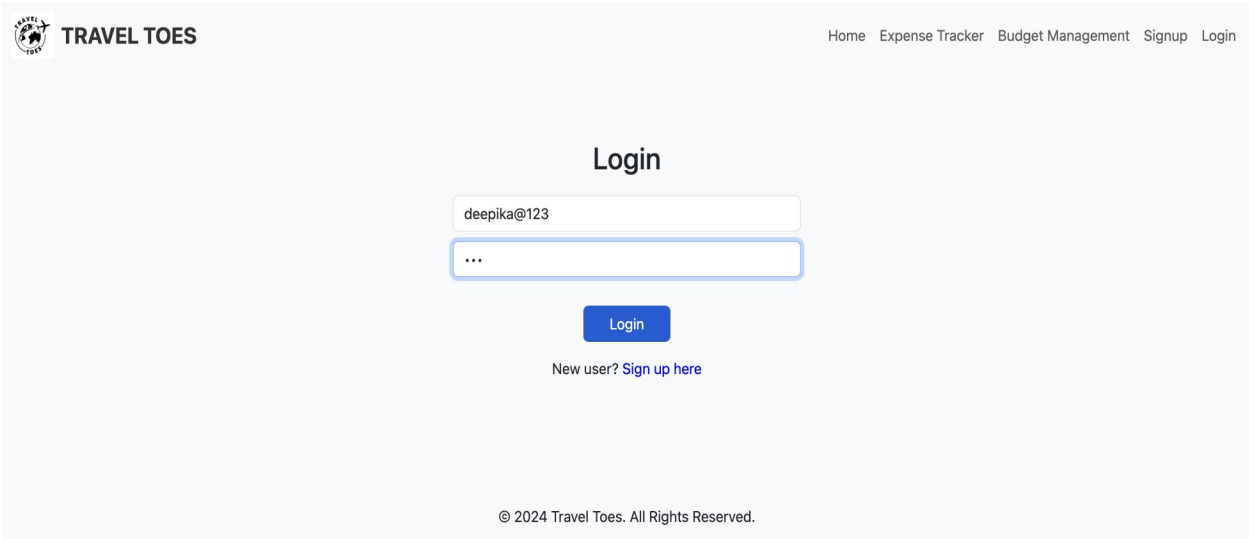
8. GUI SCREENSHOTS:

SignUp Page:

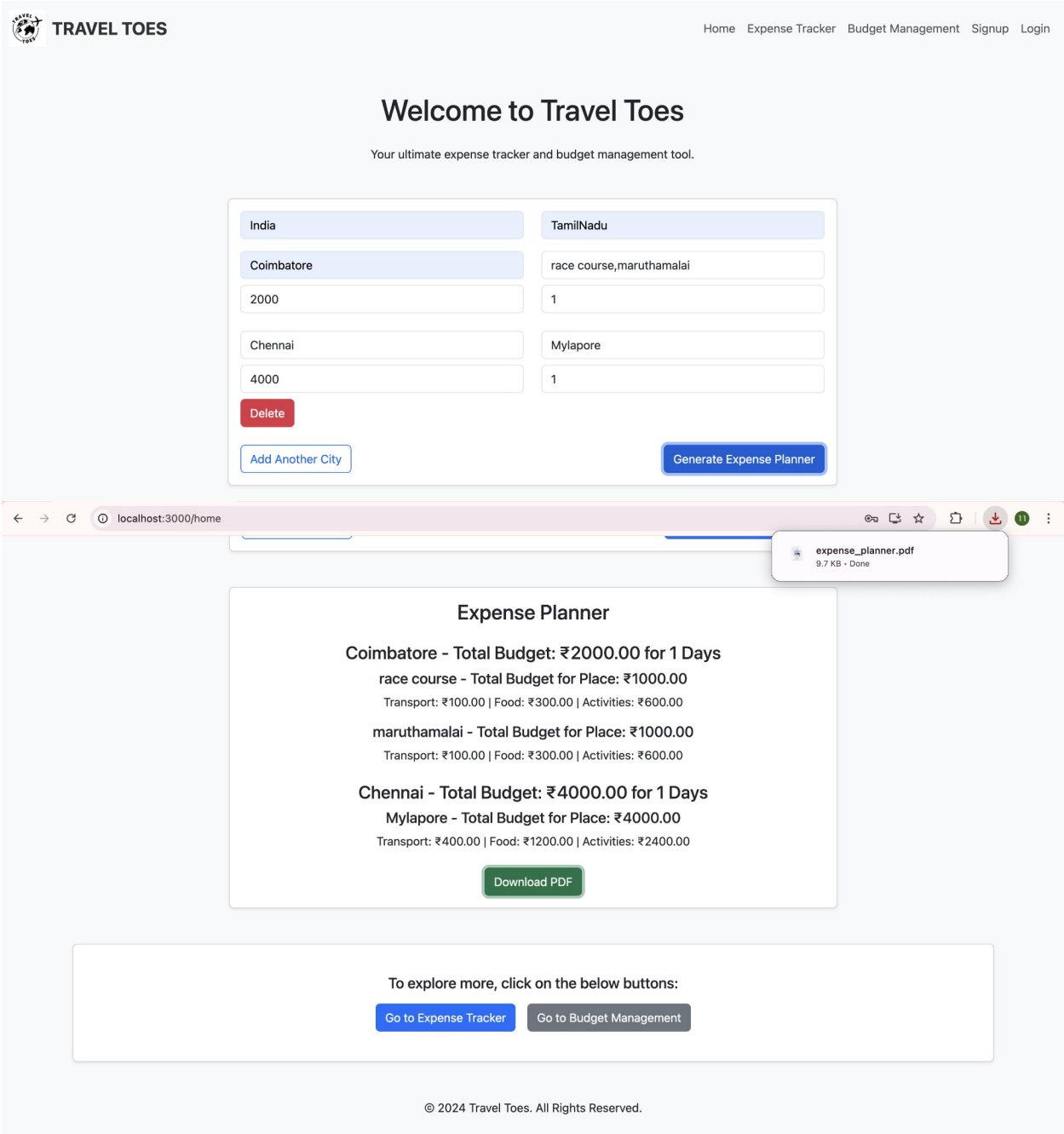



```
_id: ObjectId('671fdd3d5e56408be50f0914')
fullName : "Deepika"
email : "deepika@123"
password : "$2a$12$R03qSmbeN7WrgIw46EhwAuOWsR0OhP4mzNNagXPXLp1OlYsZjPnoS"
__v : 0
```

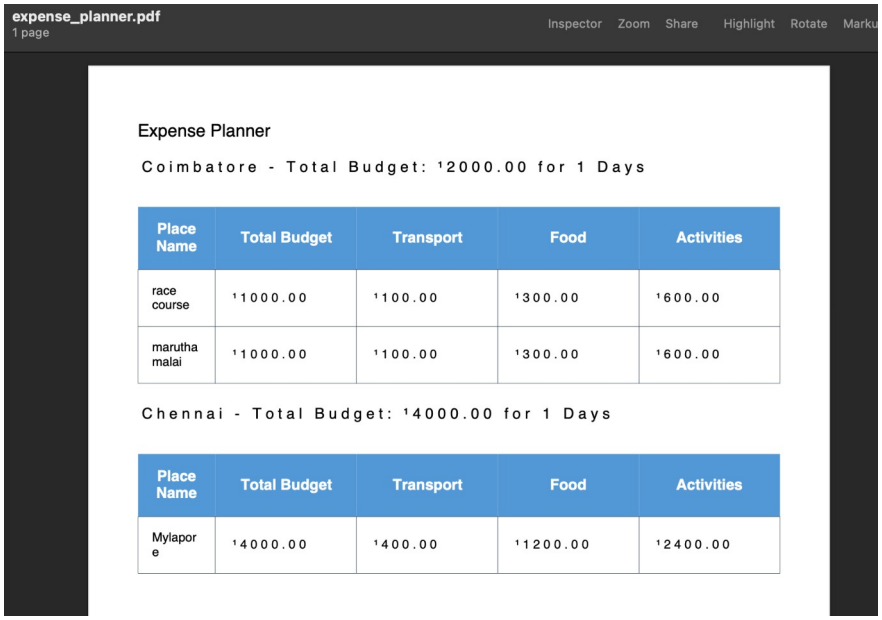

Login Page:



Home Page:



Downloaded pdf :



expense_planner.pdf
1 page

Inspector Zoom Share Highlight Rotate Marku

Expense Planner

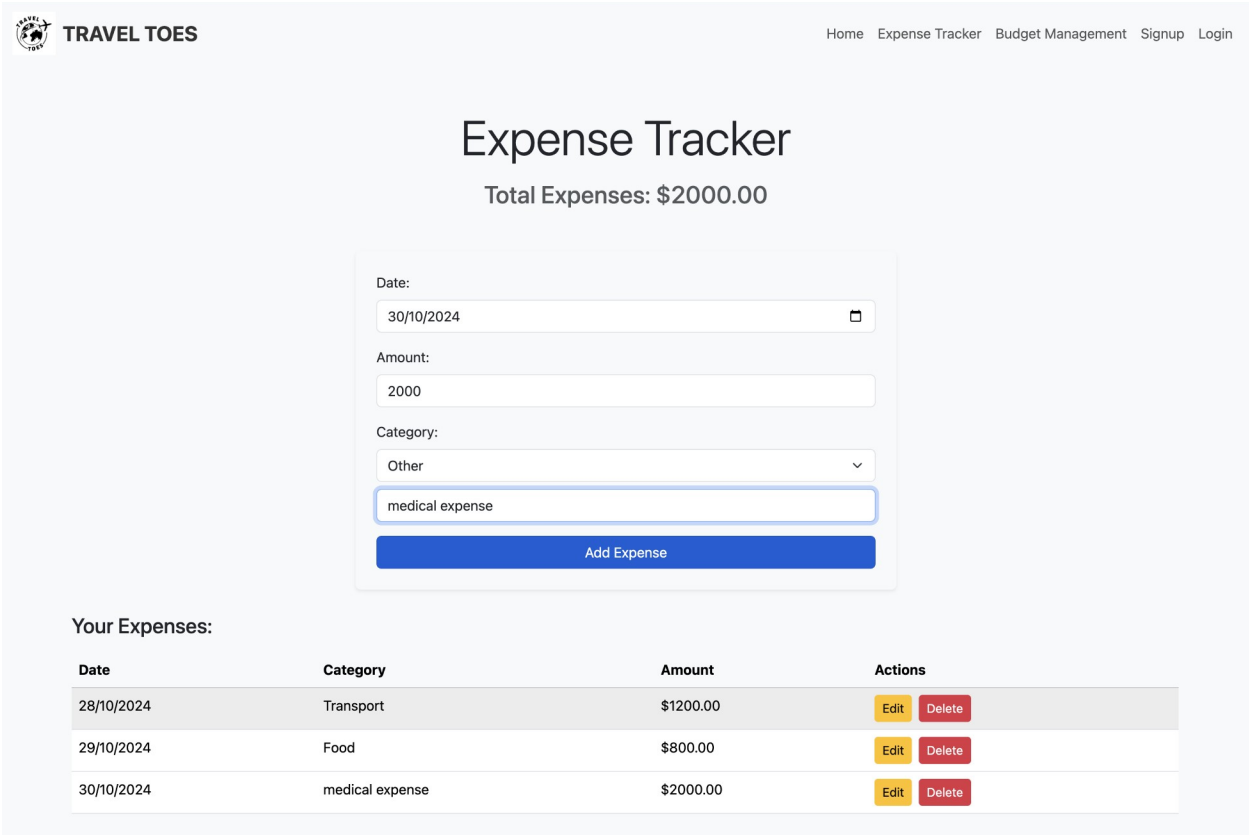
Coimbatore - Total Budget: ¹2000.00 for 1 Days

Place Name	Total Budget	Transport	Food	Activities
race course	¹1000.00	¹100.00	¹300.00	¹600.00
marutha malai	¹1000.00	¹100.00	¹300.00	¹600.00

Chennai - Total Budget: ¹4000.00 for 1 Days

Place Name	Total Budget	Transport	Food	Activities
Mylapore	¹4000.00	¹400.00	¹1200.00	¹2400.00

Expense Tracker Page:



TRAVEL TOES

Home Expense Tracker Budget Management Signup Login

Expense Tracker

Total Expenses: $2000.00

Date:
30/10/2024

Amount:
2000

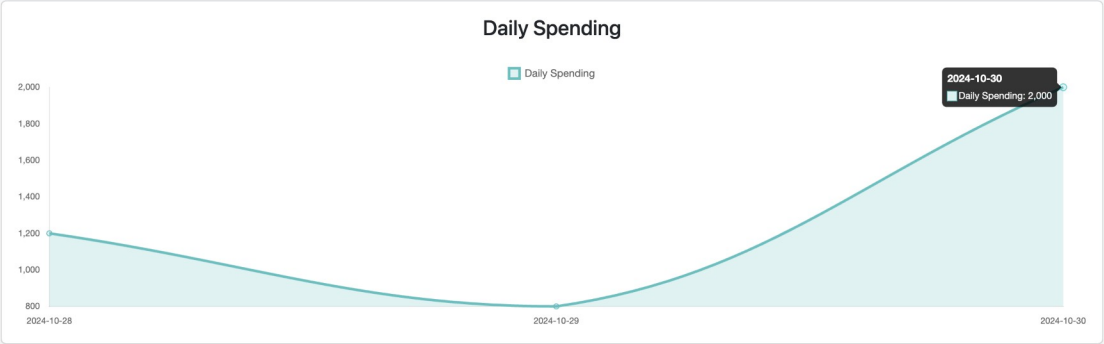
Category:
Other

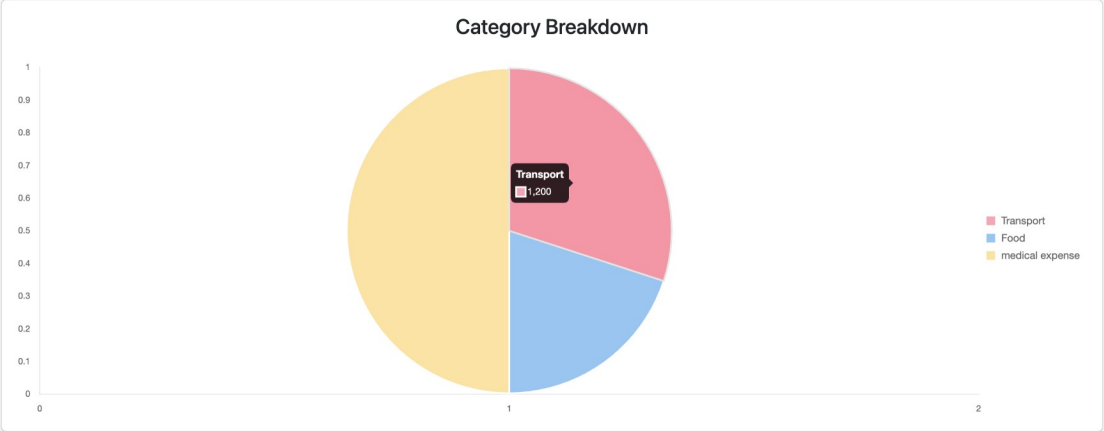
medical expense

Add Expense

Your Expenses:

Date	Category	Amount	Actions
28/10/2024	Transport	$1200.00	Edit Delete
29/10/2024	Food	$800.00	Edit Delete
30/10/2024	medical expense	$2000.00	Edit Delete





Tips for Better Expense Management

Track your spending regularly to stay aware of your financial situation.

Set a budget for different categories and stick to it.

Review your expenses at the end of the month to identify areas for improvement.

Use cash for discretionary spending to help limit overspending.

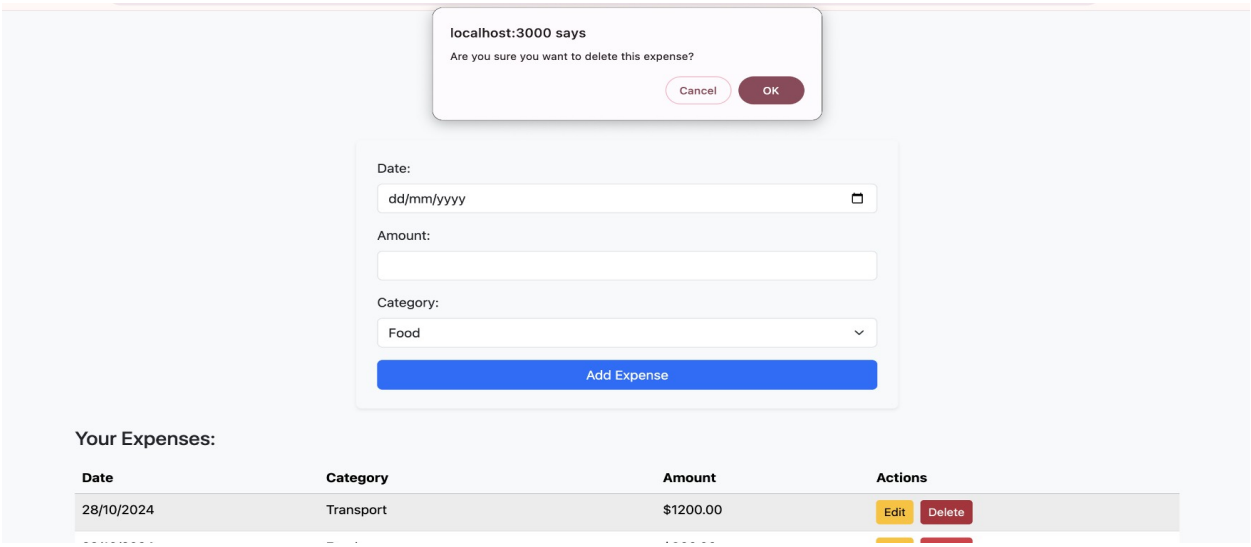
Consider creating an emergency fund to handle unexpected expenses.

Prioritize needs over wants when making purchases to avoid impulsive buying.

To explore more, navigate to budget page:

Go to Budget Management

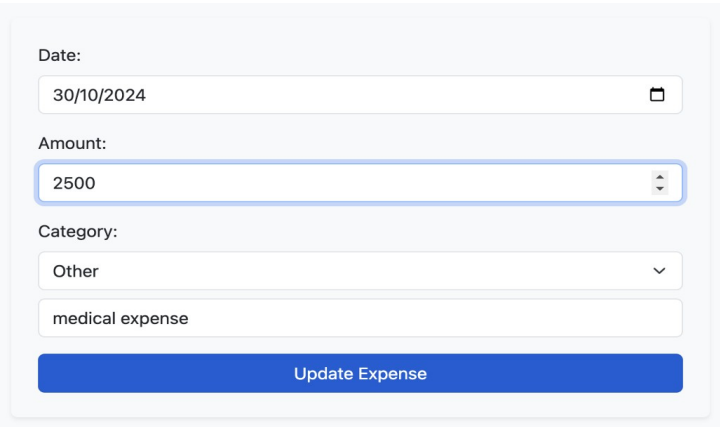
Alert message while deleting expense:




```
_id: ObjectId('671fdfc75e56408be50f092a')
userId : ObjectId('671fdd3d5e56408be50f0914')
date : "2024-10-29"
category : "Food"
amount : 800
__v : 0
```

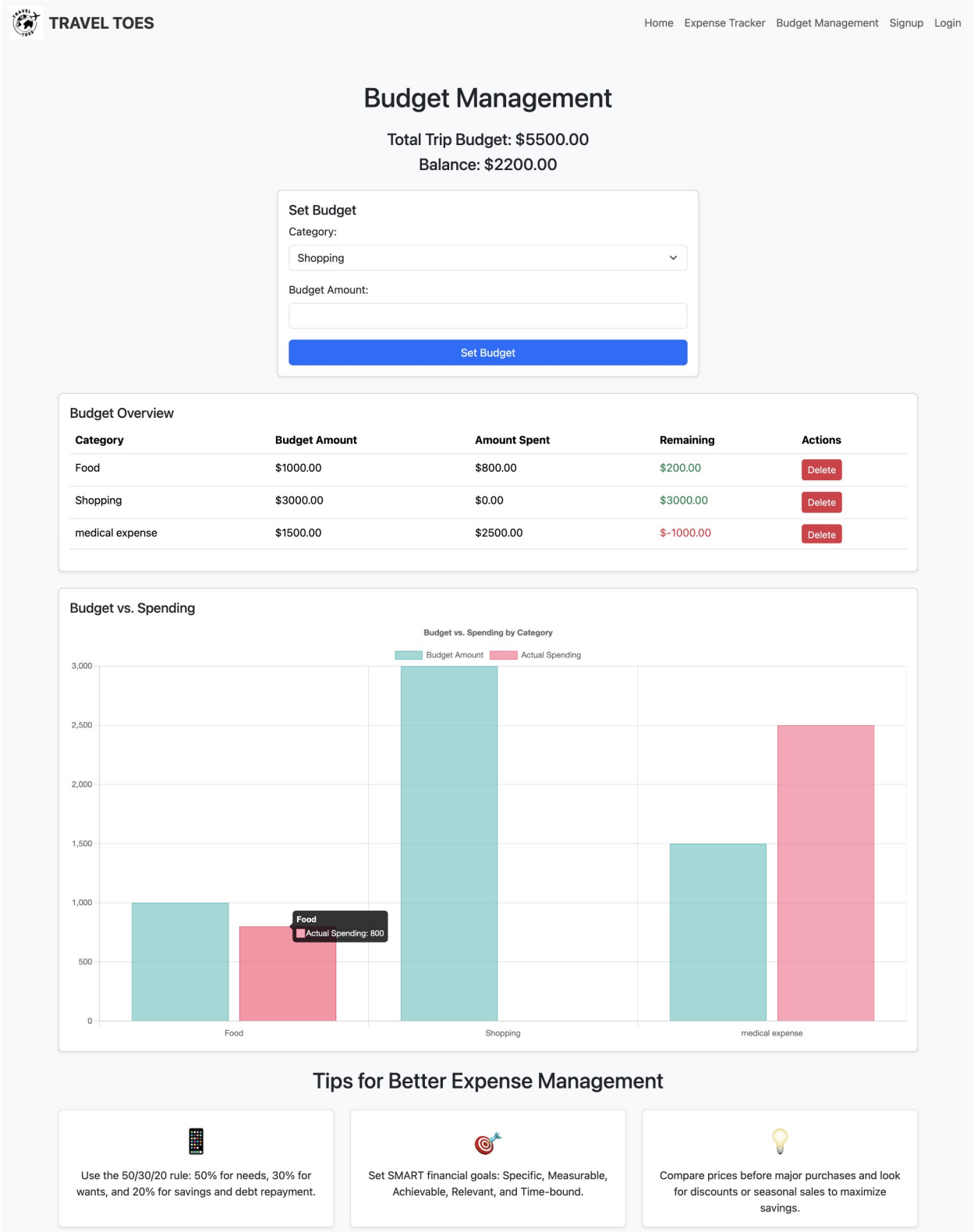
```
_id: ObjectId('671fdff75e56408be50f092d')
userId : ObjectId('671fdd3d5e56408be50f0914')
date : "2024-10-30"
category : "medical expense"
amount : 2000
__v : 0
```

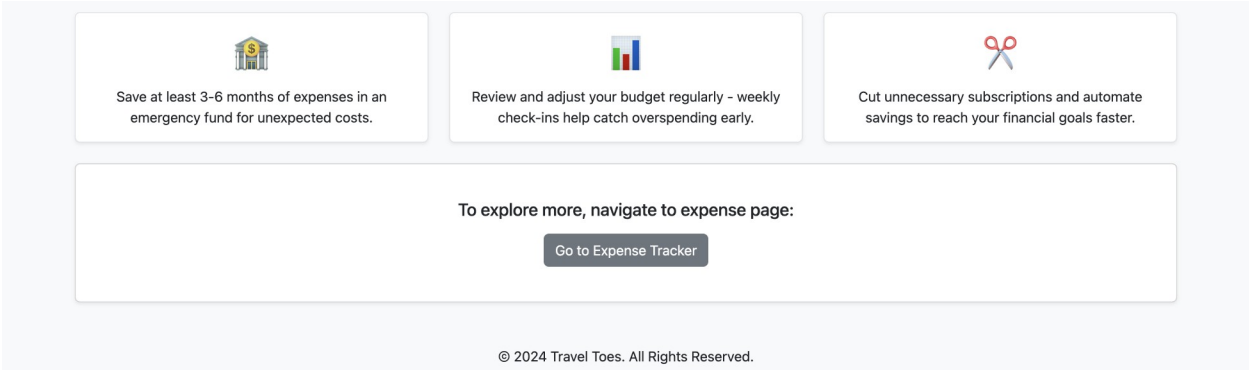
Editing expense:




```
_id: ObjectId('671fdff75e56408be50f092d')
userId : ObjectId('671fdd3d5e56408be50f0914')
date : "2024-10-30"
category : "medical expense"
amount : 2500
__v : 0
```

Budget Management Page:





Shopping deleted :

```
_id: ObjectId('671fe2545e56408be50f093d')
userId : ObjectId('671fdd3d5e56408be50f0914')
category : "Food"
amount : 1000
__v : 0
```

```
_id: ObjectId('671fe2695e56408be50f0940')
userId : ObjectId('671fdd3d5e56408be50f0914')
category : "medical expense"
amount : 1500
__v : 0
```

9. CONCLUSION:

- "Travel Toes" successfully simplifies travel expense management with features like budget planning, real-time tracking, and detailed reports.
- Its user-friendly interface and integration of modern tech make it a valuable tool for travelers.
- Future enhancements could add advanced analytics and multi-currency support to broaden its appeal.
