# Beverage Booker
# Architecture Notebook

## Version Information

| Version | Date | Remarks | Author |
|---|---|---|---|
| 1.0 | 13/04/20 | Initial Architecture Notebook created for LCOM submission | Jake Durnford |
| 1.1 | 13/06/20 | Added version information<br><br>Removed LCOM Use-Case Model and replaced it with updated LCAM Use-Case Model | Benn Curby |
| 1.2 | 14/06/20 | Added more formatting (spacing) to the entire document to improve readability<br><br>Made minor edits to the section 2: Architectural goals and philosophy.<br>Added more specific information under Compatibility about the Android environment we are developing the app for.<br><br>Removed Table Booking mechanism and Event Booking mechanism from section 6: Architectural mechanisms. This is to conform with the revised scope of our project.<br><br>Removed Booking a Table and Booking an Event from the Operational Views in section 9: Architectural Views. This is to conform with the revised scope of our project.<br><br>Updated Operational Views. Added and refined steps in 'Creating an Account', 'Logging into an Account' and 'Placing an Order' based on how our implementation is now working. | Benn Curby |
| 1.3 | 24/06/20 | Added Staff App Class Diagram, updated component diagram and 'Viewing Active Order (cafe staff)' | Jake Durnford |

| | | | |
| --- | --- | --- | --- |
| | | operational view to section 9. Architectural views.<br><br>Updated sections 8. Layers or architectural framework to better reflect how the system functions.<br><br>Added to sections 4. Architecturally significant requirements and 5. Decisions, constraints, and justifications to address there being 2 separate apps, for the customers and then for the staff. | |
| 2.0 | 24/06/20 | Section 1. Purpose updated.<br>Section 2. Architectural goals and philosophy updated.<br>Tweaked some sentences slightly to increase understanding. | Jake Durnford |
| 2.1 | 16/07/20 | Section 5. Fixed a spacing issue.<br>Talked about the decision to use a queueing system for receiving orders.<br>Section 2. Mentioned the reliability of the queueing system for customer wait times and also for the employees. | Jake Durnford |
| 3.0 | 17/07/20 | Changed Full Use Case Model to the updated version. | Jake Durnford |

## 1. Purpose

This document describes the philosophy, decisions, constraints, justifications, significant elements, and any other overarching aspects of the Beverage Booker system that shape the design and implementation. The main concerns of the project are usability, availability and security. This is reflected in the architecture being chosen, the Three-Layered Architecture. Separate layers of the project allow for a UI to be installed on an Android phone to allow users to request orders from the system. The logic layer will check for correct and valid requests and allow data from the database layer to be used or updated to complete the request. The UI layer will then display a result for the user. This functionality allows data to be secure, an increased usability and availability through the UI and reliability in the data sent through the logic layer. Furthermore, having 2 separate apps for the customers and the

staff increases the security by having certain functions unavailable to use by customers.

## 2. Architectural goals and philosophy

Beverage Booker goals and philosophies stem from our high priority non-functional requirements (NFRs) which are:

- **Usability**: It is a major requirement that the Beverage Booker application is easy to use, easy to understand and allows users to place an order effectively. Actions the user can take should be obvious, easy to understand and quick to use. This means that the user interface (UI) should have clearly labelled buttons, a simple font of appropriate size and easy to change actions made like being able to edit an order after selecting an item.

- **Availability**: The system must be available for it to be used effectively, meaning it needs to be available during the cafe's business hours while also being available for any required maintenance while the cafe is closed. There should be no delays or poor response times that make users have a less than optimal experience.

- **Security**: The system has to be able to keep user information safe and protected from all other users where only a user can know their own information. It is important this is inline with relevant privacy policies and laws. Certain users should be allowed access to different options such as a manager having the ability to customise the menu but prevent unauthorised users from being able to do the same thing. Using a separate app for staff activities allows an extra layer of security that disallows customers from performing the same actions that staff and admins can like: check off completed items from active orders or complete the order even if it isn't ready.

- **Reliability**: The reliability of the system directly impacts a customers opinion of the cafe and if the application is not reliable, it will not keep or attract any new customers. This could also deter frequent customers or face-to-face customers from trusting the cafe as a whole if they have a particularly bad experience on the application. This also goes for the employees. A reliable system for receiving orders like a queue will help the employees reliably fill orders and keeps waiting times of customers reasonable.

- **Audit**: The system needs to keep track of orders due to it dealing with transactions and to keep a record of purchases for each customer. The system will also keep track of the status of the order to keep the user up to date on how long they will need to wait for their order.

- **Integrity**: The system requires valid data so that it will be able to rely on accurate and up to date information. The database and its tables should keep sensitive data such as user data and payment information safe. If accurate and reliable data is not maintained then issues such as duplicate accounts could occur. This in turn could lead to issues with the ordering system or serious transaction issues occuring.

- **Compatibility**: The application is being developed solely for Android, so it is being implemented to run smoothly on the latest version of the Android OS. We are also trying to offer legacy support for people with older phones that are running an older version of Android. The minimum SDK in the development environment is set to Android Oreo, which allows for approximately 98% compatibility with most users phones in the market.
  Only using a compatible phone also ensures a smoother user experience, and prevents other issues occurring such as failed transactions, failed order placements, or images not loading.

## 3. Assumptions and dependencies

**Dependencies**:

- The flexibility of the project team when programming a solution. How well the team can coordinate and effectively solve issues during programming.
- The responses to the questionnaires for barista, cafe manager, customer, kitchen staff and register operator (and if we get answers).
  The ability of the team to roleplay for roles that cannot be filled with an expert in the role like cafe manager.
- The software to create the application is available and hardware (or a simulation of hardware) is available to test it.

**Assumptions**:

- That users understand English as the application will be displayed in English.
- The user has a connection to the server through the Internet.
- The team can code in Java.
- The team has Android devices to test the application out.

## 4. Architecturally significant requirements

- The database has to be able to be edited by a manager to change menu items or state that they are out of stock.
- The system needs to store user information and transaction information while making sure it is all correct and valid data.
  Sensitive data must be stored in a database that is not local to users to keep it safe and protected from users with ill intent.
- The usability of the software will need to be tested by people with minimal / without assistance to check if the application satisfies usability goals.
- General users (customers) should not have the same options as staff or admins.

## 5. Decisions, constraints, and justifications

- A decision made for the architecture is that it has to support Java as a programming language as everyone in the group knows Java and it will boost the programming efficiency a lot if we know the language. The language is also an extremely popular language that allows for issues to be solved from simple google searches and quick troubleshooting.

- A constraint to use Android devices. This constraint was put in place based on a limitation to what devices we own, them being Android devices. This limitation is more of a forced constraint to use Android studio to create our application instead of programming for multiple devices with different software.

- The system is constrained to using a tiered architecture to allow for protecting sensitive data behind an extra layer so that the sensitive data isn't accessible directly by unauthorised personnel.

- The UI must be easily understood, clear and simple to interpret. This is a major requirement for usability but is also important to be accessible to larger audiences. It is critical to have good usability as it creates good experiences for users and can attract new audiences if it is simple and easy to understand.

- A decision to increase usability by increasing font sizes and button size to allow for users to understand the UI easier and help users who have trouble seeing understand the options easier.

- A decision to increase security by having 2 separate apps, a customer app that allows for the ordering of cafe items while the staff app is used to view the orders and complete them for the customer.
- A decision to use a queue system for receiving orders on the employee side to help streamline managing the orders and being able to keep wait times consistent for customers.

## 6. Architectural Mechanisms

### Ordering Mechanism

The ordering mechanism is for making an order to the cafe and the transaction process when ordering. Some attributes will be:

- Cart, table
- payment information, (sent to payment system)
- the menu items, objects (listed in the cart)
- price, double (2 decimal places)

The mechanism would start by the user selecting items that are confirmed by the user and added to the cart. They can then check their cart for a total price and option to checkout (purchase). This information is then sent to the payment system for a secure transaction.

### Admin Mechanism

The admin mechanism is used by managers / admins who have authorisation to change menus, prices and other details about the menu.
Some attributes will be:

- menu items, objects
- prices, double (2 decimal places)
- menus, lists
- login details (authorised username and password), Strings

The admin can log in and have unique access to modifying the menus by adding and deleting submenus and the items they contain. The prices can also be changed and these changes are reflected onto the menus all users can order from.

### Account Mechanism

The account mechanism is used when that application starts so that the user can log into their account and check and edit their details. Attributes for this mechanism are:

- username, String
- password, String
- order history, list of orders
- saved payment information, record

The user will be able to check their account details once they sign in. This is access granted only to them and other users have the same access to their own details but not other users. An account's details can be edited and saved again.

## 7. Key abstractions

- Layer: A separated bunch of code that communicates with other code bunches (layers) to complete a request.
- Request: An action a user makes that is sent through layers of code for processing
- Cart: A list of items that the user selects to purchase from a menu for items.
- Order: The request sent from a user to purchase menu items which they added to their cart.

## 8. Layers or architectural framework

The Three-Layer architecture is an architecture pattern where there are 3 layers of code being a top layer, middle layer and bottom layer. This is done in our app/s by having the client be made up of the UI and the logic layers. This allows the user to communicate to the data layer without direct interaction since the top layer has to talk to the middle layer which has the capability to talk to the bottom layer. Authentication is used to allow or disallow access of clients to certain actions that use the data layer.

Top layer: This layer is the User Interface layer, it is where users can input data and is also the layer where they can receive results or responses for their actions. A lot of usability features occur on this layer.

Middle layer: This layer is the Business or Logic layer, a layer for where all business rules and logic occurs. This will check for valid and correct data inputs to prevent incorrect or invalid data from users which can decrease the reliability, integrity and security of the system. Authentication is taken into consideration when actions are made to prevent users who cannot make actions from making them.

Bottom layer: This layer is the Data layer, this layer contains access to a database for storing sensitive data like transaction data and user data. This is an important layer for integrity and reliability since if the middle layer works correctly it provides correct and valid data that allows users to trust the application and therefore the whole cafe.

The customer and staff use 2 different apps to operate on. The customer app is for ordering the coffee while the staff app is for viewing the order and fulfilling it. This prevents security issues by not allowing the same functions to a customer that the staff of the cafe would have.

## 9. Architectural views

- **Logical Views:**

**Component Diagram:**
Component Diagram

**Class diagrams:**

Customer App

Staff App

- **Operational Views:**

**Creating an Account**:

**1.** The user opens the application through the application button.
**2.** The login screen appears. The user presses the text below the login input fields prompting them to press here if they would like to create an account.
**3.** The user enters their required information into the input fields, namely: email, password, first name, last name and phone number.
**4.** The user hits the register button.
**6a.** The input data is checked by the application and is found to be valid.
**6b.** The input data is invalid. The field with invalid data is focused on the UI and a prompt toast message will appear instructing the user how to correct the invalid input. User repeats steps **3.** and **4.**
**7.** The register request is sent to the server.
**8.** The request is checked for empty parameters and that the email address isn't already registered.
**9a.** The request is accepted and the user account is registered to the database. The user can now use the details they provided to log in.
**9b.** The request is declined because there is a technical problem and some of the parameters are empty. The user starts again from step **4.**
**9c.** The request is declined because the email address is already registered to a user in the database. The user may start again from step **3.** with a different email address.
**10.** A response is sent back to the user to tell them the request was accepted or declined based on the results in step **9**.
**11.** The Login screen is displayed.

**Logging into an Account:**

**1.** The user opens the application through the application button.
**2.** The login screen is displayed.
**3.** The user enters a username and password.
**4.** The user presses the login button.
**5a.** The input data is checked by the application and is found to be valid.

**5b.** The input data is invalid. The field with invalid data is focused on the UI and a prompt toast message will appear instructing the user how to correct the invalid input. User repeats steps **3.** and **4.**
**6.** The login request is sent to the server.
**7.** The request is checked for empty parameters, that the user exists, and that the email and password match.
**8a.** The request is accepted and the user account is logged in.
**8b.** The request is declined because there is a technical problem and some of the parameters are empty. The user starts again from step **3.**
**8c.** The request is declined because the email address is not registered to a user in the database. The user may start again from step **3.**
**8d.** The request is declined because the email address and password do not match those registered to a user in the database. The user may start again from step **3.**
**9.** A response is sent back to the user to tell them the request was accepted or declined based on the results in step **8**.
**10.** The Profile Activity screen is displayed with a welcome message for the user.

**Placing an order:**

**1.** Assumptions: the user has created an account and logged in (‘Creating an Account and Logging into an Account). The user is on the Profile Activity screen and presses the Primary Menu button.
**2.** The Primary Menu UI is displayed.
**3.** The user presses the ‘Main Menu’ button.
**4.** The Main Menu (cafe food/drinks) UI is displayed.
**5.** The user browses the menu.
**6a.** A menu item is added to the cart by hitting the ‘Add to Cart’ button.
**6b.** If the menu item is already in the users cart a toast message is displayed conveying this to the user.
**7.** Repeat steps **5.** to **6.** until all items required are added to the cart.
**8.** The user presses the ‘Cart’ button.
**9a.** The Cart Activity UI is displayed.
**9b.** If the cart is empty a toast message is displayed conveying this to the user. The user can return to step **5.**
**10.** The user checks their cart contents.
**11.** The user presses the ‘Place Order’ button.
**12a.** The Payment Activity UI is displayed.
**12b.** If the cart is empty a toast message is displayed telling the user to add items to their cart before pressing the ‘Place Order’ button. The user returns to step **5.**
**13.** The user enters their credit card details into the provided fields.
**14a.** The ‘Continue’ button is active because data has been entered into all the fields.
**14b.** The ‘Continue’ button is inactive because data has not been entered into all the fields.The user returns to step **13.**
**15.** The user presses the ‘Continue’ button.
**16a.** The credit card input data is checked by the application and is found to be valid.
**16b.** The credit card input data is invalid. The field with invalid data is focused on the UI and a prompt toast message will appear instructing the user how to correct the invalid input. User repeats steps **13.** and **14a.**
**17.** The place order request is sent to the server.
**18.** The request is checked for empty parameters, that the payment is valid, and lastly that the order execute SQL statement is successful.
**19a.** The request is accepted and the order is placed in the database.
**19b.** The request is declined because there is a technical problem and some of the parameters are empty. The user may start again from step **13.**
**19c.** The request is declined because the payment is invalid. The user may start again from step **13.**

**19d.** The request is declined because there is a technical issue placing the order in the database. The user may start again from step **13.**
**20.** A response is sent back to the user to tell them the request was accepted or declined based on the results in step **19**.
**21.** The Order Confirmation Activity screen is displayed with a message for the user thanking them for their order and giving them an estimated time for pickup.

**Editing account details:**

**1.** Assuming the user has logged into an account and is on the main menu (follow signing into an account), the user can use the "account" button to be taken to a new page with details on the user.
**2.** The user can view their details here but can also hit the "edit" button to change certain fields such as "email", "password" or "payment details" (if any are saved).
**3.** Once the user has made any changes required, the "save changes" button will send an update request through the system.
**4a.** The data is checked and is valid, the data is updated in the database and a "saved changes" response is sent to the user.
**4b.** The data is checked and is invalid, a response is sent to the user informing them of the data being incorrect or not valid for the field, the data is not updated.

**Viewing an Active Order (cafe staff):**
**1.** Under the assumption that the cafe staff has a valid account and enters their Staff ID into the provided space on the Login page of the staff app.
**2.** The employee hits the 'Click to View Orders' button.
**3a.** The Active Orders UI appears and displays any active orders.
**3b.** The Active Orders UI appears but has no active orders and therefore the employee cannot continue and must wait for an order to be placed.
**4. (After 3a.)** The employee hits the 'Start Order' button on an active order to view it.
**5.** The Current Order UI will appear with the order that was clicked and shows the items for the employee to make to fulfil the request.

- **Use case: Link**