SAVOR

Fine Dining Restaurant Mobile App

Software Requirements Specification (SRS)

Version 1.0

February 2026

Project	SAVOR Restaurant App
Platform	React Native + Expo (iOS & Android)
Document Type	Software Requirements Specification
Status	**Final - v1.0**

1. Introduction

1.1 Purpose

This Software Requirements Specification (SRS) document describes the functional and non-functional requirements for the SAVOR Restaurant Mobile Application. The application is built using React Native with Expo and targets both iOS and Android platforms. This document is intended for developers, project managers, QA engineers, and stakeholders involved in the development lifecycle.

1.2 Scope

SAVOR is a frontend-only mobile restaurant application that provides customers with a seamless dining experience. The application encompasses:

- User authentication (registration and login)
- Interactive digital menu with category filtering and search
- Today's Special showcase with discount and chef highlights
- Shopping cart management with quantity control
- Order placement with Dine-In / Takeaway selection
- Table reservation with date, time, and guest selection
- User profile management

The application uses static/dummy data for all menu items and does not include any backend integration in this version.

1.3 Definitions, Acronyms, and Abbreviations

Term / Acronym	Definition
SRS	Software Requirements Specification
RN	React Native — a JavaScript framework for building mobile apps
Expo	A platform and set of tools built around React Native
UI	User Interface
UX	User Experience
API	Application Programming Interface
FR	Functional Requirement
NFR	Non-Functional Requirement
Context API	React mechanism for sharing state across the component tree without prop drilling

1.4 References

- React Native Documentation — https://reactnative.dev
- Expo SDK 51 Documentation — https://docs.expo.dev

- React Navigation v6 — https://reactnavigation.org
- IEEE Std 830-1998 — IEEE Recommended Practice for SRS

1.5 Overview

The remainder of this document is organized as follows:

- Section 2: Overall Description — product perspective, assumptions, and constraints
- Section 3: Functional Requirements — detailed feature requirements
- Section 4: Non-Functional Requirements — performance, security, usability
- Section 5: Use Case Diagram — UML diagram with actor-use case relationships
- Section 6: System Architecture — component and navigation structure
- Section 7: Data Requirements — data models and static data
- Section 8: Constraints & Assumptions
- Section 9: Appendix

2. Overall Description

2.1 Product Perspective

SAVOR is a standalone frontend mobile application. It operates as an independent system with no backend dependency in its current version. The app simulates a fully functional restaurant ordering and reservation system using in-memory state management via React Context API and static JSON data.

The system interfaces involved include:

- Mobile operating systems: iOS 14+ and Android 10+
- Expo Go runtime environment for development and testing
- React Navigation for screen-to-screen transitions
- React Native SafeAreaContext for device-safe layout rendering

2.2 Product Functions

At a high level, the SAVOR app provides the following core functions:

#	Function	Description
F1	Authentication	Register, login, and logout securely
F2	Menu Browsing	View all food items grouped by category with search and filters
F3	Today's Special	View chef's highlighted daily special with discount info
F4	Cart Management	Add, remove, and update quantities of selected items
F5	Order Placement	Select order type and confirm order with total summary
F6	Reservation	Book a table by selecting date, time, guests, and special requests
F7	Profile	View account details, order stats, and logout

2.3 User Classes and Characteristics

2.3.1 Customer (Primary User)

The primary user of the application. Customers are restaurant patrons who use the app to browse the menu, place orders, make reservations, and manage their profiles. They are assumed to have basic smartphone literacy.

2.3.2 Restaurant Staff (Secondary Actor)

Restaurant staff may view reservations and confirmed orders in future versions. In the current version, their role is represented as a system actor for confirmation flows.

2.4 Operating Environment

- Mobile Platforms: iOS 14+ (iPhone) and Android 10+ (API level 29+)
- Development Environment: Visual Studio Code with Expo CLI
- Runtime: Expo Go app or standalone APK/IPA build
- Language: JavaScript (ES2020+)
- Framework: React Native 0.74.x, Expo SDK 51

- State Management: React Context API + useState hooks
- Navigation: React Navigation v6 (Native Stack)

2.5 Design and Implementation Constraints

- The application is frontend-only — no REST API, no database
- All data is static and defined in /data/menuData.js
- No external UI libraries — only React Native core StyleSheet
- Images are loaded from Unsplash CDN URLs (requires internet)
- Authentication is simulated with in-memory state (no JWT/sessions)
- Cart state resets upon logout

2.6 Assumptions and Dependencies

- The user has the Expo Go app installed and up to date
- The development machine and phone are on the same Wi-Fi network (or tunnel mode is used)
- Node.js 18+ is installed on the development machine
- Internet access is required for CDN image loading

3. Functional Requirements

3.1 Authentication Module

FR-101: User Registration

Requirement ID	FR-101
Title	User Registration / Sign Up
Priority	High
Description	The system shall allow new users to create an account by providing their full name, email address, password, and password confirmation.
Inputs	Full Name, Email, Password, Confirm Password
Validation	Name is required; email must match RFC format; password must be 6+ characters; confirm password must match password
Output	User is logged in and navigated to Home Screen

FR-102: User Login

Requirement ID	FR-102
Title	User Login
Priority	High
Description	The system shall authenticate users using email and password credentials.
Validation	Email format validation; password minimum 6 characters
Output	Successful login navigates to Home Screen; failed validation shows inline error messages

FR-103: User Logout

The system shall allow authenticated users to log out from the Profile screen. Upon logout, cart state is cleared and the user is redirected to the Login screen.

3.2 Home Screen

FR-201: Dashboard Navigation

The Home Screen shall display:

- Personalized greeting with the logged-in user's first name
- A restaurant hero banner with branding (logo, tagline, stats)
- Quick access navigation cards for: Menu, Today's Special, Reservation, Cart, and Profile
- A live cart badge showing the current item count on the Cart nav card

3.3 Menu Module

FR-301: Category Filtering

The Menu screen shall display a horizontal scrollable list of food categories (Burgers, Pizza, Drinks, Desserts). Tapping a category filters the visible menu items to that category only.

FR-302: Menu Item Display

Each menu item card shall display:

- Food image (from CDN)
- Item name and description (max 2 lines)
- Price formatted to 2 decimal places
- Star rating
- Popular badge (if applicable)

FR-303: Search

A search bar shall allow users to filter items by name or description keywords within the selected category. A clear button shall appear when the search field is non-empty.

FR-304: Add to Cart from Menu

Each menu item shall have an "Add" button. Once added, the button is replaced with a quantity selector (+/−) allowing inline quantity adjustment. Reducing quantity to 0 removes the item from the cart.

3.4 Today's Special Module

FR-401: Special Item Display

The Today's Special screen shall prominently display the chef's featured item with:

- Full-width hero image
- Discount badge showing percentage off
- Original price with strikethrough and discounted current price
- Savings amount highlighted
- Chef's name, prep time, and calorie info
- Star rating and review count

FR-402: Add Special to Cart

Users shall be able to add the special item to the cart with an 'Add to Cart' button. Once added, a quantity selector and a 'Go to Cart' shortcut button appear.

3.5 Cart Module

FR-501: Cart Item List

The Cart screen shall list all added items with their image, name, unit price, quantity selector, and line total. An empty cart state shall show an illustration and a "Browse Menu" CTA button.

FR-502: Quantity Management

Users can increment or decrement item quantity directly in the cart. Decrementing to zero removes the item from the cart.

FR-503: Remove Item

A delete button on each cart row shall remove that item entirely from the cart.

FR-504: Order Type Selection

Users shall select between Dine-In or Takeaway before placing an order. The selected option affects the displayed service/packaging charge.

FR-505: Order Summary

The cart shall display a real-time order summary including: subtotal, tax (10%), service/packaging charge ($2.00), and total amount.

FR-506: Place Order

A "Place Order" button shall show a confirmation alert with order details and estimated time. Upon confirmation, the cart is cleared and the user is redirected to the Home screen.

3.6 Reservation Module

FR-601: Date Selection

A horizontally scrollable date picker shall display the next 7 days. The user selects a date by tapping the appropriate card. Today is labelled "Today".

FR-602: Time Selection

Available time slots shall be presented as a grid of tap-selectable chips. A validation error is shown if the user attempts to confirm without selecting a time.

FR-603: Guest Count

Users shall select the number of guests (1–8) from a grid of circular buttons.

FR-604: Special Request

An optional free-text field allows users to specify allergies, seating preferences, or special occasions.

FR-605: Confirmation

Upon tapping "Confirm Reservation", the system validates required fields (name and time) and displays a summary alert with all reservation details.

3.7 Profile Module

FR-701: Profile Display

The Profile screen shall display: user avatar, full name, email, member-since date, total orders, star rating, and membership tier.

FR-702: Navigation Options

The profile menu shall list options for: My Orders, Favorites, Saved Addresses, Payment Methods, Notifications, Rate Us, Help & Support, and Privacy Policy. Tapping any option shows a "Coming Soon" alert.

FR-703: Logout

A logout button with confirmation dialog shall clear the user session and cart, then redirect to the Login screen.

4. Non-Functional Requirements

4.1 Performance

- App initial load time shall not exceed 3 seconds on mid-range devices
- Screen transitions shall complete within 300ms
- FlatList rendering shall support smooth 60fps scrolling for up to 100 items
- Cart state updates shall reflect in UI within 100ms

4.2 Usability

- The UI shall follow a consistent red & black color theme throughout all screens
- All interactive elements shall have a minimum touch target of 44x44 points
- Error messages shall appear inline, adjacent to the relevant input field
- Empty states shall be communicated with meaningful illustrations and action prompts
- All screens shall be scrollable and responsive across device sizes (small phones to tablets)

4.3 Reliability

- The app shall not crash upon invalid user input — all inputs shall be validated gracefully
- Cart state shall persist across screen navigation within a session
- Removing an item when cart is empty shall not cause errors

4.4 Security

- Passwords shall be masked by default with a toggle to reveal
- No sensitive data shall be stored in local storage or AsyncStorage in this version
- The app shall not transmit user data to any external server

4.5 Maintainability

- Code shall be organized in /screens, /components, /navigation, /data, and /context directories
- Reusable components (Header, PrimaryButton, InputField, MenuItemCard, CartBadge) shall be used consistently
- A centralized theme file (constants/theme.js) shall define all colors, fonts, spacing, and shadows
- Static data shall be isolated in /data/menuData.js for easy updates

4.6 Portability

- The application shall run on iOS 14+ and Android 10+ without modification
- The codebase shall be compatible with Expo SDK 51 and support both Expo Go and standalone builds

5. Use Case Diagram

The following UML Use Case Diagram illustrates the interactions between actors (Customer, System Backend, Restaurant Staff) and the system use cases within the SAVOR application boundary.

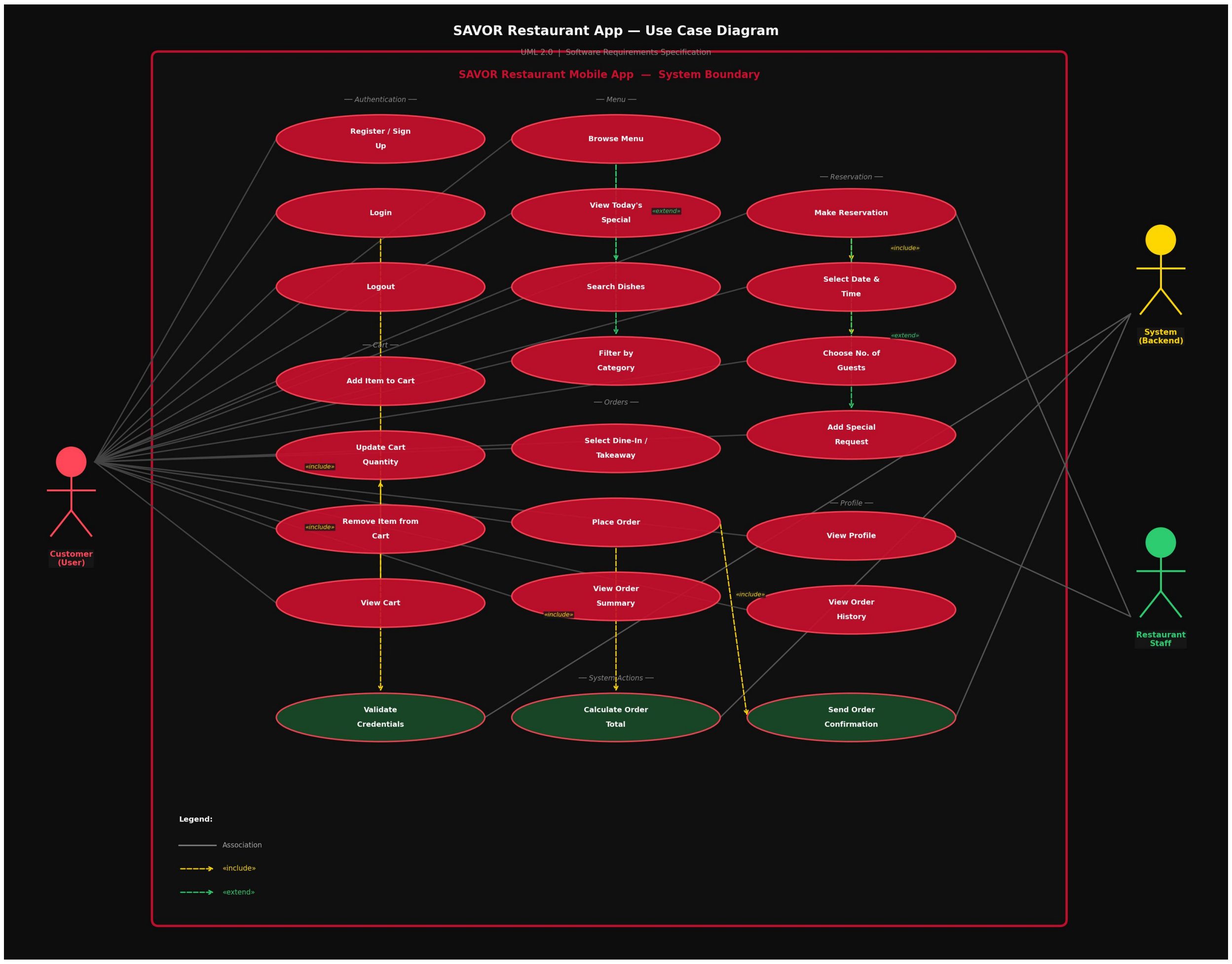

Figure 1: SAVOR Restaurant App — UML Use Case Diagram

5.1 Actors

Actor	Description
Customer (User)	Primary user of the app. Can authenticate, browse menu, manage cart, place orders, make reservations, and view profile.
System (Backend)	Represents automated system actions: credential validation, order total calculation, and confirmation alerts.
Restaurant Staff	Secondary actor who may view reservations and profile data in future versions.

5.2 Use Case Relationships

Relationship	Type	Description
Login → Validate Credentials	«include»	Login always triggers credential validation
Place Order → Calculate Total	«include»	Order placement always computes the final total
Place Order → Send Confirmation	«include»	A confirmation alert is always sent upon order placement
Make Reservation → Select Date & Time	«include»	Date and time selection is mandatory for reservations
Make Reservation → Add Special Request	«extend»	Special requests are optional extensions to reservations
Browse Menu → Search Dishes	«extend»	Search extends the browse behavior optionally
Browse Menu → Filter by Category	«extend»	Category filtering extends browsing optionally

6. System Architecture

6.1 Folder Structure

The project follows a modular, feature-oriented structure:

Directory / File	Purpose
App.js	Root entry point — wraps app in SafeAreaProvider, AuthProvider, CartProvider
index.js	Expo registerRootComponent entry
constants/theme.js	Centralized design tokens: colors, fonts, spacing, radius, shadows
context/AuthContext.js	Global authentication state (user, login, logout)
context/CartContext.js	Global cart state (items, add, remove, update, totals)
data/menuData.js	Static menu items, categories, and Today's Special data
navigation/AppNavigator.js	Auth-gated Native Stack navigator — switches between auth and app stacks
components/	Reusable UI components: Header, PrimaryButton, InputField, MenuItemCard, CartBadge
screens/	8 full screens: Login, Signup, Home, Menu, TodaysSpecial, Cart, Reservation, Profile

6.2 Navigation Flow

The app uses a conditional navigation structure based on authentication state:

- Unauthenticated: Login Screen → Signup Screen
- Authenticated: Home → Menu → TodaysSpecial → Cart → Reservation → Profile

All authenticated screens can navigate back via the Header back button. Cart is accessible from Menu, TodaysSpecial, and Home screens via the CartBadge icon.

6.3 State Management

Context	State Fields	Methods Exposed
AuthContext	user, isLoggedIn	login(userData), logout()
CartContext	cartItems, cartTotal, cartCount	addToCart, removeFromCart, updateQuantity, clearCart, getItemQuantity

7. Data Requirements

7.1 Menu Item Data Model

Field	Type	Description
id	String	Unique identifier (e.g., "b1", "p2")
categoryId	String	Foreign key to category (e.g., "1" = Burgers)
name	String	Display name of the dish
description	String	Full text description of the dish
price	Number	Price in USD (e.g., 14.99)
image	String (URL)	Unsplash CDN image URL
rating	Number	Average rating out of 5
popular	Boolean	Flag for popular badge display

7.2 Cart Item Data Model

Field	Type	Description
id	String	References menu item ID
name	String	Copied from menu item at add time
price	Number	Unit price at add time
image	String	Image URL
quantity	Number	Current quantity in cart (min 1)

7.3 Menu Categories

ID	Name	Items Available
1	Burgers	Classic Smash, BBQ Bacon Blaze, Mushroom Swiss Deluxe, Spicy Chicken Crisp
2	Pizza	Margherita Royale, Inferno Diavola, Truffle Funghi, BBQ Pulled Pork
3	Drinks	Signature Lemonade, Berry Blast Smoothie, Cold Brew Coffee, Mango Tango Mojito
4	Desserts	Lava Chocolate Bomb, New York Cheesecake, Tiramisu Classico, Crème Brûlée

8. Constraints and Assumptions

8.1 Technical Constraints

- No backend — all data is static and in-memory
- No persistent storage — app state resets between sessions
- No push notifications in this version
- No payment gateway integration
- Images require internet access (Unsplash CDN)

8.2 Business Constraints

- The application is a demonstration/prototype for a restaurant scenario
- Order placement results in an in-app alert (no real order processing)
- Reservation confirmation is in-app only (no SMS/email notification)

8.3 Assumptions

- Users have a stable internet connection for image loading
- The device supports React Native and Expo SDK 51
- Users are comfortable with standard mobile app navigation patterns
- Menu items and pricing are static for the duration of the session

9. Appendix

9.1 Technology Stack Summary

Technology	Version	Role
React Native	0.74.5	Core mobile UI framework
Expo SDK	51.0.28	Development platform and toolchain
React	18.2.0	UI component library
@react-navigation/native	6.1.18	Navigation container
@react-navigation/native-stack	6.11.0	Stack-based screen navigation
react-native-screens	3.31.1	Native screen optimization
react-native-safe-area-context	4.10.5	Safe area padding for notched devices
expo-status-bar	1.12.1	Status bar control

9.2 Screen Inventory

Screen	Route Name	Access	
Login Screen	Login	Unauthenticated users only	
Signup Screen	Signup	Unauthenticated users only	
Home Screen	Home	Authenticated users	
Menu Screen	Menu	Authenticated users	
Today's Special Screen	TodaysSpecial	Authenticated users	
Cart Screen	Cart	Authenticated users	
Reservation Screen	Reservation	Authenticated users	
Profile Screen	Profile	Authenticated users	

9.3 Revision History

Version	Date	Author	Changes
1.0	Feb 2026	SAVOR Dev Team	Initial SRS release

End of Document — SAVOR Restaurant App SRS v1.0