# Technical Documentation for Personal Expense Tracker

1. **Project Structure**
   * **Frontend**: Built with React and TypeScript for component-based architecture, ensuring modular and reusable UI elements.
   * **Components**: Key components include:
     + ExpenseForm: Handles user input for logging expenses.
     + ExpenseChart: Renders monthly spending data visually.
     + ExpenseSummary: Provides a summary of expenses, including totals and averages.
   * **State Management**: Uses React hooks for managing component states, ensuring efficient updates and data flow.
2. **Data Management**
   * **Local Storage**: Expense data is stored in the browser’s localStorage, allowing persistence across sessions. Data is stored in JSON format, making it easy to parse and manipulate.
   * **Data Validation**: Input validation is implemented within ExpenseForm, enforcing correct data types and required fields for expense entries.
3. **Key Libraries and Tools**
   * **React**: Utilized for building the user interface.
   * **TypeScript**: Provides static typing, enhancing code safety and readability.
   * **Recharts**: A library for data visualization, used to display bar charts and other visual elements in ExpenseChart.
4. **Data Flow**
   * **Expense Entry**: Users add expenses via ExpenseForm, which updates the main application state and saves data in localStorage.
   * **Data Visualization**: Data from localStorage is read and visualized in ExpenseChart to show spending trends.
   * **Summary Calculation**: ExpenseSummary calculates total and average expenses using stored data, displaying it in the user interface.
5. **Deployment and Development**
   * **Development Setup**: Clone the repository, run npm install, then start the development server with npm start.
   * **Future Enhancements**: Plans include transitioning data storage from local storage to a backend database for scalability.
6. **Best Practices**
   * **Code Modularity**: Each component is designed with a single responsibility, making the code easy to maintain.
   * **Error Handling**: Minimal error handling currently implemented. Future improvements should include error boundaries and logging for better user feedback.