Department of Computer Science & Engineering
The University of Texas at Arlington

Architectural Design Specification
CSE 4316: Senior Design I
Summer 2021

Coding Avengers
Wine Inventory Management

Tuyen Vo
Sita Lama
Mokshada Upreti
Linh Tran
Greg Whatley

Revision History

Revision	Date	Author(s)	Description
0.1	08.06.2021	GW	document creation
1.0	08.13.2021	TV, SL, MU, LT, GW	First draft
1.1	08.16.2021	TV, SL, MU, LT, GW	Final draft

Contents

List of Figures

List of Tables

1 Introduction

Imagine that you are an avid wine lover, who is passionate about collecting different types of wine grapes from different regions. As the time goes by, the amount of wine bottles in your collection surprisingly increases and it is frustrating that you can't find the perfect bottle in your own cellar because you do not remember where the exact bottle is stored, in which bottle hole or which bin. In addition, we all know that the taste of the wine gets better as it ages. However, if a bottle has been opened, it gets put away in random places and can't be found when you want it.

Expensive wine end up going past prime, gifted wine bottles get put in random bottle holes that can not be found easily. On the other side, if you own a small winery business,wine inventory management could be time consuming and requires a decent amount of manpower by manually counting how many bottles are left in stock, which one is running out or you want to check which bottles have been opened and put back in the cellars. That brings us the opportunity to work on solving such problem with managing the inventory with details of the bottles. It will you a help for a better experience in finding lovely wine bottles, check its condition and consume it before it goes bad.

Wine Inventory app will be built to perform some tasks such as scan and add the wine bottles into the inventory with a specific location provided by the user along with its descriptions: color, origin, winery, vintage/year produced, styles: read/white/rose/sparking white/sparking rose. Its condition: opened or sealed, opened date, is it full, half or almost consumed. The app will help to mitigate the problems with locating the wine or liquor bottle, its condition, check to see if the bottle is getting expired(if the bottle is opened).

Wine Inventory app will be made mostly for personal and general uses only. The final product will be publicly available on App Store or Google Play Store with my purchase required. In addition, this app is particularly designed per the requirements from Christopher Conly PhD.

2 System Overview

Our wine inventory management app is divided into 3 distinct layers: the UI (frontend), the database (backend), and the camera (hardware). The user's first interaction with our app is the frontend. The frontend is what our user is able to see, from how a screen is formatted to how a button reacts when being clicked. The backend will mostly be how our database interacts with the app, what is stored, what is deleted, etc. The backend will also include the actual functionalities of our app and the API for the barcode scanner. Users are able to use their phone cameras to scan a bottle's barcode for an easier way to register a bottle.

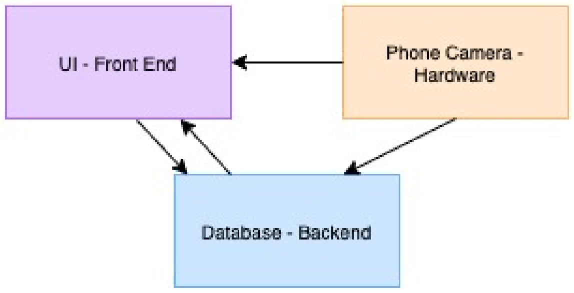

Figure 1: Architectural layer diagram

2.1 UI - Frontend

The app will have three system. The application will allow users to sign up and create account. Once the account is created, it will take users in the home page where they can search the list of bottles. The users can choose the specific bottle and find the details of bottle like origin, year, style, storage location, status of bottle, and pairing. This will be the UI frontend of the application.

2.2 Phone Camera - Hardware

The second layer is system layer will be phone camera. The app will have barcode scanner API. The user can scan the barcode of bottle using their phone camera to auto fill the information of bottle in the inventory. Another thing is that users can take picture of bottle using their phone camera. Barcode scanner API will have image of the bottle and it automatically register bottle information.

2.3 Firebase - Backend

Firebase is an application server that integrates several services into one product. For the Wine Inventory Management app, we will be using three of these services. First, Firebase Realtime Database will handle storage and retrieval of user data such as bottle information. Realtime Database is a NoSQL database that stores data as JSON. Because we are building our app with JavaScript and React Native, JSON data can be integrated easily. The frontend will read and write data to/from the database. We will configure proper security rules in the Firebase Console to prevent other users (or unauthenticated users) from accessing data belonging to another account. Second, Firebase Storage will handle photo uploads for our users. Storage has a simple API that will allow our frontend to upload images (e.x. bottles or locations) and view them later. Like Realtime Database, we will configure security rules for the storage bucket. Third, Firebase Authentication will be the interface for signing in our users either with email and password or with third-party authentication services (Google, Facebook, Sign in With Apple, etc).

Authentication provides an API to create accounts and send password reset emails. Firebase UI is an optional framework that includes prebuilt UIs for sign in; however, we may decide to build our own.

3 Subsystem Definitions & Data Flow

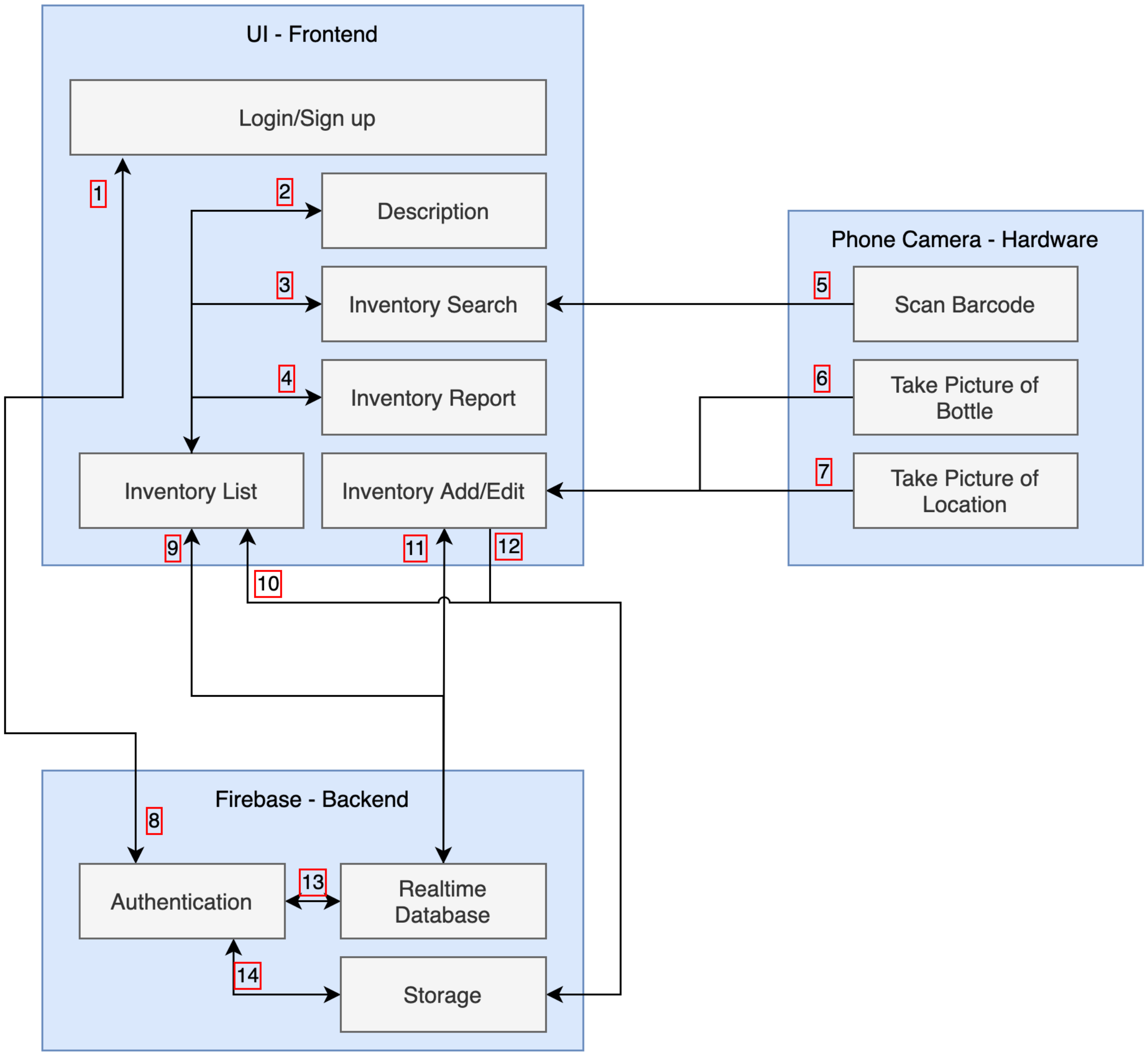

Figure 2: Data flow diagram

4 UI Front-End

In this section, layer X aka the UI (front end) which is Wine Inventory app front where user will interact with the system including: User Login, Sign-up, Forgot Password and Change Password; Inventory Display List; Bottle's Description; Inventory Search; Add/Edit Bottle's Description - Location; Inventory Report.

4.1 User Login, Sign-up, Forgot Password and Change Password

This particular subsystem require users to identify themselves before entering into the app's database with registered and authenticated user name and password. If a user is new to the system, then he/she has to sign up with his email and password. In addition, user can also request a new password if he/she forgot it as well as changing the password.

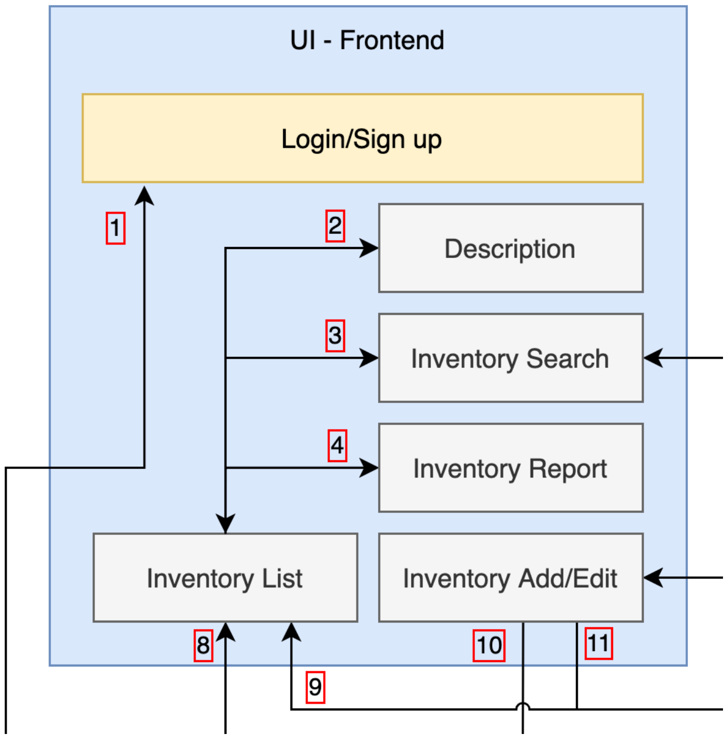

Figure 3: User Login, Sign-up, Forgot Password and Change Password Subsystem

4.1.1 Assumptions

1. User has to login with authenticated username and password.
2. Username can be string or email address type.
3. Password has to contain Uppercase, Lowercase and a special character.
4. New user has to sign-up with an email.
5. A link will be sent to the authenticated email when user forgot password.

4.1.2 Responsibilities

1. As soon as the user is identified when login into the app. It will pull out user's correct data such as their inventory list in the database.
2. In case, when a new user sign up, this system will be capable of sending out an confirmation link to

the email that was provided by user. If the link is not confirmed, then user's account is still unauthenticated.
3. If user forgot their password, then the system will pull out user's login data in the database by the email user provided and prompts user for a new password, then stored that new password to the database.
4. If user wants to change their password, then the system will pull out user's login data in the database by the email user provided and prompts user for a new password, then stored that new password to the database.

4.1.3 Subsystem Interfaces

Table 2: Login/Sign-up Subsystem interfaces

ID	Description	Inputs	Outputs
#1	Login	Input 1, Input 12, Input 13	Output 9
#2	Sign-up	Input 1, Input 12, Input 13	Output 9
#3	Forgot Password	Input 1, Input 12	Output 1
#4	Change Password	Input 1, Input 12	Output 1

4.2 Inventory List Display

The Inventory List Display subsystem is capable of displaying user's inventory after user successfully logged into the system. It will display all the bottle user has in the inventory.

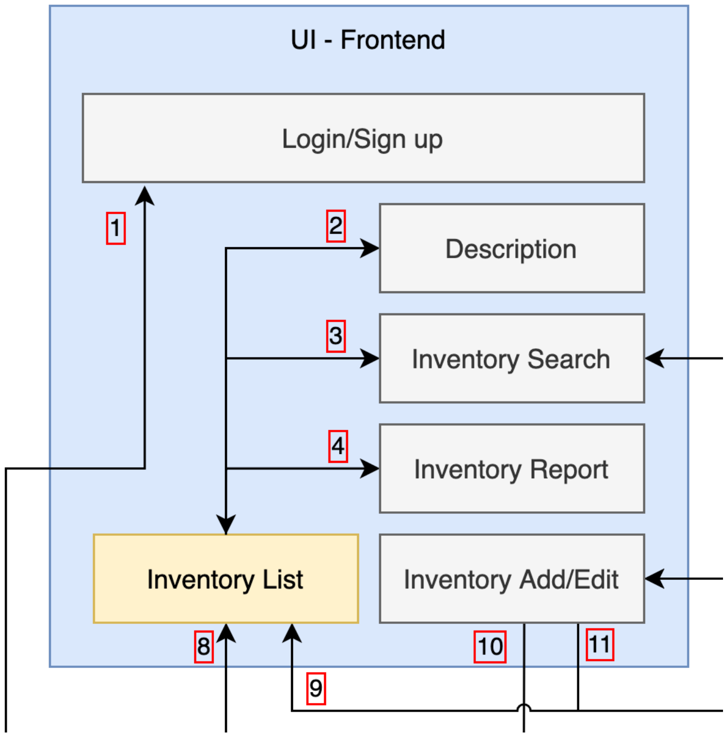

Figure 4: Inventory List Display Subsystem

4.2.1 Assumptions

1. User has to successfully logged in to the system with their authenticated account.
2. If user is new, then the app will suggest user to add some bottles to the inventory.
3. Their will be some buttons on this screen for user to sort the list or the way how user wants to display it.

4.2.2 Responsibilities

1. This subsystem only activated when user successfully logged into the app with their authenticated account.
2. Will allow user to modify how they want to display the list of item they have in the inventory.
3. Include a feature to add/edit item by clicking on it.

4.2.3 Subsystem Interfaces

Table 3: Inventory List Display Subsystem interfaces

ID	Description	Inputs	Outputs
#1	Inventory Display	Input 1, Input 12, Input 13	Output 9

4.3 Bottle's Description

The Inventory List Display subsystem is capable of displaying bottle's information such as winery type, wine name, wine style, vintage/year produced, picture of the bottle, storage location, bottle's status and opened date after user successfully logged into the system. It will display all the bottle user has in the inventory.

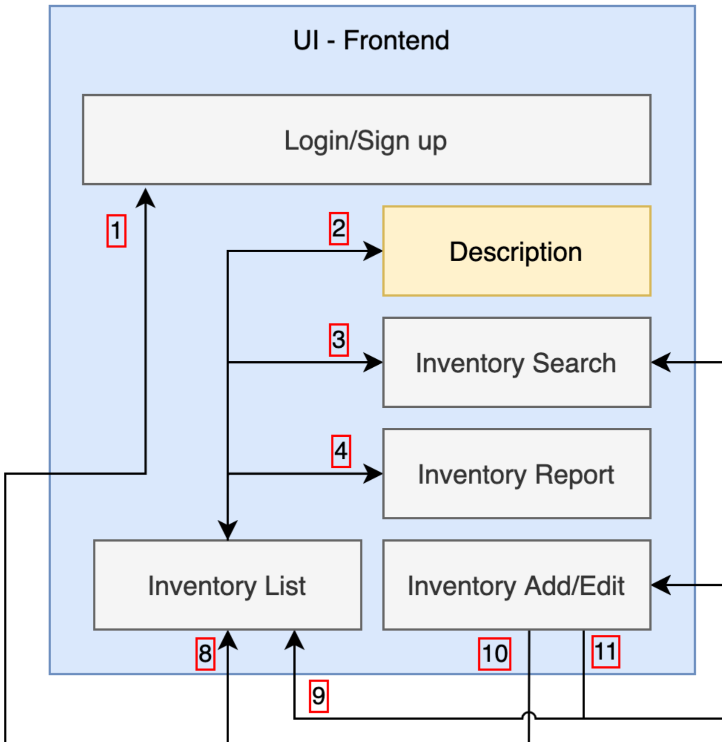

Figure 5: Inventory List Display Subsystem

4.3.1 Assumptions

1. This subsystem is only activated when user has some item in the inventory.
2. Bottle is stored in the inventory with all the required information including location.
3. User has to click onto the bottle in order to access to this subsystem.

4.3.2 Responsibilities

1. This subsystem has to display all the information that were stored with the bottle in the inventory such as wine name, picture, opened date, wine type, bottle status,...
2. Will include a button for user to edit the bottle's description.
3. Will include a button for user to remove the bottle.

4.3.3 Subsystem Interfaces for Bottle Description

Table 4: Bottle Description subsystem interfaces

ID	Description	Inputs	Outputs
#1	Login and Inventory List Display	Input 1, Input 12, Input 13	Output 9
#2	Bottle's Description	Input 1, Input 12, Input 13, Input 9	Output 2

4.4 Inventory Search

The Inventory List Display subsystem is capable of searching for bottle in the inventory by names, food pairing, winery type after user successfully logged into the system. It will display all the bottle user has in the inventory.

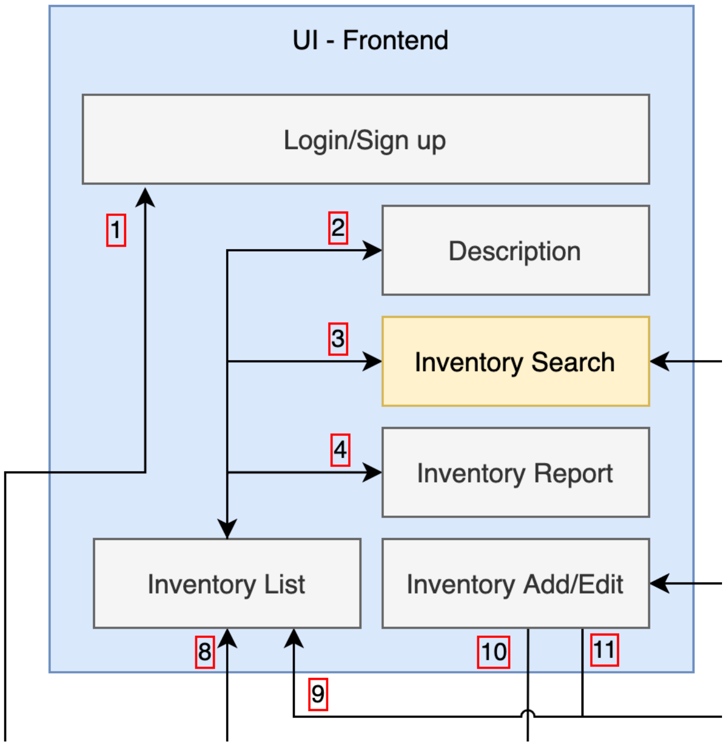

Figure 6: Inventory List Display Subsystem

4.4.1 Assumptions

1. This subsystem is only activated when user has some item in the inventory.
2. User can search for the bottle by it's name.
3. User can search for bottle by choosing the options to see how many bottles are opened and unopened.

4.4.2 Responsibilities

1. If the name provided by user matches some in the inventory then it app will display all of them as the result.
2. If user choose to search by food pairing then the app will display all the bottle that match the criteria.

4.4.3 Inventory List Display Subsystem Interfaces

Table 5: Inventory Search Subsystem interfaces

ID	Description	Inputs	Outputs
#1	Login and Inventory List Display	Input 1, Input 12, Input 13	Output 9
#2	Inventory Search	Input 1, Input 12, Input 13, Input 9	Output 3

The user will allow to scan the bar code with their mobile phone. It will let user to add their wine in the inventory which includes year of manufacture, type of brand, color, style, name, variety of wine. User can add or edit the location of the wine in the cells per the need. The user can easily monitor the status of the bottle, if it is open or sealed.

4.5 Add/Edit bottle and Location

This particular subsystem prompts the input from the user about the bottles and the location.The user can add the bottle and where it is located.

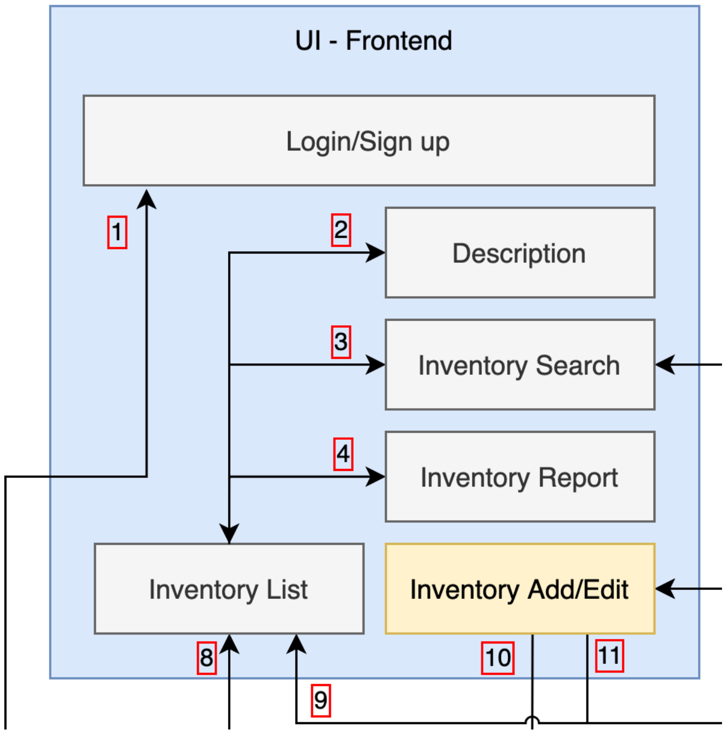

Figure 7: Add/Edit Bottle and Location subsystem

4.5.1 Assumptions

1. User will use this subsystem to add or edit wine/s in the inventory.
2. User has a camera in phone which works well.
3. User has an internet connection.

4.5.2 Responsibilities

1. If the name input by the user preexists, the quantity and a new location will be updated.
2. After the bar code is scanned in case of a new product, the option to add the location, description, picture will be provided.
3. After a product is added or edited, the display page of so will be available to the user.

4.5.3 Add-Edit bottle and Location Subsystem Interfaces

Table 6: Add-Edit Bottle and Location subsystem interfaces

ID	Description	Inputs	Outputs
#1	Inventory Display	Input 1, Input 12, Input 13	Output 9
#2	Add/Edit Inventory	Input 1, Input 12, Input 13	Output 11

4.6 Inventory Report - Sorting

The user can check what condition the bottle is in. The information if the bottle is sealed, opened or finished can be known through this functionality.

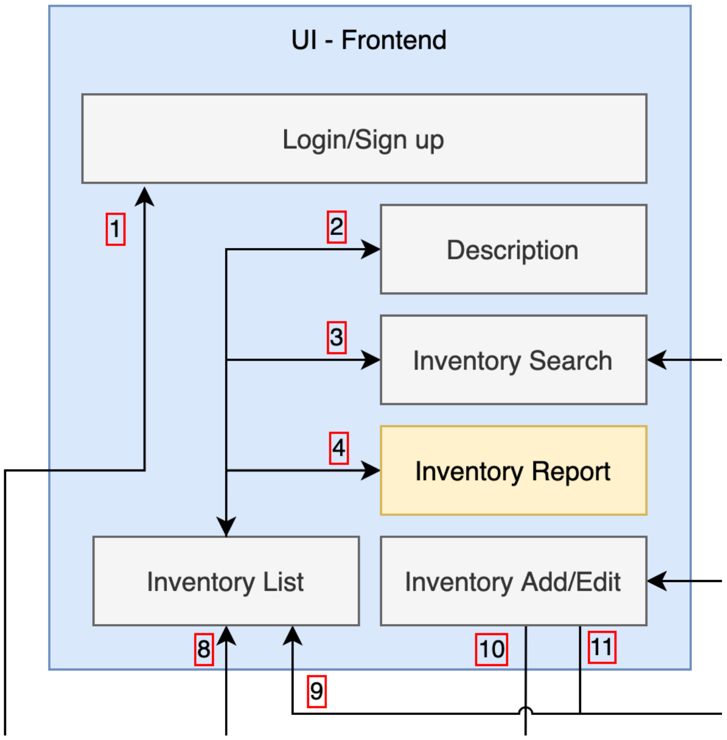

Figure 8: Inventory Report - Sorting subsystem

4.6.1 Assumptions

1. User has an internet connection.
2. The request made by user can be successfully executed through this layer of the subsystem.
3.This will fetch the data being query as per the API request and send the result back to the application.

4.6.2 Responsibilities

1. Must generate the correct report.
2. This subsystem will be responsible to process the product information in the inventory.

3. This subsystem is responsible to generate the detailed inventory report which as requested by the user.

4.6.3 Inventory Report - Sorting Subsystem Interfaces

Table 7: Inventory Report - Sorting subsystem interfaces

ID	Description	Inputs	Outputs
#1	Inventory Report	Input 1, Input 12, Input 13	Output 4

5 Phone Camera

The user will allow to scan the bar code with their mobile phone. It will let user to add their wine in the inventory which includes year of manufacture, type of brand, color, style, name, variety of wine. User can add or edit the location of the wine in the cells per the need. The user can easily monitor the status of the bottle, if it is open or sealed.

5.1 Scan Bottle Barcode

The user will allow to scan the barcode with their mobile phone. It will let user to add their wine in the inventory which includes year of manufacture, type of brand, color, style, name, variety of wine. User can add the location of the wine in the cell. The user can easily monitor the status of the bottle open or sealed.

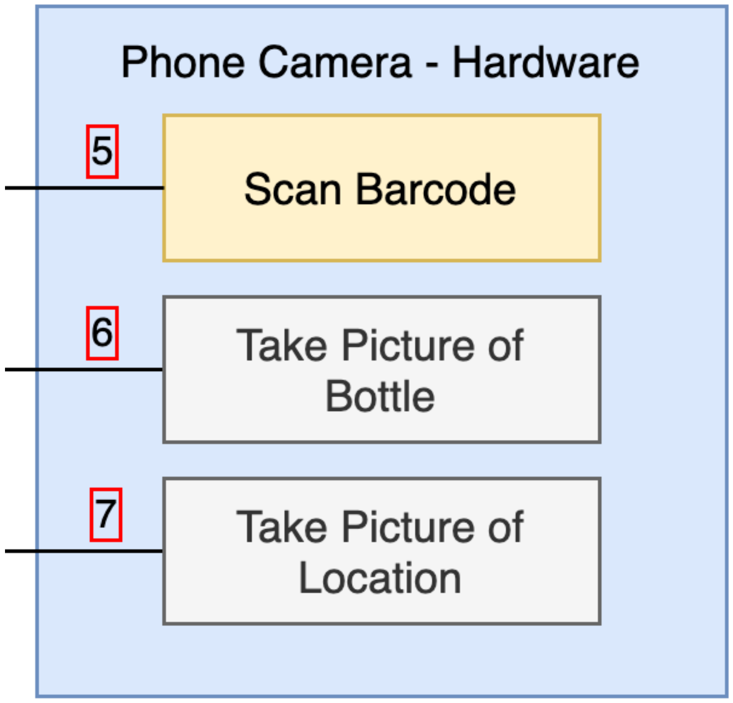

Figure 9: Scan Bottle Barcode subsystem

5.1.1 Assumptions

Barcode will be scan by the phone camera and auto generate the information of the wine.

5.1.2 Responsibilities

The barcode scanner API needs to create and so we will have all the information of the wine in the system like year, style, color, status of bottle, etc.

5.1.3 Subsystem Interfaces

Table 8: Scan Bottle Barcode subsystem interfaces

ID	Description	Inputs	Outputs
#1	Send scanned data	N/A	Output 5

5.2 Take Picture of Bottle

The user can take the picture of bottle using their phone. This picture or image will allow user to add information of bottle easily. The barcode scanner API uses the camera to scan the barcode. It will auto generate information about the bottle including year, style, type, as well as status of bottle through its image.

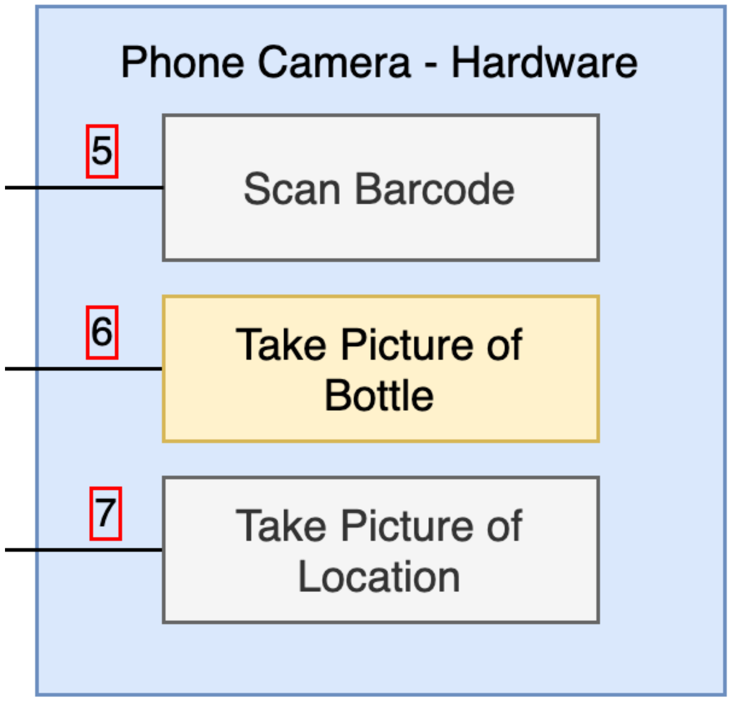

Figure 10: Take Picture of Bottle Subsystem

5.2.1 Assumptions

The user will take picture of bottle and scan in a barcode. It will auto generate information of bottle. The users do not need to do manually.

5.2.2 Responsibilities

The barcode scanner API with image/picture needs to create and add all the information about bottle of wine. When the user takes the picture of bottle then automatically fill the information.

5.2.3 Subsystem Interfaces

Table 9: Take Picture of Bottle subsystem interfaces

ID	Description	Inputs	Outputs
#1	Send captured photo	N/A	Output 6

5.3 Take Picture of Location

The user will allow to take picture of location. It will allow user another way to keep track of inventory as well. The picture of location will allow users to have photo of location and the barcode scanner API uses the camera to scan the barcode. It will auto generate all the information of the bottle while registering it.

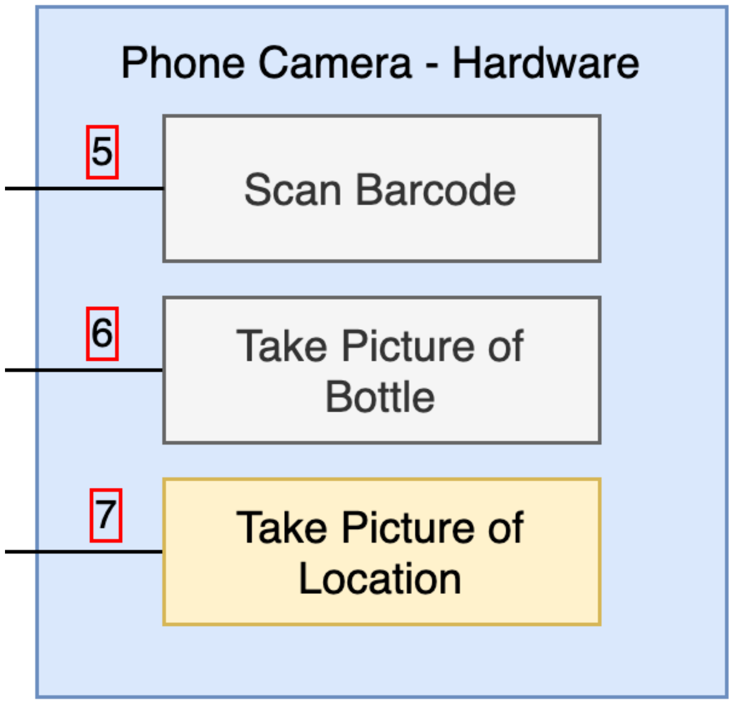

Figure 11: Take Picture of Location Subsystem

5.3.1 Assumptions

The user will take the picture of the location and get the information of the wine in the inventory.

5.3.2 Responsibilities

The barcode scanner API with picture of location need to create. It should include information required to add in the inventory.

5.3.3 SUBSYSTEM INTERFACES

Table 10: Take Picture of Location subsystem interfaces

ID	Description	Inputs	Outputs
#1	Send captured photo	N/A	Output 7

6 Firebase - Backend Subsystems

6.1 Realtime Database

Realtime Database is a NoSQL JSON database that can update in realtime. The database will communicate with the frontend layer. Note that Firebase Realtime Database and Authentication are linked regarding security rules. Session information received from Authentication is supplied through the API to database requests and Firebase internally decides whether or not to honor the request based on the security rules.

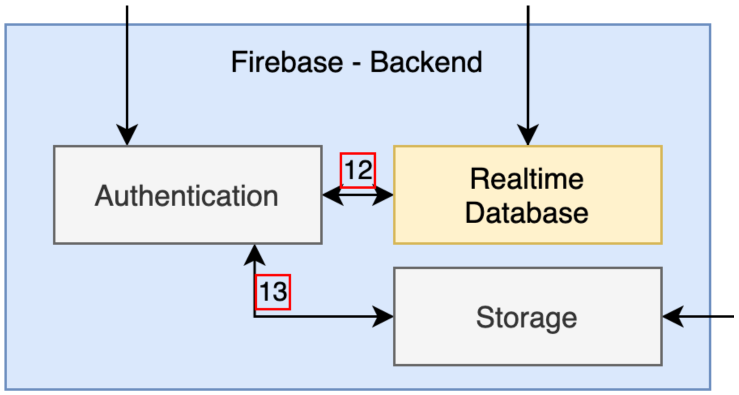

Figure 12: Realtime Database subsystem

6.1.1 Assumptions

JSON data is formed of basic types (objects or arrays containing strings, numbers, Booleans, and other objects or arrays).

6.1.2 Responsibilities

Prior to launch, we will need to set up security rules so that data is properly protected and isolated. Session information generated by Authentication is provided with database API calls; Firebase internally decides whether to accept the request based on security rules. Realtime Database only stores JSON data; the Storage subsystem will handle images.

6.1.3 Subsystem Interfaces

Table 11: Realtime Database subsystem interfaces

ID	Description	Inputs	Outputs
#1	Retrieving inventory items	Input 8	Output 8
#2	Inserting and modifying items	Input 10	N/A
#3	Authorizing read/write requests	Input 12 Input 13	Output 12 Output 13

6.2 Storage

Firebase Storage handles file uploads and downloads for a cloud bucket.

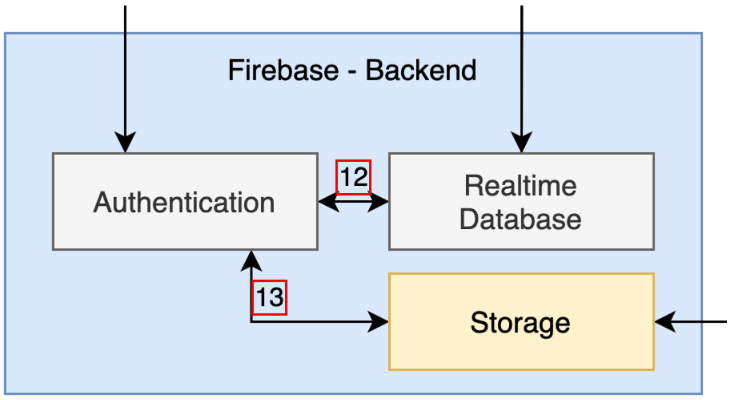

Figure 13: Storage subsystem

6.2.1 Assumptions

Files should be less than 1GB in size. We are only handling images, so this should be plenty.

6.2.2 Responsibilities

Prior to launch, we will need to set up security rules so that data is properly protected and isolated. Realtime Database only stores JSON data; the Storage subsystem will handle images. Like Realtime Database, we will need to configure security rules. Authentication will send session information with API calls to Storage.

6.2.3 Subsystem Interfaces

Table 12: Storage subsystem interfaces

ID	Description	Inputs	Outputs
#1	Downloading images to show on the inventory screen	Input 9	Output 9
#2	Saving images for new or modified items	Input 11	N/A
#3	Authorizing read/write requests	Input 12 Input 13	Output 12 Output 13

6.3 Authentication

Authentication provides an interface for our app to create and authenticate users.

6.3.1 Assumptions

Users may choose email/password authentication or third-party services like Facebook, Google, or Sign in With Apple.

6.3.2 Responsibilities

Error codes returned from Authentication need useful error messages (i.e. wrong credentials, user already exists, etc). Passwords should be sufficiently strong.

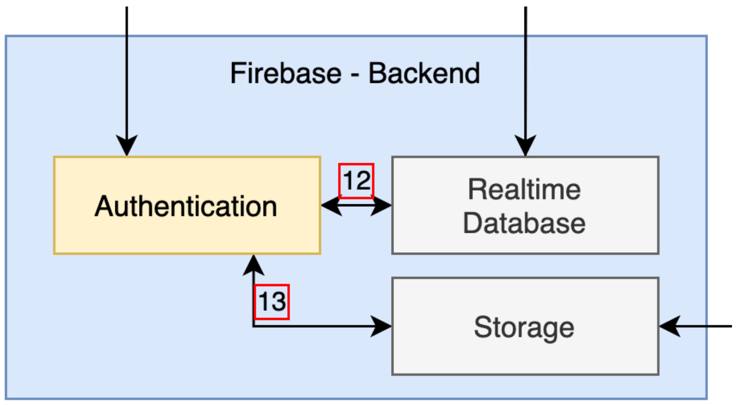

Figure 14: Authentication subsystem

6.3.3 Subsystem Interfaces

Table 13: Authentication subsystem interfaces

ID	Description	Inputs	Outputs
#1	Authenticating users, creating accounts, and resetting passwords	Input 1	Output 1
#2	Authorizing read/write requests	Input 12 Input 13	Output 12 Output 13