**ShopEase - Grocery Shopping List & Budgeting App**

**1. Purpose**

ShopEase is designed to help users create and manage grocery shopping lists while keeping track of their budget. The app will provide an easy-to-use interface that allows users to categorize items, mark items as purchased, and monitor their spending throughout the shopping process.

**2. Scope**

The app will be built using **React Native** for cross-platform support (iOS and Android) with **Firebase** for data storage and user authentication. It will provide users with the ability to create shopping lists, track expenses, and get insights on their grocery spending. Optional features include offline functionality, list sharing, and future expansion for smart shopping suggestions.

**3. Functional Requirements**

**3.1 User Roles**

1. **Registered User**:
   * Can create and manage multiple shopping lists.
   * Can set a budget for each list and track expenses.
   * Can categorize items (e.g., Produce, Dairy, etc.).
   * Can mark items as purchased.
   * Can view past shopping trips.
   * Can log in and sync data across multiple devices.
2. **Guest User**:
   * Can create a temporary shopping list.
   * No cloud sync or budget tracking (local storage only).
   * Limited to basic shopping list features.

**3.2 Features**

**3.2.1 Shopping List Management**

* **Create New List**:
  + Users can create new shopping lists and name them (e.g., Weekly Groceries, Birthday Party Shopping).
* **Add/Edit Items**:
  + Users can add grocery items to the list, assign a quantity (e.g., 2kg, 3 packs), and choose a category (e.g., Produce, Dairy, etc.).
  + Items can be edited or deleted as needed.
* **Mark Items as Purchased**:
  + As users shop, they can mark items as "Purchased," automatically calculating the total cost.

**3.2.2 Budget Tracking**

* **Set Budget**:
  + Users can set a budget for each list (e.g., $100 for weekly groceries).
* **Track Expenses**:
  + As users add prices to purchased items, the app tracks total spending against the budget and provides visual feedback (e.g., green for under budget, red for over budget).
* **Budget Alerts**:
  + Notifications are sent when a user approaches or exceeds the budget.

**3.2.3 Categories**

* **Default Categories**:
  + Provide users with default categories (e.g., Produce, Dairy, Meat, Household Items).
* **Custom Categories**:
  + Allow users to create custom categories for organizing their lists.

**3.2.4 Offline Mode**

* **Offline Shopping List**:
  + Users can add and mark items without an internet connection. Data is stored locally and will sync with the cloud when back online.

**3.2.5 List Sharing (Optional)**

* **Share List**:
  + Users can share their list with others (e.g., family or roommates) by generating a shareable link or inviting them via email.
* **Collaborative Editing**:
  + Shared users can add, edit, or remove items in real-time.

**3.2.6 Expense Insights**

* **Shopping History**:
  + Users can view a history of past shopping trips and see how much they spent.
* **Spending Insights**:
  + Graphs and analytics showing users’ spending habits over time (e.g., monthly reports on grocery spending).

**3.2.7 Notifications**

* **Task Reminders**:
  + Reminder notifications for incomplete lists or when a budget is close to being exceeded.

**4. Non-Functional Requirements**

**4.1 Performance**

* The app must load shopping lists within 2 seconds and handle multiple lists simultaneously without performance degradation.

**4.2 Security**

* All user data (especially budgets and shopping histories) should be encrypted.
* Authentication via Firebase to securely manage user logins and prevent unauthorized access.

**4.3 Usability**

* A simple, intuitive user interface suitable for users of all ages.
* Support for both light and dark modes to improve readability.

**4.4 Scalability**

* The system should handle a large number of concurrent users and lists without slowdowns or crashes.

**4.5 Compatibility**

* The app must be fully functional on Android (5.0 and up) and iOS (12 and up).

**4.6 Reliability**

* The app should have 99.9% uptime with seamless sync between local and cloud data when online.

**5. High-Level Architecture**

**5.1 Technology Stack**

* **Frontend**: React Native (JavaScript), Redux for state management.
* **Backend**: Firebase for authentication, real-time database, and cloud storage.
* **API**: RESTful API for any future third-party integration (optional, e.g., barcode scanning).
* **Notifications**: Firebase Cloud Messaging for push notifications.

**5.2 Data Model**

* **User**:
  + ID
  + Email
  + Shopping Lists
* **Shopping List**:
  + List ID
  + List Name
  + Budget
  + Total Cost
  + Items (Array of Items)
* **Item**:
  + Item Name
  + Quantity
  + Category
  + Price
  + Purchased (True/False)

**5.3 Key App Flows**

1. **List Creation**:
   * User logs in → creates new shopping list → adds items to the list → sets a budget → completes shopping.
2. **Expense Tracking**:
   * User adds prices to purchased items → app updates total cost → compares against the budget and provides feedback.
3. **Offline Mode**:
   * User can create lists and mark items offline → app syncs data when the user is back online.

**6. UI/UX Design**

The UI should be clean, minimalistic, and user-friendly with a focus on usability. Here’s a general layout suggestion:

1. **Home Screen**:
   * List of all shopping lists with buttons to create a new list or view history.
2. **List Screen**:
   * View and edit the current list, with sections for items and categories.
   * "Add Item" button at the bottom for quick access.
3. **Budget and Analytics Screen**:
   * Budget progress bar on the top of the list.
   * Past spending and visual graphs on the history page.
4. **Settings**:
   * Customize categories, set notification preferences, and manage user profile.

**7. Future Expansion**

* **Smart Suggestions**: Suggest commonly purchased items based on user history.
* **Barcode Scanning**: Allow users to scan items while shopping for easy list management.
* **Recipe Integration**: Suggest recipes based on items in the shopping list.

**8. Development Timeline (Estimate)**

| **Phase** | **Duration** |
| --- | --- |
| **UI/UX Design** | 1-2 weeks |
| **Frontend Development** | 3-4 weeks |
| **Backend Development** | 2-3 weeks |
| **Testing & Debugging** | 2 weeks |
| **Launch & Optimization** | 1 week |

This document should serve as a guide to start development on **ShopEase**. Let me know if you'd like to dive deeper into any of these sections or need additional details!