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


# Detailed Design Specification
# CSE 4317: Senior Design II
# Fall 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 | 09.20.2021 | GW | document creation |
| 1.0 | 10.25.2021 | TV, SL, MU, LT, GW | First draft |

# CONTENTS


**1 Introduction** **5**

**2 System Overview** **5**
- 2.1 UI - Frontend . . . 6
- 2.2 Phone Camera - Hardware . . . 6
- 2.3 Firebase - Backend . . . 6

**3 Subsystem Definitions & Data Flow** **7**

**4 UI - Frontend Subsystems** **8**
- 4.1 Layer Hardware . . . 8
- 4.2 Layer Operating System . . . 8
- 4.3 Layer Software Dependencies . . . 8
- 4.4 User Login, Sign-up, Forgot Password and Change Password . . . 8
- 4.5 Inventory List Display . . . 9
- 4.6 Bottle's Description . . . 10
- 4.7 Inventory Search . . . 11
- 4.8 Add/Edit bottle and Location . . . 12
- 4.9 Inventory Report - Sorting . . . 13

**5 Phone Camera - Hardware Subsystems** **15**
- 5.1 Layer Hardware . . . 15
- 5.2 Layer Operating System . . . 15
- 5.3 Layer Software Dependencies . . . 15
- 5.4 Scan Bottle Barcode . . . 15
- 5.5 Take Picture of Bottle . . . 16
- 5.6 Take Picture of Location . . . 17

**6 Firebase - Backend Subsystems** **19**
- 6.1 Layer Hardware . . . 19
- 6.2 Layer Operating System . . . 19
- 6.3 Layer Software Dependencies . . . 19
- 6.4 Realtime Database . . . 19
- 6.5 Storage . . . 20
- 6.6 Authentication . . . 21

## List of Figures


1 Architectural layer diagram . . . . . 5
2 Data flow diagram . . . . . 7
3 User Login, Sign-up, Forgot Password and Change Password Subsystem . . . . . 8
4 Inventory List Display Subsystem . . . . . 9
5 Bottle's Describtion System . . . . . 10
6 Inventory Search Subsystem interfaces . . . . . 11
7 Add/Edit Bottle and Location subsystem . . . . . 12
8 Inventory Report - Sorting subsystem . . . . . 13
9 Scan Bottle Barcode subsystem . . . . . 15
10 Take Picture of Bottle subsystem . . . . . 16
11 Take Picture of Location subsystem . . . . . 17
12 Realtime Database subsystem . . . . . 19
13 Storage subsystem . . . . . 20
14 Authentication subsystem . . . . . 21


## 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.

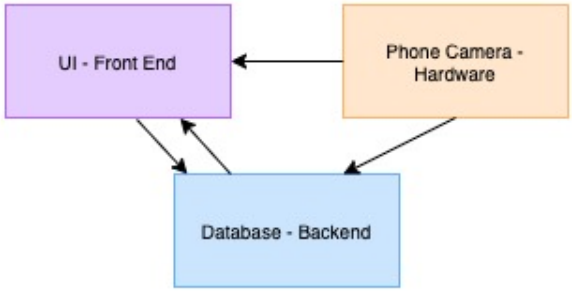


Figure 1: Architectural layer diagram

### 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

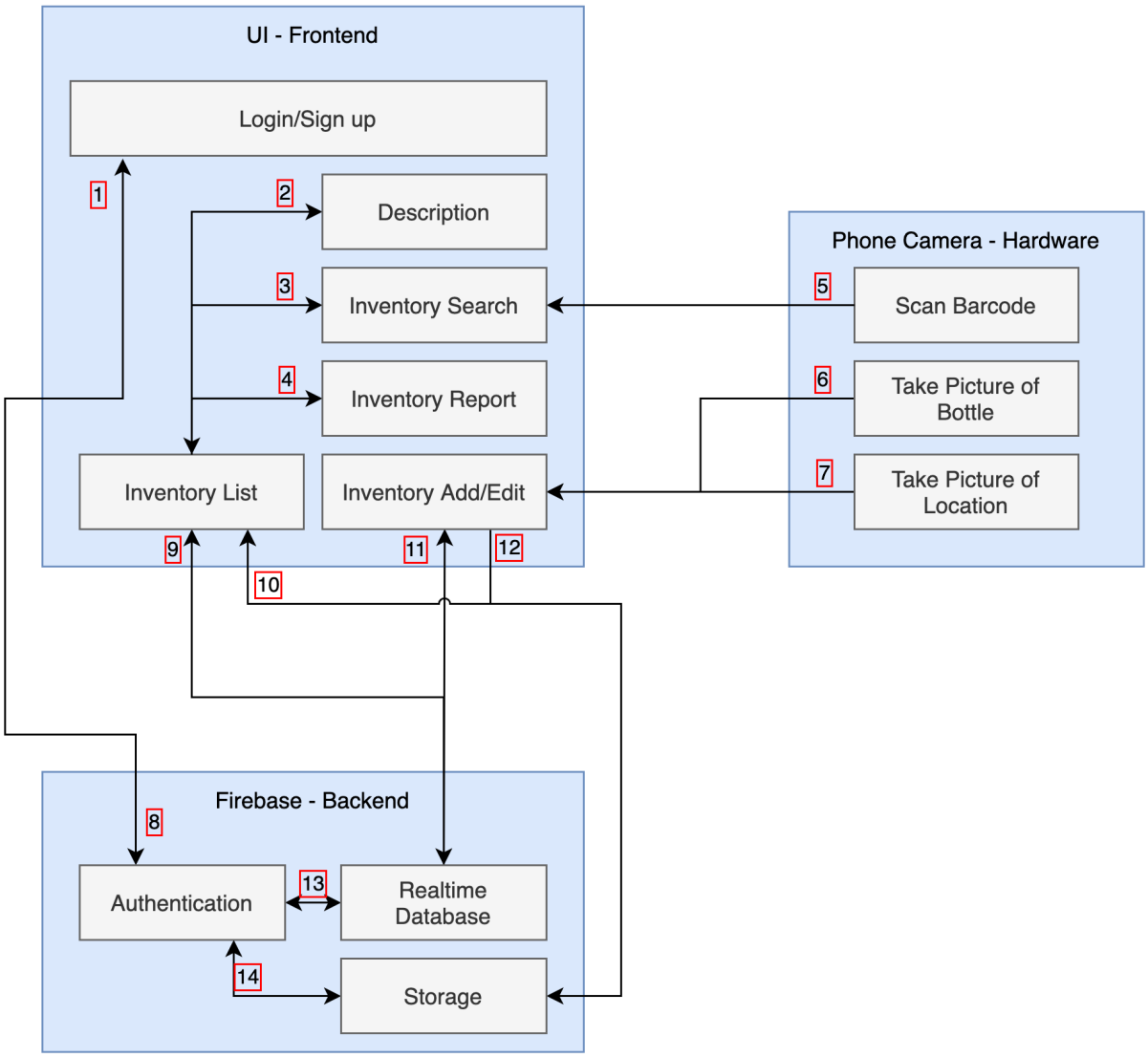


Figure 2: Data flow diagram

# 4 UI - Frontend Subsystems

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 where the user inputs the username and password to sign, Sign-up incase of a new user, Forgot Password and Change Password for various reasons; Inventory Display List which displays the list of wines in the inventory; Bottle's Description where the type of the wine, what it is good with and the condition is stored; Inventory Search where the user can search the desired wine; Add/Edit Bottle's Description - Location; Inventory Report where the report of all the inventory can be generated.

## 4.1 Layer Hardware

This layer uses the screen of the device in which the app will run.

## 4.2 Layer Operating System

This layer uses the operating system (android or iOS) of the device in which the app will run.

## 4.3 Layer Software Dependencies

This layer uses React Native framework.

## 4.4 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.

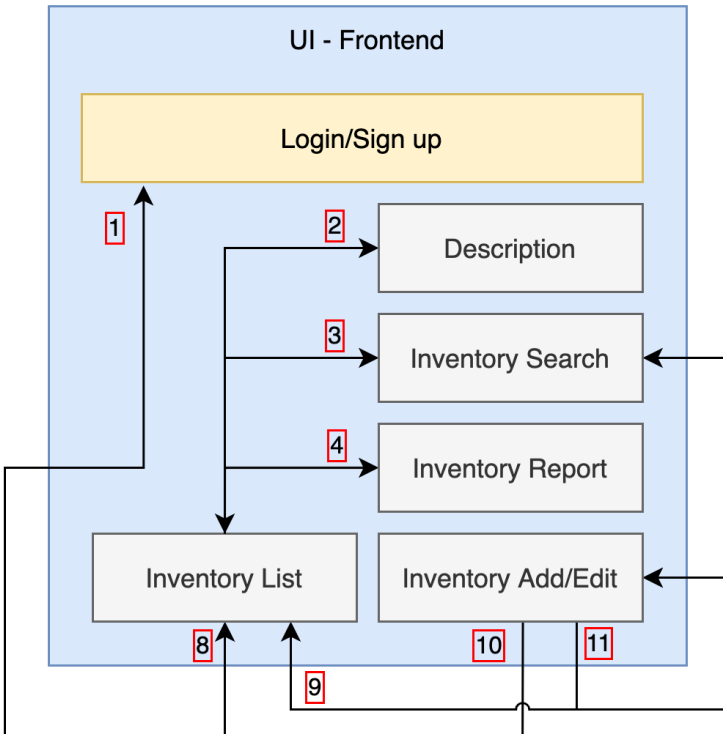


Figure 3: User Login, Sign-up, Forgot Password and Change Password Subsystem

### 4.4.1 Subsystem Hardware

See section 4.1.

### 4.4.2 Subsystem Operating System

See section 4.2.

### 4.4.3 Subsystem Software Dependencies

See section 4.3.

### 4.4.4 Subsystem Programming Languages

Will be written in JavaScript.

### 4.4.5 Subsystem Data Structures

No particular data structure is required.

### 4.4.6 Subsystem Data Processing

No particular data processing algorithm is required.

## 4.5 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.

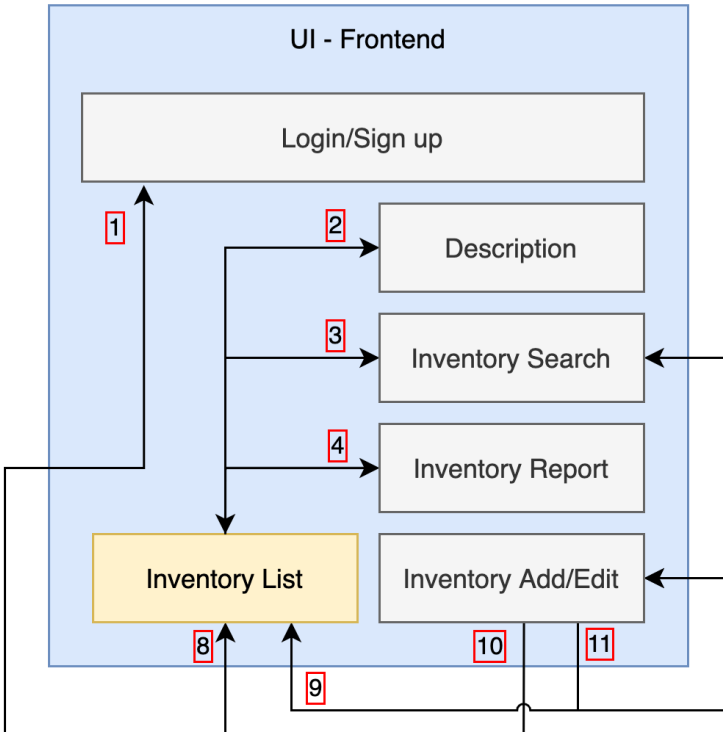


Figure 4: Inventory List Display Subsystem

### 4.5.1 Subsystem Hardware

See section 4.1.

### 4.5.2 Subsystem Operating System

See section 4.2.

### 4.5.3 Subsystem Software Dependencies

See section 4.3.

### 4.5.4 Subsystem Programming Languages

Will be written in JavaScript.

### 4.5.5 Subsystem Data Structures

No particular data structure is required.

### 4.5.6 Subsystem Data Processing

No particular data processing algorithm is required.

## 4.6 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.

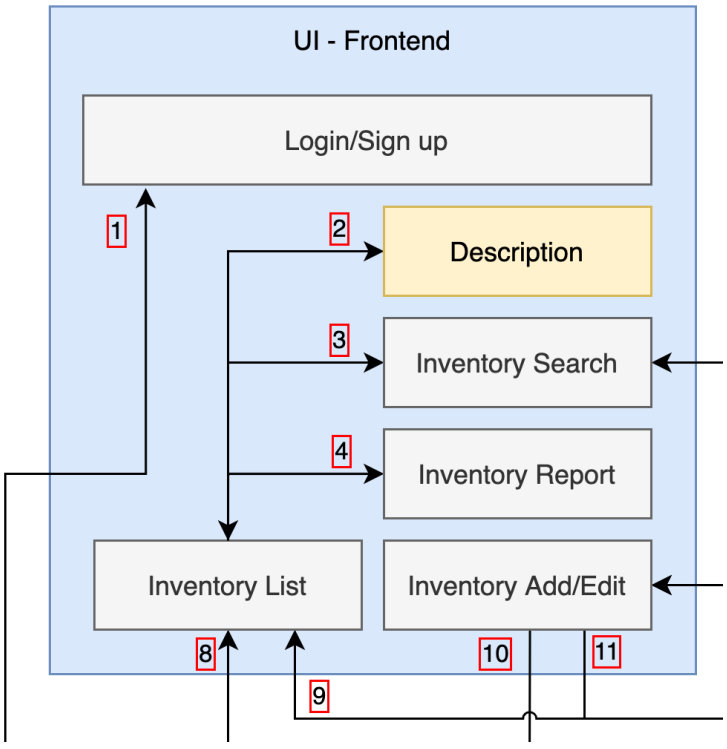


Figure 5: Bottle's Describtion System

### 4.6.1 Subsystem Hardware

See section 4.1.

### 4.6.2 Subsystem Operating System

See section 4.2.

### 4.6.3 Subsystem Software Dependencies

See section 4.3.

### 4.6.4 Subsystem Programming Languages

Will be written in JavaScript.

### 4.6.5 Subsystem Data Structures

No particular data structure is required.

### 4.6.6 Subsystem Data Processing

No particular data processing algorithm is required.

## 4.7 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.

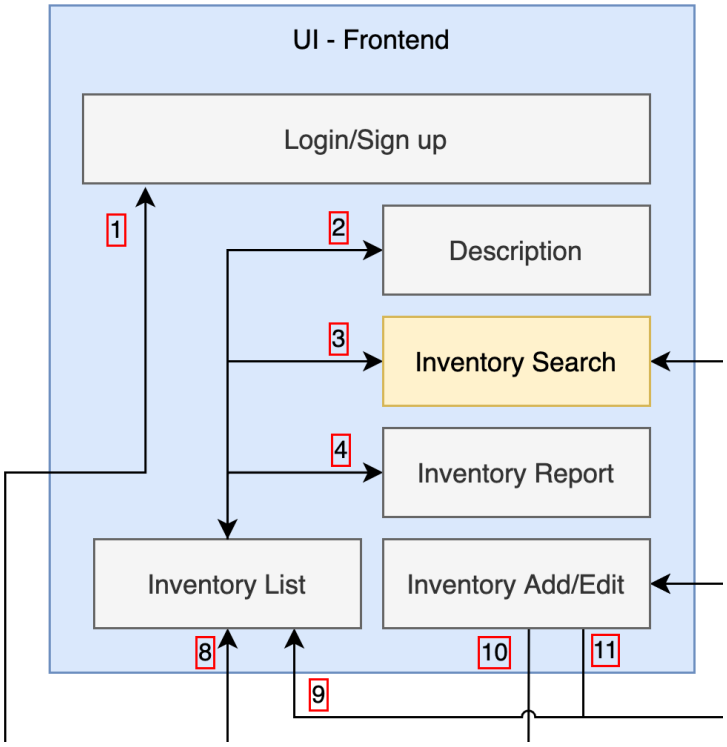


Figure 6: Inventory Search Subsystem interfaces

### 4.7.1 Subsystem Hardware

See section 4.1.

### 4.7.2 Subsystem Operating System

See section 4.2.

### 4.7.3 Subsystem Software Dependencies

See section 4.3.

### 4.7.4 Subsystem Programming Languages

Will be written in JavaScript.

### 4.7.5 Subsystem Data Structures

No particular data structure is required.

### 4.7.6 Subsystem Data Processing

No particular data processing algorithm is required.

## 4.8 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.

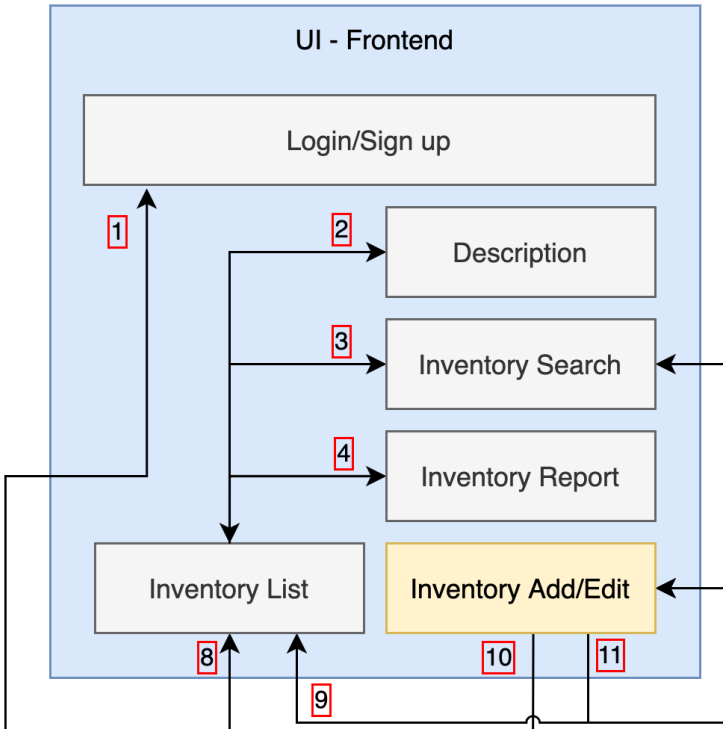


Figure 7: Add/Edit Bottle and Location subsystem

### 4.8.1 Subsystem Hardware

See section 4.1.

### 4.8.2 Subsystem Operating System

See section 4.2.

### 4.8.3 Subsystem Software Dependencies

See section 4.3.

#### 4.8.4 Subsystem Programming Languages

Will be written in JavaScript.

#### 4.8.5 Subsystem Data Structures

No particular data structure is required.

#### 4.8.6 Subsystem Data Processing

No particular data processing algorithm is required.

### 4.9 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.

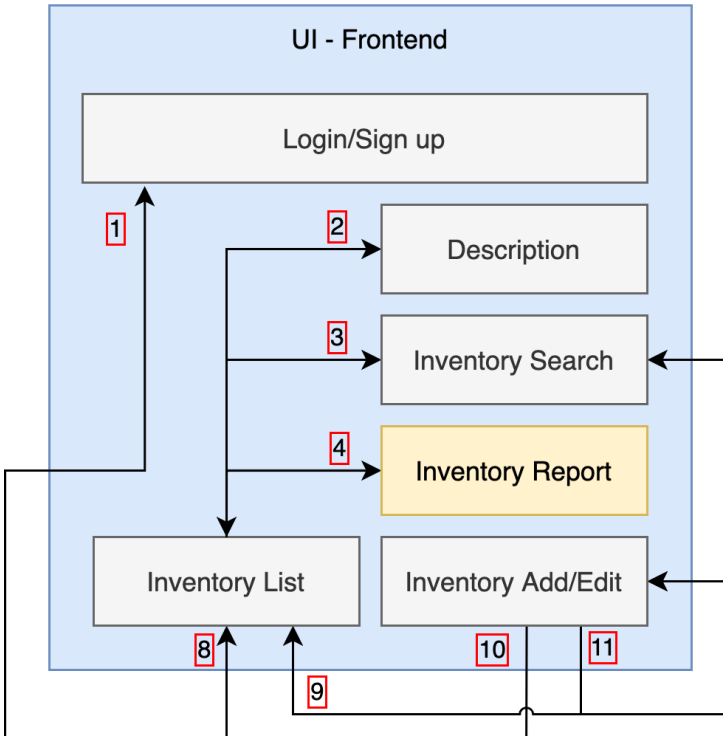


Figure 8: Inventory Report - Sorting subsystem

#### 4.9.1 Subsystem Hardware

See section 4.1.

#### 4.9.2 Subsystem Operating System

See section 4.2.

#### 4.9.3 Subsystem Software Dependencies

See section 4.3.

#### 4.9.4 Subsystem Programming Languages

Will be written in JavaScript.

### 4.9.5 Subsystem Data Structures

No particular data structure is required.

### 4.9.6 Subsystem Data Processing

No particular data processing algorithm is required.

# 5 Phone Camera - Hardware Subsystems

This app will have feature to scan barcode to register bottle in the inventory. App will use the phone camera. The user does not have to enter wine manually. The user can scan bottle barcode or take picture of bottle or take picture of location to create and add information about the wine. It will be easy and faster to keep track of wine. The user can use their phone camera to scan barcode. This barcode scanner will automatically include information in inventory.

## 5.1 Layer Hardware

The camera hardware will be pre-installed in the devices running the app.

## 5.2 Layer Operating System

The device's own operating system automatically manages the camera.

## 5.3 Layer Software Dependencies

The Expo SDK provides access to device and system functionality such as camera. We will use Expo SDK API for react native to access camera to scan the barcode. We will install modules for the Expo SDK and will import modules from it in our JavaScript code.

## 5.4 Scan Bottle Barcode

The user will be able 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. The user can easily monitor the status of the bottle open or sealed. Barcode will be scanned by the phone camera and auto generate the information of the wine.

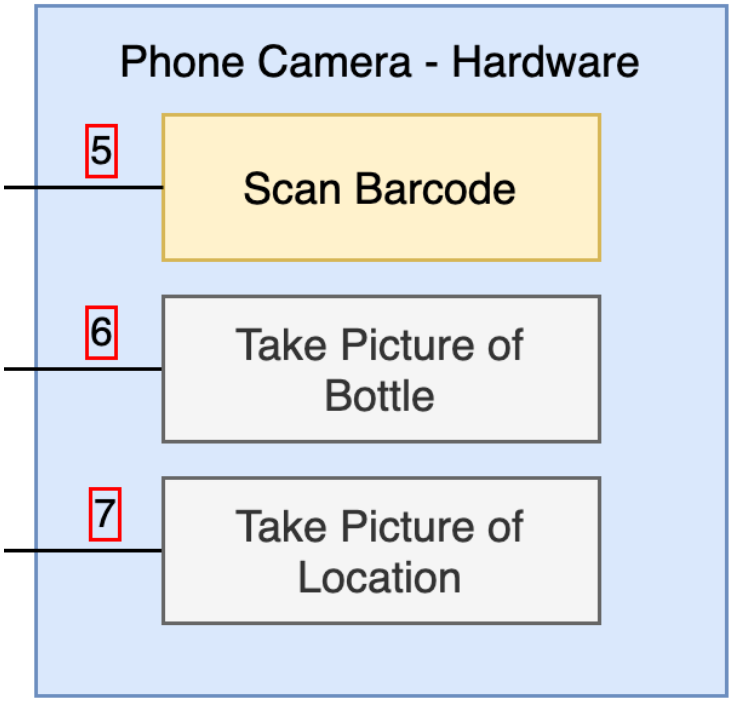


Figure 9: Scan Bottle Barcode subsystem

### 5.4.1 Subsystem Hardware

See section 5.1.

### 5.4.2 Subsystem Operating System

See section 5.2.

### 5.4.3 Subsystem Software Dependencies

In addition to the Expo SDK described in section 4.3, we will use a barcode scanning library to read the encoded data.

### 5.4.4 Subsystem Programming Languages

We will use JavaScript to interact with these APIs.

### 5.4.5 Subsystem Data Structures

No special data structures.

### 5.4.6 Subsystem Data Processing

The barcode scanning library will automatically detect and decode multiple types of barcodes (code39, code93, datamatrix, etc).

## 5.5 Take Picture of Bottle

To help identify bottles, the user can capture a photo of them.

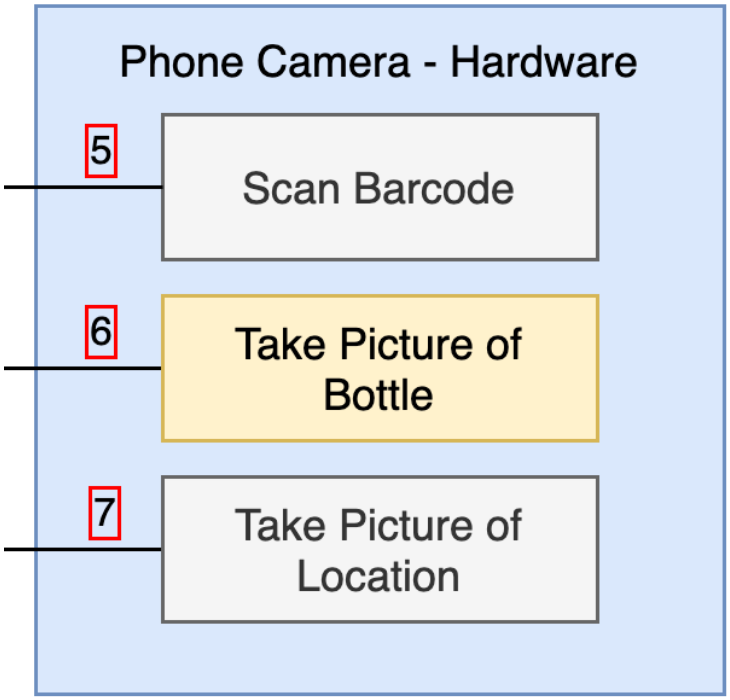


Figure 10: Take Picture of Bottle subsystem

### 5.5.1 Subsystem Hardware

See section 4.1.

### 5.5.2 Subsystem Operating System

See section 4.2.

### 5.5.3 Subsystem Software Dependencies

In addition to the Expo SDK described in section 4.3, we will use a camera library to interact with the device's camera easily.

### 5.5.4 Subsystem Programming Languages

We will use JavaScript to interact with these APIs.

### 5.5.5 Subsystem Data Structures

No special data structures.

### 5.5.6 Subsystem Data Processing

No special data processing algorithms.

## 5.6 Take Picture of Location

Similar to section 4.4, the user can capture a photo of a bottle's general location to help find it later.

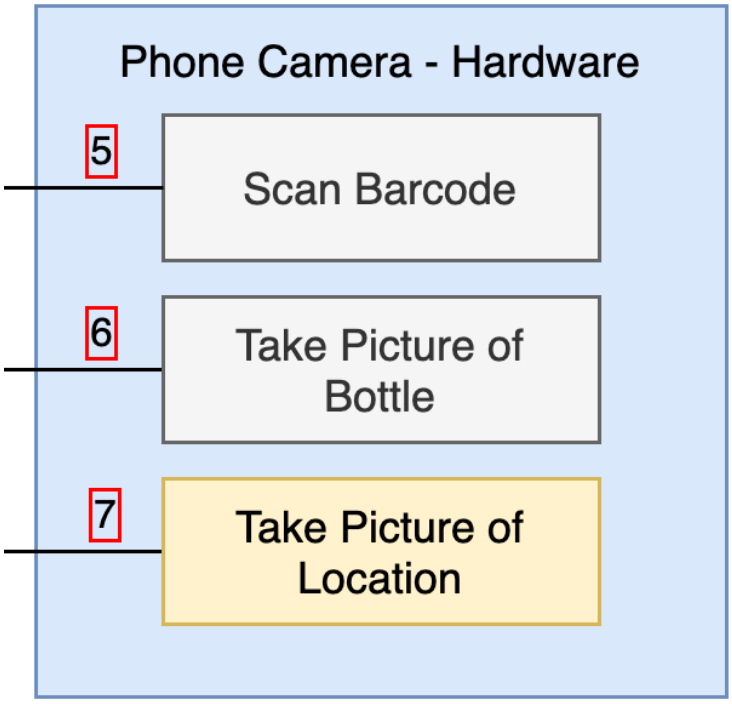


Figure 11: Take Picture of Location subsystem

### 5.6.1 Subsystem Hardware

See section 4.1.

### 5.6.2 Subsystem Operating System

See section 4.2.

### 5.6.3 Subsystem Software Dependencies

In addition to the Expo SDK described in section 4.3, we will use a camera library to interact with the device's camera easily.

### 5.6.4 Subsystem Programming Languages

We will use JavaScript to interact with these APIs.

### 5.6.5 Subsystem Data Structures

No special data structures.

### 5.6.6 Subsystem Data Processing

No special data processing algorithms.

# 6 Firebase - Backend Subsystems

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.

## 6.1 Layer Hardware

Since we are using Firebase, there are no particular hardware requirements on our end. Firebase already runs on Google hardware in the cloud and we communicate with it entirely through software API libraries provided by Google.

## 6.2 Layer Operating System

The server set up of Firebase/Google Cloud Platform is proprietary and not disclosed; however, we are not concerned with the operating system(s) of the servers anyway as Firebase/Google manages this.

## 6.3 Layer Software Dependencies

Firebase provides API libraries for multiple programming languages and environments. In our case, we'll use the JavaScript library which provides all of the interfaces we need.

## 6.4 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.

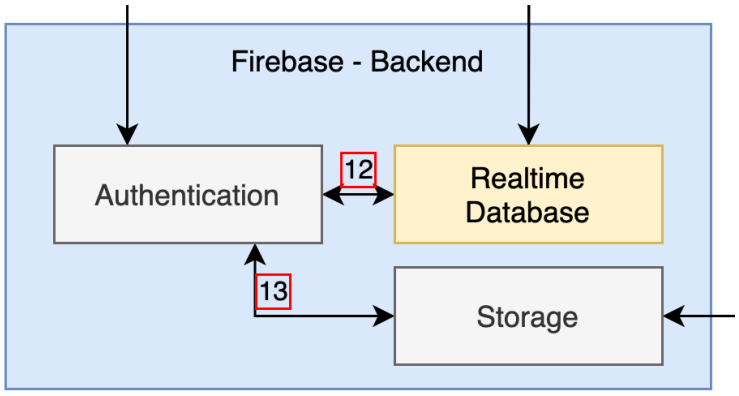


Figure 12: Realtime Database subsystem

### 6.4.1 Subsystem Hardware

We do not manage hardware for this subsystem. See section 5.1.

### 6.4.2 Subsystem Operating System

We do not manage the operating system for this subsystem. See section 5.2.

### 6.4.3 Subsystem Software Dependencies

API interfaces to the Realtime Database are included with the Firebase SDK for JavaScript. See section 5.3.

### 6.4.4 Subsystem Programming Languages

We will be using JavaScript with the Firebase SDK to communicate with the Realtime Database.

### 6.4.5 Subsystem Data Structures

Data is stored and retrieved from the database in the form of JSON objects and arrays, integrating nicely with JavaScript.

### 6.4.6 Subsystem Data Processing

No particular data processing algorithms are necessary for our project to use the Realtime Database. The bulk of our project will be transforming user input to well-formed JSON objects and representing JSON objects in a readable manner.

## 6.5 Storage

Firebase Storage handles file uploads and downloads for a cloud bucket. Similar to Realtime Database, Storage works in conjunction with Authentication and storage rules to manage access to uploaded files.

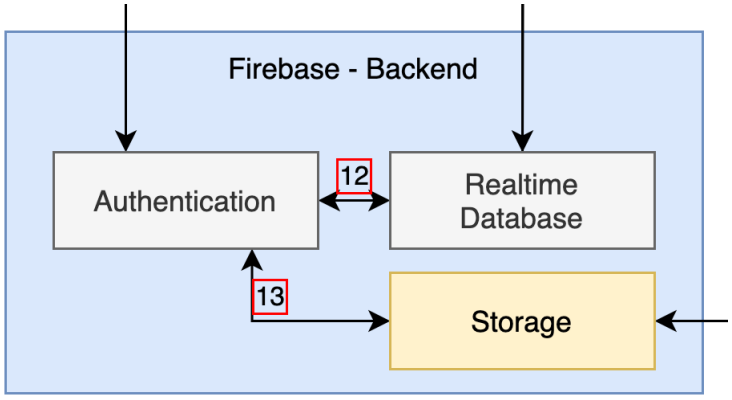


Figure 13: Storage subsystem

### 6.5.1 Subsystem Hardware

We do not manage hardware for this subsystem. See section 5.1.

### 6.5.2 Subsystem Operating System

We do not manage the operating system for this subsystem. See section 5.2.

### 6.5.3 Subsystem Software Dependencies

API interfaces to Storage are included with the Firebase SDK for JavaScript. See section 5.3.

### 6.5.4 Subsystem Programming Languages

We will be using JavaScript with the Firebase SDK to communicate with Storage.

### 6.5.5 Subsystem Data Structures

Files are uploaded in form of binary, base 64, or Blobs. JavaScript conveniently contains APIs for using these structures.

### 6.5.6 Subsystem Data Processing

No particular data processing algorithms are necessary for our project to use Storage. We will only be uploading/downloading/deleting user images of wine bottles. Any conversion to/from Blobs/binary/base64 encodings is provided by JavaScript itself or the Firebase SDK.

## 6.6 Authentication

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

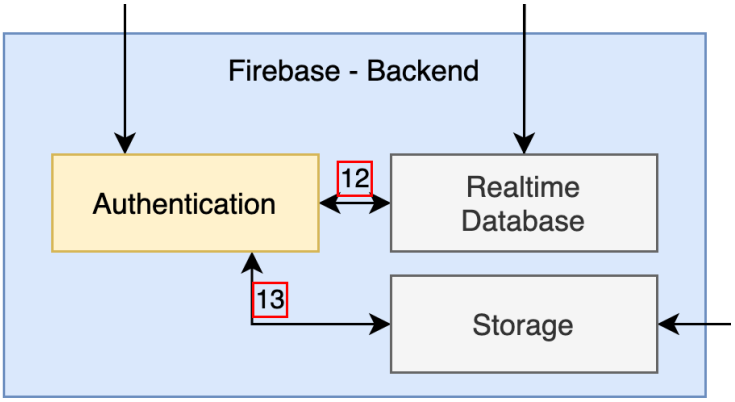


Figure 14: Authentication subsystem

### 6.6.1 Subsystem Hardware

We do not manage hardware for this subsystem. See section 5.1.

### 6.6.2 Subsystem Operating System

We do not manage the operating system for this subsystem. See section 5.2.

### 6.6.3 Subsystem Software Dependencies

API interfaces to Authentication are included with the Firebase SDK for JavaScript. See section 5.3.

### 6.6.4 Subsystem Programming Languages

We will be using JavaScript with the Firebase SDK to communicate with Authentication.

### 6.6.5 Subsystem Data Structures

No particular data structures are necessary for Authentication. The bulk of our project will be passing in usernames/passwords or 3rd party authentication tokens to the API.

### 6.6.6 Subsystem Data Processing

No particular data processing algorithms are necessary on our end to use Authentication. Any necessary work is done by the Firebase SDK and/or Google's servers.

## References