# SOFE 3650U: Software Design & Architecture

# Final Project - ADD Iteration 2

**Date:** November 19, 2021


**Group Members:**

Foram Gandhi (100699245)

Kinjal Shah (100743551)

Rutvi Shah (100747171)

Danial Shaikh (100698628)

## Iteration 2: Identifying Structures to Support Primary Functionality

### Step 2: Establish Iteration Goal by Selecting Drivers

The goal of this iteration is to address the general architecture concern of identifying structures to support primary functionality. The architect considers CRN-3 which is the allocation of team members, as well as the following primary use cases:

- UC-1: System Availability
- UC-6: Date Selection
- UC-7: Ticket Selection
- UC-8: Seat Selection
- UC-9: Payment Options

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

The elements to be refined in this iteration are the modules in the reference architecture model from Iteration 1. The modules are located in the different layers and provide support of functionality for the system by collaborating different components associated with the modules.

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

| **Design Decision and Locations** | **Rationale and Assumptions** |
|---|---|
| Create a **Domain Model** for the application | Before starting a functional decomposition, it's necessary to create an initial Domain Model for the system, identifying relationships and major entities in the domain. There are no good alternatives and the domain model must eventually be created or it will emerge in a suboptimal fashion, this leads to an complex ad hoc architecture that's hard to understand and maintain. |
| Identify **Layers** that map to functional requirements | Each layer of the application needs to have a distinct and specific responsibility. The interactions between all the layers have to be highly constrained with only unidirectional dependencies between them.<br>One possible alternative is to consider domain objects from the start to eliminate the risk of not considering a requirement. |
| Decompose **Layers** into general and specialized **Components** | Each self-contained and coherent responsibility within a layer is categorized as a separate domain object (modules that can be developed independently).<br>These modules are "components" in this pattern. Their |

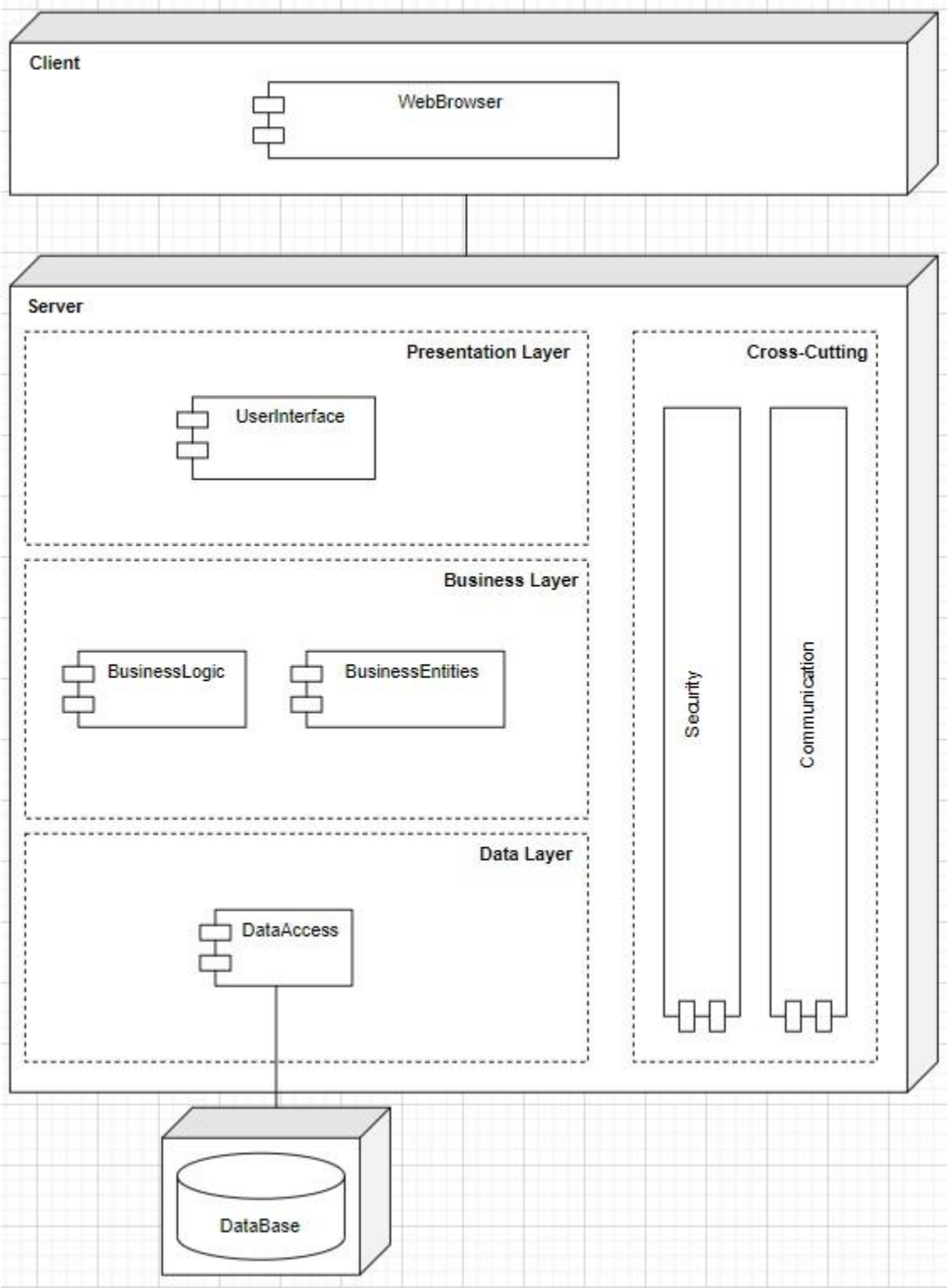
| | |
|---|---|
| | specialization is associated with the layers where they are located, such as UI Modules. |

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

| **Design Decision and Locations** | **Rationale and Assumptions** |
|---|---|
| Create initial domain model | To accelerate this phase of design, only an initial domain model is created using the participating entities in the primary use cases. |
| System use cases mapped to layers | Analyzing the system use cases to initially identify domain objects. |
| Identify layer-specific modules with an explicit interface by decomposing the domain objects across the layers | Ensure that all modules that support the functionalities are identified. |

**Step 6: Sketch Views and Record Design Decisions**

Model that supports the primary use cases.

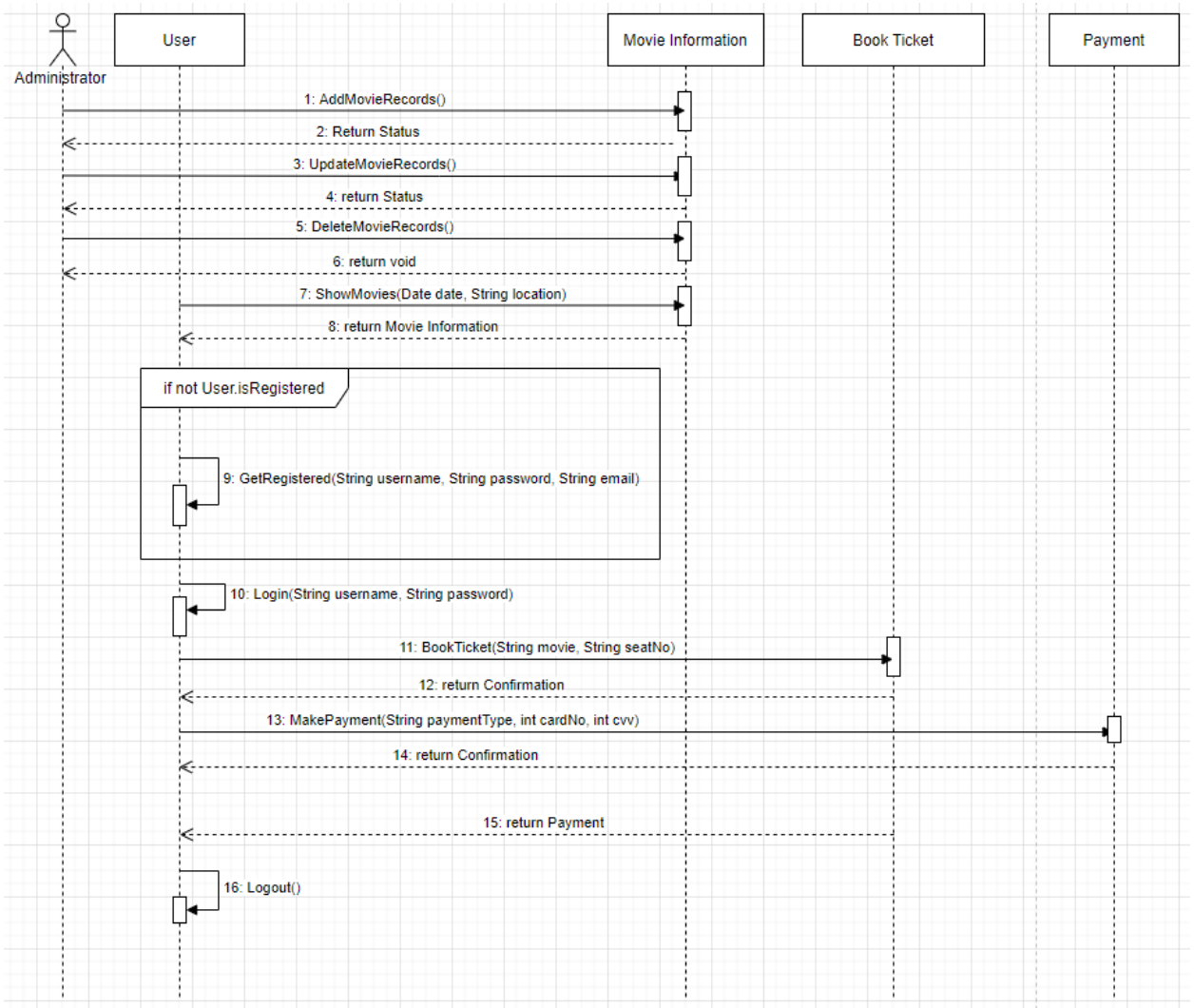


| Layer | Module | Responsibility |
|---|---|---|
| Client | WebBrowser | Movie Ticket Booking Application runs on the client's web browser. |
| Server - Presentation Layer | User Interface | Receives user interactions and presents the information to the users. It contains UI elements as buttons and text fields as well as a form in the ticket checkout page. |
| Server - Business Layer | BusinessLogic<br>- Payment<br>- Ticket Booking | Retrieves and processes application data and applies business rules on the data (such as seat booking page and checkout page). |
| Server - Business Layer | BusinessEntities<br>- User<br>- Movie Information | Represents the entities from the business domain and the associated business logic. |
| Server - Data Layer | DataAccess<br>- MySQL database<br>- OMDB API | Provides the common components needed to retrieve and store information. It uses the OMDB API for retrieval of movies and movie info, as well as retrieval from the database to verify/check login/signup info. |
| Server - Cross-Cutting Layer | Security<br>- User authentication<br>- Password encryption/decryption | Handles security aspects such as user authorization and authentication, as well as password encryption/decryption |
| Server - Cross-Cutting Layer | Communication<br>- Data layer communicates with database | Handles communication mechanisms across layers and physical tiers |
| External Data Sources | DataBase | Stores all data from the system, later accessed by the |

| | | |
|---|---|---|
| | | DataAccess module in the Data Layer (using MySQL to store databases). |

UML Sequence Diagram for UC-1

UML Sequence Diagram for UC-7




| Method Name | Description |
|---|---|
| **Element:** User | |

| | |
|---|---|
| void Login (String username, String password)<br><br>void GetRegistered (String username, String password, String email)<br><br>void Logout () | Logs users into the system allowing them access to book movies.<br><br>If the user is trying to book a ticket and is not already registered into the system they are prompted to register by entering their desired username, password, and email.<br><br>Logs out the user denying them access to book tickets. |
| **Element:** Movie Information | |
| boolean AddMovieRecords ()<br><br>boolean UpdateMovieRecords ()<br><br>boolean DeleteMovieRecords ()<br><br>String ShowMovies (Date date, String location) | The admin has authority to add, update and delete movies from the ticket booking system which returns a boolean status value indicating if their action was successful or not.<br><br>The user can request to see the movies the available movies to watch at a certain location on a certain day, these movies are returned and displayed onto the UI. |
| **Element:** Book Tickets | |
| boolean BookTicket (String movie, String seatNo) | To book a ticket the movie and seat that the user is requesting is passed to make the booking. The function returns a confirmation status if the booking was made successfully. |
| **Element:** Payment | |
| boolean MakePayment(String, paymentType, int cardNo, int cvv) | For the user to make a payment they must enter the type of payment they'd like to make the payment with plus they must also enter their card number and cvv. This function returns a confirmation status upon verifying the payment. |

UML Sequence Diagram for UC-9

**Step 7: Perform Analysis of Current Design and Review Iteration**

| Not Addressed | Partially Addressed | Completely Addressed | Design Decisions Made During the Iteration |
|---|---|---|---|
| | | UC-1 | Modules across the layers, supporting this use case have been identified. |
| | UC-6 | | Modules partially support this use case across the layers. |
| | UC-7 | | Modules partially support this use case across the layers. |
| | UC-8 | | Modules partially support this use case across the layers. |
| UC-9 | | | No relative decisions made in this iteration for the use case. |
| | | QA-2 | The elements that support the associated use cases have been identified. |
| QA-4 | | | No relative decisions made in this iteration for the quality attribute. |
| QA-5 | | | No relative decision has been made for this particular quality attribute |
| CON-1 | | | Decisions for the work efficiency of the technician have not been made yet. |
| | CON-2 | | Decisions for the communication relationship between the developer and technician have not been made yet. |
| CON-3 | | | Constraint is referenced by the architecture model although no relative decision has been made. |
| CON-6 | | | No relative decision has been made for this constraint. |
| | CRN-2 | | Technologies and reference architecture that have been considered up to this point have taken into account the knowledge |

| | | | |
|---|---|---|---|
| | | | of the developers. |
| | | CRN-3 | Modules associated with all the use cases have been identified. |