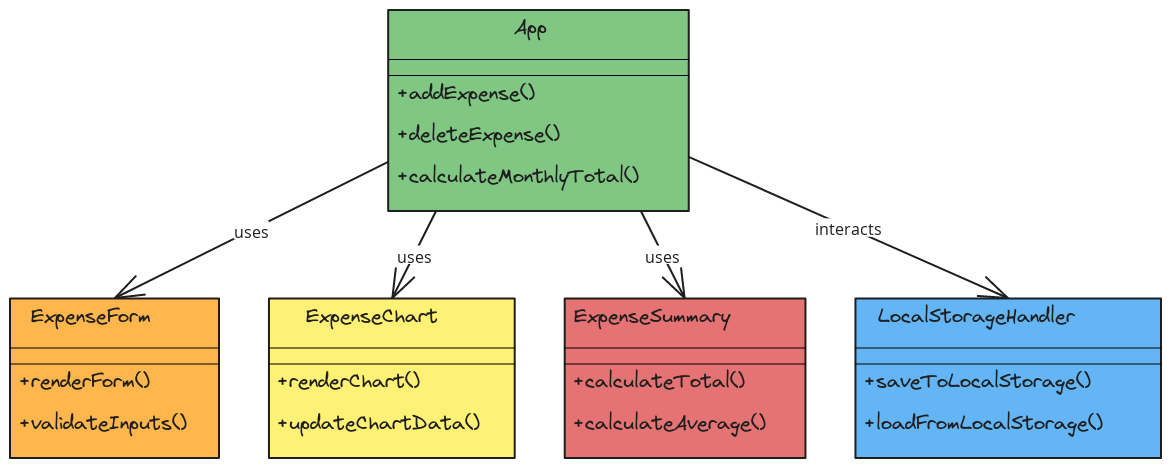
# Software Design Document for Personal Expense Tracker

1. **Architecture Overview**
   * **Frontend**: Built with React and TypeScript, featuring modular, component-based architecture. This design ensures scalability and reusability of components.
   * **Data Storage**: Utilizes localStorage for maintaining persistence across sessions. All expense data is stored as JSON, facilitating easy data handling.
   * **Visualization**: Integrated recharts library for dynamic data visualization, supporting responsive charts and interactive data insights.



1. **Core Components**
   * **ExpenseForm**: Manages user input for expenses, including validation to ensure accuracy.
   * **ExpenseChart**: Renders bar charts displaying monthly spending, aiding users in visualizing financial trends.
   * **ExpenseSummary**: Calculates and displays total and average expenses, providing a quick summary of user spending.
   * **App State Management**: React hooks are used for managing state within components, ensuring efficient data flow and reactivity.
2. **Design Decisions**
   * **Choice of Local Storage**: Selected for its simplicity in maintaining data between sessions for single-user applications. This could be upgraded to a cloud solution in future releases.
   * **TypeScript**: Chosen to enhance code safety and readability, ensuring type safety and minimizing runtime errors.
   * **Component-Based UI**: Component structure supports modular development, making the UI easier to maintain and extend.
3. **Error Handling**
   * **Client-Side Validation**: Input fields in ExpenseForm validate data in real-time, preventing invalid entries.
   * **Local Storage Errors**: Basic checks are implemented for storage failures, though additional safeguards may be required in future releases for complex storage solutions.
4. **Future Enhancements**
   * **Backend Integration**: Plan to transition from local storage to a database for improved data security and scalability.
   * **Enhanced Data Validation**: Implement server-side validation in future cloud or database-based architectures for additional data integrity.