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

# Architectural Design Specification
# CSE 4316: Senior Design I
# Summer 2020


## The Brew Crew
## Beverage Management

Bishal Paudel
Nirjal Phaiju
Sima Raymajhi
Lokendra B. Chhetri
Kunal Samant

## Revision History

| Revision | Date | Author(s) | Description |
|---|---|---|---|
| 0.1 | 08.10.2020 | BP | document creation |
| 0.2 | 08.11.2020 | BP | Added architectural layer diagram and section 2 |
| 0.3 | 08.11.2020 | LC | Added section 1 |
| 0.4 | 08.12.2020 | LC | Working on section 6 |
| 0.5 | 08.12.2020 | NP | Added section 4 |
| 0.6 | 08.12.2020 | KS | Added section 5 |
| 0.7 | 08.12.2020 | SR | All sections Revision |
| 1.0 | 08.12.2020 | SR | Official Release |

# CONTENTS

## List of Figures



## List of Tables

# 1 INTRODUCTION

Beverage Management is an Android OS based application designed to manage the beverages in the household environment with significantly large collection in multiple setups. The product will virtually manage the collection allowing the user to store the beverages in different locations with customized racks.

Main intended purpose including keeping track of name, storage location, brewery, style, volume, manufactured date, best before date, and some other functionality including search and sort the beverage according to the date, style, etc.

Additional product concept includes the bar-code scanning capability, which ensure the initial setup of the inventory, and saving it the local data-set,a friendly user interface to keep it sleek and notification system to notify the user about certain important information.

Conclusively, the initial application release will on Android OS, however the team will also be working on iOS and web platform and will be released in future releases.

## 2 System Overview

The diagram (Figure 1) below shows the basic architectural layer diagram of the Beverage Management app. The overall structure of our app can be described using the popular three-layer architecture which consists of presentation layer, application layer and data access layer. The presentation layer is the top-most layer of our system which allows user to interact with the system. Application layer acts as an interface between the presentation layer and data access layer. This layer supports all of the core functions of our application. The data access layer is the layer where all the data and information are stored or retrieved from the database. In other words, the presentation layer takes input from the user and pass it to the application layer. The application layer then process those commands and pass the information to the data access layer. The data access layer either store the information on the database or retrieve the requested information from the database and pass it back to the application layer, and eventually to the presentation layer where the result is displayed in a user understandable format.

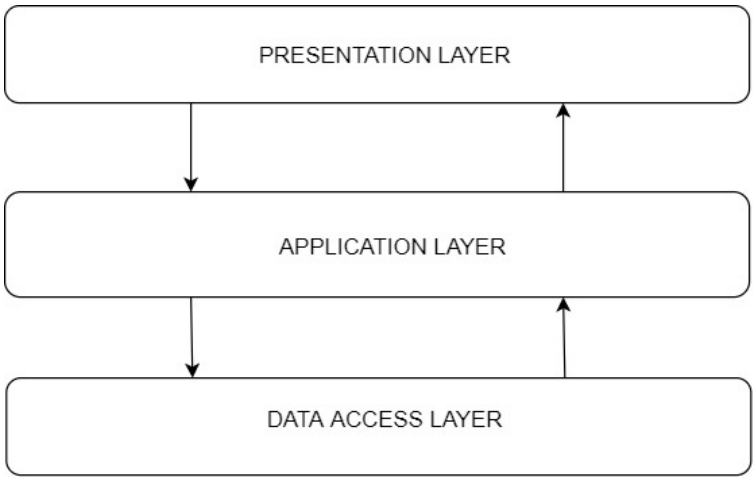


Figure 1: A simple architectural layer diagram

### 2.1 Presentation Layer Description

This layer will allow our application to successfully communicate with the user. The features will include the display of login page, page for storing new products in the inventory, and also displaying the desired output for the user. This user level layer is connected with our application layer which will help to display or retrieve the required information that our user is looking for.

### 2.2 Application Layer Description

This layer will serve as a bridge between the presentation layer and data access layer of our application. No matter what command the user gives in the presentation layer level, these commands will be interpreted by the application layer. Depending on the type of command, this layer will execute the command by accessing the database layer and then channels the accurate and expected output to the presentation layer.

### 2.3 Database Access Layer Description

This layer is the most fragile yet the most critical aspect of our application. This is where the information from the user is stored so that it can be accessed when required in future. We will be using subsytem such as Firebase, SQLite and Global barcode inventory. This layer will be the fulcrum to our team creating and effective management system. Multiple information will be recorded in this layer. This

includes login credentials, a productâs name, flavor, type and so on. In brief, this layer will execute the query given by the user in presentation layer and process the result back to application layer which will finally process the information to be displayed in the presentation layer.

## 3 SUBSYSTEM DEFINITIONS & DATA FLOW

This section breaks down our layer abstraction to another level of detail. Altogether, our system is divided into 3 different layers, Presentation Layer, Application Layer, Data Access Layer. Each of these layers are further divided into multiple subsystems. The Presentation Layer has Request, XML and Response. The Application Layer is divided into Java, Database Connection and Data Access Query whereas the Data Access Layer consists of Firebase, SQLite and Global Barcode Inventory. When an user opens the app, they will see the presentation layer consisting of text, buttons, forms, etc made using XML. When user makes some request like checking whether "India pale ale" is present in the inventory, this layer communicates with the application layer which formats a query and passes it to the data access layer. The data access layer then runs this query and returns the results (list of all beverage matching that style) to the application layer which then converts these results into appropriate format so that it can be displayed in the presentation layer.

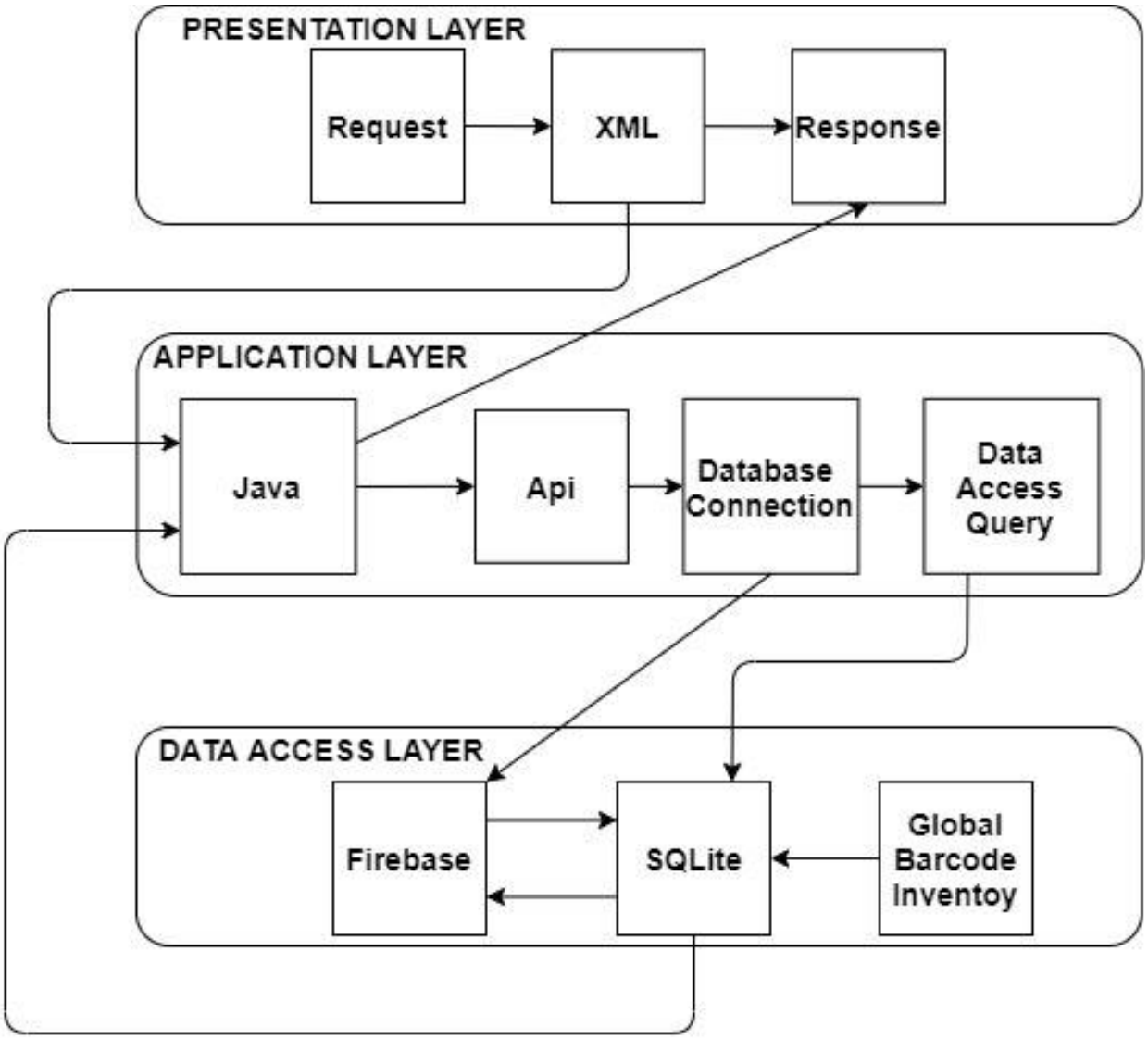


Figure 2: A simple data flow diagram

# 4 Presentation Layer

Presentation layer is where the interaction between human and machine takes place. This layer is further divided into 3 subsystems that continuously interact with each other to successfully communicate with the user. The screen of a mobile device will help the user to explore the various features of the application. Request subsystem is involved in receiving the commands of the user, XML subsystem will receive the command from Request subsystem. This subsystem will help to decode the instructions and complete the task. The response subsystem will display the result after the command is executed.

## 4.1 Request

Request is a fundamental subsystem of the Presentation layer. This subsystem will help to process the user given command. The commands may vary; the variety includes, signing up for new account, logging in to an already existing account, add to item to inventory, delete items from inventory, and search the inventory.

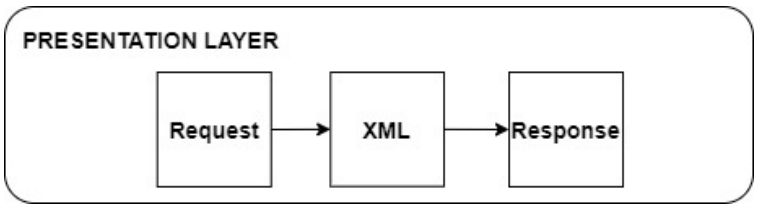


Figure 3: Example subsystem description diagram

### 4.1.1 Assumptions

We assume that all the data entered by the user are accurate and valid for the application layer to process.

### 4.1.2 Responsibilities

This subsystem will be responsible to process the instructions entered by the user to the application layer.

### 4.1.3 Subsystem Interfaces

Each of the inputs and outputs for the subsystem are defined here. Create a table with an entry for each labelled interface that connects to this subsystem. For each entry, describe any incoming and outgoing data elements will pass through this interface.

Table 2: Subsystem interfaces

| ID | Description | Inputs | Outputs |
|---|---|---|---|
| #1 | The request subsystem will interact with the user to understand what the wants from the application. This will include SignUp, Login, store information or retrieve information from database | Email ID, User name, password, Age, First Name and Last Name | Registered or denied |
| #2 | The request subsystem will interact with the user to understand what the wants from the application. This will include Login. | username, password | login success or denied due to incorrect credentials |

## 4.2 XML

XML is the unseen mechanics of our Presentation layer. The layouts, images or buttons seen in the application are functional because of XML. There are various kinds of XML files. Manifest XML file will help to define the functionality of buttons, layout XML file will help to determine the layout and many more XML files are available to help make the user-system interaction easy.

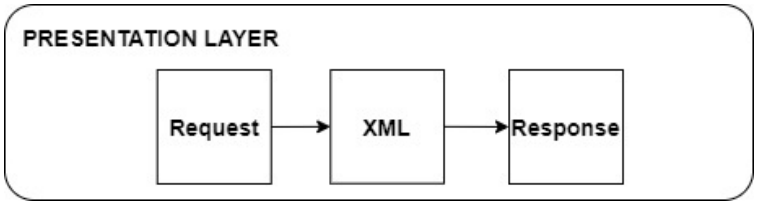


Figure 4: Example subsystem description diagram

### 4.2.1 Assumptions

The user will issue right commands.

### 4.2.2 Responsibilities

The main responsibility of this subsystem is to form a connection with application layer from the presentation layer depending on what the user wants to do.

### 4.2.3 Subsystem Interfaces

Each of the inputs and outputs for the subsystem are defined here. Create a table with an entry for each labelled interface that connects to this subsystem. For each entry, describe any incoming and outgoing data elements will pass through this interface.

Table 3: Subsystem interfaces

| ID | Description | Inputs | Outputs |
|---|---|---|---|
| #1 | User wants to search for an item(s) already in the inventory | Items to search by name, expiry date, flavor | Lists of product that match the search criteria. |

## 4.3 RESPONSE

This subsystem is essential in order to successfully display the output that is asked by the user. The information gained from the request layer is processed by the application layer using database access layer if necessary in order to produce the desired out for the response subsystem.

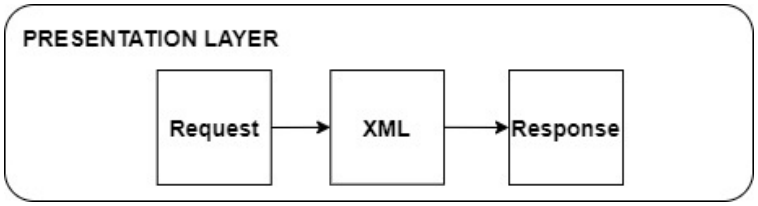


Figure 5: Example subsystem description diagram

### 4.3.1 ASSUMPTIONS

We assume that the query made by user can be successfully executed by application layer.

### 4.3.2 RESPONSIBILITIES

This subsystem is responsible to display the result that is required from the users query.

### 4.3.3 SUBSYSTEM INTERFACES

Each of the inputs and outputs for the subsystem are defined here. Create a table with an entry for each labelled interface that connects to this subsystem. For each entry, describe any incoming and outgoing data elements will pass through this interface.

Table 4: Subsystem interfaces

| ID | Description | Inputs | Outputs |
|---|---|---|---|
| #1 | The user wants to look in to inventory for a certain product. | Name of the product or search by specific requirement. | Detailed description of the product. |

# 5 Application Layer

This layer processes the data scanned from the presentation layer and stores the data in local variables, i.e. name of the beverage and expiry date. This data is added to the current user's beverage database and send the updated data back to the presentation layer for further input.

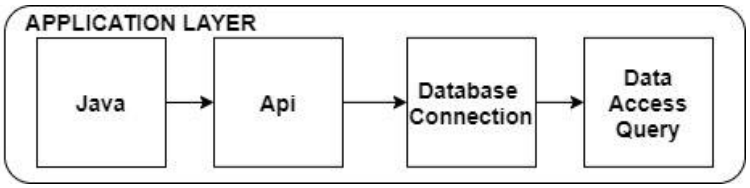


Figure 6: Example subsystem description diagram

## 5.1 Java Subsystem

The Java Subsystem is responsible for taking the XML data provided from the Presentation layer and and send it to the API subsystem to connect to the dataset and accurately get the required information. After the data is retrieved from the dataset, this subsystem passes the result to the Response subsystem in the Presentation layer.

### 5.1.1 Assumptions

- The xml input is in the proper format.

### 5.1.2 Responsibilities

The application must successfully connect to the database.

### 5.1.3 Subsystem Interfaces

Table 5: Subsystem interfaces

| ID | Description | Inputs | Outputs |
|---|---|---|---|
| #1 | XML to Java | XML file | API |
| #2 | SQLite3 to Java | Beverage Data | Formatted Data |
| #3 | Java to Response | Formatted Data | |

## 5.2 API Subsystem

The API subsystem passes the barcode to the Data Access Layer and retrieves the information of the beverage being scanned. The data retrieved is added to the current dataset of beverages which the user owns.

### 5.2.1 Assumptions

- The API exists in the dataset.

### 5.2.2 Responsibilities

The API must be able to establish fetch data on the database.

#### 5.2.3 Subsystem Interfaces

Table 6: Subsystem interfaces

| ID | Description | Inputs | Outputs |
|---|---|---|---|
| #1 | Java to API | Product ID | API fetche call |

### 5.3 Database Connection Subsystem

This subsystem establishes the connection between the database (Firebase) and the application.

#### 5.3.1 Assumptions

- A database is present and operational.

#### 5.3.2 Responsibilities

The connection must be present at all times to allow for the various features to be implemented properly.

#### 5.3.3 Subsystem Interfaces

Table 7: Subsystem interfaces

| ID | Description | Inputs | Outputs |
|---|---|---|---|
| #1 | API to Database | API call | Database connection |

### 5.4 Data Access Query Subsystem

This will fetch the data being query as per the API request and send the result back to the application.

#### 5.4.1 Assumptions

- The query is always valid.

#### 5.4.2 Responsibilities

Must return the correct data as per the query.

#### 5.4.3 Subsystem Interfaces

Table 8: Subsystem interfaces

| ID | Description | Inputs | Outputs |
|---|---|---|---|
| #1 | Query to Database | Data in the dataset | JSONified data |

# 6 Data Access Layer Subsystems

This layer is the most fragile yet the most critical aspect of our application. This is where the information from the user is stored so that it can be accessed when required in future. We will be using subsytem such as Firebase, SQLite and Global barcode inventory. This layer will be the fulcrum to our team creating and effective management system. Multiple information will be recorded in this layer. This includes login credentials, a productâs name, flavor, type and so on. In brief, this layer will execute the query given by the user in presentation layer and process the result back to application layer which will finally process the information to be displayed in the presentation layer.

## 6.1 Firebase

Firebase is Google's mobile platform a cloud-hosed database where data is stored as JSON and is synchronized in the realtime. Beverage Management will be build on the Android OS, all of our client will share one Realtime Database instance and will receive updates with the newest data.

This subsystem will be our main component, it will store and retrieve the user's access credentials, will engage with Database connectivity of the application layer, and SQLite subsystem to store and retrieve data locally.

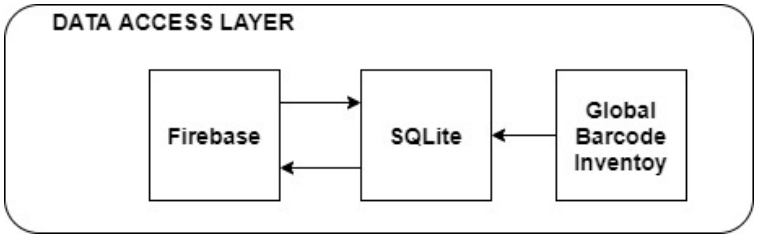


Figure 7: Example subsystem description diagram

### 6.1.1 Assumptions

We are going to assume that the:

- The connection with Application layer will be made successfully via Database Connectivity to initiate the firebase.
- The connection between Firebase and the SQLite is established successfully.

### 6.1.2 Responsibilities

This subsystem will be responsible to process the access to the SQLite information if the credentials are correct.

### 6.1.3 Subsystem Interfaces

Table 9: Firebase interfaces

| ID | Description | Inputs | Outputs |
|---|---|---|---|
| #1 | Firebase will have direct relation with the application layer. | Takes credentials information as a input | Gives access rights as an output to the SQLite. |

## 6.2 SQLite

SQLite is a programming library which implements a relational database management system. The SQLite database concept is, in contrast to other client-server systems, to be linked into the applications code, instead of providing a standalone daemon with which an application can communicate to request or write data. Because of the small size of the library itself, and the ease of use, it is esepcially interesting for embedded systems. SQLite supports a variety of SQL- (Structured Query Language) commands with some exceptions and does not provide any access or user-management. That means, that everyone, who can access the database file, can access the data as well as write (change, delete, add) data, if he can write the database file. It therefore inherits the access permissions of the filesystem.

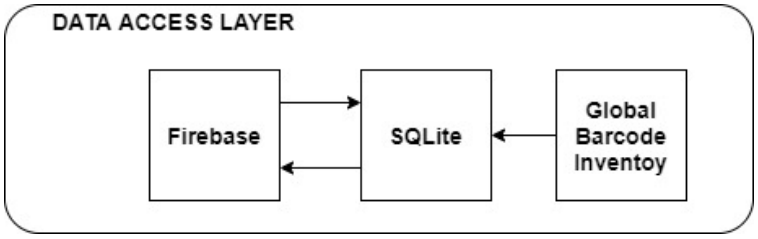


Figure 8: Example subsystem description diagram for SQLite

### 6.2.1 Assumptions

We are going to assume that we will be reading and writing data to the SQLite as a local database at this moment.

### 6.2.2 Responsibilities

This subsystem will be responsible to process the information that will be stored in the database locally and the application layer can access the information.

### 6.2.3 Subsystem Interfaces

Table 10: SQLite interfaces

| ID | Description | Inputs | Outputs |
|---|---|---|---|
| #1 | SQLite will communicate via Firebase subsystem component and directly from Data Access Query. | Takes instruction from Application layer in the form of queries. | Gives the appropriate response to the instruction provided to the Java Application layer. |
| #2 | SQLite will also intended to communicate with Global Barcode Inventory subsystem component. | Takes the response from Global Barcode Inventory. | Gives the appropriate response to the instruction provided to the Java Application layer. |

## 6.3 Global Barcode Inventory

Barcodes are symbols that can be scanned electronically using laser or camera-based systems. They are used to encode information such as product numbers, serial numbers and batch numbers. We will be using some third party global barcode inventory that will provide us with all beverages in the industry.

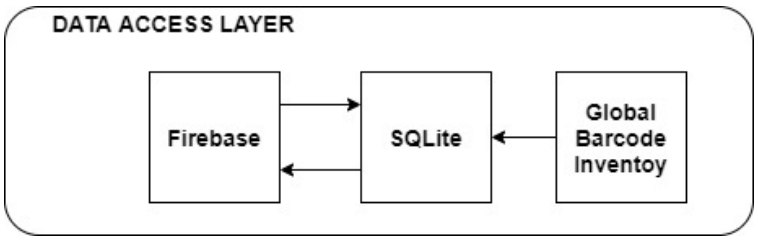


Figure 9: Example subsystem description diagram for Global Barcode Inventory

### 6.3.1 Assumptions

We are going to assume that the inventory are accurate.

### 6.3.2 Responsibilities

This subsystem will be responsible to process the product information in the industry currently.

### 6.3.3 Subsystem Interfaces

Table 11: Global Barcode Inventory interfaces

| ID | Description | Inputs | Outputs |
|---|---|---|---|
| #1 | SQLite will access the product information form the inventory subsystem. | Takes instruction from Application layer in the form of queries. | Gives the appropriate response to the SQLite and will to the Application layer. |

## References