

Recipe Generator Mobile App

Design Document

Ronan Green

C00270395

Contents

[Section 1: Introduction 3](#_Toc182827473)

[1.1 Project Overview 3](#_Toc182827474)

[1.2 Document Purpose 3](#_Toc182827475)

[Section 2: System Design 3](#_Toc182827476)

[2.1 High-Level System Overview 3](#_Toc182827477)

[2.2 Use Case Overview 3](#_Toc182827478)

[Section 3: Technologies Used 3](#_Toc182827479)

[3.1 Android Studio 3](#_Toc182827480)

[3.2 Kotlin 3](#_Toc182827481)

[3.3 MobileNet with TensorFlow Lite 3](#_Toc182827482)

[3.4 Spoonacular API / Open Food Facts 3](#_Toc182827483)

[3.5 Firebase 3](#_Toc182827484)

[3.6 External APIs (e.g., Barcode Scanning, AI Chatbot) 3](#_Toc182827485)

[Section 4: System Architecture 4](#_Toc182827486)

[4.1 Overview of System Architecture 4](#_Toc182827487)

[4.2 Component Architecture 4](#_Toc182827488)

[4.3 Data Architecture 4](#_Toc182827489)

[4.4 Integration with External Services 4](#_Toc182827490)

[Section 5: Detailed Use Cases 4](#_Toc182827491)

[Section 6: Critical Algorithms 4](#_Toc182827492)

[6.1 AI-Based Ingredient Identification 4](#_Toc182827493)

[6.2 Recipe Recommendation (Collaborative Filtering, Reinforcement Learning) 4](#_Toc182827494)

[Section 7: Sequence Diagrams 4](#_Toc182827495)

[7.1 Ingredient Identification Process 4](#_Toc182827496)

[7.2 Recipe Recommendation Interaction 4](#_Toc182827497)

[7.3 Barcode Scanning 4](#_Toc182827498)

[Section 8: Class Diagrams 4](#_Toc182827499)

[8.1 Major Components (User, Recipe, Ingredient, etc.) 4](#_Toc182827500)

[Section 9: Database Design 5](#_Toc182827501)

[9.1 Entity-Relationship Diagram 5](#_Toc182827502)

[9.2 Data Flow 5](#_Toc182827503)

[Section 10: UI/UX Design 5](#_Toc182827504)

[10.1 Wireframes and Screens 5](#_Toc182827505)

[10.2 Login and Registration Interfaces 5](#_Toc182827506)

[10.3 Ingredient and Recipe Interaction Screens 5](#_Toc182827507)

[10.4 Shopping List Management Screens 5](#_Toc182827508)

[Section 11: Security Considerations 5](#_Toc182827509)

[11.1 User Data Handling (GDPR compliance) 5](#_Toc182827510)

[11.2 Authentication and Authorization 5](#_Toc182827511)

[Section 12: Development Methodology 5](#_Toc182827512)

[12.1 Agile Approach and Iterations 5](#_Toc182827513)

[Section 13: Conclusion 5](#_Toc182827514)

[13.1 Summary of the App Design 5](#_Toc182827515)

[13.2 Next Steps in Development 5](#_Toc182827516)

# Section 1: Introduction

## Project Overview

EasyChef is an Android application designed to help users quickly find tasty recipes. The app's main feature is ingredient identification through image recognition, allowing users to take pictures of their ingredients and discover matching recipes.

## Document Purpose

The purpose of this document is to provide a detailed blueprint for EasyChef's internal design. It covers a wide range of components involved in the system, including the architecture, technologies, algorithms, diagrams, and designs that are crucial to the application.

This document illustrates the interactions between different system components using sequence diagrams, class diagrams, and wireframes, providing a detailed overview of both the UX and database design.

Additionally, it outlines the estimated time required for each section of the application, the importance of each feature, and potential challenges that may arise during the development process.

# Section 2: System Design

## 2.1 System Overview

As stated EasyChef is going to be created as an Android Application. The main features of the application are as follows:

* Image Recognition: Identifies ingredients captured in a photo from the user's Android device:
  + **Camera X**: Used to take pictures of ingredients at the user's discretion, providing the image input for identification.
  + **MobileNet**: A pre-trained model (trained on ImageNet and further on the Food-101 dataset) is used for accurate recognition of ingredients. Once recognised, the ingredients are added to a list and sent to the Spoonacular API.
* Recipe Retrieval: Fetches recipes based on the user's inputted ingredients:
  + **Ingredients for Recipes**: Users can add ingredients either manually or by scanning a barcode:
    - **ZXing**: An open-source barcode scanning library used to scan barcodes from food items.
    - **Open Food Facts API**: Once a barcode is scanned, it is sent to the Open Food Facts API to identify the matching ingredient or food item.
  + **Spoonacular API**: Recognised and inputted ingredients are sent to the Spoonacular API to find recipes that match the available ingredients.
  + **Open Food Facts API**: Used to retrieve additional information about ingredients, such as calories, carbohydrates, proteins, etc.
* **Shopping List Creation**: Allows users to manage the ingredients for their recipes and suggests items based on previous recipes:
  + **User Interface**: The shopping list is managed through a simple, intuitive interface where users can easily add, remove, and organise items.
  + **Component Flow:** The app utilises the data layer to fetch previously identified ingredients and recipes, generating personalised suggestions for what items to buy.

## 2.2 Data Storage Design

EasyChef will utilise Firebase as the primary data storage solution to ensure a reliable and seamless experience for users. The following is covered in the data storage design:

* **Database Type**: Firebase will be used as the data storage for EasyChef. Firebase offers a NoSQL database that is well-suited for storing hierarchical data like recipes, ingredients, and user information, as well as providing real-time synchronisation capabilities.
* **Data Categories**:
  + **User Data**: User profiles, including login information and preferences, will be securely stored in Firebase using its integrated authentication service.
  + **Ingredients and Recipes**: All recognised ingredients, manually added items, and recipe details will be stored in Firebase, enabling efficient retrieval and updates.
  + **Shopping Lists**: Shopping lists will also be stored in Firebase, allowing for easy management and access across multiple devices.
* **Data Synchronisation**: Firebase’s real-time capabilities will ensure that data is synchronised across devices automatically. This feature allows users to access and update their data seamlessly.
* **Data Security**:
  + **User Authentication**: Firebase Authentication will be used to ensure only authorised users can access their personal information.
  + **Image Deletion**: To ensure privacy, all images used for ingredient identification will be permanently deleted after processing. This helps prevent unnecessary storage of user data and adheres to privacy best practices.

# Section 3: Technologies Used

## 3.1 Android Studio

## 3.2 Kotlin

## 3.3 MobileNet with TensorFlow Lite

## 3.4 Spoonacular API / Open Food Facts

## 3.5 Firebase

## 3.6 External APIs (e.g., Barcode Scanning, AI Chatbot)

# Section 4: System Architecture

## 4.1 Overview of System Architecture

## 4.2 Component Architecture

## 4.3 Data Architecture

## 4.4 Integration with External Services

# Section 5: Detailed Use Cases

# Section 6: Critical Algorithms

## 6.1 AI-Based Ingredient Identification

## 6.2 Recipe Recommendation (Collaborative Filtering, Reinforcement Learning)

# Section 7: Sequence Diagrams

## 7.1 Ingredient Identification Process

## 7.2 Recipe Recommendation Interaction

## 7.3 Barcode Scanning

# Section 8: Class Diagrams

## 8.1 Major Components (User, Recipe, Ingredient, etc.)

# Section 9: Database Design

## 9.1 Entity-Relationship Diagram

## 9.2 Data Flow

# Section 10: UI/UX Design

## 10.1 Wireframes and Screens

## 10.2 Login and Registration Interfaces

## 10.3 Ingredient and Recipe Interaction Screens

## 10.4 Shopping List Management Screens

# Section 11: Security Considerations

## 11.1 User Data Handling (GDPR compliance)

## 11.2 Authentication and Authorization

# Section 12: Development Methodology

## 12.1 Agile Approach and Iterations

# Section 13: Conclusion

## 13.1 Summary of the App Design

## 13.2 Next Steps in Development