Faculty of Engineering and Applied Science

SOFE 3650U Software Design and Architectures

Bookstore Design: Final Project Report

Group Members

Saaransh Sharma	100701820
Michal Frackowiak	100401611
Jean-Paul Saliba	100741759
Faisal Alsheet	100639174

Date: 12/6/2021

Group Number: 5

Table of Contents

1.1 System Requirements

1.1.1 Use Case Model

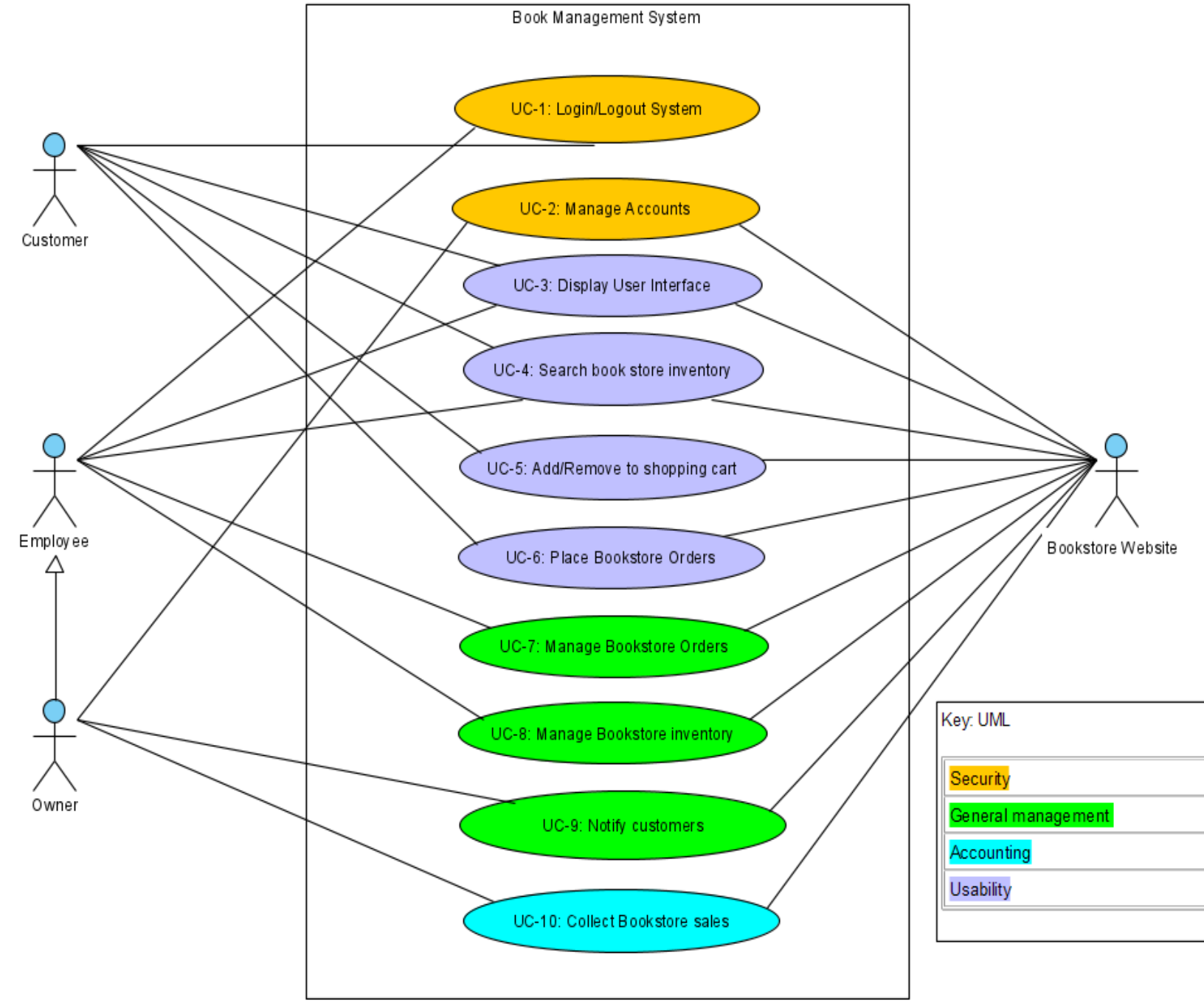

Figure 1: Use case model for Online Bookstore System

1.1.2 Use Cases

Use Case	Description
UC-1: Login/Logout System	A user provides username and password credentials to log in to the bookstore. If the login attempt fails, or they do not have an existing account, they will be prompted to create an account. A successful login will provide them with the relevant user display, depending on if they are a customer, employee or owner.
UC-2: Manage Accounts	The owner of the bookstore website has the ability to manage customer and employee accounts and permissions.
UC-3: Display User Interface	The website will display a tailored user interface for each stakeholder. (customers, employees, owner).
UC-4: Search bookstore inventory	Stakeholders can utilize the search function to look up available books in the bookstore's database system.
UC-5: Add/remove to shopping cart	Customers can add or remove books to an online shopping cart that will keep track of their orders.
UC-6: Place bookstore orders	Customers can process their bookstore order by checking out their online shopping cart.
UC-7: Manage bookstore orders	The owner or employees can use the bookstore system to access the customer orders. They are able to update, cancel or fulfill any orders.
UC-8: Manage bookstore inventory	Owners and employees have the ability to update, add or remove books from the bookstore's inventory
UC-9: Notify customers	The owner of the bookstore website can send notifications to customers regarding special offers and discounts on products.
UC-10: Collect bookstore sales	The owner of the bookstore collects and stores the total sales made from the bookstore.

1.1.3 Quality Attribute Scenarios

ID	Quality Attribute	Scenario	Associated Use Cases
QA-1	Security	When an unknown user attempts to login more than five times, the system responds by locking the account and emailing the account owner. This prevents 100% of the user's data from being compromised.	UC-1, UC-2
QA-2	Availability	A user makes a request to interact with the bookstore system, such as inventory, shopping, or searching under normal conditions. The system responds to 100% of the requests 24/7, unless there is a scheduled outage.	ALL
QA-3	Usability	A user should be able to navigate efficiently through their specific interface to the website. The interface requires at most 5 clicks to return the desired information.	ALL
QA-4	Performance	When a user interacts with the bookstore database system, it responds by adding, removing or modifying that element. This request should be processed within 500ms.	UC-2 to UC-10
QA-5	Performance, Usability	An end-user adds or removes a book into the shopping cart system. The system responds by providing the user with an updated balance, given that the book is available. This response should be within 500ms.	UC-5, UC-6, UC-7

1.1.4 Constraints

ID	Constraint
CON-1	All transactions must be recorded for bookstore financials.
CON-2	To make a purchase or interact with a database the user must be registered in the system.
CON-3	The website must be accessible on different web browsers (Firefox, Chrome, Edge).
CON-4	An existing relational database server must be used.This server cannot be used for other purposes than hosting the database.
CON-5	The system's network connection is generally reliable on all servers.

1.1.5 Architectural Concerns

ID	Concern
CRN-1	Establish the website’s overall structure, which interacts with a database system.
CRN-2	Utilize the team's knowledge of web development and database implementation using technologies such as SQL, JavaScript, Python, CSS, PHP and HTML.
CRN-3	Distribute tasks amongst team members accordingly and ensure deadlines are met.
CRN-4	Software is reliable and conforms to all business rules and requirements.
CRN-5	Database performs all operations and queries as intended providing proper data.

1.2 ADD Iteration 1 - Generate Software System Structure

1.2.1 Review Inputs

<table>
<tr><th>Category</th><th>Details</th></tr>
<tr><td>Design Objectives</td><td>This is a greenfield system for a mature domain. The purpose is to develop a detailed architectural design used for the development of the bookstore system.</td></tr>
<tr><td>Primary Functional Requirements</td><td>From the use cases in section 1.1.2, the primary functionalities of this system have been determined:
UC-1: Provides core functionality
UC-3: Provides core functionality
UC-4: Provides core functionality
UC-5: Provides core functionality
UC-6: Provides core functionality</td></tr>
<tr><td>Quality Attribute Scenarios</td><td><table><tr><th>Scenario ID</th><th>Customer Importance</th><th>Difficulty to make</th></tr><tr><td>QA-1</td><td>High</td><td>Low</td></tr><tr><td>QA-2</td><td>High</td><td>Medium</td></tr><tr><td>QA-3</td><td>High</td><td>Medium</td></tr><tr><td>QA-4</td><td>Medium</td><td>High</td></tr><tr><td>QA-5</td><td>High</td><td>High</td></tr></table>From this list, QA-1, QA-2, QA-3 and QA-4 have been selected as QA drivers.</td></tr>
<tr><td>Constraints</td><td>All the constraints have been selected as drivers</td></tr>
<tr><td>Concerns</td><td>All the concerns have been selected as drivers</td></tr>
</table>

1.2.2 Establish Iteration Goal by Selecting Drivers

The goal of this iteration is to establish an overall system structure to satisfy CRN-1. Addressing this concern is required in order to proceed with the second iteration. Although this iteration is concerned with the websites general architectural concern, the team will be mindful of the following drivers:

- QA-1: Security
- QA-2: Availability
- QA-3: Usability
- QA-4: Performance
- CON-4: The website must be accessible on different web browsers (Firefox, Chrome, Edge)
- CRN-2: Utilize the team's knowledge of web development and database implementation using technologies such as SQL, JS, Flask, Python, CSS and HTML.

1.2.3 Choose One or More Elements of the System to Refine

Given that this is a greenfield development, the elements that will be refined include the entire online bookstore system. Refinement will be performed through decomposition.

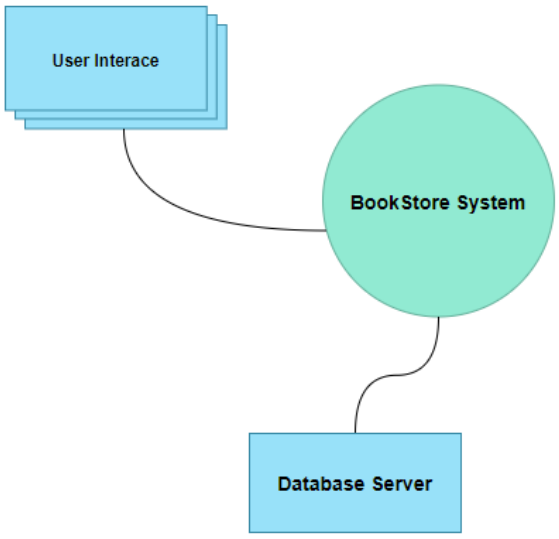

Figure 2: Context Diagram for Bookstore system

1.2.4 Choose One or More Design Concepts That Satisfy the Selected Drivers

<table>
<tr><th>Design Decisions And Location</th><th>Rationale</th></tr>
<tr><td>The system will be logically structured on the Web Application architecture.</td><td>Web applications are typically initiated from a web browser that communicates with a server using the HTTP protocol. The architecture is composed of three layers: presentation, business and data layers. Since we are developing an online bookstore, this architecture is suitable as all components are separated into layers. It also supports the cross-cutting functionality, which includes aspects of logging, security and management. This architecture completely addresses CON-4 as this structure is used in many web browsers. The cross-cutting partially addresses QA-1 by supporting logging for security.

No other alternatives will be considered or discarded, as the architects were familiar with the reference architecture and considered it to be fully adequate to meet the requirements.</td></tr>
<tr><td>The deployment will be physically structured on the three-tier deployment pattern as well as leveraging a load-balancer.</td><td>As per CON-4, the system must be accessible through a web browser. The system will also require a relational database that is hosted from a server (CON-5), making the three-tier deployment an ideal pattern. As our bookstore service expands, a load balancer will be introduced to reinforce QA-4.

<table>
<tr><th colspan="2">Discarded Alternatives</th></tr>
<tr><th>Alternative</th><th>Reason for Discarding</th></tr>
<tr><td>Two-Tier Deployment</td><td>Referencing CON-5, an existing database server must be implemented in the system, and can only be used for the purpose of hosting the database. Such deployment would violate this constraint.</td></tr>
<tr><td>N-Tier Deployment (N>3)</td><td>No other servers will be necessary for this implementation, therefore these deployments can be discarded.</td></tr>
</table>
</td></tr>
</table>

1.2.5 Instantiate Architectural Elements, Allocate Responsibilities and Define Interfaces

Design Decisions and Location	Rationale
Create a module dedicated to accessing the bookstore user databases in the data layer of the web application reference architecture.	The service agents component from the web application architecture is modified to depict how the server collects and authenticates information from the database to process requests.

1.2.6 Sketch Views and Record Design Decisions

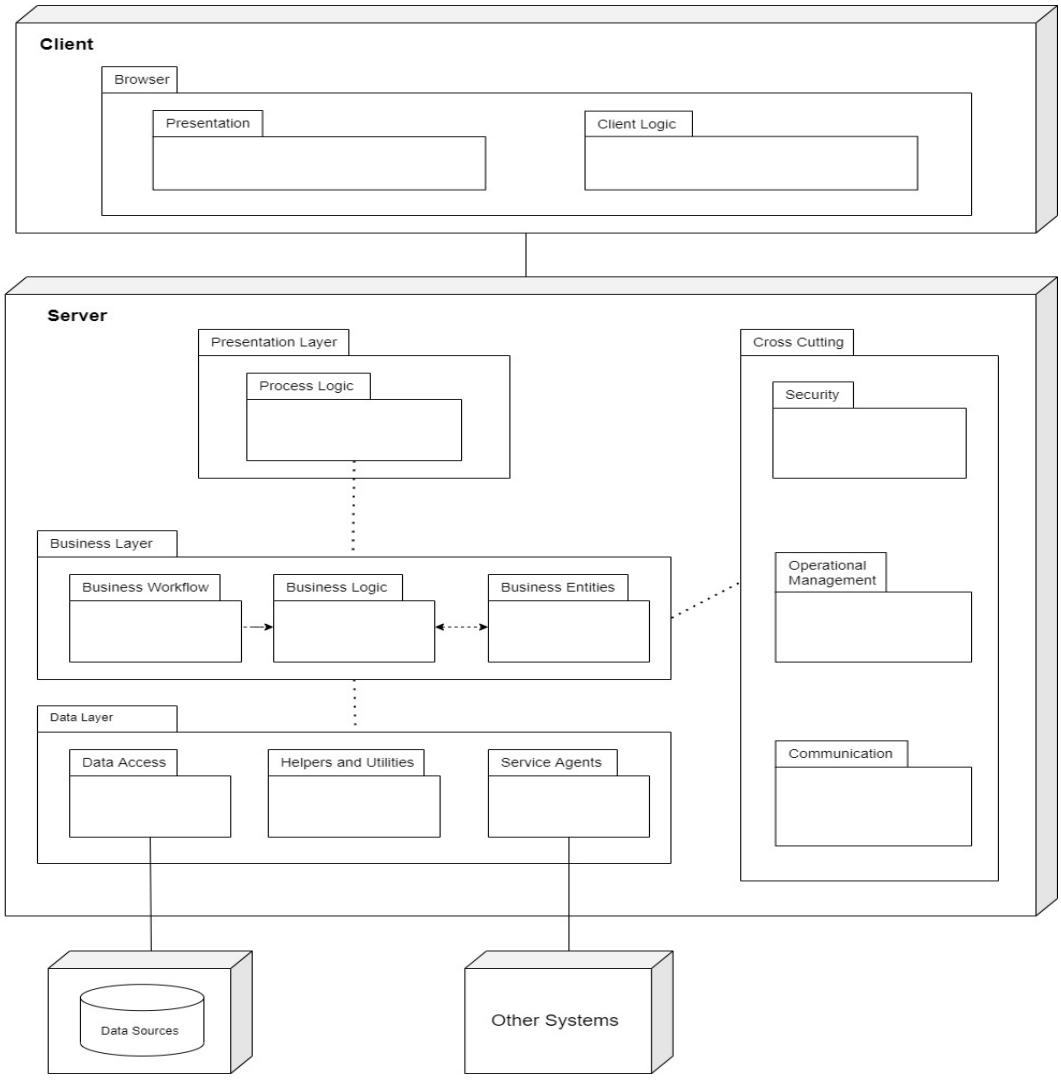

Figure 3: Module View of the selected Reference architectures (Key: UML)

Element	Responsibility
Browser Layer	Contains a web browser that is operating on the client machine and leverages the client machine's user interface.
Presentation Layer	Holds the modules that are needed to handle input from the user.
Business Layer	Holds the logical components that are important to the functional requirements of the system.
Data Layer	Holds all the components needed to connect to the database and interact with its data.
Cross-Cutting Layer	Holds all the components that must be addressed on the overall system, that impact several layers and the functionality of the system.
Presentation	A module that handles user interaction between components.
Process Logic	This component takes care of the control flow of information retrieved from the client-side and ensures this information is distributed for handling.
Client Logic	This component is responsible for handling all the use cases where access or connection to the server is not necessary.
Business workflow	Manages several repeatable business processes.
Business entities	This component depicts entities from the business domain and their related business logic.
Business Logic	This module grabs and processes application data, this data is validated through the business rules.
Data Access	Component captures consistent mechanisms and provides regular operations utilized to store and retrieve information.
Helpers and Utilities	The module deals with the connection to the database and the querying of data.
Service Agents	The component provides a method of communication to handle changes in data for external systems.

Security	The module ensures that users are authorized and that the information provided is authenticated.
Operational Management	The component handles logging and exception management.
Communication	Handles communication across layers and other components in the system.

Figure 4: Initial Deployment Diagram for the Online Bookstore System (Key: UML)

Element	Responsibility
Client: Web Browser	The user's personal computer, that handles the client-side logic of the web application and aids in connecting to the server.
Web/App Tier	Utilizing Flask to process requests from the clients to the database server.
Database Server	The server dispatches requests made from the web server and hosts the connection to the bookstore database.

Relationship	Description
Between presentation/business layer and data layer	Communication with the database will be done using the HTTP protocol.

1.2.7 Perform Analysis of Current Design and Review Iteration Goal and Achievement of Design Purpose

Not Addressed	Partially Addressed	Completely Addressed	Design Decision made during the iteration
	UC-1		Picked a reference architecture that establishes components that support this functionality.
	UC-3		Picked a reference architecture that establishes components that support this functionality. This functionality is depicted on the client-side.
	UC-4		Picked a reference architecture that establishes components that support this functionality.
	UC-5		Picked a reference architecture that establishes components that support this functionality.
	UC-6		Picked a reference architecture that establishes components that support this functionality.
	QA-1		Although the initial steps have been constructed with the selection of architecture, this quality attribute is not addressed in a detailed manner during this iteration.
	QA-2		Although the initial steps have been constructed with the selection of architecture, this quality attribute is not addressed in a detailed manner during this iteration.
	QA-3		Selected a reference architecture that contains modules that accomplish this functionality.
	QA-4		Although the initial steps have been constructed with the selection of architecture, this quality attribute is not addressed in a detailed manner during this iteration.
CON-1			No relevant design decisions were made during this iteration.
CON-2			No relevant design decisions were made during this iteration.
		CON-3	Selected a reference architecture that contains modules that accomplish this functionality.
		CON-4	Selected a reference architecture that contains modules that accomplish this functionality.
		CON-5	Selected a reference architecture that contains modules that accomplish this functionality.

		CRN-1	Through the selection and design of the reference architectures, the website's overall structure has been depicted.
	CRN-2		The concern of using the team's knowledge of web development and database implementation has been partially addressed through the selected architectures.
CRN-3			No relevant design decisions were made during this iteration.
CRN-4			No relevant design decisions were made during this iteration.
		CRN-5	Selected a reference architecture that contains modules that accomplish this functionality through the communication between the database modules and the server.

2.1 ADD Iteration 2 - Identify Structures to Support Primary Functionality

---

This section presents the results of the activities that are performed in the second iteration of the design process for the Bookstore website. This iteration will focus on identifying structures to support the primary functionality of our architectural concerns.

2.1.1 Establish Iteration Goal by Selecting Drivers

The main goal of this section is to confront the primary use cases, UC-1, UC-3, UC-4, UC-5, UC-6. All of these primary use cases and drivers impact the underlying structure of the system. Through the second iteration establishing the website's overall structure that interacts with a database system as well as CRN-3 distributing tasks among team members.

The primary use cases are listed below as follows:

- UC-1: Login/Logout System
- UC-3: User Interface
- UC-4: Search Bookstore inventory
- UC-5 Add/remove to shopping cart
- UC-6: Place Bookstore Orders

2.1.2 Choose one or more Elements of the System to Refine

The elements that will be refined in this iteration are the primary use cases that were outlined above located in the presentation and business layer of the web application architecture.

2.1.3 Choose One or More Design Concepts That Satisfy the Selected Drivers

During this step, several design patterns, specifically architectural design patterns, were selected from the book, Pattern-Oriented Software Architecture, Volume 4.

Design Decisions and Location	Rationale
Construct a Domain Model for the system	Prior to breaking down the functional components, it is vital to build an initial domain model for the system that demonstrates the important entities in the domain, as well as their relationships.
Identify the Domain Objects that map to the functional requirements	Every unique functional element of the system must be encapsulated. It is important to capture all functional elements, to prevent missing out on any requirements.
Break down the Domain Objects into general and focused components.	Domain objects are used to partition the functionality of the complete system, however, this functionality is supported by the detailed elements that are located within the layers of the architecture.
Use Python Flask web framework	Flask is a microframework that aims to keep the core simple but extensible. Flask is flexible and modifiable with design decisions such as what database or template engine to use. Flask was selected as the development team was already familiar with it, resulting in greater and earlier productivity.

2.1.4 Instantiate Architectural Elements, Allocate Responsibilities and Define Interfaces

Design Decisions and Location	Rationale
Construct only an initial domain model	The entities that are involved in the primary use cases must be identified and structured, however only the initial domain model is built to speed up this process of design. This will contain the presentation layer, business and data layer.
Map the system use cases to domain objects.	The mapping of domain objects can be fulfilled by examining the system's use cases. In order to address CRN-3, domain objects are identified by all the members of the team.
Break down the domain objects and depict them across layers	This decision will assist in refining the objects, ensuring that each module that aids all of the functionalities are examined and identified.
Connect components associated with the modules using Flask	Using this framework, we develop classes that will connect to the components which will be tested on the presentation layer.

2.1.5 Sketch Views and Record Design Decisions

As a result of the decisions made in step five, several diagrams are created that include the initial domain model, domain objects, and module package diagram.

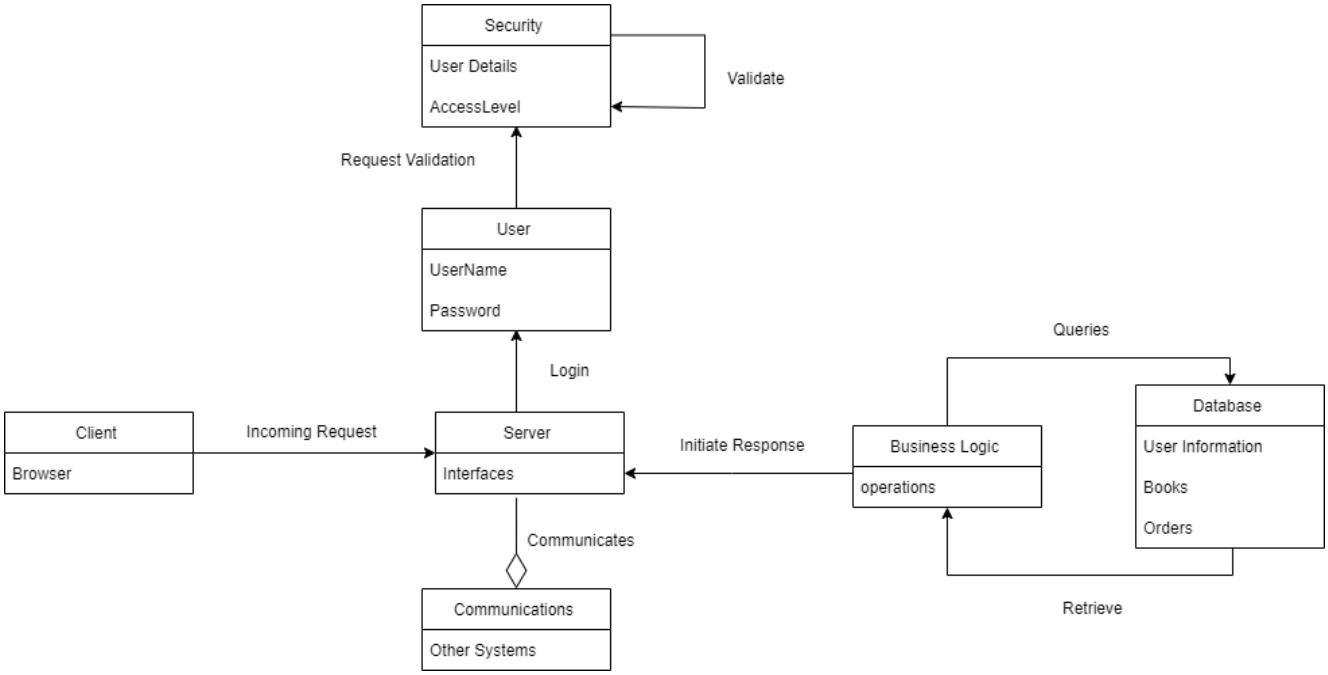

Figure 5: Initial domain model (Key: UML)

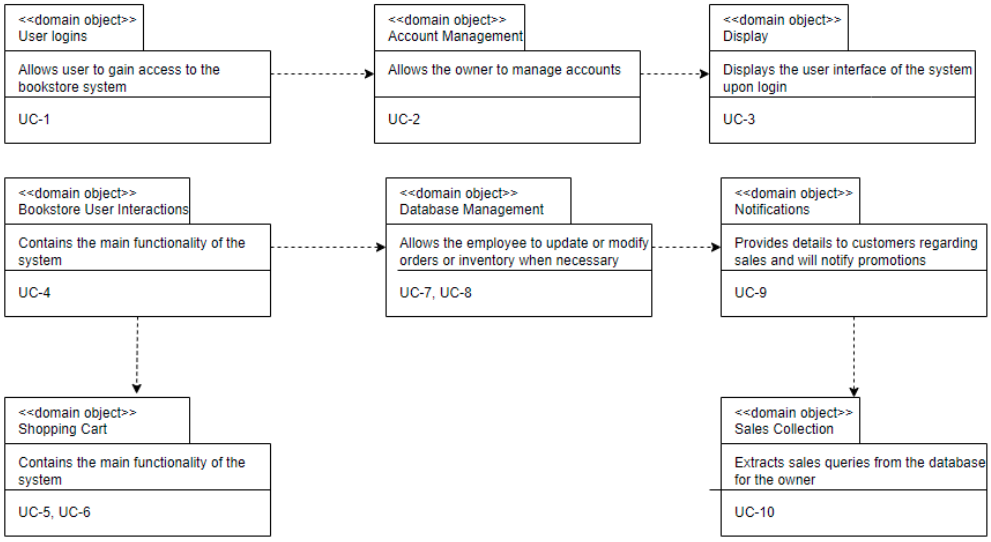

Figure 6: Domain Objects associated with use case (Key: UML)

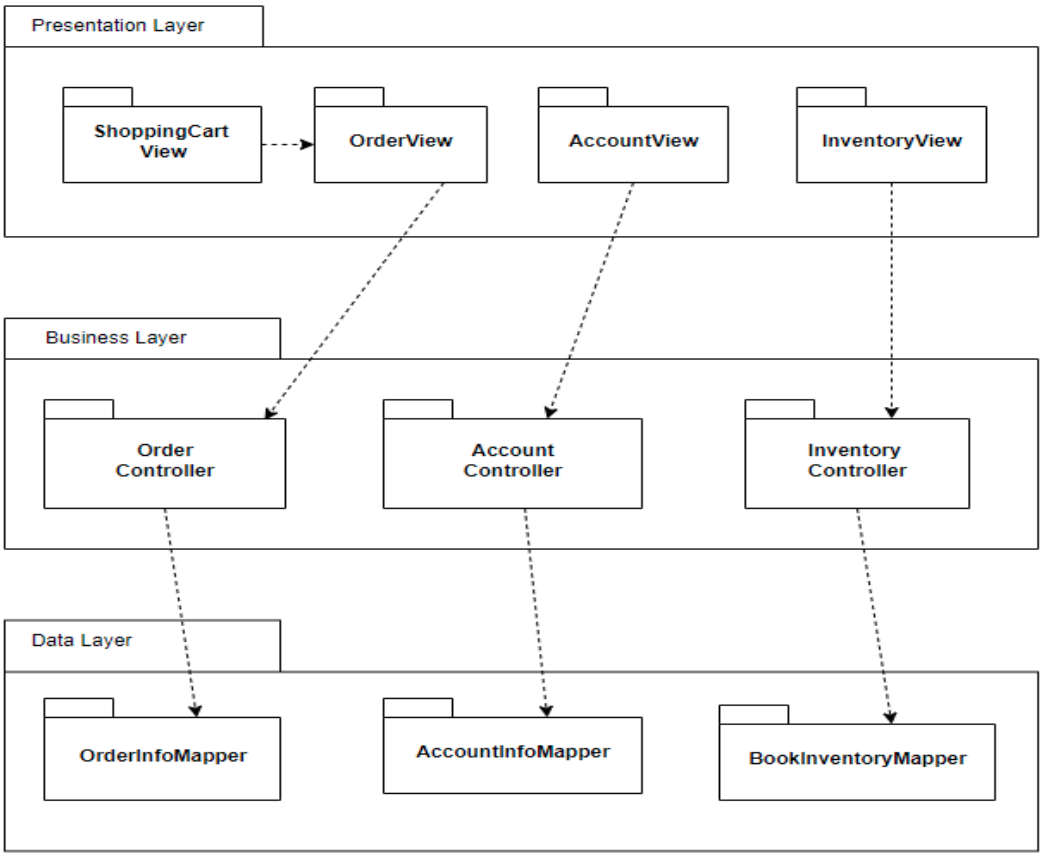

Figure 7: Modules to support the primary use cases (Key: UML)

The responsibilities of the modules identified in Figure 6 are summarized in the table below.

Element	Responsibility
Shopping Cart View	Displays products in your shopping cart, updates price content based on requests sent to the server.
Order View	Displays Orders and updates information based on requests to the server.
Account View	Displays Account information and updates information based on requests sent to the server.
Inventory View	Displays Inventory on the main website and updates views based on requests sent to the server.
Order Controller	Contains business logic related to the order information.
Account Controller	Contains business logic related to the account information of users.

Inventory Controller	Contains business logic related to the inventory control of the system.
OrderInfoMapper	Responsible for persistence operations (CRUD) related to the events.
AccountInfoMapper	Responsible for persistence operations (CRUD) related to the events.
BookInventoryMapper	Responsible for persistence operations (CRUD) related to the events.

UC-4, UC-5, UC-6: Search, Add/Remove, Order Book

Figure 8 shows a sequence diagram for the UC-4 (search book inventory), UC-5 (add/remove book to cart) and UC-6 (place an order). This interaction starts with a customer searching for a book they would like to purchase. They place the book into their cart and it updates the price. Once ready, the customer can form their order. This is sent to the order controller then to the order mapper which records the order in the database.

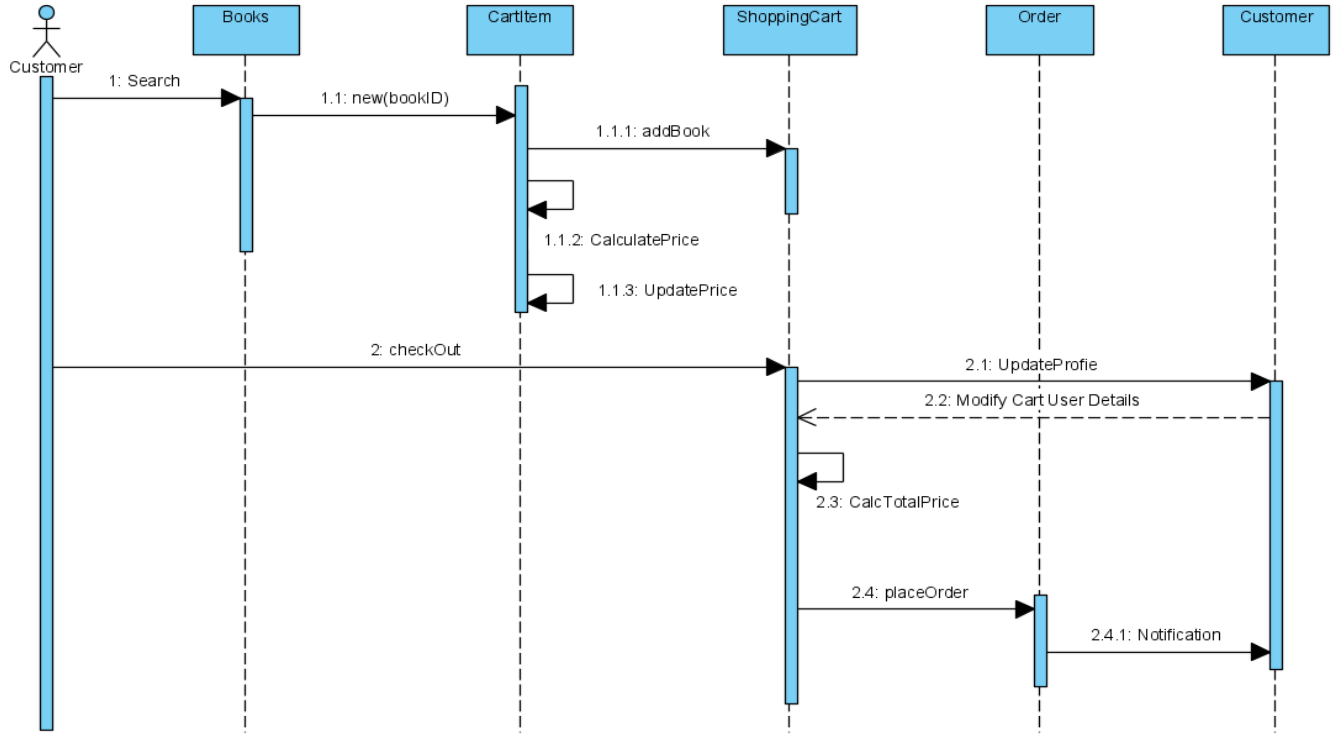

Figure 8: Sequence Diagram illustrating use case UC-4, UC-5 and UC-6 (Key: UML)

UC-1, UC-3: Login and redirect to Corresponding Interface

Figure 9 shows a sequence diagram for the UC-1 (login), UC-3 (user interface). A user will input their credentials to the Account view and the request will be sent to the Account controller. The account controller will send the result to the Account Mapper in the database to validate the user. Once a user is validated they will be directed to their corresponding web page (interface).

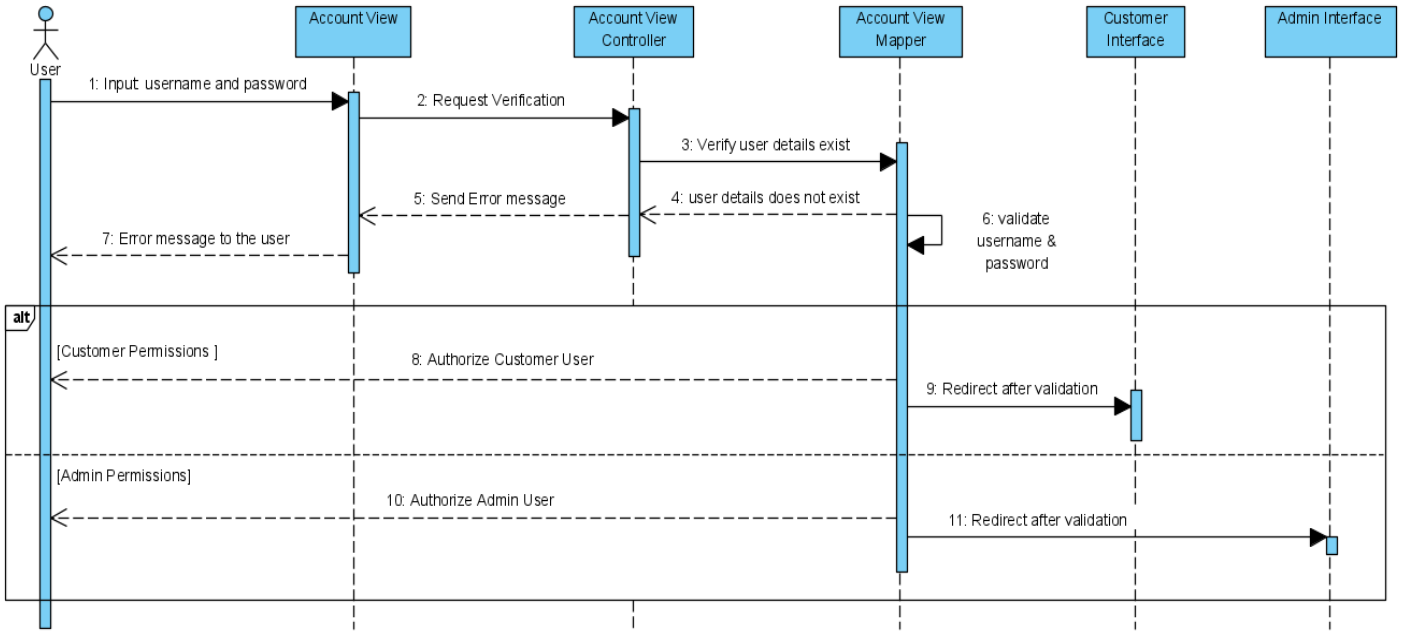

Figure 9: Sequence Diagram illustrating use case UC-1 and UC-3 (Key: UML)

2.1.6 Perform Analysis of Current Design and Review Iteration Goal and Achievement of Design Purpose

Not Addressed	Partially Addressed	Completely Addressed	Design Decision made during the iteration
		UC-1	Modules across the layers and preliminary interfaces to support this use case have been identified.
		UC-3	Modules across the layers and preliminary interfaces to support this use case have been identified.
		UC-4	Modules across the layers and preliminary interfaces to support this use case have been identified.
		UC-5	Modules across the layers and preliminary interfaces to support this use case have been identified.

		UC-6	Modules across the layers and preliminary interfaces to support this use case have been identified.
	QA-1		The elements that support the associated use case (UC-1) have been identified.
	QA-2		The elements that support the associated use case (ALL) have been identified.
	QA-3		The elements that support the associated use case (ALL) have been identified.
	QA-4		The elements that support the associated use case (UC-3, UC-4, UC-5, UC-6) have been identified.
		CON-2	Selected a reference architecture that accomplishes this design need.
		CON-4	The modules for fulfilling this need have been identified in the reference architectures.
		CRN-1	Established the website's overall structure through the reference architectures selected.
		CRN-2	The Flask web framework has been selected to support this concern.
		CRN-3	All domain objects were defined and objectives were distributed amongst team members accordingly.
CRN-4			No relevant design decisions were made.

3.1 ADD Iteration 3 - Assessing Quality Attribute Scenario Driver

---

This section presents the results of the activities that are performed in the third iteration of the ADD design process. Building on the decisions made in iteration 1 and 2 we will now focus our design towards the quality attributes.

3.1.1 Establish Iteration Goal by Selecting Drivers

The main purpose of this iteration is to analyze and reason for the achievement of some of the important quality attributes. During this iteration, we will be focusing on QA-2 and QA-4 quality attributes.

Scenarios:

QA-2: A user makes a request to interact with the bookstore system, such as inventory, shopping, or searching under normal conditions. The system responds to 100% of the requests 24/7 unless there is a scheduled outage.

QA-4: When a user interacts with the bookstore database system, it responds by adding, removing or modifying that element. This request should be processed within 500ms.

3.1.2 Choose One or More Elements of the System to Refine

For this quality attribute scenario, the elements that will be refined are the layers that were identified during the first iteration:

- Client Tier (Presentation layer of web application)
- Web/App Tier (Business layer of web application)

3.1.3 Choose One or More Design Concepts That Satisfy the Selected Drivers

The design concepts used in this iteration are the following:

Design Decisions and Location	Rationale
Introduce **active redundancy tactic** by replicating web app tier of 3-tier physical deployment pattern	By replicating the web/app tier we can ensure the critical elements of the system can withstand high traffic requests from clients without affecting functionality.

3.1.4 Instantiate Architectural Elements, Allocate Responsibilities and Define Interfaces

The instantiation design decisions are summarized in the following table:

Design Decisions and Location	Rationale and Assumptions
Implement active redundancy of the web/app tier	Docker will be used for replicating the web/app tier components, which will distribute server load during high traffic requests.
Implement load balancing and redundancy using Nginx web server	Since two replicas of the web/app tier server are active at any time the Nginx web server will balance the load of client requests at any given time.

3.1.5 Sketch Views and Record Design Decisions

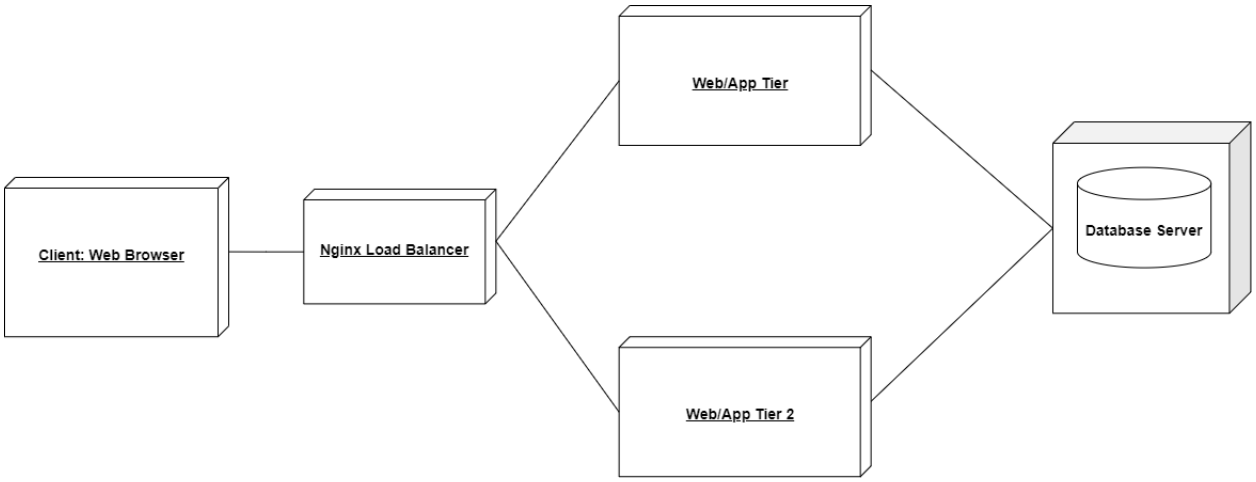

Figure 10: Refined deployment diagram (Key: UML)

Element	Responsibility
Nginx Load Balancer	The Nginx load balancer receives requests coming from clients to the application servers and balances the load among the multiple web/app servers.
Web/App Tier 2	Introduced active redundancy in a second web/app tier. This is in conjunction with the load balancer, which will utilize Flask to process requests from the clients to the database server.

3.1.6 Perform Analysis of Current Design and Review Iteration Goal and Achievement of Design Purpose

Not Addressed	Partially Addressed	Completely Addressed	Design Decision made during the iteration
	QA-1		No relevant design decisions were made.
		QA-2	By implementing active redundancy, we minimize the risks of failure within the system and ensure that client requests are met. Should the load balancer fail, this redundancy replicates our web/app tier.
	QA-3		No relevant design decisions were made.
		QA-4	Integrating a load balancer will reduce the processing load of requests made by the client tier, improving end-user performance by reducing network traffic.
CRN-4			No relevant design decisions were made.