Beverage Bookers
Architecture Notebook

1. Purpose

This document describes the philosophy, decisions, constraints, justifications, significant elements, and any other overarching aspects of the Beverage Bookers system that shape the design and implementation.

[Always address Sections 2 through 6 of this template. Other sections are recommended, depending on the amount of novel architecture, the amount of expected maintenance, the skills of the development team, and the importance of other architectural concerns.]

2. Architectural goals and philosophy

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

- **Usability**: It is a major requirement that the Beverage Bookers 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 open 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 and safe and protected from all other users where only a user can know their own information. It is important this falls in lines of 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.
- **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.
- **Audit**: The system needs to keep track of orders due to it dealing with transactions and to keep a record of purchases from customers. 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 correct databases that keep sensitive data such as user data and payment information safe. If inaccurate and unreliable data is not maintained then issues such as duplicate accounts can lead to users from having inaccurate orders or serious transaction issues occuring.
- **Compatibility**: The application should be able to run smoothly on the user device of choice so it needs to be used only on compatible devices to prevent failed transactions, failed order placements and images not loading which could lead to a negative user experience. The compatible devices should be able to use more modern versions of Android as that is the targeted platform but should also allow for a large potential audience.

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 application 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.

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.

6. Architectural Mechanisms

This

[List the architectural mechanisms and describe the current state of each one. Initially, each mechanism may be only name and a brief description. They will evolve until the mechanism is a collaboration or pattern that can be directly applied to some aspect of the design.]

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, floating point

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, floating point
- 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.

Table Booking Mechanism

The table booking mechanism is for booking available tables in the cafe.

Event Booking Mechanism

The event booking mechanism is used when booking an event at the cafe.

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.

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.

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.
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

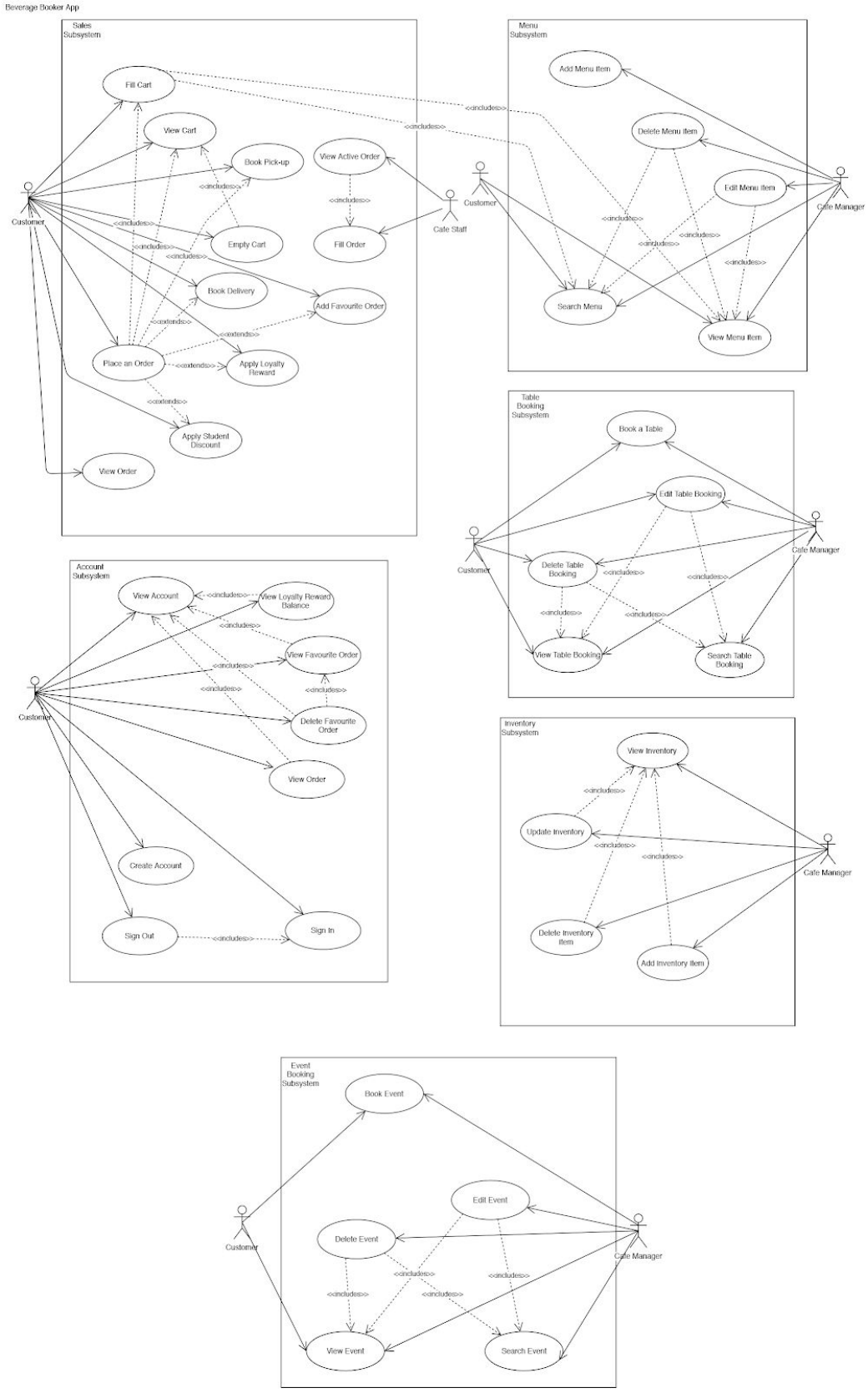
reliability, integrity and security of the system.
Bottom layer: This layer is the Data layer, this layer contains access to a database for storing sensitive data like transaction data and user data. This is an important layer for integrity and reliability since if the middle layer works correctly it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

9. Architectural views

[Describe the architectural views that you will use to describe the software architecture. This illustrates the different perspectives that you will make available to review and to document architectural decisions.]

Recommended views

- **Logical:** Describes the structure and behavior of architecturally significant portions of the system. This might include the package structure, critical interfaces, important classes and subsystems, and the relationships between these elements. It also includes physical and logical views of persistent data, if persistence will be built into the system. This is a documented subset of the design.
- **Operational:** Describes the physical nodes of the system and the processes, threads, and components that run on those physical nodes. This view isn't necessary if the system runs in a single process and thread.
- **Use case:**


Beverage Booker App
Sales Subsystem
Fill Cart
View Cart
Book Pick-up
View Active Order
Empty Cart
Fill Order
Book Delivery
Add Favourite Order
Place an Order
Apply Loyalty Reward
Apply Student Discount
View Order
Customer
Cafe Staff
Menu Subsystem
Add Menu Item
Delete Menu Item
Edit Menu Item
Search Menu
View Menu Item
Cafe Manager
Table Booking Subsystem
Book a Table
Edit Table Booking
Delete Table Booking
View Table Booking
Search Table Booking
Account Subsystem
View Account
View Loyalty Reward Balance
View Favourite Order
Delete Favourite Order
View Order
Create Account
Sign Out
Sign In
Inventory Subsystem
View Inventory
Update Inventory
Delete Inventory Item
Add Inventory Item
Event Booking Subsystem
Book Event
Edit Event
Delete Event
View Event
Search Event
<<includes>>
<<extends>>