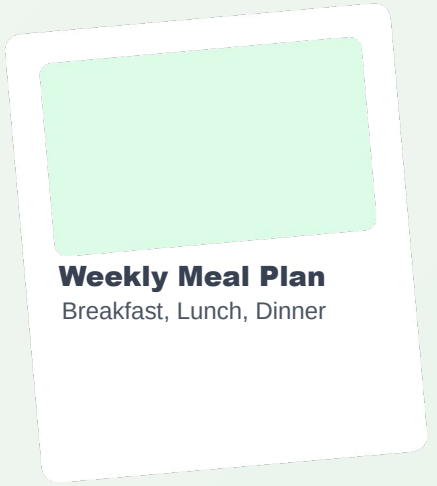
Decidish

Project Requirements Overview



Personalized Recipes

Smart Shopping

Personalization

Comprehensive Project Documentation

Motivation

Decision Fatigue

Deciding what to cook or finding the right recipes is too time and energy consuming.

→ Results in money being spent on food delivery or eating out, which is both more expensive and less healthy.

Lack of Personalization

Internet is overflowing with non-personalized recipes that ignore individual needs.

→ Recipes rarely consider budget, taste preferences, dietary restrictions, or regional ingredient availability.

Dietary Monotony

Users often return to the same simple meals they already know due to lack of time or motivation.

→ Creates nutritional imbalances and misses opportunities to discover new foods.

Resource Waste

Users waste food, time, and money due to poor meal planning and shopping management.

→ Food spoils when users don't know what they have at home, leading to double buying or forgetting essential items.

Project Goals & Solution

Overall Goal

Build a **personalized**, **time-saving**, and **budget-friendly** meal planner that transforms how users approach food.

Key Solution Aspects

Complete Personalization
Considers user preferences, budget, allergies, and dietary needs

Time Efficiency
Streamlined recipe selection and shopping list creation

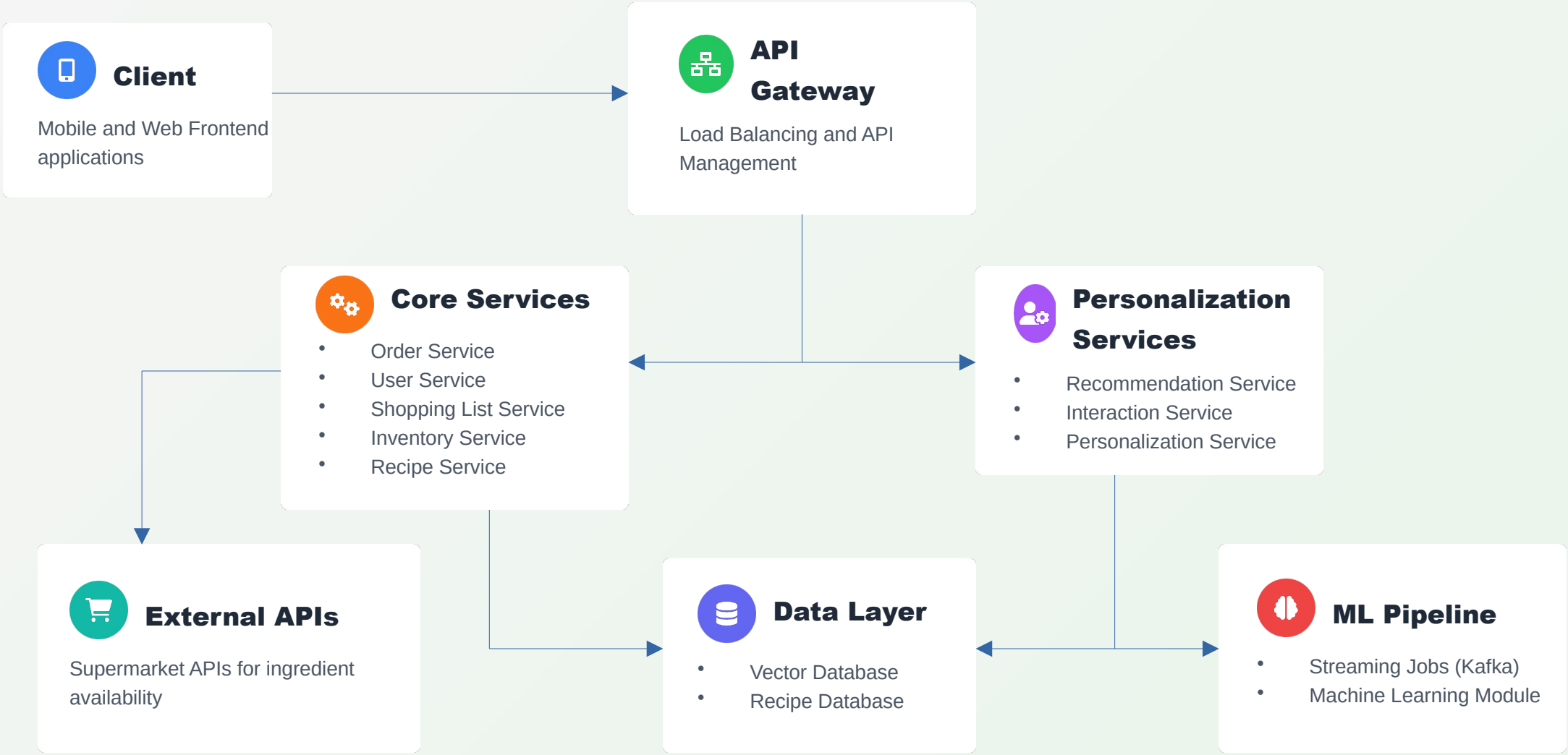
Minimize Waste
Smart inventory management and meal planning

Diet Diversity
Expands food choices based on user preferences

Expected Outcomes:

- Reduced food costs
- Healthier eating habits
- Less time spent cooking
- More meal variety

System Architecture Overview



Technology Stack

Frontend

- **Framework:** React Native
- **Language:** TypeScript
- **API:** REST API

API Gateway

- **Tool:** Nginx
- **Role:** Load Balancing
- **Function:** API Gateway

Backend Services

- **Personalisation:** Go
- **Recipe Service:** Java
- **Order Service:** Java

Database Layer

- **Primary:** PostgreSQL
- **Vector DB:** Milvus
- **Storage:** 50 MB/user

ML Pipeline

- **Purpose:** Update vector database
- **Stack:** Python (PyTorch, Scikit-learn)
- **Messaging:** Kafka

Performance Specs

- **Response:** P95 < 300 ms
- **Transaction:** < 150 ms
- **Uptime:** 95%

Functional Requirements

Seven core system capabilities that define the meal planner application

1

Recipe Suggestions

The system shall suggest suitable recipes based on the user's available ingredients.

2

Local Market Products

The system shall retrieve available products from local markets.

3

Personalized Recommendations

The system shall generate personalized recipe recommendations.

4

Nearest Markets

The system shall find nearest markets (based on user location).

5

Shopping List Generation

The system shall generate a shopping list based on chosen recipes.

6

User Profile Storage

The system shall store user features, preferences, and profile information.

7

Recipe History Tracking

The system shall track and store the user's recipe history (e.g., liked/disliked meals).

Performance & Quality Standards

Usability

User Experience Requirements

- Users must generate shopping lists within **10 clicks** from dashboard

Robustness

System Stability Requirements

- Continue operating despite component failures
- Microservices must degrade gracefully
- Maintain data consistency under concurrent access

Performance

System Speed Requirements

- Support horizontal scaling
- Handle **50 MB** data per user
- Transactions under **150 ms**
- P95 response time under **300 ms**

Availability

System Uptime

- System must be available **95%** of the time

Portability

Deployment Requirements

- Deployable on:
  - Android
  - iOS
  - Web Browsers

Quality Standards

The system must balance these quality requirements while maintaining a seamless user experience across all platforms.