Technical Requirements – January Recipe

Challenge is accessible on Memberspot: https://pl-coding.mymemberspot.io/library/jx3b7Qik9ip5qpNI8IF2/BwAxp75QNXGaDGF2ZJEG/uoUAeShOlYCFvoq2dzEx/details

🎭 Scenario

This mini-challenge simulates a seasonal recipe browsing experience focused on winter comfort dishes. The user scrolls through a recipe feed, searches for recipes, marks favorites, refreshes the list, and views recipe details in a dialog.

🎨 Figma Mockups

> https://www.figma.com/design/XKuvJCxCybCUpKZP0LelAs/New-Year-Fresh-Start?node-id=1-4

🔤 Fonts - Instrument Sans, Instrument Serif

🎯 Feature Goal

Practice building a state-driven, single-screen app using Jetpack Compose, focusing on list rendering, search filtering, pull-to-refresh behavior, dialogs, snackbars, and simple adaptive layouts.

📌 Requirements

Main Screen (Recipe Feed)

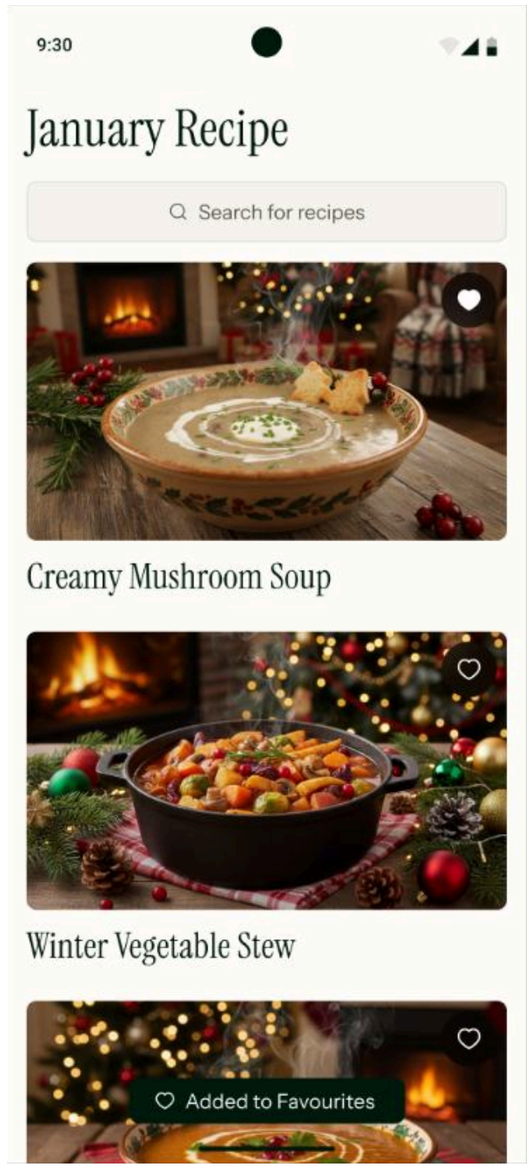
The main screen displays a feed of winter-themed recipes in a vertical list format. This screen is the primary and only main screen of the app and is used for browsing, searching, and interacting with recipes.

Top Bar

- Positioned at the top of the screen.
- Displays the title *“January Recipe”*.
- Contains no navigation buttons or actions.

Search Field

- Positioned below the header.
- Implemented as a search text field.
- Displays the placeholder text *“Search for recipes”*.
- Typing in the field filters the recipe list instantly.
- Filtering happens in real time without any confirmation action.



Recipes List

- Positioned below the search field.
- Implemented as a vertical scrolling list.
- Each list item represents a single recipe.

Recipe Card

Each recipe card includes:

- A food image occupying the main area of the card.
- The recipe title displayed below the image.
- A heart icon positioned on the image, used to mark the recipe as a favorite.

Favorite Behavior

- Tapping the heart icon marks the recipe as a favorite.
- Favorited recipes have a visual indication.
- Adding a recipe to favorites triggers a snackbar confirmation.
- Favorite recipes are pinned to the top of the list and keep their position during refresh.

Pull-to-Refresh

- The list supports pull-to-refresh interaction.
- Refresh simulates receiving new seasonal recipes.
- After refresh:
  - all non-favorite recipes are **shuffled** in the list;
  - favorite recipes remain pinned at the top.
- This behavior visually demonstrates pull-to-refresh without performing a real network request.

Recipe Details Dialog

The dialog opens on top of the main screen when the user taps a recipe card. The background content is dimmed to emphasize focus on the dialog.

Dialog Appearance

- Displayed in the center of the screen.
- Has rounded corners.

Dialog Content

The dialog displays the following elements arranged vertically:

- A large recipe image at the top of the dialog.
- The recipe title displayed below the image.
- A short ingredients list presented as text.
- A brief cooking description consisting of a few sentences.
- A confirmation button at the bottom of the dialog.

Action Button

- A button labeled *“Done”* is displayed at the bottom of the dialog.
- The button is used to close the dialog.
- Tapping *“Done”* closes the dialog and returns the user to the main screen.

Interaction Behavior

- The dialog opens when the user taps anywhere on a recipe card.
- The dialog can be dismissed by:
  - tapping the *“Done”* button;
  - tapping outside the dialog on the dimmed background.
- The dialog content is read-only.

Wide Screen Layout Behavior

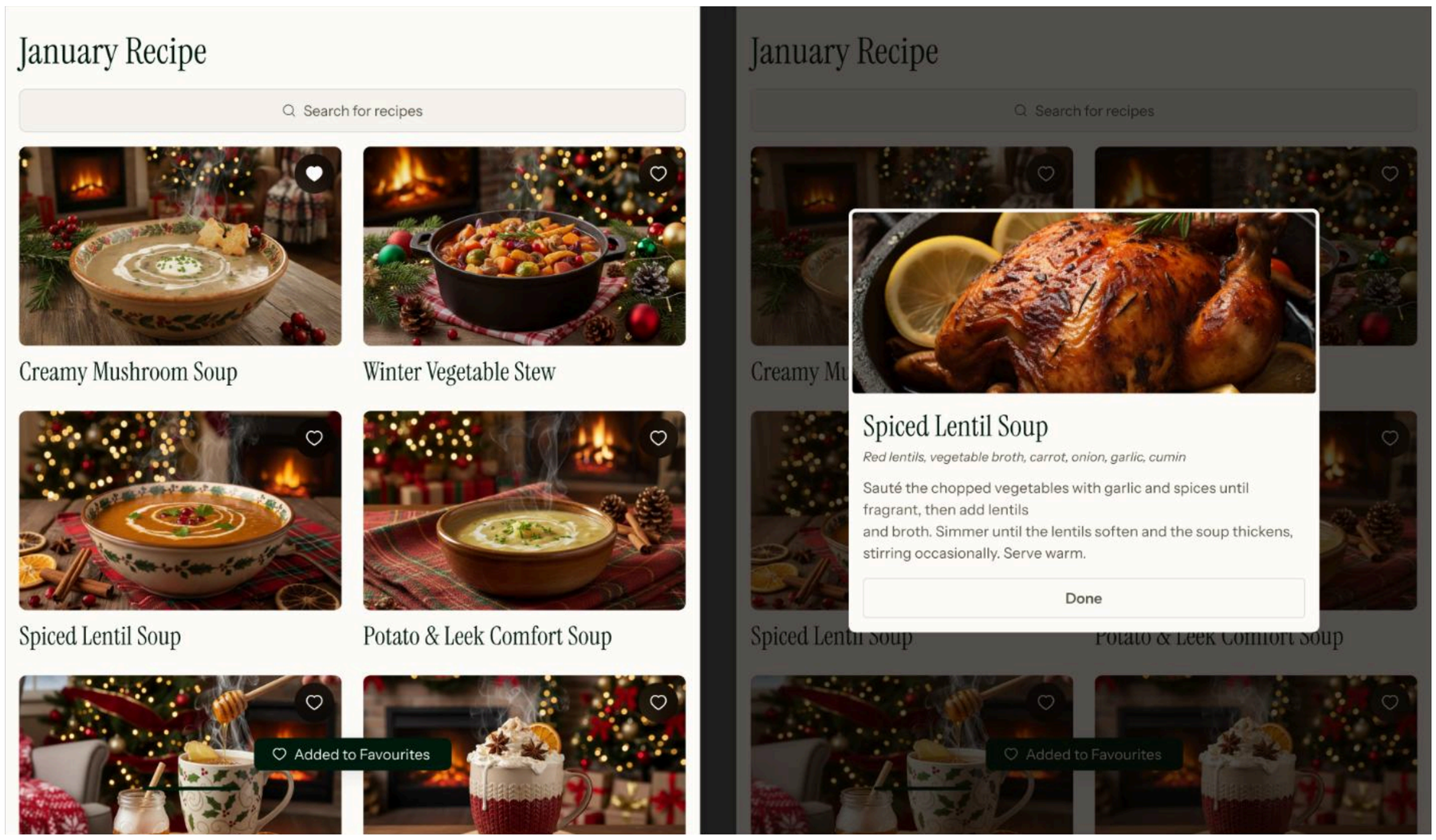
On wider screens, the overall structure and interactions remain the same as on the mobile version. The only differences are:

Main Screen

- The recipe list is displayed as a **two-column grid**.

Recipe Details Dialog

- The dialog has a fixed width of **520 dp**.
- The dialog is centered on the screen.



🧪 Initial Recipe Data

The app starts with a predefined list of winter-themed recipes used for the main feed and recipe dialogs. The data can be implemented as a simple in-memory list and does not need to be persisted in a database.

A ready-to-use Kotlin file with the full recipe list and data model is provided in the attached materials in the members area, along with the recipe images.

🤔 What's Allowed?

- Standard Android/Jetpack libraries
- No 3rd party libraries are allowed or would be required to complete this challenge
- Simple in-memory data (hardcoded recipe list).
- Basic state handling for search, favorites, and refresh behavior.

⚠️ What's not important

- Responsiveness across every device size or orientation is not mandatory.
- Light/Dark mode support.
- Persisting data after app restart.
- Networking or real backend integration.
- Complex animations or transitions.
- Advanced error handling or edge cases.

🔗 Useful Links for This Challenge

- Lists and grids
- Text in Compose
- Dialog
- Snackbar
- Pull to refresh
- Build adaptive apps
- State and Jetpack Compose
- Stateful vs. Stateless Composables
- State Hoisting in Compose
- Managing State in Jetpack Compose (Codelab)

🏆 Submission & Rewards

- Successfully submitting this challenge via the **/submit-challenge** command on Discord grants you **200 XP**.
- Your submission must include:
  a. A **Gist link** with your implementation.
  b. A **screen recording** (max 30 seconds) showing:
- Opening the app and displaying the main recipe feed.
- Filtering the list using the search field.
- Marking recipes as favorites and showing that favorites move to the top.
- Displaying a snackbar after marking a recipe as favorite.
- Performing a pull-to-refresh gesture and showing that non-favorite recipes are shuffled.
- Opening a recipe details dialog and closing it.
- For wide screens, attach two screenshots showing the main screen and the recipe details dialog.