University of Bahrain
Department of information technology

ITSE302 - SOFTWARE DESIGN & ARCHITECTURE

Online Library Management System

Software Design & Architecture

- **Section: 1, Group:**

▪ Husain Ali Ahmed	–	202106117
▪ Ali Abbas Ali	–	202107809
▪ Haitham Abdullah Taher	–	202107520
▪ Abdullah Mohammed	–	202104275

Submitted To : MS AFRAH

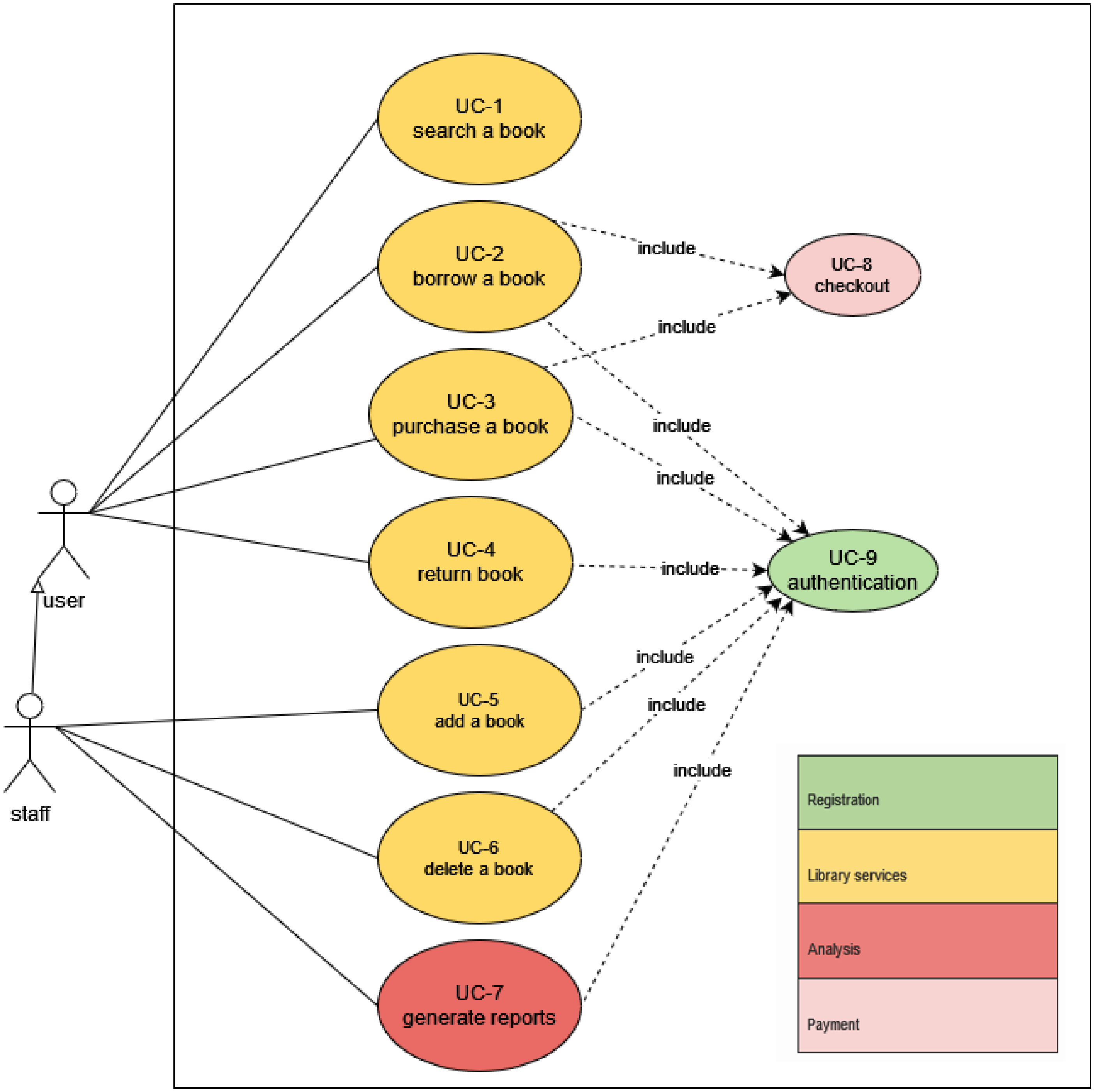
1.1.1 Description:

In the modern time, technology has significantly impacted all the aspects of our lives and changed the way the products are being produced and how the services are being delivered. One of the major industries that got affected by the modern technology is the libraries industry where the libraries management before technology used to be manual but with the help of modern technology most of the libraries use Library Management System (LMS) to provider their services to their customers in a convenient manner.

The purpose of the Library Management System is to automate the library services to enhance user experience. With the help of the Library Management System, the user can serve himself more and that will reduce operational cost for the library. LMS that will be developed will provide functionality for searching, adding, deleting, purchasing, and borrowing books and more. The LMS also provides well-structured procedure to track the activities that happen within the system such book status is it available or not, number of copies, and supplier etc. The user can borrow the books as well as purchase them where the system provides a payment process that can be used to pay borrowing cost or book cost in case of purchasing. The system provides analysis tools that can be used to generate reports and help in decision making.

ID	Use case title	Description
UC-1	**Seach a Book**	**The Search Book should allow the user to search for a book using the book's ISBN number or title.**
UC-2	**Borrow Book**	**This function is used to borrow books from the library by the users.**
UC-3	**Purchase Book**	**This function is used by the user to purchase books from the library.**
UC-4	**Return Book**	**This function is used by the users to return borrowed books.**
UC-5	**Add Book**	**The Librarians should be able to use this function to add a book to the system within 1 window.**
UC-6	**Delete Book**	**This function is used by the librarians to delete books from the system**
UC-7	**Generate Reports**	**This function is used by librarians to generate reports for analysis purposes.**
UC-8	**Checkout**	**This function is used by the users to finalize the payment.**
UC-9	**authentications**	**This is used by the system to validate user activities to check if registration is needed.**

1.1.2 Use Case Diagram:



1.1.2 Use Cases Description:

Use Case 1: Search Book

Summary: The user searches for a book in the system.

Primary Actor: User

Preconditions: The user must have internet access.

Description: The user first starts by navigating to the main page of the system. Once on the user is on main page, the user locates the search bar, which serves as the gateway to their quest for a specific book. In order to find the desired book, the user enters the book's accession number or title into the search bar, providing a precise identifier to locate the item. After inputting the details, the user takes the next step by clicking on the search icon, triggering the search process. At this point that the system actively engages, swiftly scanning its database for the relevant information. Finally, once the search is complete, the system displays comprehensive book details to the user, presenting them with valuable information about the book they were looking for, such as its title, author, availability, location, and any other pertinent information, facilitating a seamless and efficient library experience.

Exceptions:

- **Book Not Found**: If the book's accession number entered by the user is not present in the library's database, the system should display a notification message indicating that the book is not available.
- **Insufficient Information**: If the user enters incomplete or invalid information in the search bar, such as a malformed accession number, the system should display an error message prompting the user to provide valid search criteria. It may also offer suggestions or examples for correct input.

Post-Conditions: The search results are presented to the user.

Use Case 2: Borrow Book

Summary: A library patron borrows a book from the library system.

Actor: customer

Precondition: The User is logged in, and the system displays the screen.

Description (successful operation): The library patron navigates to the "Borrow Book" function within the system. The system provides a search bar or a list of available books. The patron can search for a specific book by title, author, or genre. Once the desired book is located, the patron selects it. The system checks if the book is available for borrowing. If the book is available, the system prompts the patron for confirmation. The patron confirms the borrowing of the book. The system updates the database, marking the book as "checked out" to the patron. A confirmation message is displayed indicating that the book has been successfully borrowed.

Exceptions (out of the ordinary situations):

- Book Unavailable: If the selected book is not available (e.g., already checked out), the system displays a notification informing the patron that the book cannot be borrowed at the moment.
- Maximum Borrowed Books: If the patron has already reached the maximum number of books allowed for borrowing, the system displays a notification indicating that no more books can be borrowed until some are returned.
- Technical Error: If there is a technical issue preventing the borrowing of a book, the system displays an error message and suggests trying again later.

Postcondition: The system updates the library database, marking the borrowed book as "checked out" to the patron. The borrowing action is recorded in the system's history for future reference and analysis.

Use Case 3: **Purchase Book**

Summary: the user purchases a book from the library system.

Actor: user

Precondition: The User is logged in, and the system displays the screen.

Description (successful operation):

Description:The user clicks on the "Purchase Book" function within the system. The system shows a search window. The user must search for a book either by its title or its ISBN. The system must return the search result to the purchase screen and the user must select the book he/she is looking to then click "Checkout". The system will process the payment procedure and return the results back to the purchase screen.

Exceptions (out of the ordinary situations):

- **No available copies to sell:** If there are no copies the system must display a message box to the user notifying him/her that there are no available copies.

Postcondition: The system displays window contains the receipt and the details of the book and the payment transaction.

Use Case 4: Return Book

Summary: A library patron returns a borrowed book to the library system.

Actor: Library Patron

Precondition: The User is logged in, and the system displays the screen.

Description (successful operation): The library patron navigates to the "Return Book" function within the system. The system displays a list of books currently borrowed by the patron, along with their due dates. The patron selects the book they wish to return. The system prompts for confirmation to ensure the return is intentional. The patron confirms the return of the book. The system updates the database, marking the book as "returned" and adjusts the due date if necessary. A confirmation message is displayed indicating that the book has been successfully returned.

Exceptions (out of the ordinary situations):

- **Overdue Book:** If the book is returned after the due date, the system may apply a late fee or display a notification to inform the patron of the overdue status.
- **Technical Error:** If there is a technical issue preventing the return of a book, the system displays an error message and suggests trying again later.

Postcondition: The system updates the library database, marking the returned book as "available" for other patrons to borrow. The return action is recorded in the system's history for future reference and analysis.

Use case 5: Add Book

Summary: The admin or manager adds a new book to the library system.

Actor: admin/manager

Precondition: The User is logged in, and the system displays the screen.

Description (successful operation): The admin/manager navigates to the "Add Book" function within the system. The system prompts the user to input the necessary details for the new book, including title, author, genre, ISBN, publication year, and available quantity. The admin/manager fills in the required information. The system validates the input and checks for any potential duplicates. If all information is valid, the system adds the book to the library database. A confirmation message is displayed indicating that the book has been successfully added. The admin/manager has the option to add another book or return it to the main menu.

Exceptions (out of the ordinary situations):

- **Incomplete Information:** If the admin/manager fails to provide essential details, the system displays an error message and prompts for the missing information.
- **Duplicate ISBN:** If the ISBN provided already exists in the system, the admin/manager is notified and prompted to review and correct the information. • Technical Error: If there is a technical issue preventing the addition of a book, the system displays an error message and suggests trying again later.

Postcondition: The system updates the library database with the newly added book, making it available for borrowing. The added book is recorded in the system's history for future reference and analysis.

Use Case 6: Delete Book

Summary: The admin or manager removes a book from the library system.

Actor: admin/manager

Precondition: The User is logged in, and the system displays the screen.

Description (successful operation): The admin/manager navigates to the "Delete Book" function within the system. The system provides a list of books currently available in the library. The admin/manager selects the book they want to delete. The system prompts for confirmation to ensure the deletion is intentional. The admin/manager confirms the deletion. The system removes the selected book from the library database and updates the records accordingly. A confirmation message is displayed indicating that the book has been successfully deleted.

Exceptions (out of the ordinary situations):

- **Book Currently on Loan:** If the selected book is currently on loan, the system displays a notification informing the admin/manager that the book cannot be deleted until it is returned.
- **Technical Error:** If there is a technical issue preventing the deletion of a book, the system displays an error message and suggests trying again later.

Postcondition: The system updates the library database, removing the deleted book from the records. The deletion action is recorded in the system's history for future reference and analysis.

Use case 7: Generate Reports

Summary: the admin or manager generates reports in the system.

Actor: admin/manager

Precondition: The User is logged in, and the system displays the screen.

Description (successful operation): The admin/manager clicks on the "Generate Reports" function within the system. The system shows a menu of options for the types of reports available (borrowing history, overdue items, popular genres…). The admin/manager selects the desired type of report. The system processes the request and compiles the relevant data. The system generates and displays the selected report in a readable format (table, graph, PDF…) as selected. The admin/manager has the option to save, print, or export the report for other uses.

Exceptions (out of the ordinary situations):

- **Not enough data to generate**: If there is a lack of data to generate a specific type of report, the system provides a notification to the admin/manager.

Postcondition: The system records the generated report for future reference and analysis as a manageable history list.

Use case 8: Checkout!

Summary: The user finalizes the payment for a book purchase in the library system..

Actor: library patron

Precondition: The User is logged in and has items in cart.

Description (successful operation): The library patron adds item in card and then clicks on the "checkout" function within the system. The system shows a item in cart and shows the price receipt. The user chooses payment method, the system processes the request and compiles the relevant data and take the user to the payment page. The system confirms the payment and sends the receipt to the user.

Exceptions (out of the ordinary situations):

- **Unavailable item in cart:** if the item is out of stock or not available anymore during checkout, the system should notify the user and provide an option to remove the unavailable items or continue without them..
- **Invalid payment:** if the user fails to complete the payment, the system sends an error message notifying the user about the issue of the payment.

Postcondition: The user successfully completes his purchase with a receipt from the system. The system updates the data of the user.

Use Case 9: Authentications

Summary: The system validates user activities and determines if user registration is required for certain actions.

Actor: System

Precondition: The system is operational, and a user interacts with it.

Description (successful operation): The system continuously monitors user interactions with the library system. When a user attempts to perform specific actions that require authentication or registration, such as borrowing books, purchasing books, or accessing certain user-specific features, the system checks whether the user is already logged in. If the user is logged in, the system proceeds with the requested action, ensuring that the user has the necessary privileges. If the user is not logged in, the system prompts the user to log in or register, depending on the action's requirements.

Exceptions (out of the ordinary situations):

- **Authentication Failure**: If the user provides incorrect login credentials, the system should display an error message and allow the user to try again or reset their password.
- **Registration Required:** If a user attempts an action that requires registration, the system should guide the user through the registration process.

Postcondition: The system grants or denies access to specific features based on user authentication status and processes user registrations as needed.

Phase 2

Quality Attributes
Constrains
Concerns

2.1 Quality Attributes

.

ID:	Attribute	Description	Associated UC
QA-01	Usability	The system should be user-friendly, with an easy-to-navigate interface, and provide help frames and menus.	All
QA-02	Performance	Quick response times, so that the user should complete a task within 10 seconds.	All
QA-03	Security	Strong data encryption and user authentication.	UC-7,8,9
QA-04	Scalability	The system should handle a growing number of users and books, and future updates and add ons.	UC-2,3,5
QA-05	Reliability	The system should be available 24/7 with maximum of 1 day shutdown time.	All
QA-06	Modifiability	The system should be easy to modify where the authorized users (librarian) should be able to modify the system will the system is available to use. modification could be done for adding new books, deleting books modify the reports formats.	UC-5,6,8

2.2 Constraints:

ID	Constraints
CON-01	Web-based application using JavaScript and PHP.
CON-02	The system should be cost-effective and within the estimated budget.
CON-03	Ensure compliance with data protection and copyright regulations.
CON-04	Ensure the system can scale to accommodate more users and data.
CON-05	Make the system compatible with major web browsers (Chrome, Firefox, Edge).
CON-06	The system woks locally in Bahrain and servelocal users.
CON-07	The payments records must be stored for 5 years for future need.
CON-08	The system must be developed and deployed within 6 months.

2.3 Concerns

ID	Description
CRN-01	Establishing an overall initial system structure.
CRN-02	Implement the system to be used in both Arabic and English languages.
CRN-03	The team members are knowledgeable in JavaScript, PHP, HTML, CSS technologies.
CRN-04	The payments done through the system are performed using Bahraini Dinar.
CRN-05	Distribute work on the software development team.
CRN-06	The system must support the latest technologies and tools.

Phase 3

Design Process

ADD Steps

Iteration 1: Establishing the System Structure

This section presents the results of the activities that are performed ineach of the steps of ADD in the first iteration of the design process.

ADD 1.1 Step 1: Review inputs:

<table>
<tr><th>Categories</th><th>Details</th></tr>
<tr><td>Design purpose</td><td>The design is for a greenfield system from a mature domain and its purpose is to automate library services, enhance user experience, and reduce operational costs. The system aims to replace manual processes with a more efficient and convenient automated solution.</td></tr>
<tr><td>Primary functionality requirement</td><td>From the user cases presented in phase 1 and 2, those are the primary functionality:
<b>UC-5:</b> It is the core function used in a library where adding the books and present them in the system is a goal to be satisfied for both librarians and library customers.
<b>UC-8:</b> Because it controls and finalize the borrow and purchase processes.
<b>UC-9:</b> Because in some of the system functions it is essential to maintain high security such as when purchasing books the users need to enter their credit card credential or when a user want borrow a book it is important to authenticate the user identity.</td></tr>
<tr><td>Quality attribute sceneries</td><td>The scenarios described in the quality attribute list have now been prioritized as follows:
<table>
<tr><th>Scenario ID</th><th>Importanceto the customer</th><th>Difficulty of implementation according to the Architect</th></tr>
<tr><td>QA-01: Usability</td><td>High</td><td>low</td></tr>
<tr><td>QA-02: Performance</td><td>High</td><td>High</td></tr>
<tr><td>QA-03: Security</td><td>High</td><td>Medium</td></tr>
<tr><td>QA-04: Scalability</td><td>Medium</td><td>High</td></tr>
<tr><td>QA-05: Reliability</td><td>High</td><td>High</td></tr>
<tr><td>QA-06: Modifiability</td><td>low</td><td>Medium</td></tr>
</table>
</td></tr>
<tr><td>Constrains</td><td>All the constraints discussed in the constrain table in phase 2 are considered as drivers for the design.</td></tr>
<tr><td>Concerns</td><td>All the concerns discussed in the concerns table in phase 2 are considered as drivers for the design.</td></tr>
</table>

ADD 1.2 Step 2: Establish Iteration Goal by Selecting Drivers:

This is the first iteration throughout the design of a greenfield system, so the goal of the iteration is to achieve the initial overall structure that supports CON-01, CON-2 and CON-08. As well as CRN-01 and CRN-03.

This iteration is driven by a general architectural issue, but all the drivers that can impact the overall system's structure must be taken into account; hence, the architect needs to be aware of the following:

- QA-01: Usability
- QA-02: Performance
- QA-03: Security
- QA-04: Scalability
- QA-05: Reliability
- QA-06: Modifiability
- CRN-02: Implement the system to be used in both Arabic and English languages.
- CON-01: Web-based application using JavaScript and PHP.
- CRN-06: The system must support the latest technologies and tools.

ADD 1.3 Step 3: Choose One or More Elements of the System to Refine:

This is a greenfield development effort. Online Library management system is the element to refine it that shown in Figure 1 below. In this case, refinement is performed through decomposition.

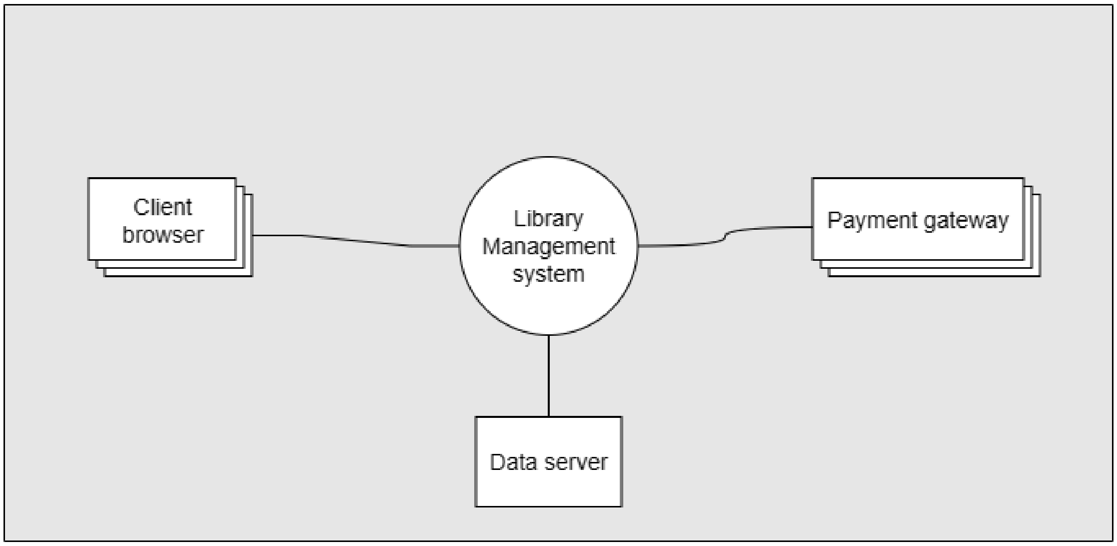


Figure 1.1: Context Diagram for Online Library Management System

ADD 1.4 Step 4: Choose One or More Design Concepts That Satisfy the Selected Drivers:

In the first iteration, we have chosen design concept; reference architectures of type web application that will provide the overall structure of the application. The following table summarizes the design decisions about choosing one design concept and the alternatives one.

<table>
<tr><th>Locations</th><th>Design decisions</th><th>Rationale</th></tr>
<tr><td>Logical Structure</td><td>Web Application Reference Architecture</td><td>
<ul>
<li>The web application reference architecture is designed to deliver the performance requirements outlined in QA-02 while adhering to the constraints and preferences specified in CON-03, CON-04, CON-06, and CRN-05. The presentation layer plays a essential role in achieving the desired user experience (QA-01) through a user-friendly web interface. This aligns with the preference for Web-based system (CON-01), while considering the need of network connection to interact with the system.</li>
<li>During the selections process for the reference architecture to be used for the library system being designed other alternative architectures were considered in the below table there is a demonstration on those alternatives and why they got discarded:</li>
</ul>
<table>
<tr><td>Alternative Architectures</td><td>Reasons for discarding</td></tr>
<tr><td>Rich Client</td><td>It was discarded because there is no need to have rich interfaces and the application will not be on the client’s machines, instead will be accessible through the web.</td></tr>
<tr><td>Rich Internet Application</td><td>It was discarded because there is no need for rich interfaces and client-side processing.</td></tr>
<tr><td>Mobile Application</td><td>It was discarded due to unreliable connectivity whereas reliable connectivity is needed for our system. Also, the resources of the handheld device may be limited.</td></tr>
<tr><td>Service Application</td><td>It was discarded because the application is used by human and requires user interface.</td></tr>
</table>
</td></tr>
<tr><td>Physical Structure</td><td>Deployment Pattern (Three-Tier)</td><td>
Since the system must be accessed from a web browser and a database server must be used, a three-tier deployment is the optimal option for the deployment process.
<ul>
<li>Discarded alternatives:
<ul>
<li>Other n-tier (n>3) deployment patterns were discarded because no additional servers are needed since the system will be accessible through web and data will be stored in data server.</li>
<li>Since the web server which acts as the gateway of the system will not be in the same server as the application, the non-distributed (2-tier) pattern was discarded.</li>
</ul>
</li>
</ul>
</td></tr>
<tr><td></td><td>Build the system using JavaScript, html, CSS, and PHP</td><td>The technologies used to build the system are those the team members familiar with (CRN-03), considering CON-01></td></tr>
</table>

ADD 1.5 Step 5: Instantiate Architectural Elements, Allocate Responsibilities, and Define Interfaces:

The instantiated design decisions considered and made are summarized in the following table.

Design Decision and Location	**Rationale**
User Authentication module	Component responsible for validating user activity during the use of the system adhering UC-09, UC-08, UC-07, UC-06. Define an interface that handles user authentication requests and responses.
Create a module to manage the books	Instantiate a layer that is responsible for adding and deleting books by the librarians of borrowed or purchased by the customers. The interface that should be designed is accessible only for librarians and admins.
Create payment module	Payment service responsible for processing payments securely and customer checkout. Define interfaces for initiating and completing payment transactions.

ADD 1.6 Step 6: Sketch Views and Record Design Decision:

These sketches were created by an online tool (diagram.net) and the following table represents the major functional responsibilities for each element in the sketches but please note that we listed the main responsibility for each one with no details

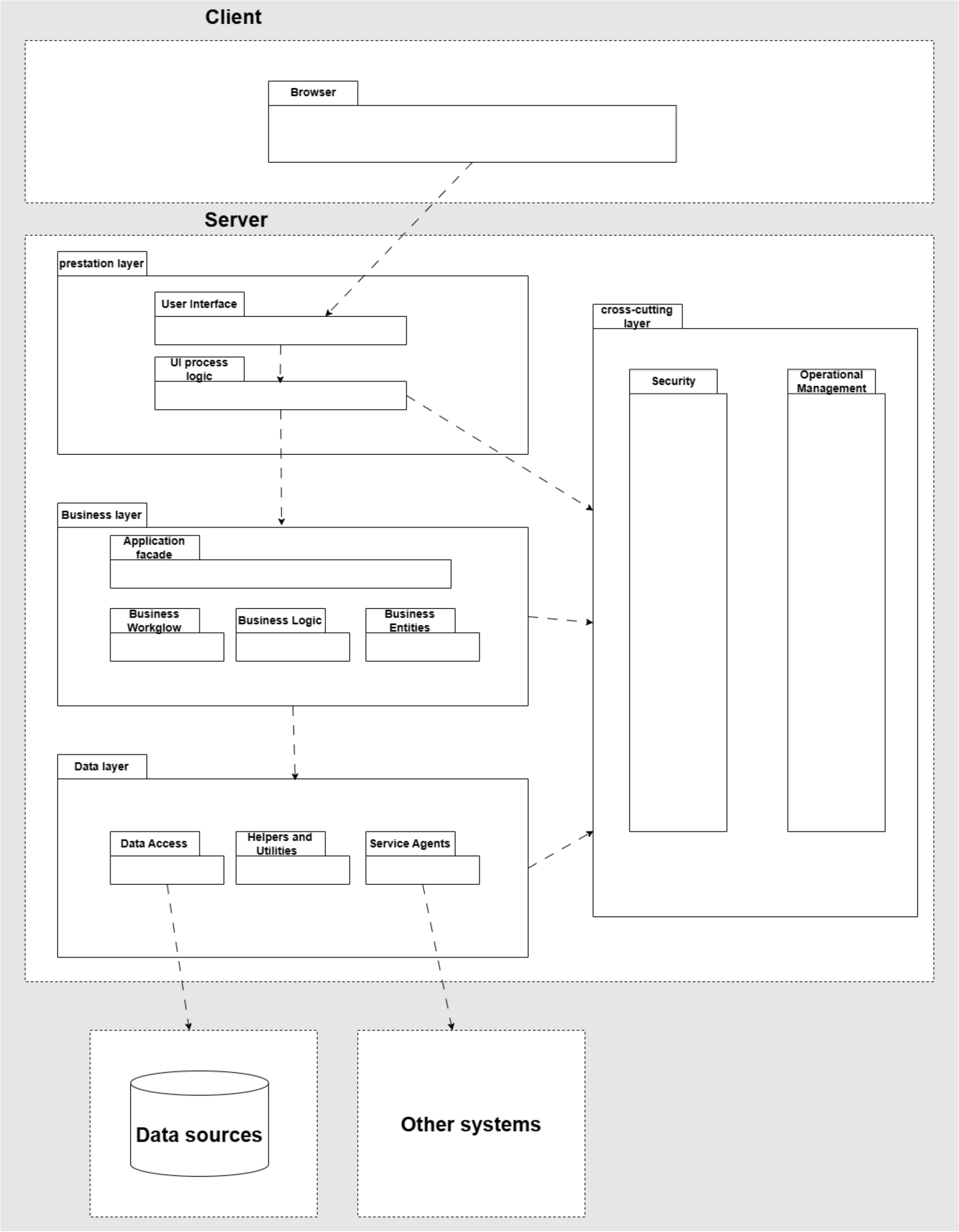


Figure 1.2: Modules obtained from the selected reference architectures.

Elements	**Responsibility**
Client side (Browser layer)	This layer is responsible for connecting the client to the server acting as a gate using browsers.
User interface module	This module includes all the interfaces that the user needs to interact with the server.
UI process logic	This module is responsible for directing the client to right interface and interacts with the needed layers and modules according to the stimulus.
Cross Cutting Layer	Layer with functionality reaching multiple Layers is included in this Layer. In other words, it supports the architecture design to meet QA-03 and UC-09.
Business layer	This layer is the heart of the server which performs all the primary functionalities UC-09, UC-08 and UC-05 responsively to the user request by execution these use case and applying business rules on the data.
Data layer	This layer is responsible to manipulate, process, save, retrieve and update data. Also, it is responsible to interact and communicate with external systems.
Data access module	This module is responsible for retrieve, update and save data with the data sources.
Helpers and Utilities module	This module contains similar functionality to other modules in the data layer but not specific to any of them.
Service agent	This module is used to transfer and exchange data with other system such as PayPal.

The deployment diagram:

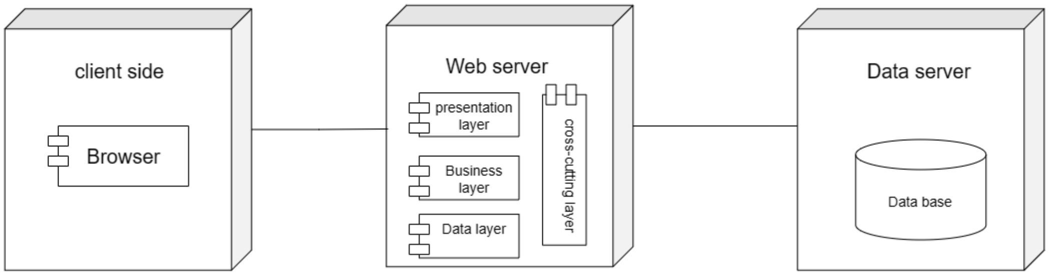


Figure 2.3: Initial deployment diagram for the system

In Figure 2.3, the deployment diagram sketches an allocation view that Shows where the components connected with the modules in the previous diagram will be deployed.

- The following table summarizes the responsibilities of the deployment pattern's elements:

Element	Responsibility
Client side	User's machines that access the application trough it
Application server	The server that hosts the server-sidelogic of the application
Database server	The server that hosts the relationshipof database.

- The communication between the Web Server and the Data Server is done using HTTP and TLS protocols.

ADD 1.7 Step 7: Perform Analysis of Current Design and Review Iteration Goal and Achievement of Design Purpose:

Not Addressed	Partially Addressed	Completely Addressed	Design Decisions Made During theIteration
QA-01			No relevant decisions made, as it is necessary to identify the elements that participate in the use case that is associated with the scenario.
	UC-02		Selected reference architecture establishes the modules that will support this functionality.
	UC-3		Selected reference architecture establishes the modules that will support this functionality.
	UC-5		Selected reference architecture establishes the modules that will support this functionality.
	UC-6		Selected reference architecture establishes the modules that will support this functionality.
	UC-9		Selected reference architecture establishes the modules that will support this functionality.
	QA-04		As an update, the system incorporates a fresh protocol for overseeing the time server. The integration of this protocol is accomplished seamlessly, without necessitating any adjustments to the current system.
CRN-01			No design decision has been made
	CRN-02		By selecting three-tier deployment patterns which helps to add more security layers between each component of the system (Firewall between the web server and the data server).
		CON-04	The use case supports the selected referencearchitecture
CON-2			Not relevant decisions were made
		CON-06	This was satisfied by selecting the reference architectures that enable the system to operate in different environment if the internet exists.

	CRN-2		The use of the expertise of the developer inHTML, CSS, JavaScript and PGP is taken into consideration, but still more decisions should be made such as choosing frameworks to ease the implementation process.
	QA-1		This partial addressed though selecting the best reference architectures that are sufficient to satisfy the user need. More work should be done one identifying the user interfaces and this will be done in the subsequent iterations
		QA-6	This quality attribute is strongly supported by the reference architecture chosen becauseit reserves the data.

Iteration 2: Identifying Structures to support Primary functionality

In this iteration the functionalities of the system are addressed in detail. The outcome of this iteration will guide the development process and help in forming the right development team that carryout the system's functionalities. The goal of the first iteration was to establish an overall initial structure. After the first iteration the goal was satisfied successfully. Now it is the time to take the right design decisions that help in creating the units of implementation which play a major role in selecting the team members, the interfaces of the system, the distribution of the development tasks, deciding whether there is a need for outsourcing.

2.1 Step 2: Establish Iteration Goal by Selecting Drivers:

The goal of this iteration is to tackle the general architectural concern to identify structures that support the primary functionality of the system.

The primary use cases that will be addressed in this iteration are:

- UC-3 (Purchase Book)
- UC-5 (Add Book)
- UC-6 (Remove Book)

2.2 Step 3: Choose One or More Elements of the system to refine:

In the previous iteration different reference architecture were selected those architectures contain different modules that exist in different layers in this step the elements that will be refined are coming from different layers which requires both collaboration and coordination between the components associated with elements to be refined.

2.3 Step 4: Choose One or More Design Concepts That Satisfy the Selected Drivers:

In this step, several design concepts were selected, and the following table summarizes the decision decisions taken in the selection process:

Design Decisions and Locations	Rational and Assumptions
Create a Domain Model for the application	Before embarking on the process of functional decomposition, it's essential to formulate an initial domain model for the system. This involves pinpointing the significant entities within the domain and mapping out their interrelations. There aren't any satisfactory alternatives to this. The creation of a domain model is inevitable. If not done properly, it will materialize in a less than perfect way, resulting in a makeshift architecture that poses difficulties in understanding and maintenance.
Identify Domain Objects that map to functional requirements	Each distinctive functional element of the application needs to be encapsulated within a standalone unit known as a domain object. An alternative strategy might involve bypassing the contemplation of domain objects and directly segmenting layers into modules. However, employing this approach increases the risk of neglecting a requirement.
Use React framework	React.js, a widely adopted library, is renowned for building dynamic and interactive applications. It enhances the UI/UX design for both web and mobile applications. Being an open-source and component-based front-end library, it takes the responsibility for the UI design. When considering alternatives for application development, Angular was a strong contender. However, React.js was eventually chosen due to its lightweight nature and the development team's familiarity with it, which resulted in greater and earlier productivity. The team did not consider other front-end libraries, as they were already satisfied with the performance of React.js.

2.4 Step 5: Instantiate Architectural Elements, Allocate Responsibilities and Define Interfaces:

The instantiation design decisions made in this iteration are summarized in the following table:

Design Decisions and Location	Rationale
Create only an initial domain model	The identification and organization of entities participating in the primary use cases are essential. However, for the purpose of streamlining the design phase, only a provisional domain model is created.
Map the system use cases to domain objects	An initial identification of domain objects can be accomplished by analyzing the use cases of the system. Domain objects are specifically identified for each of the use cases outlined in the first phase (All use cases needed for the system)
Decompose the domain objects across the layers to identify layer-specific modules with an explicit interface	This approach guarantees the recognition of modules that address all functionalities. The architect will handle this responsibility exclusively for the primary use cases, allowing another team member to identify the remaining modules and distribute the workload accordingly. Upon defining the set of modules, the architect acknowledges the need to test these modules, introducing a new architectural consideration: which stipulates that a majority of modules should undergo unit testing. This consideration encompasses most modules, excluding those implementing user interface functionality, as they present challenges when tested in isolation.
Associate frameworks with a module	Establish connections between frameworks and modules within the presentation layer: The presentation layer houses modules that encapsulate UI components. React framework is linked to these modules. This arrangement enables us to make optimal use of Reacts component-based structure, incorporating HTML, CSS, and JavaScript to proficiently construct interactive user interfaces CRN-3.

Note: The structures and interfaces are not identified in this step, they will be captured in the next step.

2.5 Step 6: Sketch Views and Record Design Decisions:

As a result of the decisions made in step 5, several diagrams are created:

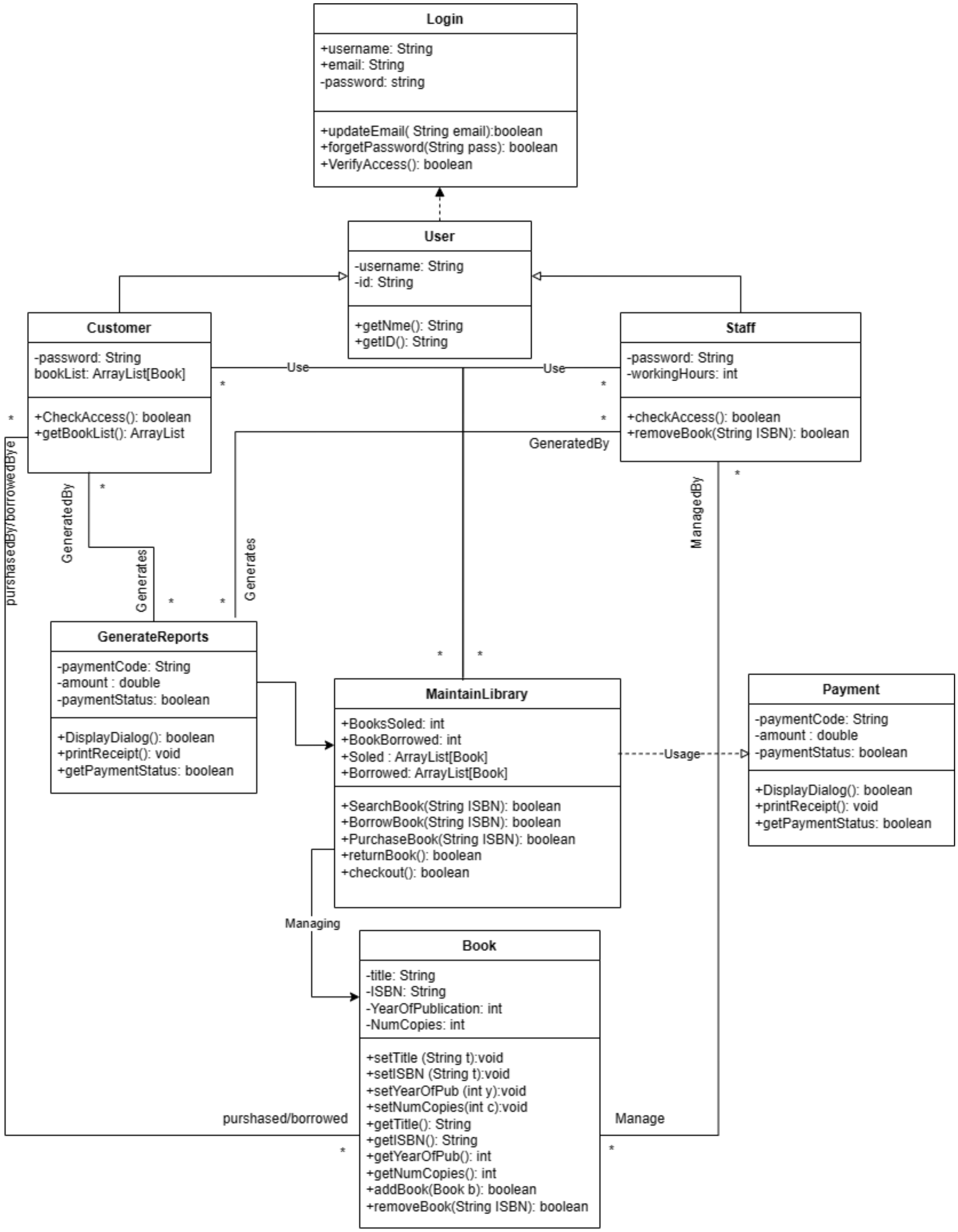


Figure 2.1: Shows initial domain model for the system.

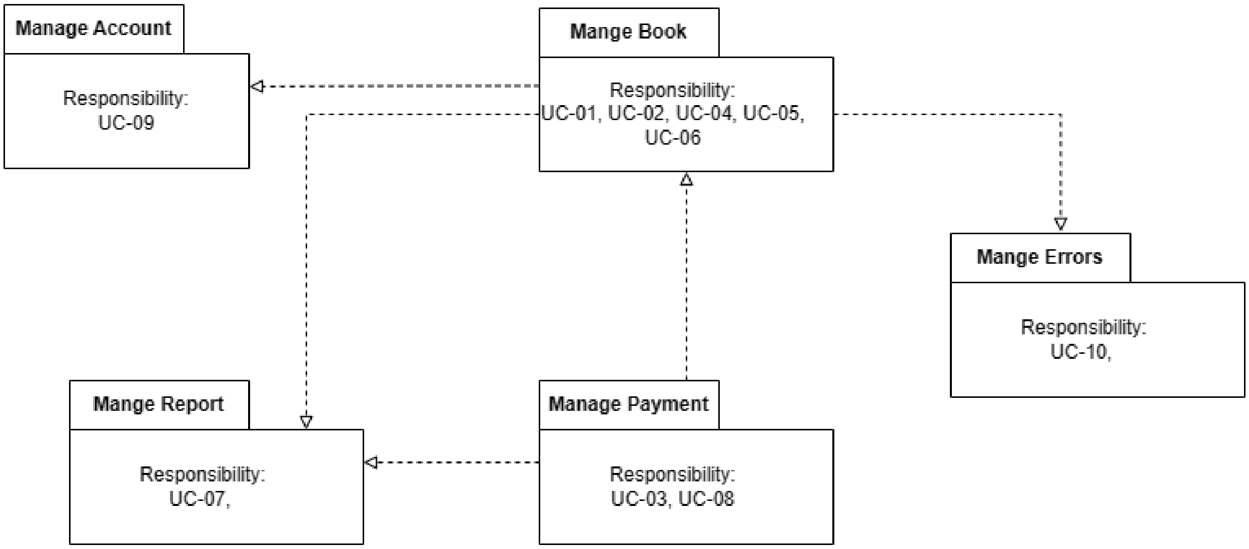


Figure 2.2: shows the domain objects that are instantiated for the use case.

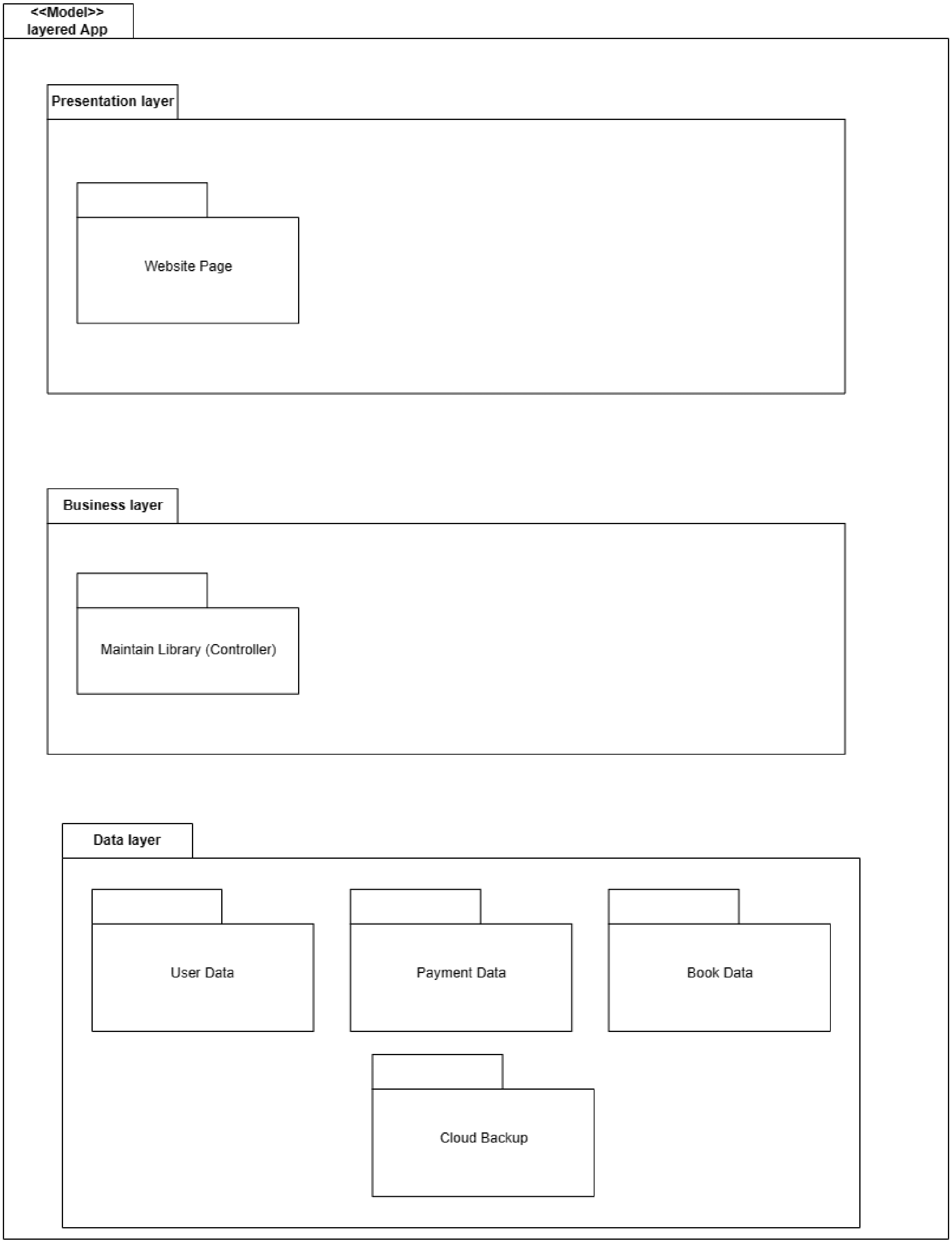


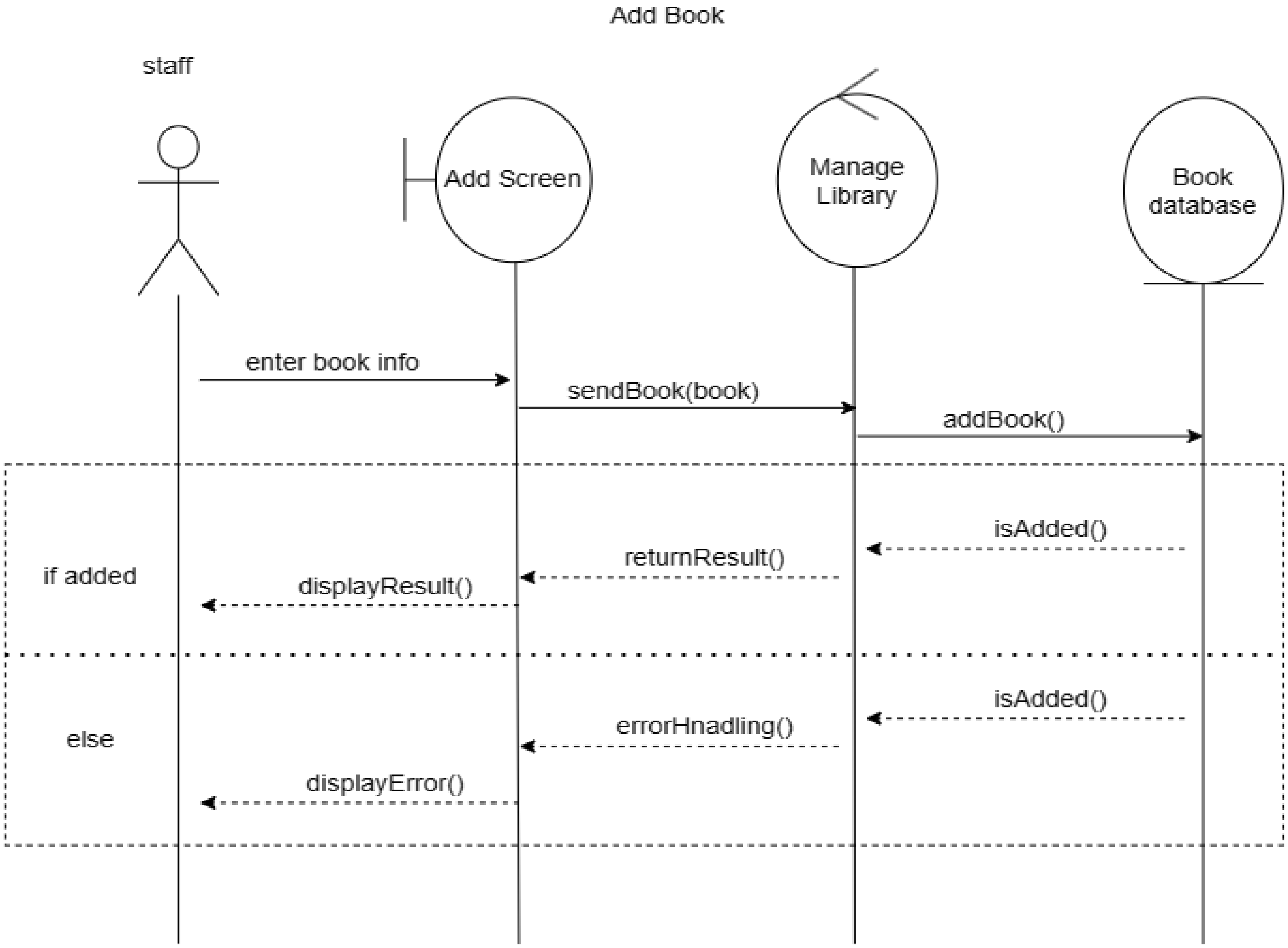
Figure 2.3: shows a sketch of a module view with modules that are derived from the business objects and associated with the primary use cases.

The responsibilities for the elements identified above are summarized in the table below:

Element	**Responsibility**
Website Page	All the functionality of the system will be displayed to user on the website page and the user will interact with the system through this page.
Maintain library (controller)	This is responsible to instantiate the desired function requested by the user. Example if the user wants to purchase a book, he/she needs to click on purchase book once the button is clicked the controller will open the purchase book page and so on.
User Data	Stores the user data whether the user is a customer or staff.
Cloud Backup	Contains a replication of all the data used in the system and it is used only used in case there is a failure in the on-premises severs

The Sequence Diagram of the primary functionalities:

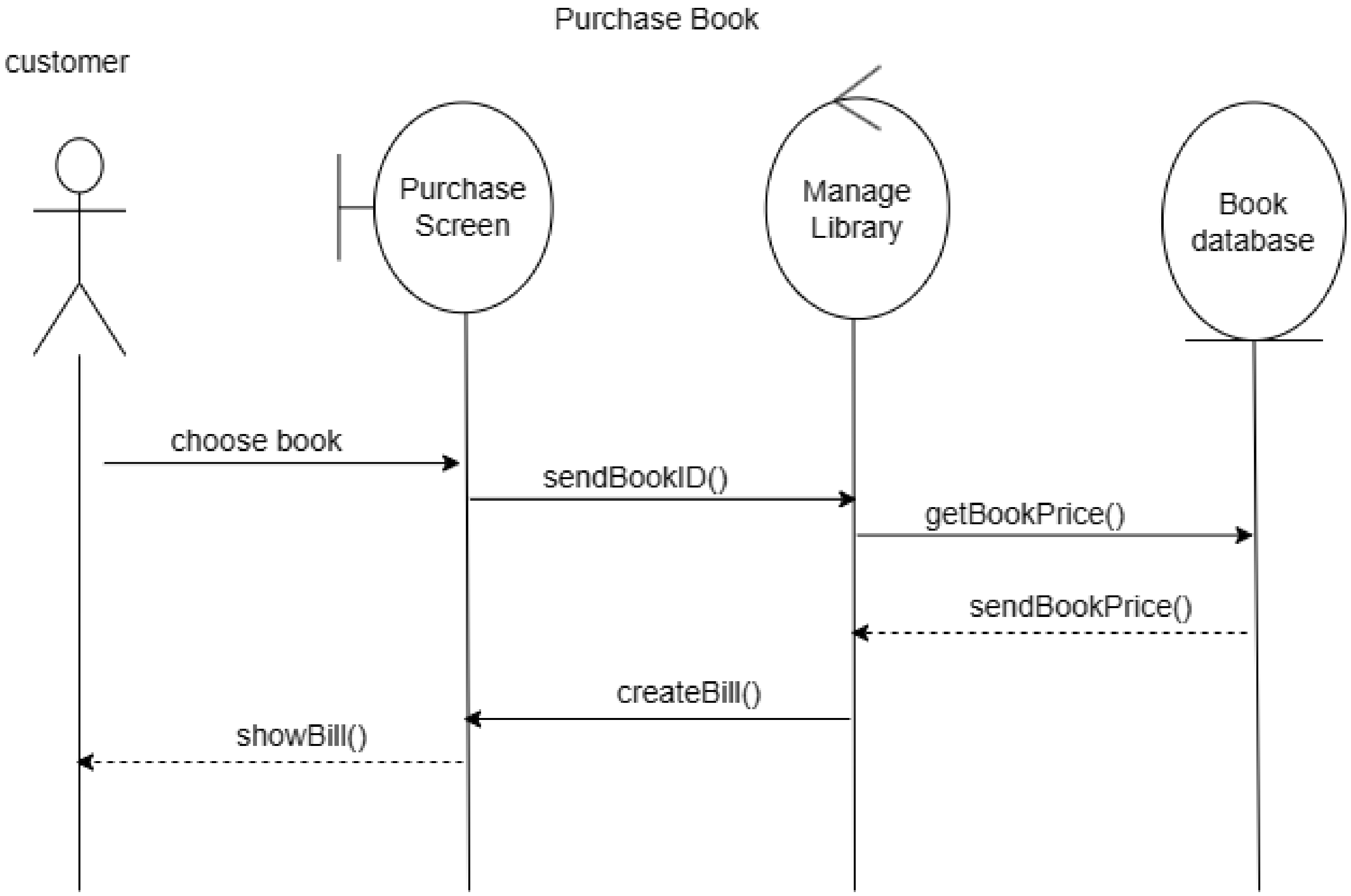
UC-5 (Add Book):



UC-5 (Add Book):

Method Name	Description
Element: Add Screen	
sendBook()	This method takes the book information and convert it to an object of type book, then send it to Manage Library element.
displayResult()	In case the add operation was successful, this method will display success message and will display the row that added to the DB.
displayError()	In case the add operation failed, this method will display an appropriate error message according to errorHandling() method.
Element: Manage Library	
addBook()	After receiving the book object from the sendBook() method, this method will convert the object to relational data that can be added to the DB and add it to the DB.
returnResult()	In normal case, this method return the row that added to the DB.
errorHandling()	In abnormal case, this method handle the error and return the reason of the error and the solution if possible.
Element: Book Data Base	
isAdded()	This method return true if the book was added, otherwise it returns false

UC-3 (Purchase Book):

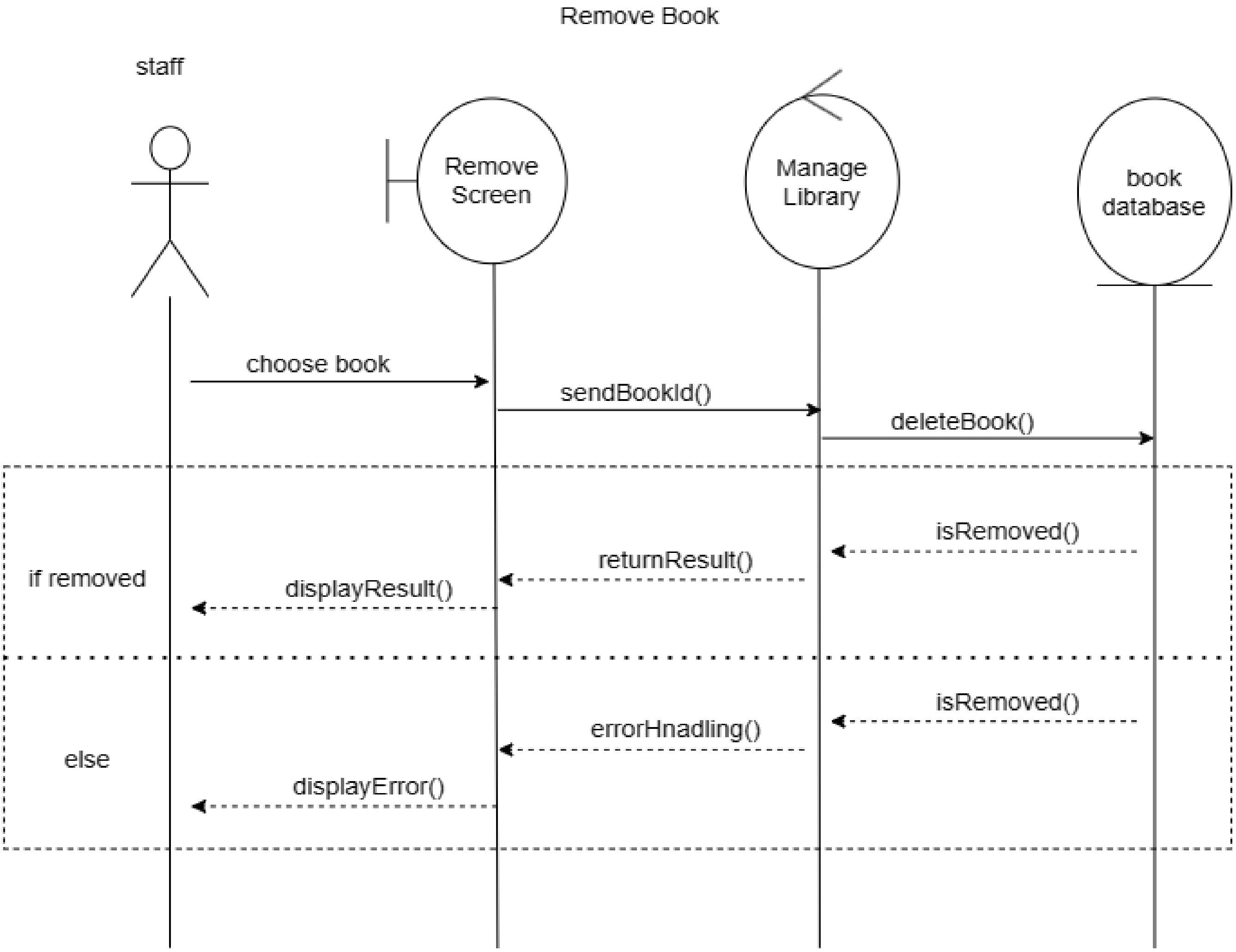


Note: the purchase use case ends after displaying the bill and all cart details, then continue with checkout use case.

UC-3 (Purchase Book):

Method Name	Description
Element: Add Screen	
sendBookId()	This method sends the ID of the book to be deleted that selected by the user.
showbill()	This method displays all the details about selected books to be purchased.
Element: Manage Library	
getBookPrice()	After receiving the book ID from the sendBookId() method, this method returns the price of the book.
createBill()	This method creates and organizes the bill to be displayed to the customer.
Element: Book Data Base	
sendBookPrice()	This method sends the book price to the library management element.

UC-6 (Remove Book):



UC-6 (Remove Book):

Method Name	Description
Element: Add Screen	
sendBookId()	This method sends the ID of the book to be deleted that selected by the user.
displayResult()	In case the remove operation was successful, this method will display success message and will display the row that deleted from the DB.
diplayError()	In case the add operation failed, this method will display an appropriate error message according to errorHandling() method.
Element: Manage Library	
deleteBook()	After receiving the book ID from the sendBookId() method, this method will delete the row containing the book selected.
returnResult()	In normal case, this method returns the row that deleted to the DB.
errorHandling()	In abnormal case, this method handles the error and return the reason of the error and the solution if possible.
Element: Book Data Base	
isRemoved()	This method returns true if the book was removed, otherwise it returns false

2.7 Step 7: Perform Analysis of Current Design & Review Iteration Goal & Achievement of Design Purpose:

Not Addressed	Partially Addressed	Completely Addressed	Design Decisions Made During theIteration
		CON-01	This was satisfied through selecting the best technologies that can be used within the development team's knowledge. React.js was selected as the framework to be used for the interfaces implementations since it can be integrated with JavaScript, HTML, and CSS.
		CON-02	This was satisfied through selecting the technologies that are light-weighted and less complex of favor React.js over Angular because it is not complicated and not costly comparing to Angular framework.
		UC-02	The initial interfaces catering to this specific use case have been identified.
		UC-03	The initial interfaces catering to this specific use case have been identified.
		UC-05	The initial interfaces catering to this specific use case have been identified.
		UC-06	The initial interfaces catering to this specific use case have been identified.
		UC-09	The initial interfaces catering to this specific use case have been identified.
	QA-01		The elements that support the associated with primary use cases have been identified
	QA-02		The elements that support the associated with primary use cases have been identified
	'QA-03		No design decisions have been made
	QA-04		No design decision has been made

Iteration 3: Addressing Quality Attribute Scenario Driver (QA-6)

This section of the design displays the results of tasks conducted at every stage of the Attribute-Driven Design (ADD) in the third iteration of the design process. Building upon the essential structural decisions from the initial two iterations, we can now begin considering the implementation of certain important quality attributes. The focus of this iteration primarily revolves around a specific quality attribute scenario.

3.1 Step 2: Establish Iteration Goal by Selecting Drivers:

In this iteration the focus was on the QA-06 quality attribute scenario: A failure occurs in the system during carrying out a process. The management system should be available 24/7 with a maximum of one day of shutdown every month.

3.2 Step 3: Choose One or More Elements of the System to Refine:

For this reliability scenario, the elements of the system that will be refined are the physical nodes that were identified during the first iteration.

- Web Server
- Data Server

3.3 Step 4: Choose One or More Design Concepts That Satisfy the Selected Drivers:

The design concepts used in this iteration are summarized as follows:

Design Decisions and Location	**Rationale and Assumptions**
Execute the active redundancy strategy by replicating the application server and critical components such as the database.	Through the replication of vital components, the system can withstand the failure of one of the duplicated parts without affecting its functionality.
Incorporate a component from the family of message queue technologies.	The application places traps received from time servers into a message queue and subsequently retrieves them. Employing a queue guarantees that the traps are handled and delivered in the sequence they were received (QA-2).
Insure all the primary functionalities are available to the users 24/7	The system will use a cloud-based technology in case there is a failure in the on-premises server the system will automatically shift to the virtual nodes provided by the cloud to continue operating.

3.4 Step 5: Instantiate Architectural Elements, Allocate Responsibilities and Define Interfaces:

The instantiation design decisions are summarized in the following table:

Design Decisions and Location	Rationale
Deploy message queue on a separate node	Establishing the message queue on a separate node guarantees that no traps are overlooked in the event of application failure. This node is duplicated through the active redundancy strategy, although only one replica processes and oversees events from the network devices.
Use active redundancy and load balancing in the application server	With both instances of the application server running concurrently, it makes sense to evenly distribute the workload between them. This can be achieved through the implementation of the Load-Balanced Cluster pattern.
Implement load balancing and redundancy using technology support	Several technological options exist for load balancing and redundancy, obviating the necessity of developing a less mature, custom solution that would pose greater maintenance challenges.

3.5 Step 6: Sketch Views and Record Design Decisions:

Figure 3.1 shows a refined deployment diagram that illustrate the introduction of redundancy in the system and how the system will be available in case of a failure on one or more of the nodes.

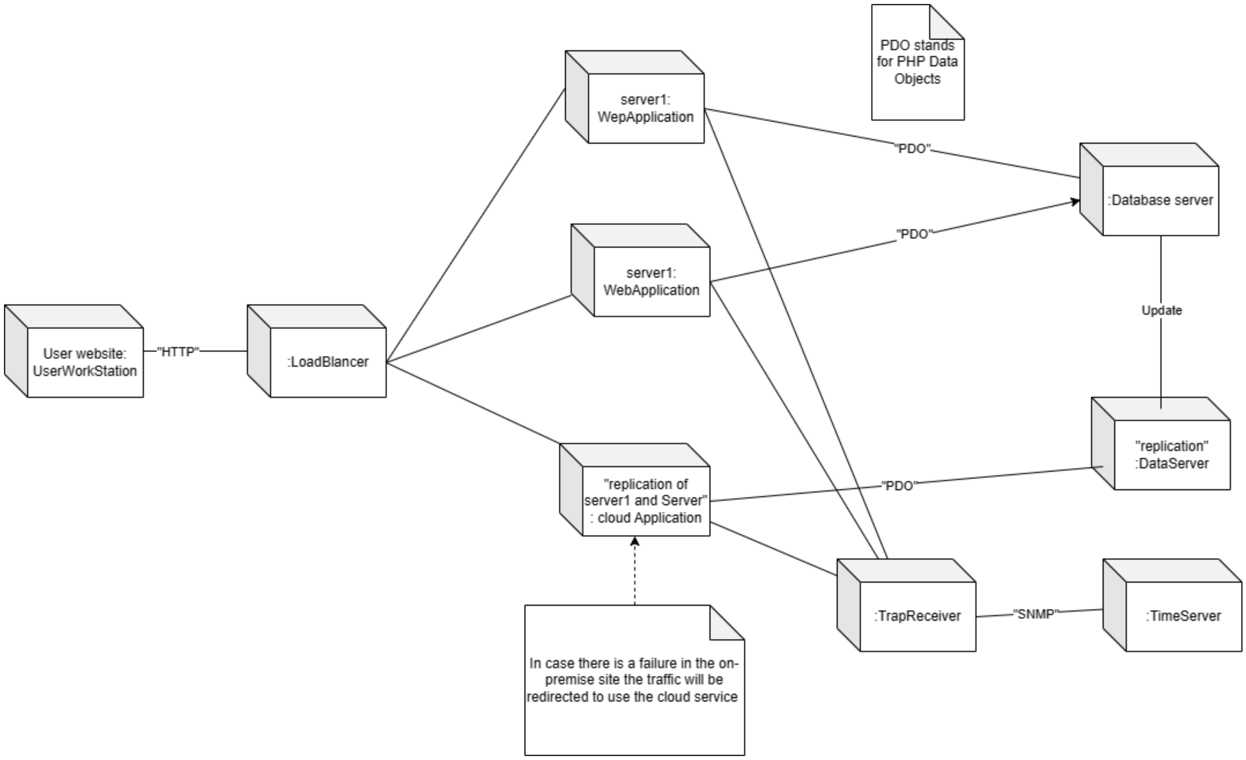

PDO stands for PHP Data Objects
server1: WepApplication
"PDO"
:Database server
server1: WebApplication
"PDO"
Update
User website: UserWorkStation
"HTTP"
:LoadBlancer
"replication" :DataServer
"replication of server1 and Server" : cloud Application
"PDO"
:TrapReceiver
"SNMP"
:TimeServer
In case there is a failure in the on-premise site the traffic will be redirected to use the cloud service

Figure 3.1: Refined deployment diagram (Key: UML)

- The following table contains a description of the elements that have not been discussed in the previous iterations:

Element	Responsibility
LoadBalancer	The load balancer not only directs and uniformly allocates incoming requests from clients to the application servers but also furnishes clients with a distinct IP address.
TrapReceiver	The system captures traps originating from the attempts to communicate with the server devices, converts them into events, and preserves these events in a resilient message queue.
CloudApplication	This will be used if there is failure in the on-premise site and all the traffics trying to access the system will be redirected to it.

- The UML sequence diagram illustrated in Figure 3.2 illustrates the collaboration of the newly introduced TrapReceiver in this phase with other components shown in the deployment diagram. This collaboration is designed to support UC-10 (Defect Errors), which is associated with both QA-6 (reliability) and QA-2 (performance). The primary aim of the diagram is to depict the communication between the physical nodes.

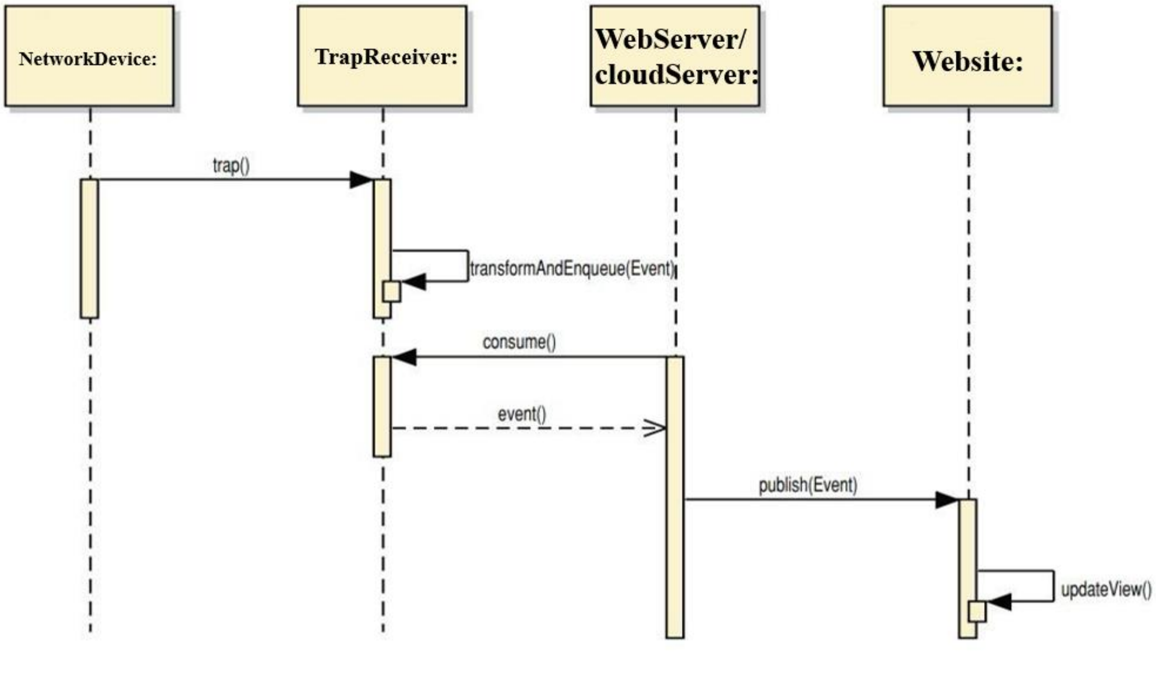


Figure 3.2: The messages exchange between the physical nodes to support UC-10

3.6 Step 7: Perform Analysis of Current Design and Review Iteration Goal and Achievement of Design Purpose:

In this phase, substantial design decisions were undertaken to address QA-6, subsequently impacting QA-2. The following table offers a synopsis of the statuses of different drivers and the decisions made during this iteration.

Not Addressed	Partially Addressed	Completely Addressed	Design Decisions Made During the Iteration
		QA-01	Introducing a separate and replicated trap receiver node ensures the comprehensive handling of all traps, even in the event of an application server failure. Additionally, by conducting trap reception on a different node, this approach reduces the processing burden on the application server.
	QA-03		To reduce the traffic coming to a web application a load balancer has been introduced during this iteration which will help the web servers to filter each request without failure and will help in detecting suspicious request.
		UC-10	The implementation of the trap receiver will help in tracking the sources of these traps and correct them.