

Recipe Generator Mobile App

Design Document

Ronan Green

C00270395

Contents

[Section 1: Introduction 3](#_Toc182840967)

[1.1 Project Overview 3](#_Toc182840968)

[1.2 Document Purpose 3](#_Toc182840969)

[Section 2: System Design 3](#_Toc182840970)

[2.1 System Overview 3](#_Toc182840971)

[2.2 Data Storage Design 4](#_Toc182840972)

[Section 3: Technologies Used 5](#_Toc182840973)

[3.1 Android Studio 5](#_Toc182840974)

[3.2 Koitlin 6](#_Toc182840975)

[3.3 Firebase Firestore 6](#_Toc182840976)

[3.4 CameraX 7](#_Toc182840977)

[3.5 Tensoreflow Lie(Lite RT) 7](#_Toc182840978)

[3.6 TensorFlow/Keras 8](#_Toc182840979)

[3.7 MobileNet 8](#_Toc182840980)

[3.8 ZXing 9](#_Toc182840981)

[3.9 Open Food Facts API 9](#_Toc182840982)

[3.10 Spoonacular API 9](#_Toc182840983)

[Section 4: System Architecture 10](#_Toc182840984)

[4.1 Overview of System Architecture 10](#_Toc182840985)

[4.2 Component Architecture 10](#_Toc182840986)

[4.3 Data Architecture 10](#_Toc182840987)

[4.4 Integration with External Services 10](#_Toc182840988)

[Section 5: Detailed Use Cases 10](#_Toc182840989)

[Section 6: Critical Algorithms 14](#_Toc182840990)

[6.1 AI-Based Ingredient Identification 14](#_Toc182840991)

[6.2 Recipe Recommendation (Collaborative Filtering, Reinforcement Learning) 14](#_Toc182840992)

[Section 7: Sequence Diagrams 14](#_Toc182840993)

[7.1 Ingredient Identification Process 14](#_Toc182840994)

[7.2 Recipe Recommendation Interaction 14](#_Toc182840995)

[7.3 Barcode Scanning 14](#_Toc182840996)

[Section 8: Class Diagrams 14](#_Toc182840997)

[8.1 Major Components (User, Recipe, Ingredient, etc.) 14](#_Toc182840998)

[Section 9: Database Design 15](#_Toc182840999)

[9.1 Entity-Relationship Diagram 15](#_Toc182841000)

[9.2 Data Flow 15](#_Toc182841001)

[Section 10: UI/UX Design 15](#_Toc182841002)

[10.1 Wireframes and Screens 15](#_Toc182841003)

[10.2 Login and Registration Interfaces 15](#_Toc182841004)

[10.3 Ingredient and Recipe Interaction Screens 15](#_Toc182841005)

[10.4 Shopping List Management Screens 15](#_Toc182841006)

[Section 11: Security Considerations 15](#_Toc182841007)

[11.1 User Data Handling (GDPR compliance) 15](#_Toc182841008)

[11.2 Authentication and Authorization 15](#_Toc182841009)

[Section 12: Development Methodology 15](#_Toc182841010)

[12.1 Agile Approach and Iterations 15](#_Toc182841011)

[Section 13: Conclusion 15](#_Toc182841012)

[13.1 Summary of the App Design 15](#_Toc182841013)

[13.2 Next Steps in Development 15](#_Toc182841014)

# 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

**Android Studio**: The official IDE for Android development.

* **Features**:
  + Tools for designing, coding, and debugging Android applications.
  + **Layout Editor**: Drag-and-drop layout editor for building UIs efficiently.
  + **Profilers**: Tools for monitoring app performance, memory usage, and network activity.
* **Why It's a Good Choice**: Android Studio is the official IDE, packed with features tailored for Android development. Its deep integration with Android tools and services makes it the best choice for building, testing, and optimising the EasyChef app efficiently.

## 3.2 Koitlin

**Kotlin**: The main programming language used for developing the EasyChef Android application.

* **Features**:
  + **Concise Syntax**: Reduces boilerplate code compared to Java.
  + **Null Safety**: Prevents null pointer exceptions, leading to more robust applications.
  + **Interoperability**: Fully interoperable with Java, allowing seamless integration of existing Android components.
* **Why It's a Good Choice**: Kotlin is officially supported for Android development and offers a modern, expressive syntax that makes coding simpler and less error-prone. Its features make the app more maintainable and reliable.

## 3.3 Firebase Firestore

**Firebase Firestore**: A NoSQL cloud database used for storing ingredient history and scanned items.

* **Integration**: Integrates directly with Android via SDKs.
* **Benefits**:
  + **Real-Time Synchronisation**: Data is synced across devices in real time.
  + **Offline Access**: Users can continue managing their ingredients and recipes even without an internet connection.
  + **Scalability**: The flexible data model is ideal for storing unstructured data like recipes and user preferences.
* **Why It's a Good Choice**: Firestore is highly scalable and offers easy integration with Android, making it suitable for storing recipe data and managing user interactions seamlessly. Real-time updates ensure the best user experience, especially for collaborative features like shopping lists.

## 3.4 CameraX

**CameraX** is a Jetpack library designed to make the process of integrating camera features into Android applications easier and more consistent across different devices.

* **Compatibility**: Supports Android 5.0 (API Level 21) and higher, ensuring broad device compatibility. The current version is Android 15.
* **Features**:
  + **Preview**: View live camera footage on the display.
  + **Image Analysis**: Access image buffers for further processing, such as ingredient identification.
  + **Image Capture**: Capture still images for use in identifying ingredients.
  + **Video Capture**: Record videos with audio.
* **Device Consistency**: Overcomes discrepancies between different cameras by using an automated test lab to standardise camera behaviour.
* **Extensions**: The Extension API includes features like HDR and night mode, enhancing the camera's capabilities.
* **Why It's a Good Choice**: CameraX simplifies camera integration across devices, minimising compatibility issues and reducing development time. Its modern, easy-to-use interface makes it ideal for a project that heavily relies on image capturing for ingredient identification.

## 3.5 Tensoreflow Lie(Lite RT)

**TensorFlow Lite** is used for running machine learning models on mobile devices.

* **Purpose**: Optimised for on-device machine learning, TensorFlow Lite allows models to run without server connections, providing a fast, private, and reliable experience.
* **Key Features**:
  + **Low Latency**: Performs inference on-device, avoiding the need for network connections.
  + **Privacy**: Keeps data on-device, enhancing user privacy.
  + **Offline Capability**: No internet connection is required for using models.
  + **Lightweight**: Reduces the model size for optimal use on mobile devices.
* **Usage**: Models can be trained in TensorFlow and converted to .tflite format, or existing TensorFlow Lite models can be utilised.
* **Metadata**: TensorFlow Lite models can include metadata, simplifying pre- and post-processing pipelines.
* **Why It's a Good Choice**: TensorFlow Lite is highly efficient for on-device processing, which is essential for ingredient recognition without relying on internet access. Its small model size and optimised performance make it perfect for a mobile-based solution.

## 3.6 TensorFlow/Keras

**TensorFlow/Keras**: Used to train a custom machine learning model for ingredient detection.

* **Keras**: A user-friendly interface for TensorFlow that simplifies creating and training machine learning models.
* **Features**:
  + Offers simple, consistent APIs that reduce the complexity of writing ML models.
  + Includes pre-trained models for quick deployment and feature extraction.
  + Tools for data processing, hyperparameter tuning, and deployment.
* **Why It's a Good Choice**: Keras is known for its simplicity and flexibility, making it an ideal choice for prototyping and training custom ingredient detection models quickly. The integration with TensorFlow also means that models can be easily optimised and converted for mobile use.

## 3.7 MobileNet

**MobileNet** is a lightweight Convolutional Neural Network (CNN) architecture designed for mobile and embedded devices.

* **Purpose**: It is used for image classification and feature extraction with minimal computational resources.
* **Optimisation**: Uses **depthwise separable convolutions** to reduce the number of parameters and operations, making it lightweight.
* **Integration**: Can be paired with object detection frameworks, such as Single Shot Detector (SSD).
* **Why It's a Good Choice**: MobileNet is specifically designed for mobile environments, making it highly efficient for running on-device inference. Its lightweight nature ensures it runs smoothly on smartphones, which is perfect for real-time ingredient identification.

## 3.8 ZXing

**ZXing** ("Zebra Crossing") is an open-source library for barcode scanning.

* **Usage**: Used to scan 1D/2D barcodes from food items for easy ingredient input.
* **Implementation**: Implemented in Java, with ports to other languages, making it easy to integrate within Android.
* **Why It's a Good Choice**: ZXing is a well-established barcode scanning library with broad support and integration options. Its reliability and ease of use make it suitable for an app that needs to scan barcodes quickly to retrieve ingredient information.

## 3.9 Open Food Facts API

**Open Food Facts API**: A community-driven database providing information on various food products.

* **Purpose**: Used to retrieve ingredient details, nutritional values, and other product information.
* **Why It's a Good Choice**: Open Food Facts provides a vast amount of data for free, allowing for rich ingredient and nutritional information to be accessed. It helps provide accurate nutritional insights to users without additional development overhead.

## 3.10 Spoonacular API

**Spoonacular API**: A comprehensive API offering recipe suggestions, ingredient information, and meal planning tools.

* **Features**:
  + Provides a vast recipe database, ingredient nutritional information, meal planning, and cost estimation.
  + Offers detailed recipes filtered by dietary requirements and allergies.
* **Cost**: This is a paid service offering advanced features.
* **Why It's a Good Choice**: Spoonacular's vast recipe database, nutritional details, and advanced filtering options make it ideal for providing users with tailored recipe recommendations based on their available ingredients. Its depth of information and meal planning features make it highly versatile.

# 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

|  |  |
| --- | --- |
| Use Case Name | Create Recipe |
| Unique ID | WW015 |
| Actors | Users, Database, Spoonacular, AI Model |
| Description | This use case is initiated when a user selects create recipe. This feature will create a new recipe using the AI Model and the Spoonacular API. It will use like items from the user's liked foods and will prompt the user for recipe details. With the information it will display a recipe the user can save. |
| Preconditions | User is registered and logged in |
| Trigger | The user selects create recipe. |
| Main Path | 1. The user prompts the system to create the recipe. 2. The recipe uses liked foods and prompts the user for extra information to create the recipe. 3. Once the user has input the data it will use the AI Model and Spoonacular API to display a recipe for the user. 4. The user accepts the recipe, and it gets added to the database for the user to view. |
| Post Conditions | The user has an added recipe in the database |
| Alternative Flows | 4a:   1. The user rejects the recipe. 2. The system gets a new recipe and prompts it to the user. |

|  |  |
| --- | --- |
| Use Case Name | Take Picture of Ingredient |
| Unique ID | WW016 |
| Actors | User, AI Model, Spoonacular API |
| Description | The user opens the camera in the app and take a picture that include all relevant ingredients they wish to be a part of the recipe. The AI Model will identify the items and search the Spoonacular API for a relevant recipe. The user is then shown a recipe and they can decide whether they like it or not. If the user likes the recipe it will be added to the database. |
| Preconditions | * The user is registered and logged in. * The user has given the app permission to access the camera. |
| Trigger | The user opens the camera in the app. |
| Main Path | 1. User opens the camera 2. The user takes a picture of the ingredients they wish to have in the recipe. 3. The AI Model then recognises the ingredients identifying each one and then making a list to send to the Spoonacular API. 4. The user will get prompted to take another picture to add more ingredients 5. The Spoonacular API then checks for a matching recipe and then it get displays to the user 6. The user then accepts the recipe and the recipe gets added to the database for the user to view. |
| Post Conditions | New recipe is added to the database |
| Alternative Flows | 2a:   1. The user takes a picture of the barcode. 2. The item then gets checked against Open Food Facts API. 3. The item is temporarily stored, and the user is prompted to scan again   3a:   1. The system cannot pick up any items. This can be due to poor picture or unclear items. 2. The system prompts the user to take another picture.   6a:   1. The user rejects the proposed recipe, and the system then finds a new one. |

|  |  |
| --- | --- |
| Use Case Name | Interact with AI Chat Bot |
| Unique ID | WW017 |
| Actors | User, AI Model |
| Description | Using the OpenAI API as the base, this chatbot allows users to interact and talk about their recipes and shopping lists. Using the information from the AI Model (which contains like/dislikes) the chatbot will be able the help the user achieves the ideal recipe and shopping lists. |
| Preconditions | The user must be registered and logged into the system. |
| Trigger | The user selects the AI chatbot |
| Main Path | 1. The user is logged in an has selected the chatbot. 2. The user prompts the chatbot with a question(“I want more protein in my diet”) 3. Using the information provided by the system, such as user recipes, shopping list, preferences the chatbot thinks of a response 4. The chatbot will respond prompting the user to reply again. (“A good solution would be to replace recipe X’s chicken thighs with chicken breast, would you like me to make this change?”) 5. The user accepts this change the recipe will be updated accordingly. (“Yes”) |
| Post Conditions | The recipe has been updated in the database |
| Alternative Flows | 5a. User Rejects Recommendation:   1. The user rejects the chatbot suggestion and no changes are made. |

# Section 6: Critical Algorithms

## 6.1 AI-Based Ingredient Identification

## 6.2 Recipe Recommendation

# 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