Key Features for Sprint 4

Requirement 9.0: Implement shared recipe collections

Feature Overview: Introduce shared recipe collections that allow users to collaboratively store, view, and manage recipes with others.Service Layer and Integration: Build backend support for shared-access logic, permissions, and synchronized updates across users.
Functionality:

- Users can create, join, and manage shared collections.
- Recipe changes update for all members in real time.
- Access control ensures only authorized contributors can edit shared lists.

Requirement 9.1: Design and implement the database structure for shared recipe collections

Feature Overview: Create a schema that supports multi-user collections with recipe links and permissions.
Structure: Define tables for collections, membership, and recipe mappings. Ensure queries scale with many contributors.
Functionality:

- Collections persist with accurate member associations.
- Recipes can belong to personal and shared lists simultaneously.
- Schema supports future collaborative features.

Requirement 9.2: Create UI for users to move saved recipes to a shared space

Feature Overview: Provide UI controls that allow users to transfer recipes from their saved list into shared collections.
UI and Flow: Include clear buttons or menus for selecting a destination shared collection.
Functionality:

- Users can select one or more recipes and move them easily.
- System prevents sending duplicate recipes to the same collection.
- Confirmation messages provide clear user feedback.

Requirement 9.3: Build a screen to view the shared recipe collection

Feature Overview: Create a dedicated screen showing all recipes inside a shared collection.
Structure: Display recipes in a grid or list, consistent with existing browsing screens.
Functionality:

- Users can browse shared recipes.
- Real-time updates refresh the list automatically.
- Navigation to full recipe details remains supported

Requirement 10.0: Create a meal planner with grocery list generation

Feature Overview: Add a weekly meal planner that allows users to assign recipes to days of the week and generate a grocery list from all assigned meals.
Service Layer and Integration: Link planner data with user accounts and ingredient aggregation logic.
Functionality:

- Users assign meals to specific days.
- Grocery list generation updates automatically.
- Planner persists changes between sessions

Requirement 10.1: Build a weekly calendar view for assigning saved recipes to the calendar

Feature Overview: Display a week-based layout where each day can hold assigned recipes.
UI and Design: Show seven interactive day sections with space to preview assigned meals.
Functionality:

- Users tap a day to add a recipe.
- Assigned recipes display titles or thumbnails.
- Calendar data saves reliably.

Requirement 10.2: Develop drag-and-drop functionality for assigning recipes to the calendar

Feature Overview: Improve usability by enabling drag-and-drop interactions for meal planning.
Behavior: Recipes can be dragged from the saved list and dropped onto a specific day.
Functionality:

- Smooth drag animations.

- Clear drop targets.
- Calendar updates immediately after placement.

Requirement 10.3: Develop a feature to automatically aggregate ingredients from the weekly plan

Feature Overview: Combine ingredients from all assigned recipes into a unified grocery list.
Data Handling: Normalize names, combine like items, and parse quantities.
Functionality:

- Grocery list updates when assignments change.
- Duplicate ingredients are merged.
- Ingredient parsing is consistent.

Requirement 10.4: Implement de-duplication logic for the aggregated grocery list

Feature Overview: Prevent repeated ingredients from appearing multiple times.
Logic: Identify similar items using normalization and unit reconciliation.
Functionality:

- List contains unique, consolidated items.
- Conflicting units are resolved when possible.
- Edge cases (e.g., plural forms) are handled.

Requirement 10.5: Create a dedicated, readable grocery list screen

Feature Overview: Provide a clear and organized screen displaying all grocery items.
UI and Structure: Scrollable list format with appropriate spacing and grouping.
Functionality:

- Items display cleanly and consistently.
- List refreshes when meal plans change.
- Supports smooth navigation

Requirement 10.6: Add ability to manually check off items on the grocery list

Feature Overview: Allow users to mark items as completed during shopping.
UI and Flow: Use checkboxes or toggle-based interactions..
Functionality:

- Checked items remain visible but marked.
- Users can uncheck items as needed.
- State persists during the shopping session

Requirement 11.0: Final UI/UX Polish and Testing

Feature Overview: Refine final design elements, animations, and transitions to unify the app's visual and interactive feel..
Structure: Apply micro-interactions, smooth animations, and consistent spacing adjustments.
Functionality:

- App feels polished and cohesive.
- Performance and usability improve across screens.
- UI inconsistencies are addressed

Requirement 11.1: Add micro-interactions and animations for a polished feel

Feature Overview: Add lightweight animations to improve clarity, responsiveness, and user satisfaction.
UI and Design: Animate taps, transitions, list expansions, and loading states.
Functionality:

- Animations run fluidly across devices.
- Feedback for interactions is clearer.
- Motion enhances usability without being distracting

Requirement 11.2: Conduct comprehensive UI/UX testing across multiple devices and screen sizes

Feature Overview: Test the interface across a wide range of device types to ensure consistency.
Testing Scope: Validate layouts, responsiveness, and interaction flows.
Functionality:

- Bugs and inconsistencies are identified.

- Screens adapt correctly to different resolutions.
- Results guide final fixes.

Requirement 11.3: Fix all critical and major UI/UX bugs identified during testing

Feature Overview: Address high-severity design and usability issues uncovered during testing.
Behavior: Prioritize issues that hinder usability or break visual consistency.
Functionality:

- UI reliability improves across the application.
- Navigation feels smooth and predictable.
- All major blockers are resolved.

Requirement 11.4: Perform usability testing with a small group of target users

Feature Overview: Validate real-world usability by observing how target users interact with the app.
Process: Collect qualitative feedback and identify areas of confusion or friction.
Functionality:

- User insights inform final refinements.
- Ensures key flows are intuitive.
- Improves readiness for broader release.