

# Recipe Finder

## Final Presentation

---

### Group 19

Justinne Baltazar

Kevin Le

Mark Vu

Brian Huang

Yan ting Leung





# Our Idea for Recipe Finder

## Original Idea & Pitch

- A smart mobile app that helps users decide what to cook based on the ingredients they already have
- Users can:
  - Manually enter leftover ingredients
  - Store ingredients in a Virtual Fridge
  - Generate recipes using AI
- The goal is to **reduce food waste**, save time, and remove the stress of daily meal planning

## Why it was worth approaching?

- Reduces food waste and encourages creative cooking by generating new recipe ideas based on user-available ingredients
- Supports personalized and inclusive cooking experiences by tailoring recipes to user preferences and dietary needs
- Strong opportunity to combine Android Development and AI
- Highly scalable idea
  - Grocery list generation
  - Meal planning

# New Things We Learned

## Android UI

- **Material 3**
  - Modern design system for buttons, text fields, cards, and dialogs
- **RecyclerView**
  - Efficient dynamic list rendering for recipes, favorites, and fridge items
- **NestedScrollView**
  - Smooth vertical scrolling for complex ingredient layouts and dynamic lists
- **Material CardView**
  - Structured UI layout for recipe and ingredient display
- **MultiAutoCompleteTextView**
  - Input field that shows drop-down suggestions as the user types and allows selecting **multiple items separated by a delimiter**



# New Things We Learned

## Persistent Storage

- **Data Store**
  - Asynchronous, lifecycle-safe user preference storage
  - User data persists even when cache is cleared

## Backend & System Integration

- **FastAPI**
  - Python-based REST API
- **Gemini API**
  - AI-powered recipe generation based on text and image input
- **Docker**
  - Containerized backend
- **Firestore**



# Project Scope – Workload Justification

We mapped our total project workload using MyRuns components as a baseline units of effort

- Our total work corresponds to about 12 MyRuns-equivalent assignments
  - Room database implementations for:
    - Favorite Recipes
    - Recipes
    - Virtual Fridge
  - DataStore for user preferences
  - Camera Integration for:
    - User Profile
    - Ingredient Input
  - Advanced UI
    - Navigation across 7 screens
    - Complex dialogs and Material 3 UI components
  - Manual Recipe Entry
    - Fast API backend
    - Gemini API
  - Automatic Ingredient Detection from Images
  - Global Database – Firestore
    - User authentication



# Challenges

## Backend & AI Integration

- Request / response formatting
- Handling slow AI response times without blocking the UI

## Data Persistence Design & Object Modeling

- Designing consistent data models that work across UI, ViewModels, and databases
- Modeling complex objects, such as:
  - A Recipe object implementing Parcelable
  - Handling lists of ingredients and instructions when:
    - Passing data between Fragments
    - Saving to database
    - Receiving from the backend
- Ensuring data consistency and type safety across all layers of the app

## Team Coordination & Version Control

- Handling:
  - Multiple feature branches
  - Merge conflicts

# Threaded Design Diagram

## Diagram



# Demo

# Code Walkthrough



# Work Breakdown

Justinne	Kevin	Mark	Brian	Yan Ting
Virtual Fridge database		Recipe generation based on user input	Global database	
Integrate virtual fridge items into recipe search			Video presentation	







# Lessons Learned



## Positives Lessons

- We gained experience in understanding the pipeline for full-stack mobile development
  - How Android UI, ViewModels, databases, and backend APIs work together
- Modern Android tools improve reliability
  - Using DataStore, MVVM, and Material 3 made our app more stable and scalable
- Backend + AI Integration adds significant value
  - Integrating FastAPI + Gemini API showed us how mobile apps can use powerful external AI services in real-time

## Negatives Lessons

- Poor early data modeling leads to refactoring later
    - Designing complex objects required refactoring when persistence was added
  - Debugging across multiple layers is difficult
    - Errors could originate across any of the layers
- 
- 



**Thank you for listening!**