TED UNIVERSITY

TED UNIVERSITY

SENG 491

High Level Design Report

Project Name: Fridge Genius

Team Name: BIS

Team Members:

İrem Güngör – 61375493682

Burcu Gül – 38113171904

Sude Sarı – 24482515562

Advisor:

Elif Kurtaran Özbudak

Jury Members:

Tansel Dökeroğlu

Eren Ulu

Table of Contents

1 Introduction

1.1 Purpose of the Report

The purpose of this High-Level Design report is to outline the architectural framework, design principles, and functional components of the Fridge Genius system. This document provides a structured overview of how the system's components will interact, focusing on the design strategies employed to deliver an efficient and scalable solution for food inventory management.

Fridge Genius is designed to revolutionize the way users manage their refrigerators by integrating image recognition, artificial intelligence, and user-friendly interfaces. Its goal is to minimize food waste, streamline meal planning, and promote sustainable consumption habits. By digitizing refrigerator management, the system offers personalized recipe suggestions and automates the creation of shopping lists, providing a seamless and intuitive experience.

This report serves to detail the system's objectives, subsystems, and technical components, ensuring alignment with its purpose of reducing waste, saving time, and fostering a sustainable lifestyle through innovative technology.

1.2 Design Goals

User-Centric Design

- Provide an intuitive user experience for users.
- An interface that is simple to navigate, ensuring quick and easy access to core functionalities such as food recognition, recipe suggestions, and shopping list management.

Performance and Responsiveness

- Fast and efficient image recognition, with processing times that do not exceed 3 seconds per image.
- Provide rapid responses for recipe generation and shopping list updates, maintaining an average completion time of 5 seconds or less.

Scalability and Flexibility

- Modular architecture that supports future feature expansions, such as Android compatibility or integration with smart appliances.
- System to handle an increasing user base and higher data loads without changing performance.

Security and Privacy

- Prioritize data security by encrypting sensitive information both at rest and in transit using industry-standard protocols.
- Data privacy regulations such as GDPR, ensuring transparency and giving users full control over their data.
- Secure authentication mechanisms to protect user accounts and ensure safe access to personal information.

Reliability and Robustness

- Application designed to function reliably under various conditions, including intermittent network connectivity or limited device resources.
- Robust error-handling mechanisms to provide meaningful feedback to users in case of failures (API errors or connectivity issues).
- Application maintains a 99.9% uptime for core functionalities such as login, food recognition, and recipe suggestions.

Maintainability

- Modular codebase to simplify the addition of new features, updates, and bug fixes.
- Clear and comprehensive documentation to support future development efforts.

Sustainability

- Encourage users to minimize food waste by providing timely notifications about expiring items and suggesting relevant recipes.
- Promote responsible consumption habits through analytics and personalized shopping suggestions.

Innovation

- Use AI and machine learning technologies to continuously improve the accuracy and relevance of food recognition and recipe suggestions.
- Integrate with emerging technologies, such as IoT-enabled appliances or advanced data analytics platforms.

1.3 Definitions, Acronyms, and Abbreviations

This section provides a comprehensive glossary of technical terms, acronyms, and abbreviations used in the Fridge Genius project. The definitions aim to ensure clarity and consistency throughout the report.

1.3.1 Definitions

- **Artificial Intelligence (AI):** A branch of computer science that enables machines to perform tasks that typically require human intelligence, such as visual perception, speech recognition, decision-making, and translation.
- **Image Recognition:** The process of identifying and classifying objects, items, or elements within digital images using AI algorithms.
- **Inventory Management:** A systematic approach to managing and monitoring the status, quantity, and location of items within a storage system, in this case, a digital representation of a fridge.
- **Recommendation System:** A system that provides personalized suggestions, such as recipes or shopping lists, based on user preferences, habits, and available inventory data.
- **Data Privacy:** Practices and technologies that ensure user data is stored securely and is not accessed or shared without permission.
- **Scalability:** The ability of the system to handle increased load (e.g., more users, larger databases) without impacting performance.

- **Cloud Integration:** The use of cloud-based services for storing, managing, and analyzing data, ensuring accessibility and scalability.

1.3.2 Acronyms

- **AI:** Artificial Intelligence
- **API:** Application Programming Interface
- **GDPR:** General Data Protection Regulation
- **IoT:** Internet of Things
- **UML:** Unified Modeling Language
- **SQLite:** Structured Query Language Lite (a lightweight local database management system)
- **TLS:** Transport Layer Security (a cryptographic protocol ensuring secure data transmission)

1.3.3 Abbreviations

- **UI:** User Interface
- **UX:** User Experience
- **DB:** Database
- **HTTP:** HyperText Transfer Protocol

1.3.4 Notes on Usage

- These terms, acronyms, and abbreviations are referenced throughout the report to describe the technical aspects of the Fridge Genius system.
- This glossary is particularly useful for readers who may not have a technical background but wish to understand the core functionalities and underlying technologies of the project.

1.4 Overview

The Fridge Genius system integrates cutting-edge hardware components and software solutions to create an intelligent food management ecosystem. Designed with a focus on efficiency and sustainability, the system leverages image recognition hardware, AI-powered analysis, and robust data management capabilities to enhance user experience.

The core of the hardware design includes a high-resolution camera module for food recognition, allowing users to capture images of their refrigerator contents. This is supported by a local processing unit that preprocesses the images before transmitting data to external AI services for analysis. A storage module, integrated with SQLite, ensures reliable data handling and facilitates cross-device synchronization via cloud integration.

To interact with users, the system features an intuitive touchscreen display or can integrate with existing smartphones and tablets, providing seamless access to personalized recipe recommendations and shopping list suggestions. Advanced connectivity features, such as Wi-Fi and Bluetooth, enable the system to communicate with external APIs (e.g., Gemini AI API) and sync with other smart home devices.

The hardware-software synergy ensures that Fridge Genius operates as a cohesive unit. Hardware components work in tandem with subsystems like AI for food recognition, recommendation engines for personalized outputs, and data management for secure and efficient storage. This comprehensive architecture aims to revolutionize food management by minimizing waste, saving time, and promoting sustainable practices.

2 Current Software Architecture

2.1 Overview of the Current Software Architecture

The Fridge Genius application follows a client-server architecture with a focus on modularity and local data handling to enhance performance and usability. To provide shopping list management, recipe recommendations, and food identification, the architecture combines many different parts.

Modules:

1. **Frontend (Mobile App):**
    - Built with Flutter, providing a cross-platform user interface and easy navigation.
    - Handles user interactions such as clicking buttons, including image capture, uploading image, database queries, and API responses.
2. **Backend Services:**

- Integrates with the Gemini AI API to handle:

    Food recognition from uploaded images.
    Recipe generation based on stored food items.

- Implements retry mechanisms and error handling for robust API communication.

3. **Local Database:**
    - SQLite database used for offline storage of recognized food items.
    - Stores user-specific inventory data for fast retrieval and API calls.
4. **Image Processing Module:**
    - Compresses and encodes images before sending them to the backend API.
    - Utilizes the image package for efficient image compression.
5. **AI Integration Module:**

- Interacts with the Gemini AI API for food recognition based on uploaded images, recipe generation using stored food items.

Features:

Sends image data or text prompts to the API.
Processes API responses for user display.
Implements retry mechanisms and error handling.

6. **Inventory Management Module:**

- Tracks and manages the user's fridge contents.
- Updates inventory when items are added.
- Allows manual edits for unrecognized items (e.g., quantity, expiration date) and remove items
- Ensures synchronization between inventory and recipe generation.

7. **Recipe Suggestion Module:**

- Generates personalized recipes based on the user's stored food items.

Features:

Retrieves inventory from the local database.
Sends prompts to the Gemini AI API for recipe suggestions.
Displays detailed recipe steps and allows users to mark recipes as favorites.

Data Flow:

1. Users interact with the mobile app to capture or select an image.
2. Images are compressed locally and sent to the Gemini AI API.
3. The API processes the image and returns recognized food items.
4. Recognized items are saved to the SQLite database for offline access.
5. Users can request recipe suggestions based on the stored inventory, which is retrieved locally and sent as prompts to the API.

2.2 Problems with the Current Software Architecture

1. **API Reliance:**
    - Due to the system's heavy reliance on external APIs (such Gemini AI), delay, rate limitations, and outages may occur.
2. **Limited Offline Functionality:**
    - All application functionalities such as food recognition and recipe generation require an internet connection, limiting the app's usability in offline scenarios.
3. **Scalability:**
    - Current architecture does not optimally handle a growing user large inventory, potentially leading to performance bottlenecks.

4. **Error Handling:**
    - Limited granularity in error messages can result in vague feedback, which makes it more difficult for users to solve problems efficiently.
5. **Performance Bottlenecks:**
    - Image compression and API communication can be resource intensive.
6. **Data Security:**
    - While some encryption is in place, storing sensitive API keys directly in the code poses a security risk.

2.3 Challenges and Constraints

Technical Challenges:

1. **Network Dependency:**
    - Functions depend on an active internet connection, which could frustrate users in areas with poor connectivity.
2. **Device Compatibility:**
    - The app must support a range of iOS devices with varying processing power and screen sizes.

Operational Challenges:

1. **API Rate Limits:**
    - API restrictions could lead to a degraded user experience during peak usage.
2. **Data Privacy and Compliance:**
    - Ensuring compliance with GDPR and other privacy regulations requires careful handling of user data.

Constraints:

1. **Platform Limitation:**
    - Currently restricted to iOS devices, with Android support planned for future iterations.
2. **Budget Constraints:**
    - Limited resources for additional API calls, server scaling, and advanced feature development.

3 Proposed Software Architecture

3.1 Overview

The Fridge Genius project is designed as a cutting-edge mobile application to transform food inventory management, reduce waste, and optimize meal preparation processes. The system

leverages advanced technologies, including **AI-powered image recognition**, **personalized recommendation systems**, and **data-driven analytics** to provide a seamless and efficient user experience. The modular architecture ensures scalability, maintainability, and flexibility for future enhancements.

System Architecture

Fridge Genius employs a **layered system architecture**, where each layer handles specific responsibilities and interacts with others to achieve a cohesive functionality. The architecture is divided into the following layers:

1. **Presentation Layer:**
    - Acts as the user-facing interface, designed using the Flutter framework.
    - Allows users to:
        - Capture or upload images for food recognition.
        - View and manage fridge inventory.
        - Request recipes and generate shopping lists.
        - Customize preferences and settings.
    - Provides a responsive and user-friendly design to cater to a diverse audience.
2. **Application Logic Layer:**
    - Orchestrates the core workflows, including:
        - Image processing and API communication.
        - Data synchronization between the local database and the Gemini AI API.
        - Notification scheduling for expiring items.
    - Handles user authentication, error management, and other business logic.
3. **Data Management Layer:**
    - Implements an **SQLite database** for local storage of recognized food items, inventory details, user preferences, and shopping history.
    - Ensures offline functionality by allowing users to manage inventory and access recipes even without an internet connection.
4. **Integration Layer:**
    - Communicates with the **Gemini AI API** for:
        - Food recognition based on uploaded images.
        - Recipe generation using AI-powered natural language processing (NLP).
    - Includes retry mechanisms and robust error handling for reliable data exchange.

Core Functionalities

1. **Food Recognition:**
    - Utilizes AI to identify food items from images captured by the user.
    - Updates the local inventory database with recognized items.
2. **Recipe Suggestions:**
    - Provides personalized recipes based on:
        - Available inventory items.
        - User preferences (e.g., allergens, favorite ingredients).
        - Items nearing expiration.
3. **Inventory Management:**
    - Tracks the quantity, expiration dates, and availability of food items.
    - Allows manual updates for unrecognized items or adjustments.
4. **Shopping List Generation:**
    - Suggests items to purchase based on:
        - Missing ingredients for favorite recipes.
        - Frequently consumed or low-stock items.

5. **Notifications:**
    - Alerts users about expiring items and relevant recipes to minimize waste.

Design Principles

- **Modularity:** Each subsystem operates independently but integrates seamlessly.
- **Scalability:** The system can handle a growing user base and larger data loads without performance degradation.
- **Security:** Data is encrypted during storage and transmission, ensuring compliance with GDPR standards.
- **Reliability:** Robust error-handling mechanisms ensure system stability under various conditions.

This architecture supports the Fridge Genius vision of being a smart, sustainable, and user-centric food management solution.

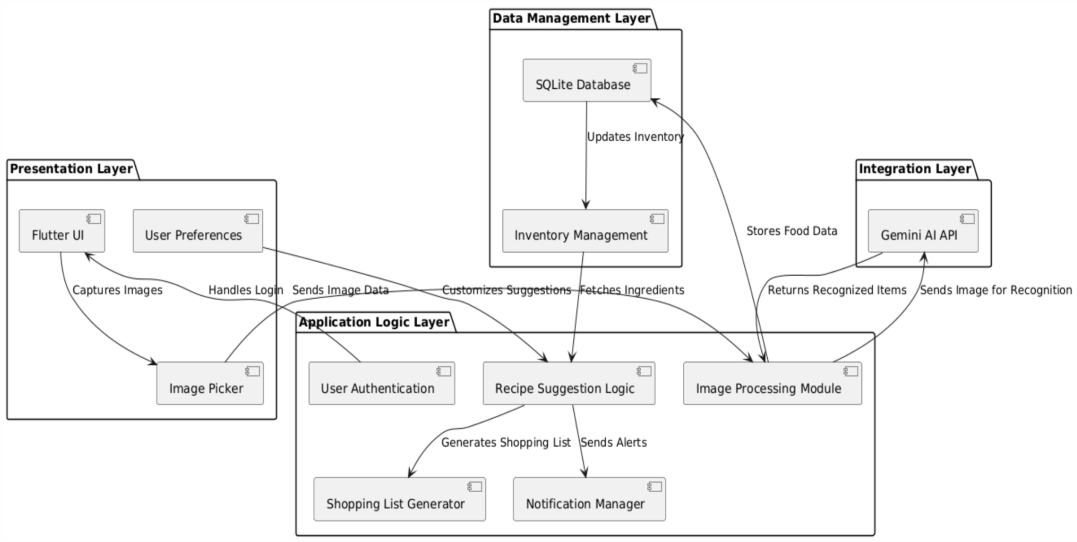

Figure 1: FridgeGenius System Overview Diagram

3.2 Subsystem Decomposition

The Fridge Genius application is structured into four interconnected subsystems, each of which plays a critical role in providing a smooth, intelligent and user-friendly experience. These subsystems — User Interface, Recommendation, Artificial Intelligence and Data Management - collaborate to transform the process of managing food stocks and meal planning.

The User Interface Subsystem serves as the primary interaction layer where users interact with the application. From uploading food images to browsing recipe suggestions and managing shopping lists, this subsystem provides intuitive and efficient navigation. Visual elements such as categorized dashboards, notifications and interactive menus enhance the user experience. For example, users receive timely alerts about products that are about to expire and trigger actions such as recipe creation or shopping list updates.

Below the surface, the Recommendation Subsystem functions as the brain of the application, using inventory data and user preferences to create contextually relevant recommendations. It uses

artificial intelligence-driven analytics to suggest recipes tailored to current ingredients and dietary restrictions. For example, it prioritizes recipes that use expired products in line with the system's goal of reducing food waste. In addition, the shopping list function automatically identifies missing or low-Octane items according to user habits, providing intelligent and personalized recommendations.

These capabilities are supported by the AI Subsystem , which integrates advanced technologies such as image recognition and natural language processing (NLP). When users upload pictures of food, the AI Subsystem processes these entries through the Gemini AI API and extracts structured data such as item names and quantities. Similarly, recipe production uses generative artificial intelligence to create detailed, customized recipes. These capabilities are supported by the AI Subsystem, which integrates advanced technologies such as image recognition and natural language processing (NLP). When users upload pictures of food, the AI Subsystem processes these entries through the Gemini AI API and extracts structured data such as item names and quantities. Similarly, recipe production uses generative artificial intelligence to create detailed, customized recipes. The Artificial Intelligence Subsystem collaborates seamlessly with the Recommendation Subsystem by feeding the processed data, enabling users to get actionable and intelligent outputs.

Supporting all these interactions is the Data Management Subsystem, which is the backbone of the Refrigerator Genius. This subsystem securely stores and manages data about inventory, user preferences, recipes and shopping history in a Supporting all these interactions is the Data Management Subsystem, which is the backbone of the Refrigerator Genius. This subsystem securely stores and manages data about inventory, user preferences, recipes and shopping history in a local SQLite database. It ensures data consistency, syncs with the cloud for cross-device access, and verifies inputs to maintain accuracy (for example, Decrying invalid expiration dates). The Data Management Subsystem is also critical for integration with external services, such as caching API responses to optimize performance and minimizing unnecessary calls to the AI Subsystem.

In summary, the subsystems work in harmony: the User Interface provides users with actionable insights, the Recommendation Subsystem determines what these insights should be, the artificial Intelligence Subsystem generates the raw data for these insights, and the Data Management Subsystem ensures that all data flows safely and efficiently. Together, these subsystems enable Fridge Genius to achieve its mission of simplifying food management, reducing waste and promoting sustainable consumption.

3.3 Hardware/ Software Mapping

The Fridge Genius project heavily relies on software components to deliver its core functionalities, such as food recognition, inventory management, and recipe generation. This section explains how the software components are organized and interact to ensure modularity, scalability, and efficiency. External servers and cloud services are utilized for AI-based image recognition and recipe generation.

Software Architecture Layers

The Fridge Genius employs a layered architecture where different layers are responsible for specific aspects of functionality. These layers are mapped to ensure modularity, maintainability, and scalability.

1. Presentation Layer

- User interface manages user interactions and displays results.
- **Components**:
  - **Flutter Framework**:
    - Builds the user interface, including buttons, menus, and result displays.
    - Uses widgets like Scaffold, ElevatedButton, and Text for intuitive UI design.
  - **Authentication**:
    - Allows users to sign in, sign up and sign out by their credentials.
  - **Image Picker Integration**:
    - Allows users to select images from the gallery or capture them via the camera.
- **Interactions**:
  - Sends user commands (e.g., image selection) to the Application Logic Layer.
  - Displays processed results from the lower layers.

2. Application Logic Layer

- Implements the core business logic for processing and managing data.
- **Components**:
  - **Login Module**:
    - Provides secure access to user-specific data and personalizes the user experience.
  - **Image Processing Module**:
    - Uses the image package to compress and encode images for API compatibility.
  - **Food Recognition Logic**:
    - Sends preprocessed image data to the AI Integration Module.
    - Processes API responses to extract recognized food items.
  - **Inventory Management**:
    - Sends stored food data to the database and manages items.
  - **Recipe Suggestion Logic**:
    - Gathers stored food data from the database and creates API prompts for recipe suggestions.
  - **Notification Management**:
    - Schedules and sends reminders for expiring items or recipe ideas.
  - **Profile And Settings**:
    - Allows users to manage their personal information, preferences, and app-specific configurations.
- **Interactions**:
  - Interfaces with both the Presentation and Data Management Layers.
  - Orchestrates workflows like food recognition, recipe generation, and notification scheduling.

3. Data Management Layer

- Handles the storage, retrieval, and management of user data.
- **Components**:
  - **SQLite Database**:

        - Stores recognized food items, their expiration dates, quantities and if it is favorite or not.
        - Provides local persistence for offline functionality.
    - **CRUD Operations**:
        - Create, read, update, and delete inventory data as users manage their fridge contents.
- **Interactions**:
    - Works with the Application Logic Layer to provide and update data.
    - Ensures efficient local data access without redundant API calls.

4. Integration Layer

- Manages communication with external APIs, processes image and gives recipe suggestions.
- **Components**:
    - **Gemini AI API Integration**:
        - Handles food recognition based on images, processes images and recipe generation from text prompts.
        - Implements retry mechanisms and error handling for network reliability.
- **Interactions**:
    - Processes requests from the Application Logic Layer and returns structured responses.
    - Handles authentication and secure communication using API keys and headers.

Software Component	Mapped Functionality	Dependencies
Flutter Framework	User interface, navigation, and event handling.	Flutter, Material Design libraries.
Image Picker	Captures images from the camera or gallery.	Camera and gallery access permissions.
Image Processing Module	Compresses and encodes images for API transmission.	image package.
SQLite Database	Local storage of recognized food items, user preferences, and shopping lists.	SQLite packages for database management.
Gemini AI API	Provides food recognition and recipe generation services.	Secure communication
Notification Module	Alerts users about expiring items or new recipes.	Flutter notification libraries.
Error Handling Mechanisms	Manages API timeouts, network issues, and user feedback in case of failures.	Exception handling in Flutter.

Software Functional Flow

1. User Login

- **Action:**
  The user logs in to access their personalized profile, inventory, and shopping habits.

- **Layers Interacted:**
  Presentation Layer: User inputs credentials on the login screen.
  Application Logic Layer: Validates credentials and manages sessions.
  Data Management Layer: Retrieves stored user data (e.g., inventory, preferences) from the database.

2. User Interaction

- **Actions:**
  The user interacts with the app to:
  View Inventory: See the list of stored items, their quantities, and expiration dates. (Inside of Fridge)
  Select an Image: Use the gallery or camera to add items to the inventory.
  Request a Recipe: Generate recipes based on available ingredients.
- **Layers Interacted:**
  Presentation Layer: Handles user actions via buttons and menus.
  Application Logic Layer: Processes images
  Data Management Layer: Retrieves inventory data for display or updates the database with changes.

3. Data Management

- **Actions:**
  Users can:
  Edit details like quantity, expiration date, or name.
  Remove items as they are consumed or expired.
- **Layers Interacted:**
  Presentation Layer: Displays inventory and provides options for manual edits**.**
  Application Logic Layer: Handles items as removal, and updates.
  Data Management Layer: Updates the SQLite Database with inventory changes.

4. Inventory Viewing

- **Actions:**
  Users can:
  View the list of stored items.
  Filter or sort items based on categories, expiration dates, or quantities.
- **Layers Interacted:**
  Presentation Layer: Displays inventory in a list or grid view.
  Application Logic Layer: Fetches inventory data and applies filters or sorting.
  Data Management Layer: Reads inventory data from the SQLite Database.

5. Shopping List Suggestions

- **Actions:**
  The app generates a shopping list based on:
  Missing items in the inventory.
  Frequently purchased items (based on shopping history).
- **Layers Interacted:**
  Presentation Layer: Displays suggested shopping list and allows user customization.
  Application Logic Layer:

Fetches inventory data.
Analyzes shopping history for frequently purchased items.

Data Management Layer: Retrieves shopping history and inventory data from the SQLite Database.

6. Image Processing

- **Actions:**
  The app processes an image to recognize new food items in the fridge.
  Compresses and encodes the image for efficient API communication.
- **Layers Interacted:**
  Presentation Layer: Captures the image or selects it from the gallery.
  Application Logic Layer: Handles image compression and encoding.
  Integration Layer: Sends the processed image to the Gemini AI API.

7. API Communication

- **Actions:**
  The app communicates with the Gemini AI API to:
  Recognize food items from an image.
  Generate recipes based on inventory data.
- **Layers Interacted:**
  Application Logic Layer: Prepares API requests and processes responses**.**
  Integration Layer:
  Sends requests to the Gemini AI API.
  Parses the API response (e.g., recognized items or recipes).

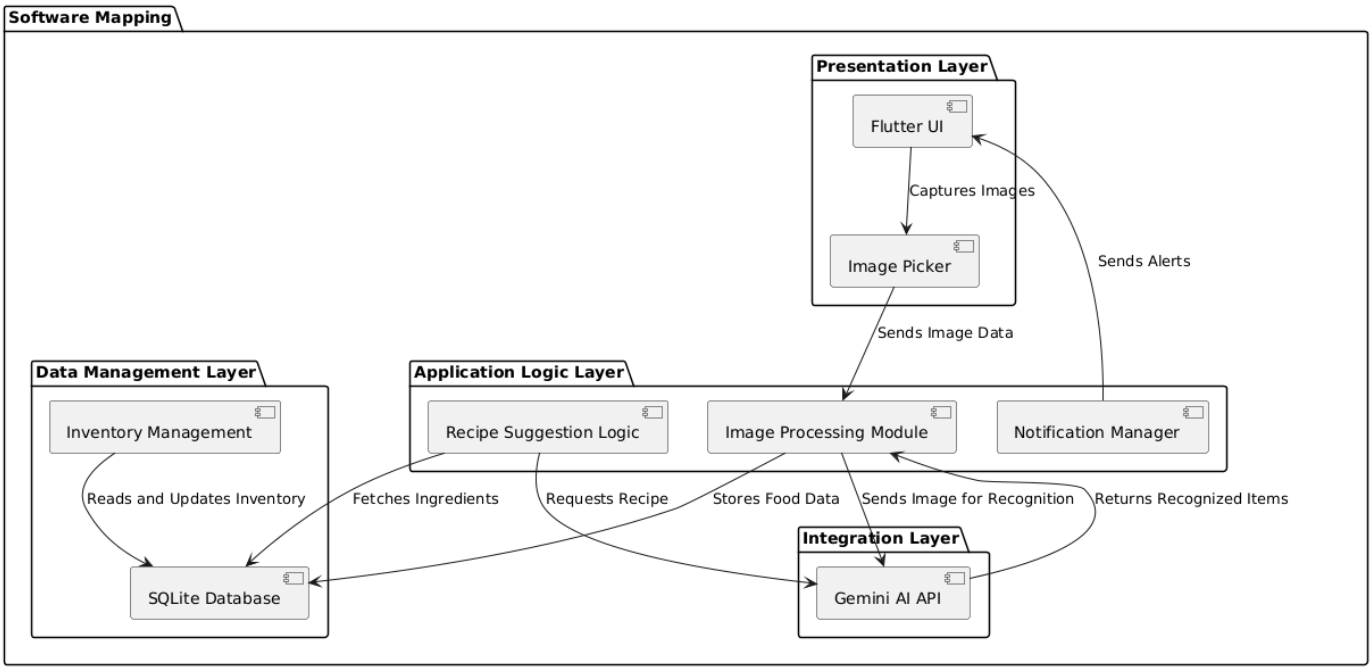

Figure 2: This image shows software mapping of this project.

Scenario:

1. **Login:**

- User logs in by using their credentials to access their inventory and preferences.
- Presentation → Application → Data Management Layers

2. **Inventory Viewing:**
   - User can view their inventory, learn what's in their fridge, and see products nearing expiration date.
   - Presentation → Application → Data Management Layers
3. **Add Items:**
   - User takes a picture of new groceries, and the app updates the inventory with recognized items.
   - Presentation → Application → Integration → Data Management Layers
4. **Shopping List:**
   - User requests a shopping list, and the app suggests items based on missing ingredients and shopping habits.
   - Presentation → Application → Data Management Layers
5. **Recipe Suggestion:**
   - User selects a recipe to prepare, and the app fetches it based on available inventory.
   - Presentation → Application → Integration → Data Management Layers
6. **Notifications:**
   - The app sends reminders about expiring items and recipes.
   - Application → Presentation Layers

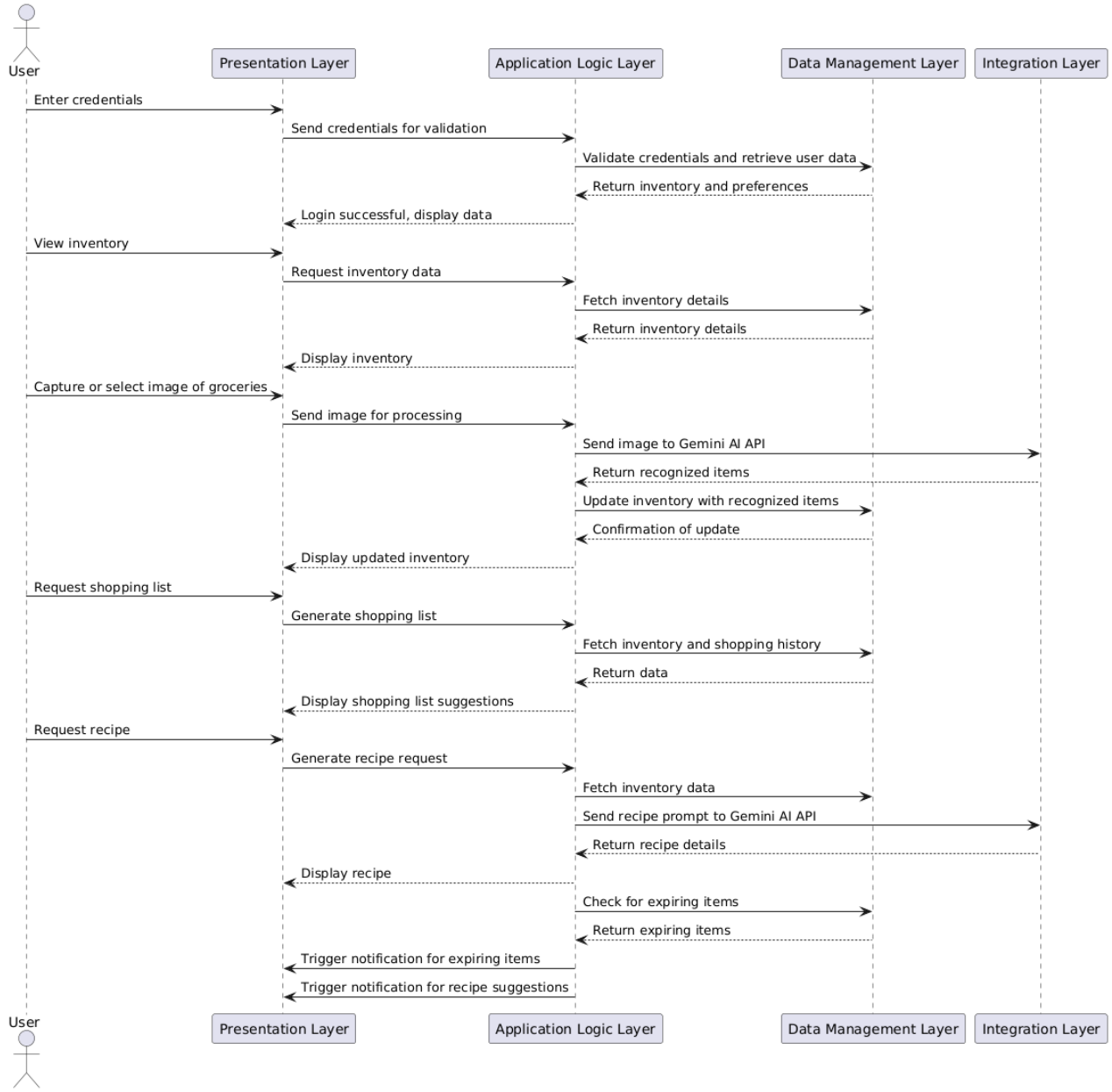

Figure 3: This image shows sequence diagram of workflow above.

3.4 Persistent Data Management

The Fridge Genius application implements a comprehensive and efficient persistent data management strategy to support its core functionalities. The design emphasizes data integrity, scalability, and integration with external systems to ensure seamless operation.

Core Components of Data Management

1. Local Database Implementation

Fridge Genius employs **SQLite**, a lightweight and serverless relational database, to handle local storage requirements.

- **Data Categories Managed:**

    - **Inventory Data:** Information about food items, including names, expiration dates, quantities, and categories.
    - **User Preferences:** Details like allergens, favorite cuisines, disliked ingredients, and dietary restrictions.
    - **Recipe Data:** Recipes retrieved from the Gemini AI API, including ingredients, preparation steps, and user ratings.
    - **Shopping History:** User-generated or recommended shopping lists, stored for review and analytics.
- **Advantages:**
    - Offline accessibility for basic functionalities, including inventory updates and viewing recipes.
    - Lightweight implementation, minimizing device storage overhead while providing robust performance.

2. Data Consistency and Integrity
To ensure data accuracy and reliability:

- Transactions are executed atomically, avoiding partial updates or corruption during failures.
- Validation rules enforce correct data entry (e.g., expiration dates cannot precede the current date).
- Consistency checks are periodically performed to align local and cloud data.

3. Cloud Synchronization
Persistent data is synchronized with a secure cloud environment to enable cross-device usage and provide a backup mechanism.

- **Synchronization Mechanism:**
    - A **delta-sync approach** minimizes bandwidth usage by transmitting only modified or newly added data.
    - Periodic synchronization is triggered automatically when a stable internet connection is detected.
- **Cloud Benefits:**
    - Enables seamless user experience across multiple devices.
    - Protects against local data loss through secure backups.

4. Integration with Gemini AI API
The Gemini AI API is used for advanced image recognition and recipe generation. Data returned by the API is stored in the SQLite database to reduce redundant API requests.

- **Example:** If a food item is already recognized and stored, subsequent calls for the same item will retrieve data locally, enhancing response time and reducing API dependency.
- **Error Handling:** In cases of API downtime, the system queues unprocessed requests and retries them using an exponential backoff strategy.

Scalability and Future-Proofing

1. Efficient Storage Design

- **Normalization:** Data tables are normalized to reduce redundancy and optimize retrieval performance.

- **Indexing:** Frequently queried fields, such as item names and expiration dates, are indexed to ensure rapid access.

2. Modular Architecture

- The persistent data management module is designed to accommodate future enhancements, such as:
    - Full migration to cloud-based databases.
    - Real-time data synchronization with IoT-enabled fridge sensors.
    - Expansion to Android platforms or other operating systems.

Data Security and Privacy

- **Encryption:** All sensitive user data, including personal preferences and inventory details, is encrypted using **AES-256** both at rest (SQLite database) and in transit (cloud storage).
- **Compliance:** The system adheres to **GDPR standards**, ensuring users have control over their data, including options for deletion.
- **Authentication:** Secure user authentication mechanisms, including multi-factor authentication (MFA), are implemented to protect access to personal data.

Error Handling and Recovery

1. Backup and Restoration

- Automatic backups are performed periodically, storing snapshots of critical data in the cloud.
- A recovery mechanism allows users to restore their data in case of accidental deletions or device issues.

2. User Feedback on Errors

- Clear error messages are displayed for issues such as failed synchronization or database write errors.
- Notifications provide actionable advice, such as retrying after resolving connectivity issues.

Benefits of the Strategy

- **Reliability:** Ensures consistent access to data across all interactions, minimizing disruptions.
- **Scalability:** Accommodates the growing user base and increased data volumes.
- **Efficiency:** Reduces reliance on external APIs through local storage of frequently accessed data.
- **Security:** Protects sensitive user information, building trust and compliance with regulations.

This persistent data management strategy lays a solid foundation for the Fridge Genius system, ensuring robust performance, scalability, and a user-centric experience.

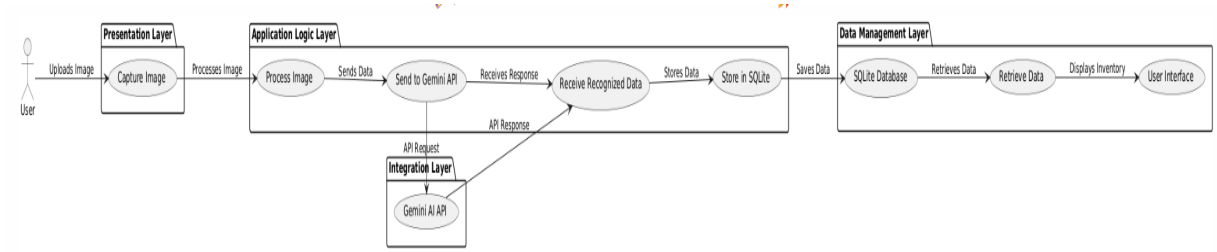

Figure 4: Persistent Data Management System Architecture

3.5 Access Control and Security

Fridge Genius implements strong access control and security measures to protect user data and ensure the secure operation of the system. The system is designed in accordance with modern security standards to ensure user privacy and the integrity of data.

1) User Authentication

- Authentication information such as e-mail and password is required for users to gain access to the system.
- Passwords are stored using strong encryption algorithms (for example, bcrypt) and are never stored in plaintext.
- The two-factor authentication option protects users' accounts as an additional layer of security.

2) Access Authorization

- The system allows users to access only their own data. Not every user is given the right to view or modify the data of other users.
- Certain authority levels are defined for different types of users (for example, regular users and administrators).

3) Data Encryption

- All user data is encrypted both during data transfer (via the TLS/SSL protocol) and during storage.
- Sensitive information is protected even in case of unauthorized access.

4) Security Monitoring and Updates

- The system is regularly monitored to detect potential security threats, and vulnerabilities are addressed quickly.
- Third-party APIs and libraries are regularly updated and audited for vulnerabilities.

5) Data Protection and GDPR Compliance

- Users can view, edit or delete their personal data at any time.
- The system ensures full compliance with the GDPR and other data protection regulations.

6) Responding to Security Issues

- When possible security breaches or threats are detected, the system activates incident response procedures. The necessary actions are taken by providing information to the users quickly.

These comprehensive access control and security measures are aimed at increasing users' confidence in Fridge Genius and guaranteeing the security of the system.

3.6 Global Software Control

In Fridge Genius, the global software control mechanism is primarily event-driven, supported by a combination of centralized and decentralized control for specific subsystems.

1. Event-Driven Control

The application listens for user actions such as logging in, capturing an image, requesting a recipe.

Each user action triggers a sequence of events managed by the respective modules.

Notifications or system-generated actions such as reminders for expiring items also follow an event-driven pattern.

- A user selects an image → triggers the Image Processing Module → invokes the Integration Layer for API communication → updates the Data Management Layer

2. Centralized Control for Core Workflow

The Application Logic Layer acts as a central controller for orchestrating major workflows. It ensures:

- Data consistency across the SQLite Database and other layers.
  Proper API calls and error handling during interactions with the Integration Layer.
- When a recipe is requested: The Application Logic Layer fetches inventory data → sends a prompt to the API → returns the recipe to the Presentation Layer

3. Decentralized Control for Independent Modules

Independent modules like the Notification Manager operate autonomously. These modules monitor specific triggers (expiring items in the inventory) and execute actions without waiting for centralized commands.

- The Notification Manager sends a reminder when it detects an item nearing its expiration date in the database.

Key Components of Global Software Control in Fridge Genius

1. Event Dispatch Mechanism

User actions and system triggers generate events that are dispatched to appropriate modules.

Events can include:

- User login.
- Image selection or capture.
- Recipe request.
- Notification triggers.

2. Central Workflow Orchestration

The Application Logic Layer serves as the orchestrator for complex workflows involving multiple layers.
Ensures:

- Proper sequencing of tasks.
- Error handling and retries.
- Interaction consistency between the Presentation, Data Management, and Integration Layers.

3. State Management

The app maintains a centralized state (e.g., current inventory, logged-in user data).
Uses state management libraries or patterns to keep the UI and underlying data in sync.

4. Notifications and Asynchronous Operations

Notifications and reminders operate independently but rely on database triggers or schedules to function effectively.
Asynchronous operations (API calls) are managed with error handling and retries to maintain system stability.

Advantages of This Approach

Scalability: Modular design allows easy addition of new features without disrupting existing workflows.
Flexibility: Event-driven mechanisms adapt well to asynchronous and user-driven interactions.
Reliability: Centralized control in the Application Logic Layer ensures consistent data handling and error management.
User-Centric Design: Decentralized notifications and state management create a seamless and responsive user experience.

3.7 Boundary Conditions

he Fridge Genius application operates within clearly defined boundary conditions that encompass technical, operational, and user-centric limitations. These boundaries ensure the application remains feasible, efficient, and scalable while adhering to external constraints, dependencies, and expectations. Below is a detailed outline of the boundary conditions:

1. Technical Boundaries

1. **Device Dependency**:
    - The application is designed exclusively for iOS devices and is compatible with iOS 14 or later.
    - A mobile device with a minimum of 12 MP camera resolution is required for accurate food recognition.
    - Optimal performance is achieved on devices with advanced hardware capabilities, such as iPhone 14 or 15.
2. **External API Usage**:

- The application relies on the Gemini AI API for food recognition and recipe suggestions.
- API rate limits (e.g., 1000 requests/day) define the maximum number of recognition operations, requiring the system to optimize request management.
- Downtime or interruptions in the API may affect functionality, handled through exponential backoff and queueing mechanisms.

3. **Database Constraints**:
   - Local storage is implemented using SQLite, with limitations on data size based on device storage availability.
   - Cloud synchronization ensures backups but requires a stable internet connection for data updates.
4. **Offline Functionality**:
   - The app supports offline inventory management and recipe viewing, but image recognition and recipe suggestions depend on an active internet connection.
5. **Network Requirements**:
   - The application requires a stable internet connection for synchronization, API calls, and cross-device data access.
   - Performance degradation may occur in low-bandwidth scenarios, particularly for high-resolution image uploads.

2. Operational Boundaries

1. **User Profiles and Personalization**:
   - Each user can maintain one personalized profile per account, including preferences for allergens, favorite cuisines, and disliked ingredients.
   - Cross-device access is limited to one active session per user to ensure data consistency.
2. **Error Handling**:
   - API errors, synchronization issues, and database write failures are logged and presented to the user with actionable feedback.
   - Critical failures, such as database corruption, trigger automated recovery mechanisms via cloud-stored backups.
3. **Privacy and Security**:
   - The application complies with GDPR standards, ensuring:
     - User control over their data, including options for deletion.
     - Data encryption in storage (AES-256) and during transmission (TLS/SSL).
   - No identifiable personal data is shared with external services or stored unencrypted.
4. **Localization**:
   - Initial development supports English language only, with planned expansions to other languages based on user demand.
   - Measurement units (metric or imperial) are configurable by the user to match regional preferences.

3. Environmental Boundaries

1. **Target Audience**:

- The app is tailored for households, busy professionals, and food enthusiasts who seek efficient food management and recipe suggestions.
- Limited compatibility with institutional kitchens or industrial-scale inventory management systems.

2. **Hardware Integration**:
   - The application does not currently integrate with smart fridges or IoT devices, though future iterations may explore such expansions.
3. **Testing and Deployment**:
   - The application undergoes rigorous testing on a range of iOS devices to ensure compatibility across screen sizes and hardware capabilities.
   - Alpha and beta testing phases involve internal developers and a select group of real-world users, respectively.
4. **Scalability**:
   - Designed to handle up to 500 concurrent users in the initial deployment phase.
   - The architecture is future-proofed for horizontal scaling to accommodate growing user bases and feature expansions.

4. Business Constraints

1. **Budgetary Limitations**:
   - Development costs, including API usage fees, licensing, and team salaries, are capped to ensure financial feasibility.
   - A fixed budget for hardware components, such as testing devices, restricts the number of supported platforms.
2. **Development Timeline**:
   - The project must adhere to a strict development schedule, with a deadline for deployment within six months from the start date.
3. **Licensing and Compliance**:
   - Open-source libraries and tools are prioritized to minimize licensing fees and legal risks.
   - App Store guidelines are strictly followed to ensure acceptance into the Apple App Store.

Impact of Boundary Conditions

1. **User Experience**:
   - Boundaries like offline limitations and API rate restrictions may slightly affect user satisfaction but are mitigated through efficient design and error handling.
2. **Scalability and Growth**:
   - Scalability is designed into the architecture, though initial boundaries, such as limited platform support, may restrict early adoption.
3. **System Longevity**:
   - Boundary conditions guide current development while leaving room for future enhancements, such as Android compatibility, IoT integration, and expanded API capabilities.

This structured approach to defining boundary conditions ensures that the Fridge Genius application delivers a reliable, efficient, and user-focused experience while remaining adaptable to future needs and technological advancements.

4 Subsystem Service

4.1 User Interface Subsystem

The user interface (UI) of the Fridge Genius application has been designed and optimized so that users can easily use the application. The user interface is a subsystem where all interactions take place and users can quickly access the basic functions of the application. In this section, it will be explained in detail how users will experience the application, which visual elements will be used and how user interactions will take place.

1) **Interface Design and User Experience (UX)**

Fridge Genius aims to provide a user-friendly interface. The user interface, in addition to being simple, clean and functional, should have an intuitive design to meet the needs of users quickly and effectively Jul. The minimum effort expenditure of users while using the application is targeted.

- **Simple and Minimalistic Design:** Unnecessary elements are avoided in the interface, the user's attention is focused only on the main functions of the application.
- **Navigation:** Users can easily switch between menus, for example, the main tabs
- **Response Time:** The user experience is not negatively affected by providing quick feedback on every interaction.

2) **Screen Streams and Navigation Paths**

The user interface is configured in such a way that the basic functions of the application can be easily accessed. Users can quickly switch to each tab or screen.

- **Home Screen:** In order for users to quickly access the refrigerator inventory, the home screen provides a user account information and icons of pages for navigating.
- **Recipes Page:** Users see recommended recipes based on the available ingredients in the refrigerator. Recipes can be filtered by categories such as ingredients and favorited.
- **Shopping List Page:** Users can list missing items and create shopping lists automatically. In addition, frequently used materials and shopping lists can be recommended according to user preferences.
- **Inventory Page:** Users can see all ingredients and inventory of fridge with quantity and expiration date. It also has removed button and add button for items needed automatically added.

Navigation paths are kept simple and clear, allowing users to easily switch from one screen to another. Each screen includes a "back" button, which will make it easier to go back to the previous screen.

3) Visual Design and Aesthetics

The visual design of the application has an aesthetically pleasing and eye-tiring structure for users to have a comfortable experience. The colors, typography and icons used have been selected in such a way that they are compatible with the functionality of the application.

- Color Palette: The application uses natural and fresh colors that are often preferred for food-related issues such as white, light blue so that users feel comfortable.
- Icons and Buttons: Clear and understandable icons are used for each function to simplify user interaction. Icons allow users to visually recognize the function they are looking for.
- Mobile Device Compatible Interface: Fridge Genius is designed to have a user interface that works with high performance on mobile devices. All screens are optimized according to different screen sizes. The application has been made compatible to run smoothly on iOS platforms.
- Responsive Design: The adaptive design according to the screen size allows users to have a consistent experience on different devices.

Figure 5: Design of Pages

4) User Feedback and Improvement

- There is a feedback mechanism in the application to receive user feedback and continuously improve the user experience. Users can easily forward problem notifications or improvement suggestions. These feedback provide valuable information for future versions of the application.

The user interface subsystem is a component that forms the basic user experience of Fridge Genius and plays a critical role in ensuring that all functions can be used effectively. This subsystem will be constantly optimized to ensure that users use the application efficiently and enjoyably.

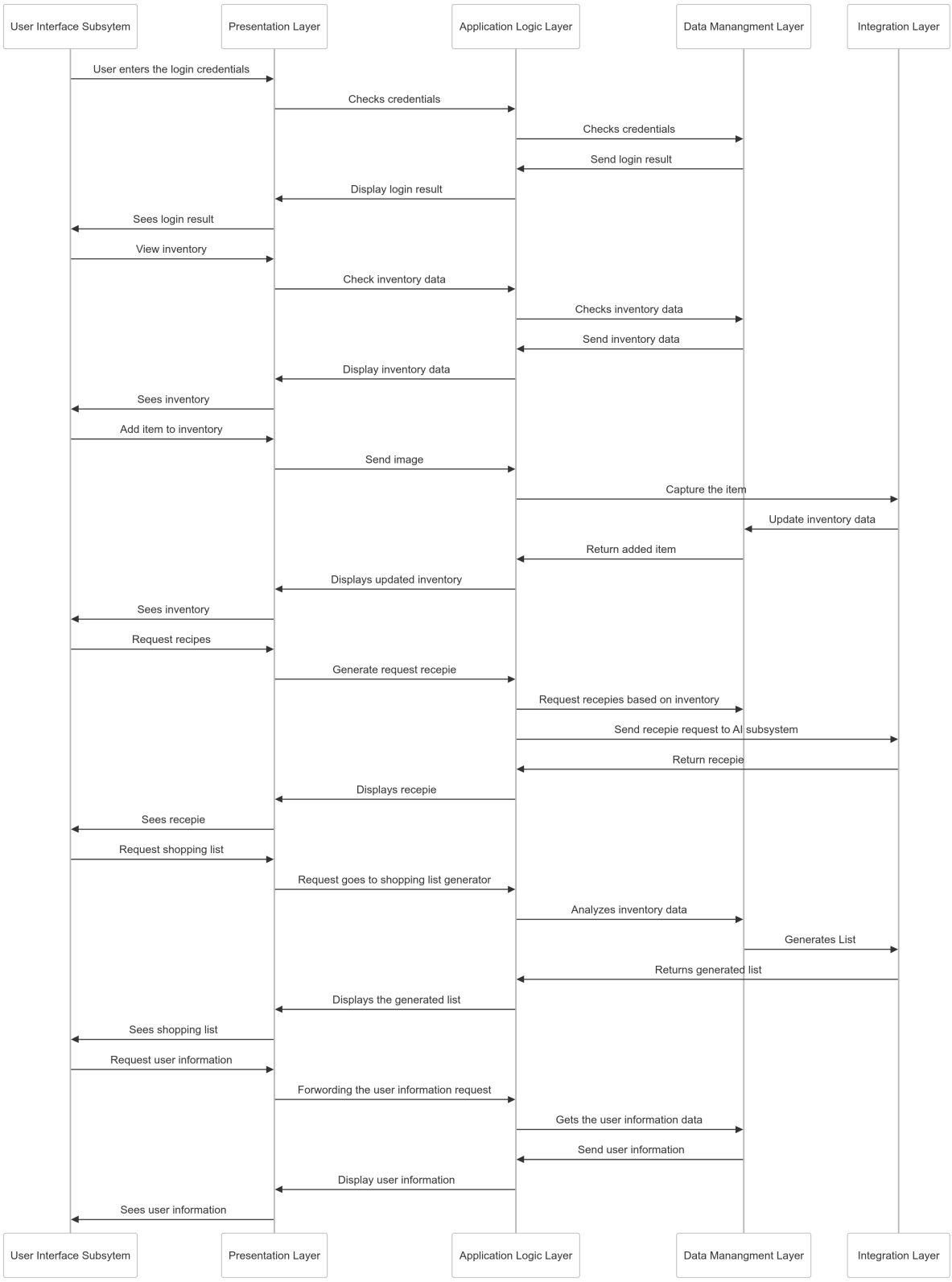

Figure 6: User subsystem interactions with layers

4.2 AI Subsystem

Fridge Genius's AI Subsystem is responsible for enabling advanced functionalities; food recognition and recipe generation, It integrates with external AI services (Gemini AI API) to provide intelligent insights and enhance the application's core features.

4.2.1. Overview of the AI Subsystem

The AI Subsystem has two primary roles:

1. **Food Recognition**:
   Identifies food items in images uploaded by the user via camera or gallery.
   Image recognition models to classify and extract item names or descriptions.
2. **Recipe Generation**:
   Creates recipes based on available inventory.
   Utilizes natural language processing (NLP) and generative AI to suggest recipes tailored to user preferences and inventory data.

4.2.2. Key Components of the AI Subsystem

2.1 Image Recognition Module

- **Purpose**:
  Identifies and classifies food items from images.
  Provides structured data (item names, quantity) for inventory management.
- **Process**:
  - Image is captured or uploaded by the user.
  - Image is preprocessed in the Image Processing Module.
  - Preprocessed image is sent to the Gemini AI API for recognition.
  - The API returns a response with identified items.
- **Integration**:
  Works with the Image Processing Module and Integration Layer for API communication.
  Updates recognized items in the SQLite Database via the Data Management Layer.

2.2 Recipe Generation Module

- **Purpose**:
  Generates personalized recipes based on inventory data and user preferences.
  Supports user-specific customization.
- **Process**:
  - User requests a recipe via the UI.
  - Inventory data is fetched from the SQLite Database.
  - A structured prompt is generated and sent to the Gemini AI API.
  - The API returns a recipe, including ingredients and step-by-step instructions.
- **Integration**:
  Tightly coupled with the Inventory Management Module to fetch available ingredients.
  Works with the Notification Module to suggest recipes for expiring items.

4.2.3. Workflow of the AI Subsystem

Food Recognition Workflow

1. **Trigger**: User captures or uploads an image of groceries.
2. **Image Processing**: The image is compressed and encoded by the Image Processing Module.
3. **API Call**: The Integration Layer sends the image to the Gemini AI API.
4. **Recognition**: The API identifies food items and returns a structured list.

5. **Database Update**: Recognized items are added to the SQLite Database for inventory tracking.

Recipe Generation Workflow

1. **Trigger**: User requests a recipe.
2. **Inventory Fetch**: The Application Logic Layer retrieves available ingredients from the SQLite Database.
3. **Prompt Creation**: A structured text prompt is generated and sent to the Gemini AI API.
4. **Recipe Generation**: The API returns a recipe with a title, ingredients, and instructions.
5. **Display**: The recipe is displayed in the UI.

Personalized Shopping List Workflow

1. **Trigger**: User requests a shopping list.
2. **Inventory Analysis**: The system compares the current inventory with commonly purchased items and missing ingredients.
3. **API Call (Optional)**: AI generates suggestions for frequently bought items or recipes requiring new ingredients.
4. **List Display**: A shopping list is displayed for user customization.

4.2.4. Key Technologies and Tools

Gemini AI API

- Food item recognition using computer vision and recipe generation using natural language processing (NLP) provided by AI.
- **API Endpoints**:
    - Image Recognition: Accepts Base64-encoded images and returns identified items.
    - Content Generation: Accepts text prompts and generates recipe content.

Flutter Libraries for AI Integration

- **HTTP Package**: Facilitates communication with the Gemini AI API.
- **Image Package**: Handles preprocessing of images before sending them to the API.

Machine Learning Models (via API)

- **Computer Vision Model**: Trained on datasets of food items to recognize and classify grocery items.
- **Generative NLP Model**: Trained on recipe datasets to generate accurate and diverse recipes.

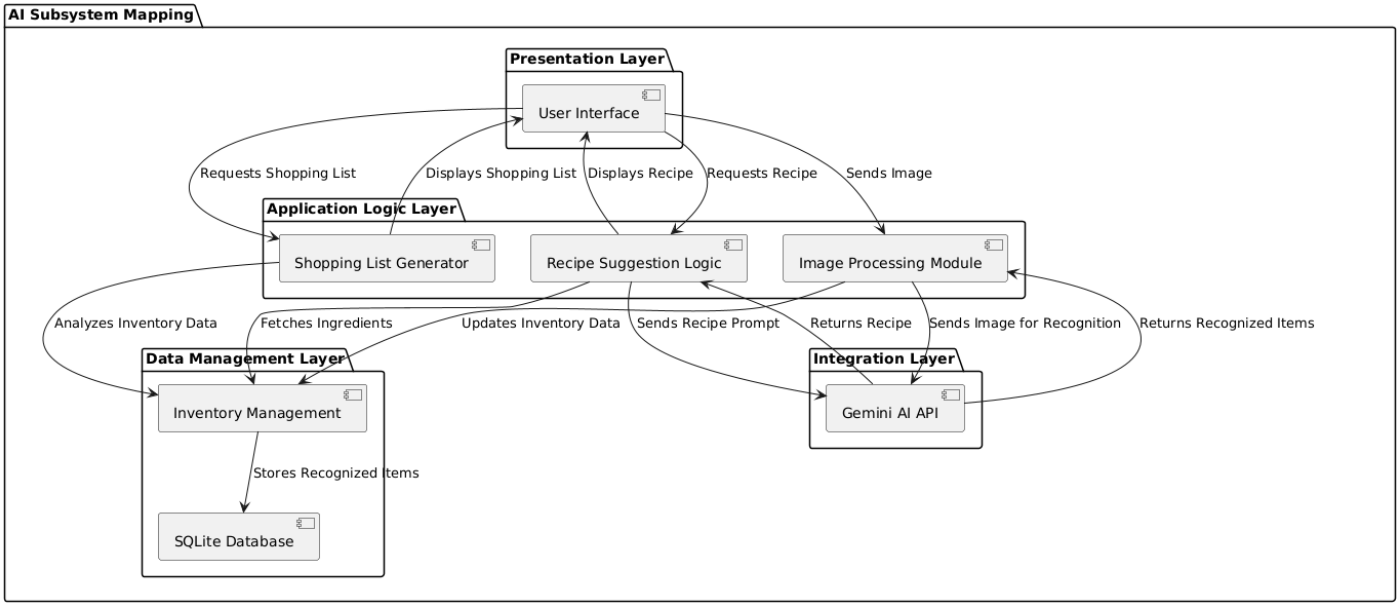

Figure 7: This image shows AI Subsystem mapping.

4.3 Data Management Subsystem

The Data Management Subsystem in the Fridge Genius application plays a vital role in ensuring the secure, efficient, and reliable handling of all data processes within the system. It is designed to support the application's core functionalities, including inventory tracking, recipe management, user preferences, and shopping suggestions. This subsystem is structured to integrate seamlessly with other components, ensuring smooth data flow and robust user experience.

Key Responsibilities

Data Storage and Retrieval

- Implements local storage using SQLite for managing inventory data, user preferences, recipes, and shopping history.
- Provides mechanisms for retrieving, updating, and deleting stored data efficiently.

Data Synchronization

- Synchronizes local data with the cloud environment to enable cross-device accessibility and maintain consistent data backups.
- Uses delta-sync techniques to minimize bandwidth usage by transmitting only modified or newly added data.

Data Integrity and Validation

- Ensures the accuracy and consistency of data by implementing validation rules (e.g., expiration dates cannot precede the current date).
- Executes all database operations atomically to avoid partial updates or data corruption.

Integration with External APIs

- Interfaces with the Gemini AI API for food recognition and recipe generation.
- Stores API responses (e.g., recognized food items or recipes) locally to reduce redundant calls and enhance performance.

Error Handling and Recovery

- Implements retry mechanisms for failed data operations, ensuring eventual consistency.
- Provides clear error messages and recovery mechanisms to handle data corruption or synchronization failures.

Architecture

Local Database Implementation

- The system uses SQLite, a lightweight, serverless relational database, for managing data locally.

Tables and Data Categories:

- **Inventory Table:** Stores details about food items, including name, quantity, expiration date, and category.
- **User Preferences Table:** Maintains user-specific data, such as allergens, favorite cuisines, and disliked ingredients.
- **Recipes Table:** Contains recipe information retrieved from the Gemini AI API, including ingredients, preparation steps, and ratings.
- **Shopping History Table:** Logs user-generated or recommended shopping lists for analytics and review.

Cloud Synchronization

- A secure cloud environment is utilized for data backup and cross-device access.

Synchronization Features:

- Delta-sync mechanism to ensure efficient data transfer.
- Periodic and event-driven synchronization processes (requires an active internet connection).

Benefits:

- Protects against data loss due to device failures.
- Enables a seamless user experience across multiple devices.

Data Flow

User Input:

- Users add food items or modify inventory data through the app.
- Captured data is stored in the SQLite database.

API Integration:

- Image data is sent to the Gemini AI API for recognition.
- Recognized data is retrieved and stored locally for future use.

Synchronization:

- Local data is synchronized with the cloud in real-time or periodically to ensure consistency and enable backups (requires an active internet connection).

Data Output:

- Data is retrieved from the database for recipe suggestions, inventory management, and shopping list generation.

Key Features

1. Data Security

Encryption:

- Sensitive data (e.g., user preferences and inventory details) is encrypted using AES-256 for storage and TLS/SSL for transmission.

Compliance:

- Adheres to GDPR standards, providing users with control over their data, including the ability to delete their information.

2. Internet-Dependent Functionality

- The application requires an active internet connection to synchronize data, retrieve updates, and access real-time features like recipe generation or AI-powered recognition.
- Without an internet connection, functionality is limited to previously synchronized or cached data.

3. Scalability

- Designed to handle increasing user data efficiently.
- Normalization techniques reduce redundancy in database tables.
- Indexed queries ensure rapid data retrieval for commonly accessed fields.

Error Handling and Recovery

Backup and Restoration:

- Regular backups are stored in the cloud, enabling recovery in case of data corruption or accidental deletions.

Retry Mechanisms:

- Failed API calls or synchronization attempts are retried using an exponential backoff strategy.

User Notifications:

- Inform users of data-related errors (e.g., synchronization issues) with actionable guidance for resolution.

Benefits

- **Reliability:** Ensures consistent and accurate data handling, enhancing user trust and system stability.
- **Efficiency:** Minimizes API calls and optimizes data retrieval processes for better performance.
- **Scalability:** Supports growing data volumes and user demands without performance degradation.
- **Security:** Protects sensitive user information, complying with industry standards and regulations.

The Data Management Subsystem forms the backbone of the Fridge Genius application, ensuring smooth, secure, and reliable operation of all data-related functionalities while providing a strong foundation for future growth and enhancements.

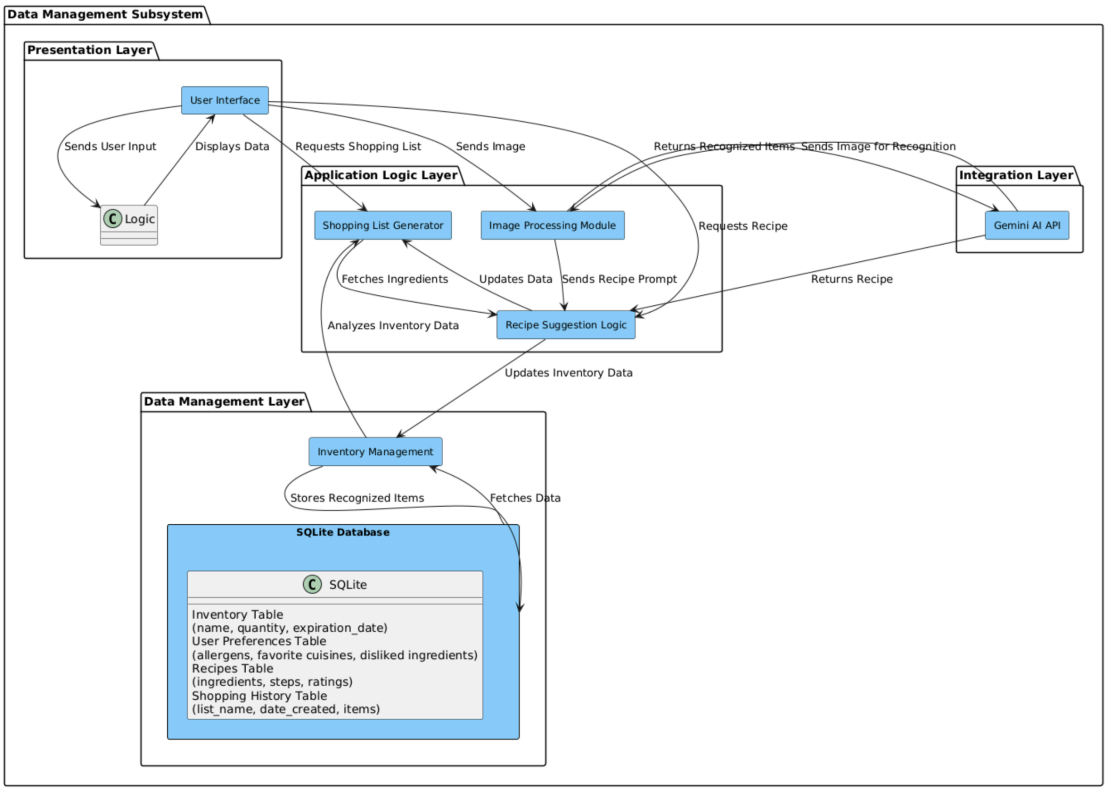

Figure 8: Data Management Subsystem Architecture Diagram

4.4 Recommendation Subsystem

The Recommendation Subsystem in the Fridge Genius application is a cornerstone of the user experience, delivering intelligent, personalized suggestions for recipes and shopping lists. This subsystem harnesses AI-powered analytics and user feedback to generate contextually relevant recommendations that align with user preferences, dietary needs, and inventory data.

Key Responsibilities

1. Recipe Suggestions

- **Core Functionality:**
  - Provides recipes tailored to the user's fridge inventory, preferences, and allergens.
  - Suggests meals that prioritize items nearing expiration to reduce food waste.
- **Personalization:**
  - Adapts suggestions based on user feedback, such as marking recipes as "Favorite" or "Tried."
  - Incorporates preferences for cuisines, dietary restrictions, and disliked ingredients.

2. Shopping Recommendations

- **Core Functionality:**
  - Analyzes inventory and user habits to suggest missing or low-stock items.
  - Automatically identifies ingredients required for favorite or planned recipes.
- **Customization:**
  - Users can edit the suggested shopping list by adding or removing items.

3. User Feedback Integration

- Tracks interactions with recipe suggestions and shopping lists to refine future recommendations.
- Employs machine learning models to adapt to evolving user behaviors and preferences.

Architecture

1. Data Sources

- **Inventory Data:**
  - Pulls real-time inventory details from the Data Management Subsystem.
- **User Preferences:**
  - Leverages stored data on allergens, cuisines, and disliked ingredients.
- **Historical Data:**
  - Analyzes past shopping lists, recipes, and consumption trends.

2. AI-Powered Analysis

- Utilizes advanced AI algorithms to:
  - Match recipes to inventory and user preferences.
  - Generate predictions for frequently used items and missing ingredients.
  - Provide real-time recommendations that adapt to user behavior.

3. API Integration

- Interfaces with the Gemini AI API for:
  - Natural language processing (NLP) to create and refine recipes.
  - Advanced analytics for more complex recommendation scenarios.

Data Flow

Input

- Inventory data, user preferences, and historical usage patterns.
- Feedback on suggested recipes and shopping lists.

Processing

- AI algorithms process input data to generate personalized recipes and shopping suggestions.
- API responses are stored locally to minimize redundant calls and improve response time.

Output

- Recommended recipes with detailed instructions and preparation times.
- Customized shopping lists categorized by priority and user preferences.

Key Features

1. Real-Time Personalization

- Dynamically adjusts recommendations based on inventory updates, user feedback, and API results.

2. Feedback-Driven Improvement

- Continuously enhances recommendation accuracy by analyzing user interactions with recipes and lists.

3. Adaptive Shopping Lists

- Automatically suggests items based on user habits and planned meals.
- Categorizes shopping lists to improve usability.

4. Error Handling

- Provides cached recommendations during API downtime or connectivity issues.
- Notifies users with actionable guidance when errors occur.

Scalability and Flexibility

1. Scalability

- Designed to accommodate a growing user base and diverse preferences.
- Supports increasing data volumes with efficient indexing and query optimization.

2. Modular Design

- Easily integrates with additional APIs or services in the future.
- Supports expansion to other platforms, such as Android.

Benefits

- **Efficiency:** Reduces the effort required for meal planning and grocery shopping.
- **Waste Reduction:** Encourages optimal use of available inventory, reducing food waste.
- **Personalization:** Delivers tailored experiences that align with user needs and preferences.
- **Scalability:** Maintains performance and accuracy as the user base grows.
- **Reliability:** Ensures consistent recommendations through robust data handling and error recovery.

The Recommendation Subsystem embodies the Fridge Genius vision of a smart, sustainable, and user-focused food management solution, simplifying meal preparation while promoting responsible consumption habits.

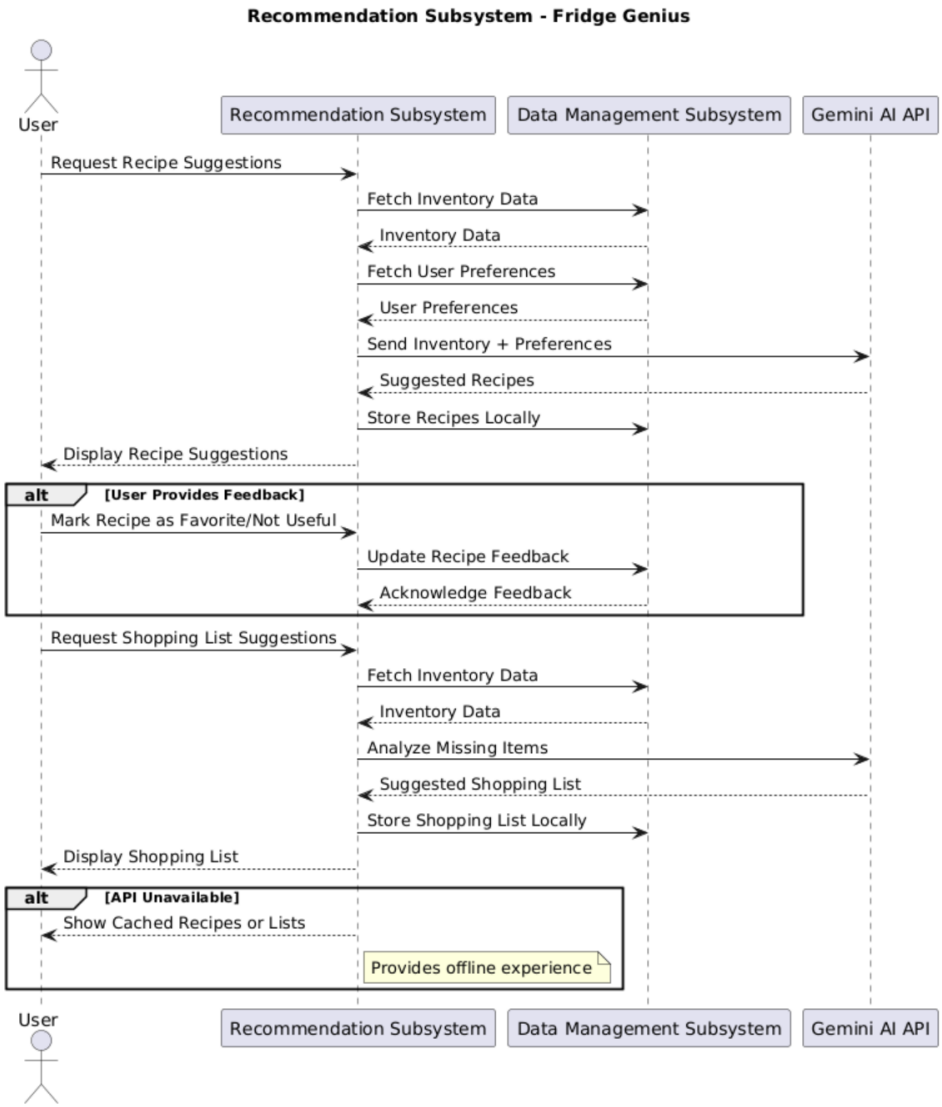

Figure 9: Sequence Diagram for Recommendation Subsystem

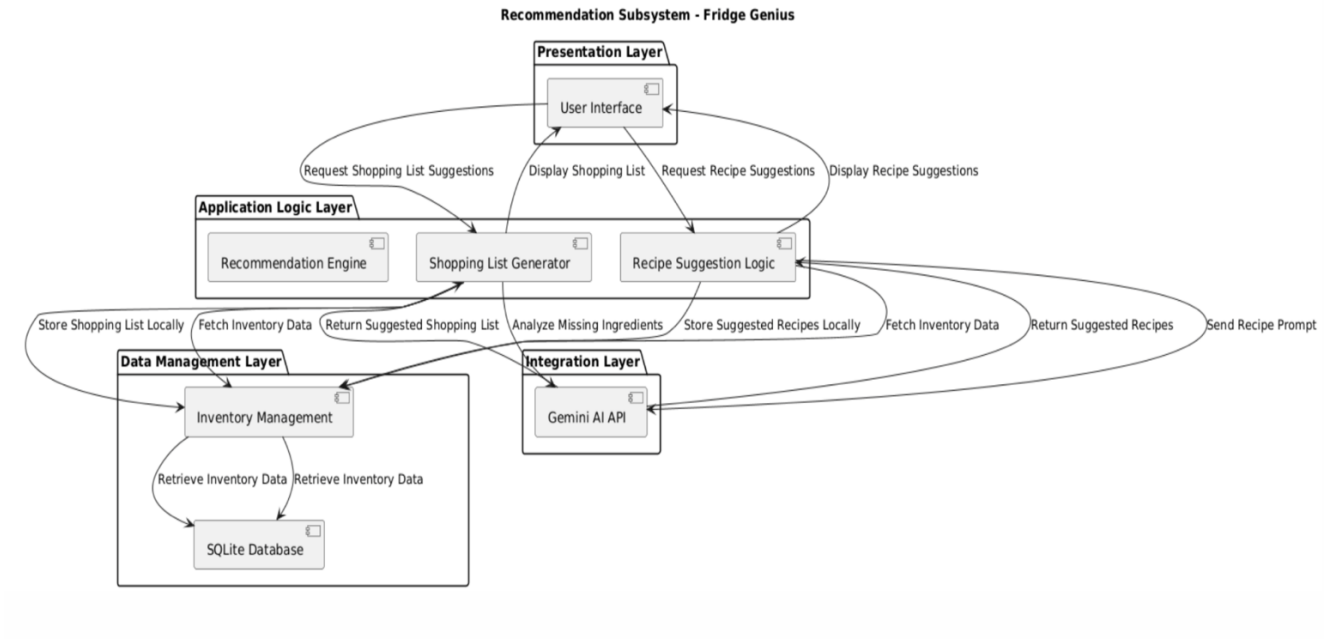

Figure 10: Layered Architecture of Recommendation Subsystem

5 Glossary

This section provides definitions for terms, acronyms, and abbreviations used throughout the **Fridge Genius** High-Level Design Report. These definitions aim to ensure consistency and clarity for technical and non-technical readers.

Definitions

- **Artificial Intelligence (AI):** The simulation of human intelligence processes by machines, especially computer systems, enabling them to perform tasks such as learning, reasoning, and decision-making.
- **Image Recognition:** A technology that processes and identifies objects, items, or patterns within digital images using computer vision and machine learning algorithms.
- **Natural Language Processing (NLP):** A field of AI focused on the interaction between computers and human language, allowing the system to understand, interpret, and generate human language text or speech.
- **Recommendation System:** A system that provides personalized suggestions based on user data, preferences, or past interactions. In the context of Fridge Genius, it includes recipe and shopping list suggestions.
- **Inventory Management:** The process of tracking and controlling items stored in a system (e.g., a refrigerator), including quantity, expiration dates, and usage patterns.
- **Data Synchronization:** The process of ensuring that data across multiple devices or systems is consistent, updated, and accurate in real-time or periodically.
- **User-Centric Design:** An approach to design that prioritizes the needs, preferences, and ease of use for the end user.
- **Scalability:** The capability of a system to handle an increasing amount of work or its potential to expand to accommodate growth.

- **Cloud Integration:** The use of cloud-based services for storage, computing, and processing, enabling enhanced accessibility and scalability.
- **Security Encryption:** Techniques used to encode data to prevent unauthorized access, ensuring data privacy and integrity.

Acronyms

- **AI:** Artificial Intelligence
- **NLP:** Natural Language Processing
- **API:** Application Programming Interface
- **GDPR:** General Data Protection Regulation
- **IoT:** Internet of Things
- **UI:** User Interface
- **UX:** User Experience
- **DB:** Database
- **TLS:** Transport Layer Security
- **HTTP:** HyperText Transfer Protocol

Abbreviations

- **SQLite:** Structured Query Language Lite – A lightweight database engine used for local storage in Fridge Genius.
- **AES-256:** Advanced Encryption Standard with a 256-bit key, ensuring robust data security.
- **MFA:** Multi-Factor Authentication – A security mechanism requiring two or more credentials for user verification.

Notes on Usage

- **Technical Jargon:** While this document is designed for technical readers, these definitions clarify terms to aid understanding for non-specialists.
- **Consistency:** Acronyms and abbreviations are consistently used throughout the report to avoid redundancy.
- **Contextual Understanding:** Where terms are used in multiple contexts (e.g., scalability), their specific meaning is clarified within the respective section.

6 Reference

1. Bishop, C. M. (2006). *Pattern Recognition and Machine Learning*. Springer.
    - Referenced for understanding AI-based machine learning methodologies employed in the Recommendation Subsystem.
2. Google Cloud. (n.d.). Google Gemini API Documentation. Retrieved from https://cloud.google.com

    - Used for integrating AI services, including food recognition and recipe generation functionalities.
3. ISO/IEC 25010:2011. (2011). *Systems and Software Engineering – Systems and Software Quality Requirements and Evaluation (SQuaRE)*.
    - Consulted for ensuring the software architecture aligns with industry-standard quality attributes such as scalability, usability, and maintainability.
4. JetBrains. (n.d.). *Datagrip Tool Documentation*. Retrieved from https://www.jetbrains.com/datagrip/
    - Used for efficient database management and SQLite development during the implementation of the Data Management Subsystem.
5. Nielsen, J. (1993). *Usability Engineering*. Morgan Kaufmann.
    - Referenced for designing the user-friendly and intuitive interface of the User Interface Subsystem.
6. SQLite Consortium. (n.d.). *SQLite Database Documentation*. Retrieved from https://sqlite.org
    - Referenced for implementing lightweight, local storage in the Data Management Subsystem.
7. The European Parliament and the Council of the European Union. (2016). *General Data Protection Regulation (GDPR)*. Official Journal of the European Union.
    - Consulted for ensuring data privacy and compliance in handling user-sensitive information.
8. PlantUML. (n.d.). *PlantUML Documentation*. Retrieved from https://plantuml.com/
    - Utilized for generating UML diagrams to model subsystems and workflows in the report.
9. Apple Inc. (n.d.). *iOS Human Interface Guidelines*. Retrieved from https://developer.apple.com/design/human-interface-guidelines/
    - Referenced for designing the user interface to adhere to iOS platform standards.
10. Flutter Developers. (n.d.). *Flutter Documentation*. Retrieved from https://flutter.dev
    - Used for building the cross-platform user interface of the application.
11. Python Software Foundation. (n.d.). *Python Documentation*. Retrieved from https://www.python.org/doc/
    - Referenced for backend scripting in data processing and API integration.
12. Nielsen Norman Group. (n.d.). *User Interface Design Basics*. Retrieved from https://www.nngroup.com
    - Consulted for designing the user-centric interface.
13. European Commission. (n.d.). *Digital Single Market - Cloud Computing and Security*. Retrieved from https://ec.europa.eu/digital-single-market/en/cloud-computing
    - Referenced for integrating cloud-based synchronization into the Data Management Subsystem.
14. JetBrains. (n.d.). *IntelliJ IDEA Documentation*. Retrieved from https://www.jetbrains.com/idea/
    - Utilized for managing and developing the modular architecture of the project.
15. W3C. (n.d.). *Web Accessibility Initiative (WAI) Guidelines*. Retrieved from https://www.w3.org/WAI/
    - Consulted for ensuring accessibility features in the user interface design.