CSC 431

Smart Food Storage

System Architecture Specification (SAS)

Team 04

Hill Yu	Developer
Angela Wang	Scrum Master
Jason Donovan	Developer
Alexander Luce	Developer

Version History

Version	Date	Author(s)	Change Comments
1.0	3/30/2022	Angela Wang	First draft

Table of Contents

Table of Tables

<Generate table here>

Table of Figures

<Generate table here>

51. System Analysis

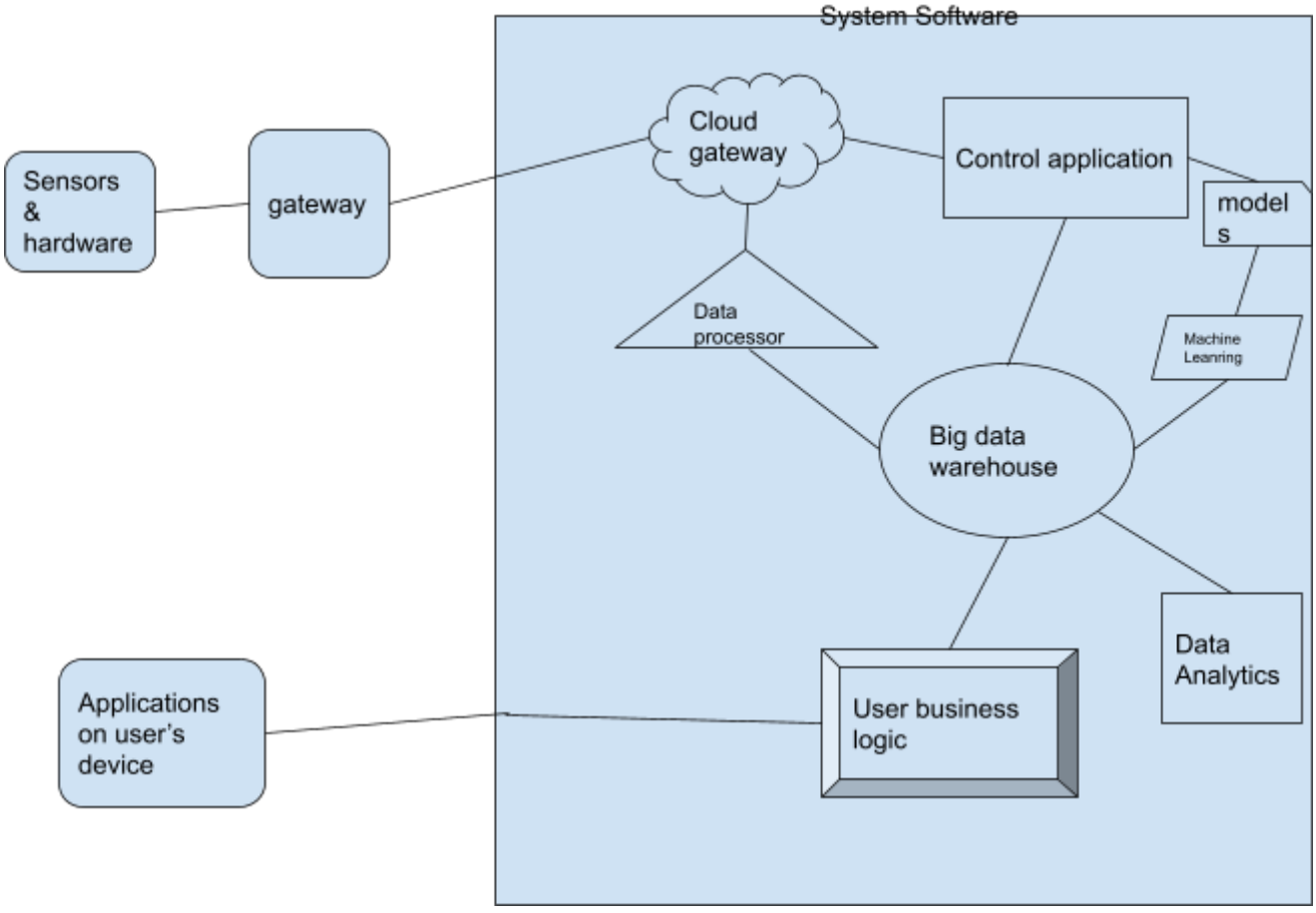
51.1 System Overview

The following document will describe the design and specifications of the Smart Food Storage devices and network. The composition of our system is mainly focused on the following parts: tracking stored food items in the database, and presenting generated recipes based on these items.

The data on food items currently stored inside the Smart Food Storage will be cached locally in the user's application, as well as stored in our database for a backup.

Structured as a traditional 3-tier application so the client can directly access the server, we first have the UI written in JavaScript. In the middle tier, information is collected from the user and used to modify data in the final tier, the data tier. In the database tier, recipes can be called upon for the user depending on the food items present.

51.2 System Diagram



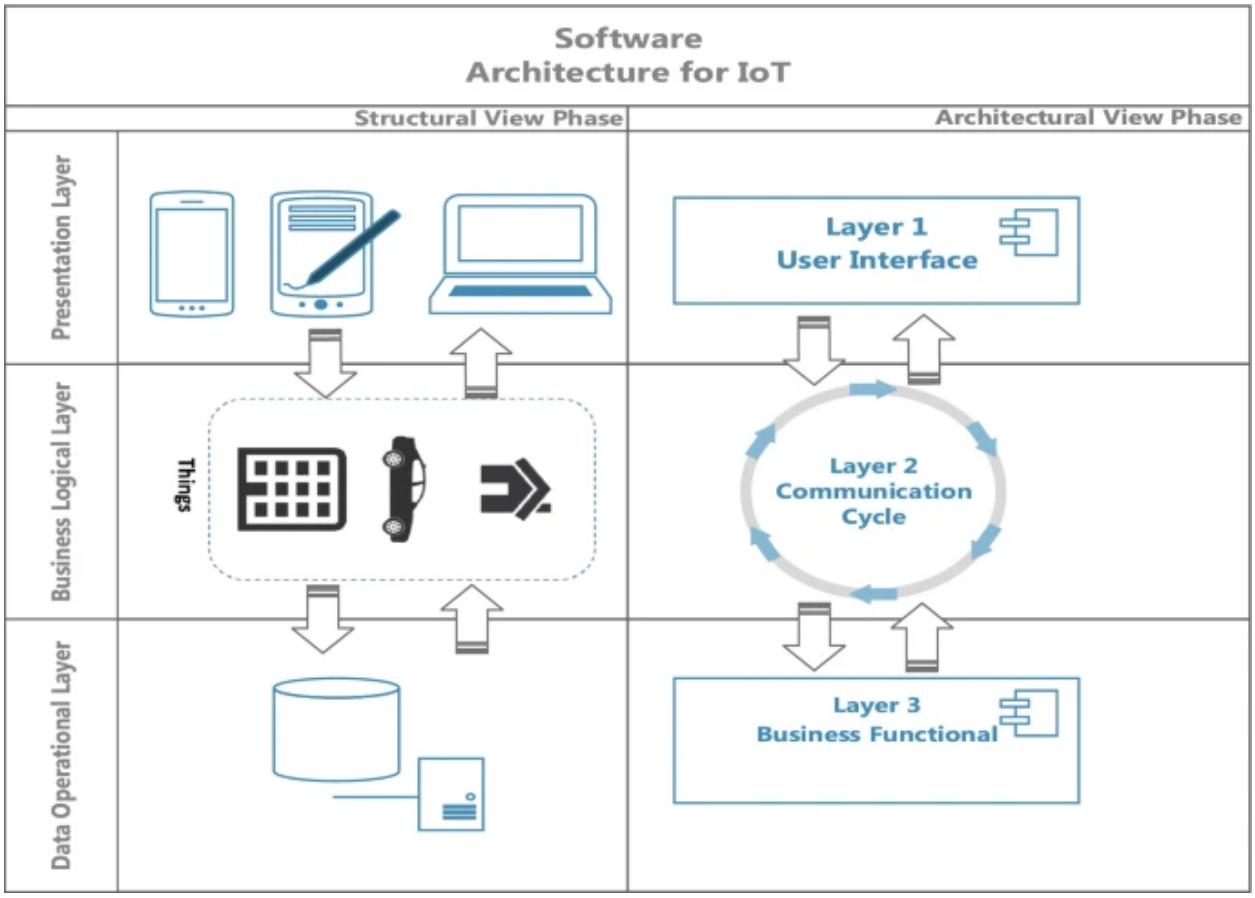
51.3 Actor Identification

Actors interacting with the system include:

- Users: Plays both the physical role of stocking the fridge, and the virtual roles of creating account on our platform, inputting food items, and requesting recipes from the database
- Device: Being either the users phone or computer, used in order to run the application
- Servers: Hold record of all our users, keep the application running smoothly
- Database: Holds all possible food items as well as possible recipe combinations

51.4 Design Rationale

1.4.1 Architectural Style—--layered architecture style



Typically, IoTs use 3 or 4 layered architecture

1.4.2 UI layer for interactions with the users. Since we have hardware, therefore it also includes the interactions with the sensors within the smart storage.
1.4.3 Gateway layer helps the communication between UI and Database.
1.4.4 Database access layer for searching ready-to-cook recipes. This way, we could maintain the integrity of quality recipes by having modify, add, delete and update all in one place.

1.4.5 Design Pattern(s)

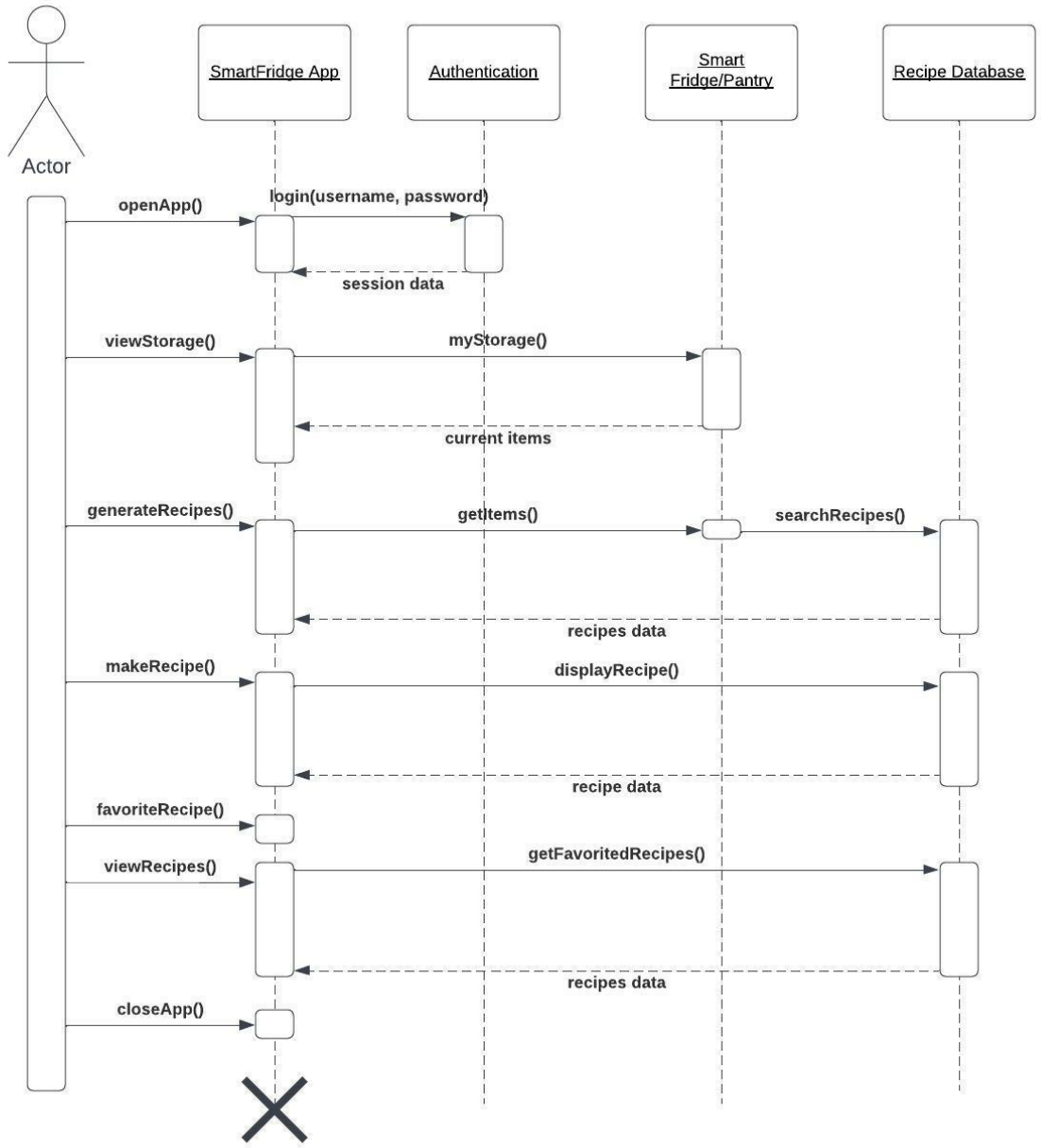
For Smart Food Storage using a Strategy design pattern will be the most efficient for our needs. A Strategy design pattern allows us to encapsulate our multiple algorithms for searching and producing recipes. Using this design pattern will also efficiently add additional algorithms in the future for evolutionary changes to Smart Food Storage. A Navigator will be used as the context class, which will implement the strategy class, RecipeSearch. RecipeSearch includes different algorithms for searching recipes, like searching for recipes based on the current food items, or searching for recipes based on substitutions.

1.4.6 Framework

Smart Food Storage will utilize the Spring framework, which includes Java, Hibernate, and MySQL. Spring also has the mobile extension, Spring Mobile, that will aid mobile development for the Smart Food Storage app. While Java will be used for the majority of development, the MySQL framework will be used for the storage of multiple recipes. The Hibernate Framework will map Smart Food Storage classes to the recipe database. Using the Hibernate framework will free Smart Food Storage from data persistence problems that may arise from changes in the recipe database, which is based on external APIs (recipe sites).

52. Functional Design

52.1 Diagram Title



53. Structural Design