Recipe Book Application Requirements Document

1. Introduction 3
    1.1 Purpose 3
    1.2 Scope 3
    1.3 Definitions, Acronyms, and Abbreviations 3
    1.4 References 4
2. Functional Requirements 5
    2.1 User Interface (UI) Requirements 5
    2.2 Recipe Management 5
    2.3 Image Handling 6
    2.4 Synchronization 6
    2.5 Timer Functionality 6
    2.6 Voice Control 6
    2.7 API for Database Management 7
3. Non-Functional Requirements 8
    3.1 Performance 8
    3.2 Scalability 8
    3.3 Usability 8
    3.4 Reliability 9
    3.5 Maintainability 9
    3.6 Portability 9
    3.7 Storage and Cost Management 9
4. System Architecture 10
    4.1 Application Layers 10
    4.2 Data Flow 10
    4.3 Technology Stack 10
    4.4 Deployment Model 11
    4.5 Security and Authentication 11
    4.6 Error Handling and Logging 11
5. Data Management 12
    5.1 Data Storage 12
    5.2 Data Access 12
    5.3 Data Synchronization 12
    5.4 Data Backup and Recovery 12
    5.5 Data Integrity and Validation 12
    5.6 Data Security 12
    5.7 Data Retention and Deletion 12

5.8 External API Integration for Nutritional Information 13
References **13**
Prompt: **15**
Formatting Guidelines 15
Iterative Development Process 16
Prompt 2: **17**
Iterative Prompt for Developing the Recipe Book Application 17
Example Workflow: 17

1. Introduction

1.1 Purpose

This document outlines the requirements for developing a cross-platform recipe book application that combines the user's passion for cooking and coding. The project serves as both a hobby and a learning opportunity, aiming to enhance coding skills and explore tools related to cloud computing, multi-device applications, and API development. The application will be accessible on both a Windows 11 PC and a Google Pixel 5 running the latest Android OS. It is designed to allow users to store, manage, and retrieve recipes with additional features such as image compression, cloud synchronization, voice control, and a custom API for database management, all while operating within the limits of free-tier cloud services.

1.2 Scope

The application will include the following features:

- **Recipe Entry and Management**: Users will be able to manually enter, edit, and manage recipes, including the addition of images.
- **Image Handling with Compression**: The application compresses images to optimize storage, and further compression is possible through ZIP archiving.
- **Synchronization Across Devices**: Recipe data and images will be synchronized between a Windows 11 PC and a Google Pixel 5 using cloud services. An offline mode will allow access to previously downloaded data.
- **Accessibility Features**: The application will include a dark mode for better visibility in bright environments and voice control to allow hands-free navigation.
- **API for Database Management**: The application will include a custom API to manage database interactions, ensuring consistent and secure data handling across platforms.

The application is intended for personal use and will be designed to operate within the limits of free-tier cloud services.

1.3 Definitions, Acronyms, and Abbreviations

UI: User Interface
UX: User Experience
API: Application Programming Interface
VM: Virtual Machine

1.4 References

Microsoft Azure and AWS Free Tier documentation.
Compression techniques and libraries such as Pillow, Sharp, and custom algorithms.

2. Functional Requirements

2.1 User Interface (UI) Requirements

- **2.1.1 Design Principles**: The user interface should be simple and easy to use, focusing on clarity and user-friendliness. The design should be professional and uncluttered, using colours strategically to guide the user's actions and emphasize important elements.
- **2.1.2 Platform Compatibility**: The UI must fit and scale across different screen resolutions, including 4K monitors, 1080p monitors, and mobile screens. The design must ensure readability and usability on all devices, particularly on smaller mobile screens where text and buttons must be easily accessible.
- **2.1.3 Animation and Interactivity**: The application should support minor animations to enhance the user experience, such as transitions between screens or animated feedback when buttons are clicked. These animations should be subtle and not distract from the application's primary functions.
- **2.1.4 Recipe Entry Interface**: The UI should support the entry of recipes with varying numbers of steps, providing a flexible and intuitive interface for users to add, edit, and manage recipes. The recipe entry screen should include clearly labelled fields for ingredients, steps, prep time, cook time, and serving size. It should also include proper headers to distinguish between different sections, such as "Prep" and "Cooking," to facilitate easy navigation.
- **2.1.5 Portion Scaling**: The UI should allow users to manually scale ingredient quantities based on the number of people eating. This feature should be easy to access and adjust and able to automatically recalculate ingredient amounts as needed.

2.2 Recipe Management

- **2.2.1 Manual Recipe Entry**: Users should be able to manually enter and edit recipes. Each recipe entry should include fields for:
    - **Ingredients**: List of ingredients with optional quantities.
    - **Steps**: Instructions for preparation, with support for multiple steps.
    - **Prep Time**: Estimated time for preparation.
    - **Cook Time**: Estimated time for cooking.
    - **Serving Size**: Number of servings the recipe provides.
- **2.2.2 Image Attachment**: Users can attach images to recipes. The application should support images in the WebP format for efficiency, using a custom lossless compression solution to minimize file size. If further compression is needed, images should be zipped before storage.
- **2.2.3 Recipe Tagging**: Implement a tagging system that categorizes recipes by:
    - **Primary Ingredient** (e.g., chicken, beef, mushroom).
    - **Dietary Preferences** (e.g., vegetarian, vegan, meat-based).

    - **Prep/Cook Time Blocks** (e.g., 0-10 minutes, 10-20 minutes, 20-30 minutes, 30+ minutes).
- **2.2.4 Search Functionality**: Users should be able to search for recipes by name or by tags, allowing for quick and easy access to the desired recipes.

2.3 Image Handling

- **2.3.1 Image Formats and Compression**: The application will primarily use the **JPEG XL** format for images due to its superior compression efficiency and advanced features. Given its widespread support and effectiveness, WebP will be used as a fallback option if necessary. Lossless compression methods should be implemented to maintain image quality while reducing file size.
- **2.3.2 Additional Compression**: Zipping will be considered as an additional step to compress images further if space becomes a critical issue, though this will be a stretch goal depending on the overall storage usage.

2.4 Synchronization

- **2.4.1 Data Sync Across Devices**: Recipe data and images should be synchronized between the Windows 11 PC and the Google Pixel 5. Upon startup, the application should check with the server for any changes and download new or updated data.
- **2.4.2 Offline Functionality**: The application should function offline, allowing users to access previously downloaded recipes and images without an internet connection. Users should also be able to manually sync data, allowing them to control when the app checks for updates to avoid excessive mobile data usage.

2.5 Timer Functionality

- **2.5.1 Timer Features**: The timer feature should include basic functionality such as start, pause, reset, and adjust during the countdown. Cross-device synchronization should ensure that timers set on one device are reflected on the other, with notifications appearing on both.
- **2.5.2 Stretch Goals**: Additional features such as multiple timers or customizable alerts (sound/vibration options) can be implemented if time and resources allow.

2.6 Voice Control

- **2.6.1 Voice Commands**: Implement voice commands for mobile devices to allow hands-free navigation through recipes. Commands should include:
    - **Scrolling**: “Scroll down/up: %” to scroll by a specific percentage of the page.
    - **Jumping to Sections**: “Scroll to: [Location]” to jump to predefined sections such as “Prep” or “Cooking.”

- **2.6.2 Activation Phrase**: The voice control system should be activated using a custom phrase like “Please Chef,” followed by the command. The system should operate similarly to virtual assistants like Siri or Google Assistant.

2.7 API for Database Management

- **2.7.1 Purpose of the API**: The application will include a custom API to manage database interactions, providing a consistent and secure way to handle data across both mobile and PC platforms.
- **2.7.2 Benefits**: The API will facilitate cross-platform consistency by ensuring that mobile and PC applications interact with the database in the same way. This approach will simplify the codebase for data handling and enhance the application's scalability and maintainability.
- **2.7.3 API Functions**: The API will support all necessary CRUD operations (Create, Read, Update, Delete) for managing recipes, images, and user data. It will also handle authentication, ensuring that only authorized users can access or modify data.
- **2.7.4 Security Considerations**: The API will implement security measures such as encryption, authentication, and access control to protect user data and ensure the security of all interactions.

3. Non-Functional Requirements

3.1 Performance

- **3.1.1 Response Time**: The application should respond quickly, with a target load time of under 1 second for all main functions, such as loading recipes, switching between screens, and retrieving data.

3.2 Scalability

3.2.1 Data Capacity: The application should be able to store at least 200 recipes, each containing one image, within a 5GB storage limit.

- **Storage Feasibility**: Use JPEG XL compression to fit within storage requirements:
    - **Lossless Compression** reduces image sizes by 20-40%, with an estimated storage range of 362MB to 962MB.
    - **Lossy Compression** reduces image sizes by 60-80%, with an estimated storage range of 122MB to 482MB.
- This approach ensures scalability and efficient use of cloud storage while balancing image quality and storage efficiency.

3.3 Usability

Include the following references directly in the **Non-Functional Requirements** section of your requirements document to specify adherence to established usability standards:

- **Nielsen's Usability Heuristics**:
  These heuristics provide principles for user interface design. They ensure a user-friendly experience by focusing on aspects such as visibility, user control, consistency, error prevention, and simplicity.

- **Web Content Accessibility Guidelines (WCAG 2.1)**:
  These guidelines focus on making web content accessible to people with disabilities. They cover perceivability, operability, understandability, and robustness.

- **Google Material Design Guidelines**:
  Provides best practices for designing intuitive, accessible, and engaging user experiences on mobile and multi-platform applications.

3.4 Reliability

- **3.4.1 Data Recovery**: Data should be easily recoverable in case of loss or corruption. The chosen cloud service must provide automatic backups within free-tier guidelines.

3.5 Maintainability

- **3.5.1 Code Quality**: The application should be well-documented and follow best coding practices.

3.6 Portability

- **3.6.1 Target Platforms**: The application must be fully compatible and perform reliably on both a Windows 11 PC and a Google Pixel 5 running the latest Android OS.

3.7 Storage and Cost Management

- **3.7.1 Storage Cost Constraints**:
    - The application must operate within a free or low-cost storage solution wherever possible, prioritizing cost-effectiveness for long-term use.
    - After the 12-month free tier period ends, the total monthly cost for storage should not exceed £5/month for up to 5GB of data.
- **3.7.2 Estimated Storage Costs**:
    - **Azure Hot Blob Storage**: This tier is priced at £0.0141 per GB per month and is suitable for frequently accessed or modified data.
        - Estimated monthly cost for **5GB**: **£0.0705** (~£0.07).
        - Estimated monthly cost for **10GB**: **£0.141** (~£0.14).
- **3.7.3 Cost Projection Over Time**:
    - **First 12 Months (Free Tier)**:
        - No storage cost due to Azure and AWS free tier offerings of 5GB.
    - **After 12 Months**:
        - For **5GB** of Azure Hot Blob Storage:
            - Monthly: **£0.07**; Yearly: **£0.84**.
        - For **10GB** of Azure Hot Blob Storage:
            - Monthly: **£0.14**; Yearly: **£1.68**.

4. System Architecture

4.1 Application Layers

- **4.1.1 Presentation Layer**:
    - Implemented using **.NET MAUI** to create a cross-platform user interface compatible with both Windows and Android devices. This layer will handle user interactions, display data, and send user input to the Business Logic Layer.
- **4.1.2 Business Logic Layer**:
    - Developed in **C#**, this layer will contain all the core functionalities, such as processing recipe data, managing synchronisation between the local device and cloud storage, handling data conversions, and implementing business rules.
- **4.1.3 Data Access Layer (DAL)**:
    - Use **Entity Framework (EF)** for standard CRUD operations with the Azure SQL Database. Direct SQL queries can be used to optimise or complex queries where necessary. The DAL will interact with the database through the API layer.
- **4.1.4 Database Layer**:
    - The Azure SQL Database will store all recipe data. The database schema will be designed to support different segments of a recipe (e.g., ingredients, prep instructions, cooking instructions) and to optimise storage and retrieval performance.

4.2 Data Flow

- **4.2.1 Initial Data Retrieval**: When the app starts, it will call the API to check for any updates and download only the necessary recipe data (excluding images initially) to minimise local storage usage.
- **4.2.2 Lazy Loading for Images**: Images will be loaded from Azure storage only when a recipe is accessed to reduce data transfer and storage costs.
- **4.2.3 Data Synchronization**: The application will periodically or when triggered by specific events (e.g., saving a new recipe) synchronise data between the local device and the Azure SQL database.

4.3 Technology Stack

- **4.3.1 Programming Language**: C# using .NET MAUI for cross-platform compatibility.
- **4.3.2 Database**: Azure SQL Database for storing recipe data.
- **4.3.3 API**: Custom RESTful API developed using **ASP.NET Core** for secure app and Azure SQL database interaction.

- **API Rate Limiting**: Implement rate limiting to prevent excessive API calls, using techniques such as token buckets or leaky buckets to manage request rates and avoid overloading the server or incurring additional costs.

4.4 Deployment Model

- The application will be deployed using **Azure** for cloud storage and database management. Both the Windows and Android versions will utilise the same backend infrastructure on Azure.

4.5 Security and Authentication

- **4.5.1 Data Transmission**: Use HTTPS to secure data transmission between the app and the Azure API.
- **4.5.2 Minimal Encryption**: Given the non-sensitive nature of the data (recipes and images), use lightweight encryption to avoid increasing data size.

4.6 Error Handling and Logging

- **4.6.1 Error Handling**: Implement robust error handling, especially for API calls, to prevent excessive requests that could incur additional costs. Use retry logic with exponential backoff for API calls.
- **4.6.2 Logging**: Utilize a logging framework in C# to capture errors, monitor app performance, and assist in debugging. Recommended logging libraries include:
  - **Serilog**: A popular logging library for .NET that allows for structured logging and easy integration with various outputs (e.g., files, databases, cloud logging services). Supports rich data formats like JSON.
  - **NLog**: Another robust logging framework for .NET applications, offering flexible configuration, various targets (file, email, database), and support for asynchronous logging to improve performance.
  - **Microsoft.Extensions.Logging**: A lightweight, built-in logging framework that integrates with ASP.NET Core. It is helpful for basic logging needs and can be extended with other logging providers like Serilog or NLog.

5. Data Management

5.1 Data Storage

- The application must store all recipe data, including ingredients, instructions, presentation details, dietary information, and any associated metadata, in a secure and scalable database.
- The storage solution must support efficient querying and retrieval of recipes, ingredient lists, and nutritional information.

5.2 Data Access

- The application must use a secure and efficient method for accessing and modifying data stored in the database. To ensure security and consistency, all data access should be performed through a custom RESTful API.
- Rate limiting should be implemented to avoid excessive database calls.

5.3 Data Synchronization

- The application must synchronise local data on the device (Windows and Android) with the cloud-based database to ensure consistency. Synchronisation should occur periodically or upon specific actions (e.g., adding or editing a recipe).

5.4 Data Backup and Recovery

- The system must perform automatic backups of recipe data weekly, retaining backups for up to five weeks to ensure recoverability in case of data loss or corruption.
- The backup process should exclude images to minimise storage costs.

5.5 Data Integrity and Validation

- The system must enforce data validation rules to prevent inconsistent or invalid data entries.
- Transaction management must be in place to ensure that all data modifications are successfully completed or appropriately rolled back.

5.6 Data Security

- All data transmissions between the application and the cloud storage must be secured using HTTPS.
- Data should be encrypted as needed to balance security with performance.

5.7 Data Retention and Deletion

- The system must retain all recipe data permanently unless manually deleted by the user.
- A "soft delete" mechanism should be implemented to prevent accidental data loss and manage deletion grace periods.

5.8 External API Integration for Nutritional Information

- The application must integrate with an external API (e.g., Spoonacular) to provide automated nutritional analysis for recipes.
- The API integration should operate within the limits of the chosen provider's free tier (e.g., up to 150 calls per day).

References

1. Nielsen, J. (1994). "10 Usability Heuristics for User Interface Design." Nielsen Norman Group. Available at: https://www.nngroup.com/articles/ten-usability-heuristics/

2. World Wide Web Consortium (W3C). (2018). "Web Content Accessibility Guidelines (WCAG) 2.1." Available at: https://www.w3.org/WAI/WCAG21/quickref/

3. Google. (2023). "Material Design Guidelines." Available at: https://m2.material.io/design/guidelines-overview

Summary Prompt

Requirements Document Summary:

1. **Overview:**
    - The project involves developing a cross-platform recipe book application as a personal learning opportunity to combine a passion for cooking and coding.
    - The app will be accessible on both a Windows 11 PC and a Google Pixel 5 running Android OS.
    - Key features include recipe entry and management, image compression, cloud synchronization, voice control, and a custom API for database management.
2. **Functional Requirements:**
    - **Recipe Management:** Allows manual entry, editing, tagging, and searching of recipes.
    - **Image Handling:** Uses JPEG XL compression (lossless and lossy) to optimize image storage within a 5GB limit.
    - **Data Synchronization:** Ensures consistency between local and cloud-based data with synchronization occurring periodically or upon specific actions.
    - **Voice Control:** Implements voice commands for mobile devices for hands-free navigation.
    - **Timer Functionality:** Timers can be set on either device and sync between them with customizable notification settings.
3. **Non-Functional Requirements:**
    - **Performance:** Target load time of under 1 second for main functions.
    - **Scalability:** Should support up to 200 recipes and images within a 5GB storage limit.
    - **Usability:** Adheres to established usability standards like Nielsen's Heuristics, WCAG 2.1, and Google Material Design Guidelines.
    - **Reliability:** Data should be recoverable; Azure SQL's backup options are utilized.
    - **Maintainability:** Code should be well-documented and follow best coding practices.
    - **Portability:** Fully compatible with Windows 11 PC and Google Pixel 5.
    - **Storage and Cost Management:** Prioritizes cost-effective storage solutions within free or low-cost tiers.
4. **Data Management:**
    - **Database Design:** Uses Azure SQL with tables for Recipes, Ingredients, Instructions, and Images, following normalization principles.
    - **Backup Strategy:** Weekly backups retained for 5 weeks, excluding images to minimize costs.

- **API Integration:** Utilizes Spoonacular API for automated nutritional analysis within free-tier limits.

Technical Documentation Summary:

1. **Database Schema:**
   - **Ingredient Reference Table:** Stores unique, standardized ingredient names and details.
   - **Recipes Table:** Contains basic information about each recipe.
   - **RecipeIngredients Table (Junction Table):** Manages the many-to-many relationship between recipes and ingredients.
   - **Instructions Table:** Stores step-by-step instructions for each recipe.
   - **Images Table:** References recipe images stored in Azure Blob Storage.
2. **Data Management:**
   - **Standardization and Fuzzy Matching:** Automatically standardizes ingredient entries upon input and uses fuzzy matching to prompt for potential duplicates.
   - **Data Synchronization:** Initial data load at startup and lazy loading for images to minimize data transfer.
   - **Error Handling:** Covers various scenarios like database connection errors, data retrieval errors, API errors, and client-side application errors.
3. **Image Compression and Storage:**
   - **JPEG XL Compression:** Both lossless and lossy methods fit within a 5GB storage limit, balancing image quality and storage efficiency.
   - **Implementation Notes:** Use of libraries like libjxl for compression; fallback to WebP if necessary.
4. **Security Considerations:**
   - **Data Security:** Use HTTPS for secure communication and minimal encryption for sensitive fields.
   - **Storage Management:** Handle storage limits and avoid early deletion fees with strategic backup and data retention policies.
5. **API Integration:**
   - **Spoonacular API:** For nutritional analysis with a simple implementation to send ingredient lists and receive back detailed nutritional information.

Next Steps:

1. **Complete Technical Documentation:**
   - Flesh out specific implementation details, particularly around data handling, synchronization, API integration, and error management.
2. **Design the User Interface (UI):**

    - Create GUI mockups for main screens, including Recipe List, Recipe Details, Add/Edit Recipe, and Manage Ingredients.
    - Ensure accessibility features are considered (e.g., dark mode, text size adjustments).
3. **Prototype and Test:**
    - Develop a basic prototype with core screens and navigation.
    - Perform initial usability testing to validate design choices and make necessary adjustments.