Iteration 2: Identifying Structures to Support Primary Functionality

This section presents the results of the activities that are performed in each of the steps of ADD in the second iteration of the design process. This iteration relates to the recommend game component for the game database system. This iteration expands on the more basic definitions made in the iteration 1 to make them more specific. This will aid in the drive of the implementation.

The expansion is part of the ADD method as initially the team cannot design everything at once vague definitions must be used to be built upon later to allow maneuverability in the system architecture. This iteration focuses on addressing risks the system can face during the building process and sharpening the structure of the system. The goal of iteration one has been met, so the goal of the second iteration must commence which will is to represent and explain the units of implementation.

Step 2: Establish Iteration Goal by Selecting Drivers

The goal of the second iteration is *identifying structures to support primary functionality*. By addressing these concerns, it aids in the understanding of how the Game_Knight system's functionality is allowed. This also explains how CRN-2-which is, the background knowledge of the team is essential in determining whether the system can be built or not. SQL knowledge is necessary, and python is also necessary in implementing the views for the statistical analysis function. It also allocates members to the respective position in the development team.

In the second iteration the architecture must also consider the following system's use cases:

- UC-1: Monitor user interaction
- UC-3: Display recommended game
- UC-7: Create statistical analysis of game data

Step 3: Choose one or more Elements of the system to refine

The specific elements selected to be refined is the game application function which displays and manages game recommendations. This includes the backed aspect which provides the recommendation generation.

Step 4: Choose one or more design concepts that satisfy the selected drivers

Within the second iteration, design concepts in the architectural design patterns are selected. The following table below builds on the design decisions.

Design Decisions and Location	Rationale and Assumptions

Use **Observer Design Pattern** to publish game recommendations and updates to users	There will be several cases in which multiple users should be pushed the same game recommendations based on their previous history. To better manage these cases, it can be advantageous to use an observer design pattern to better manage and more efficiently push updates to user accounts.
Introduce the **save (tactic**) in the system to monitor user interaction	The save tactic is introduced to be able to receive, process and save several events to log user interaction. Implementing a save tactic is essential in the logging of user data, and allows more advanced statistical analysis to proceed
Use of systems **hover pattern** through the website to investigate the function operability and return it as a Boolean value	The system monitors the end user's interactions in the Game_Knight website. When the end user hovers in the vicinity of an interactable service provided by the systems it will provide a visual cue. The users can expand and collapse game directories and any related information.
Create a **Domain model** for the system and identify objects	Create an initial domain model to identify the major entities in the system along with the relationship between entities. Each object must be identified in the system for later use

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

The instantiation of the architectural elements, responsibilities and definitions are explained in the table below.

Design Decisions and Location	Rationale and Assumptions
Web application to disconnect local sources of data	With a strong and stable connection, it is viable to not store any data locally. The data layer is where server communication is handled. Communication between the many client components are directed via local method calls without any other support.

System use cases are mapped to domain objects	By analyzing the system's use cases an initial identification of the domain objects can be made.

Step 6: Sketch Views and Record Design Decisions

- Figure 1.5 initial domain model for the system.
- Figure 1.6 the domain objects that are instantiated for the use case model in Section 1.2.1.
- Figure 1.7 shows a model of a module view with modules that are derived from the business objects and linked with the primary use cases.

The description for the elements provided in figure 1.7 are explained in the table below it.

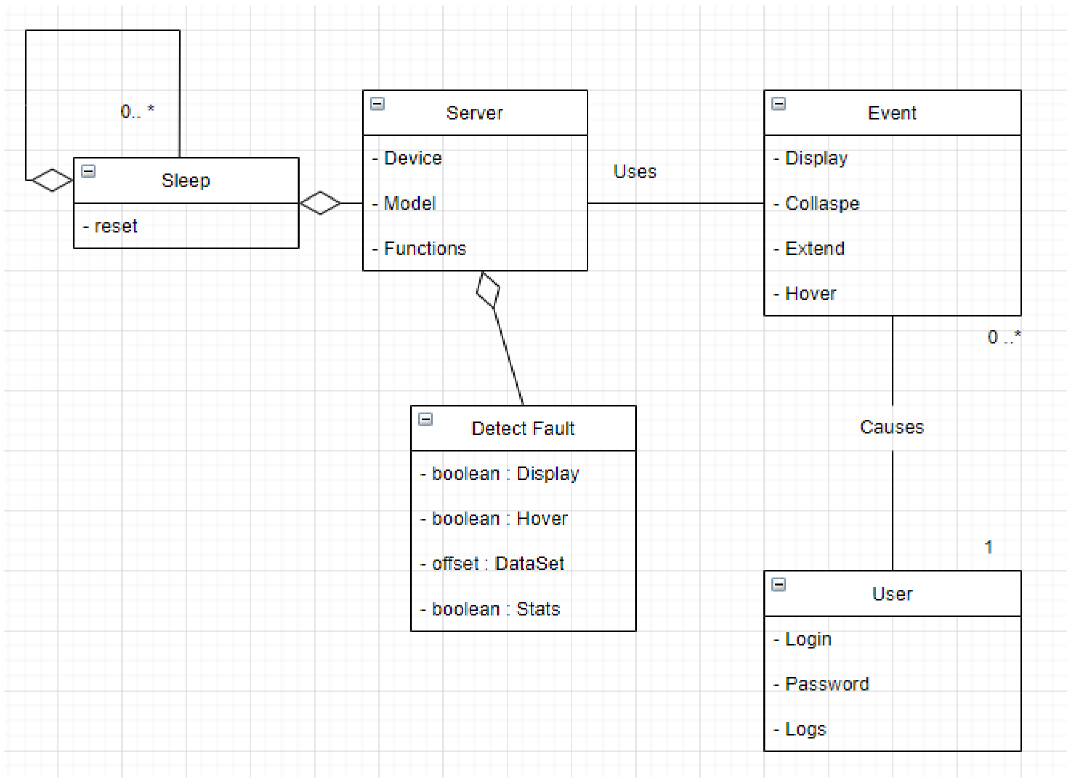

FIGURE 1.5 Initial domain model

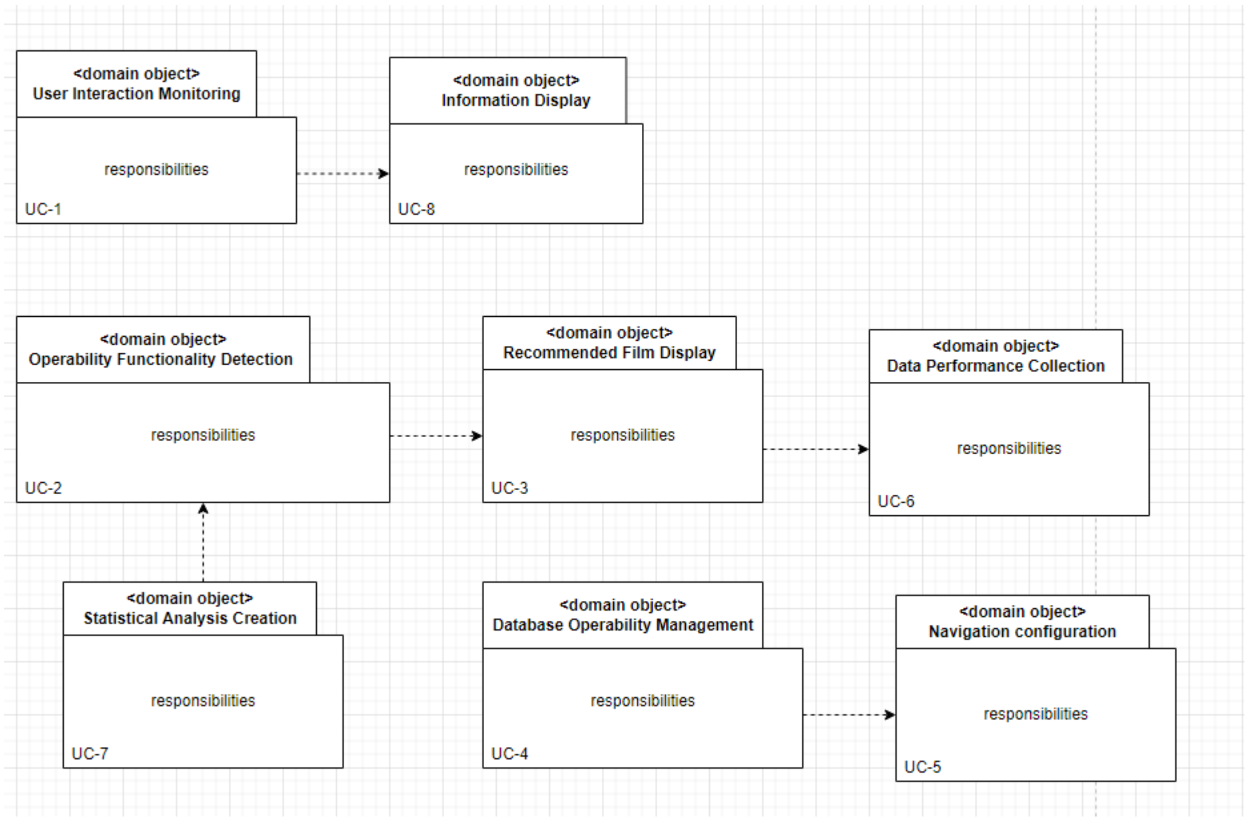

FIGURE 1.6 The domain objects linked to the use case model

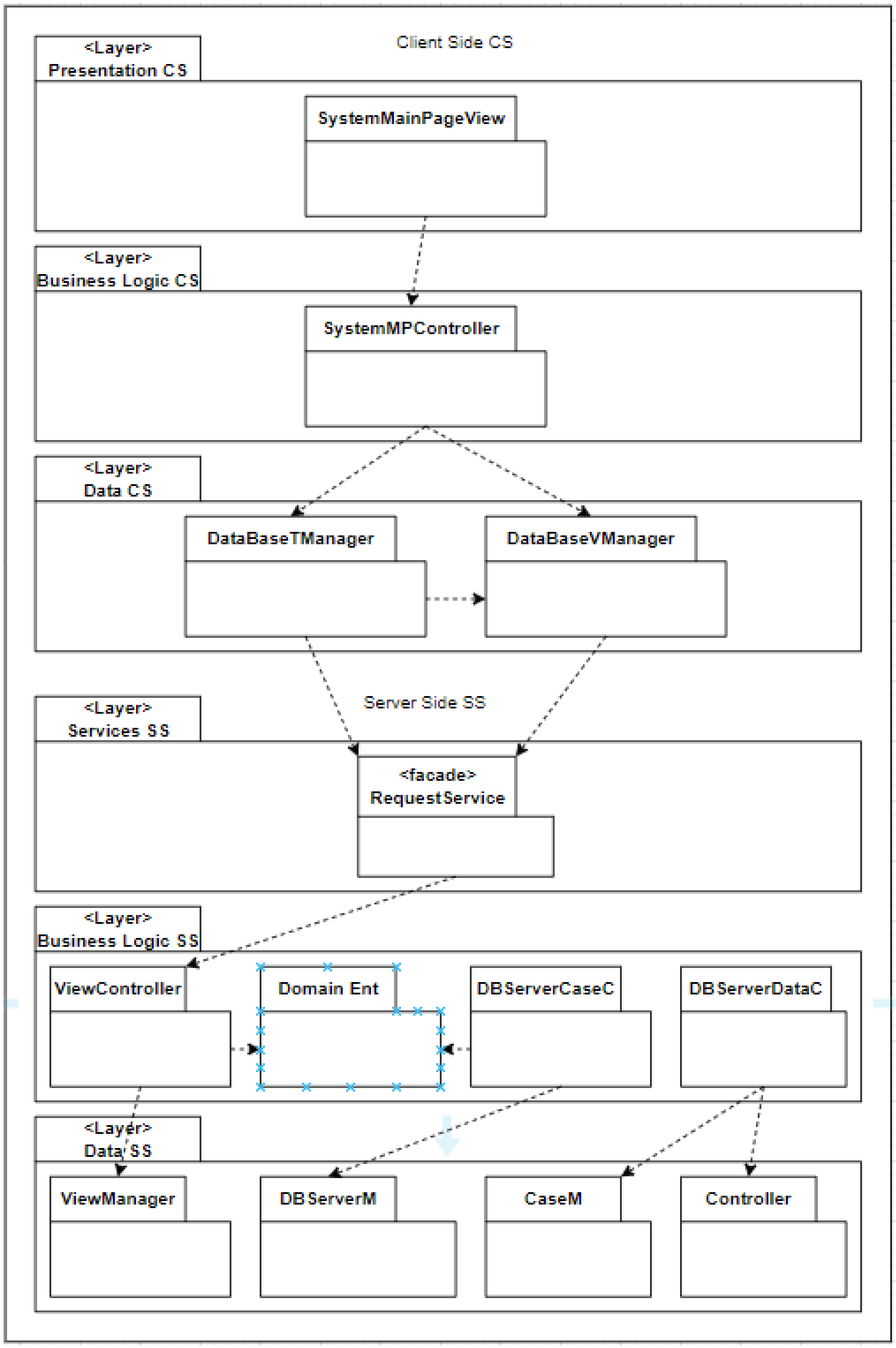

FIGURE 1.7 Modules that support the essential use cases

Element	Responsibility
SystemMainPageView	Displays the main page of the Game_Knight system represent a view of the system that inherits the compilation of the systems functions and services
SystemMPController	Responsible for controlling the interactions with the main page as well as controlling the flow of information accessible to the end user
DataBaseTManager	Database tool manager responsible for managing the functions available to the system provided by the tuples in the database
DataBaseVManager	Responsible for managing the views in the system available from the database
RequestService	Provides a façade that receives prompts from the end user
ViewController	The responsibility of this element is to control the views of the database it can provide to the systems page based on if a user selects the function
Domain Ent	Contains the server-side domain model entities
DBServerCaseC	Controls the systems different cases
DBServerDataC	Controls the flow of data from the database linked to the system allows tuples to be shown from the various tables
ViewManager	Manages the views in the system
DBServerManager	Responsible for the server-side maintenance operatable without admin interaction
CaseM	Case mapper illustrates the various cases provided from the systems’ functions
Controller	The controller for the linked server

The sequence diagram below for UC-3 was created to define interfaces in the system. Similar diagrams were created for the UC-1, and UC-7 but were not needed as they repeated interfaces, or their interfaces were deemed valid not to be represented.

UC-3: Display Recommended Game

Figure 1.8 shows the initial diagram for UC-3 (Display Recommended Game). It shows how the user views the systems web applications page after a user request a certain genre or searched a game. The SystemMainPageView on the server requests the DataBaseVManager on the server to retrieve tuples that fit the prompt of the user. The CaseM then displays the view on the user's page.

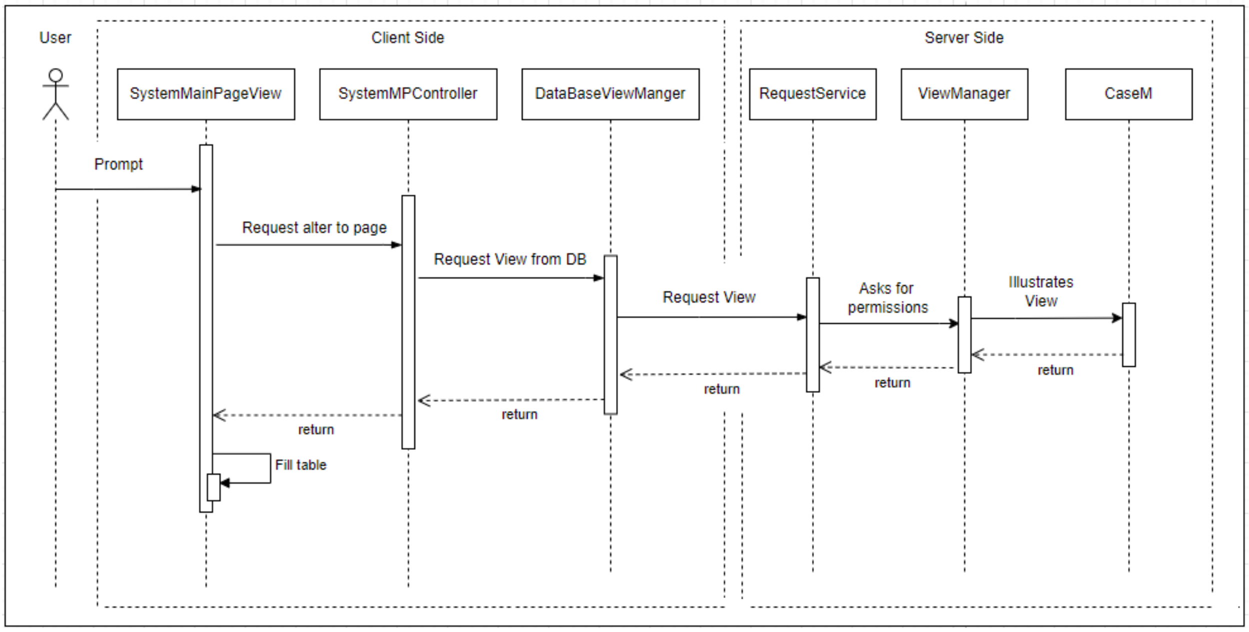

FIGURE 1.8 Sequence Diagram for UC-3 Display Recommended Game

Method Name	**Description**
Element: SystemMainPageView	
Boolean functcall ()	Provides a function that test whether one of the systems services have been prompted by the user
Hover getHoverElement ()	Gets the service the user hovered then clicked
Element: SystemMPController	
Response allow ()	Allows the page to change
Block ()	Blocks change
Element: DataBaseViewManager	
Show_View ()	This method responsibility is to manage views within the database server
Element: RequestService	
Response sendPrompt (Request x)	This method request various functions from the Game_Knight system.
Element: ViewController	
Region requestView ()	Request the view. This method returns the view the end user wants to display on the main page of the system allows the expansion and the collapse of the view
Element: CaseM	
Region retrieve (VARCHAR (45) tuple)	Displays the tuples associated with the view the user has requested

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

In the table below the design decisions made during the second iteration are address as being, not addressed, partially addressed, or completely addressed. Design decisions completely addressed in the previous iteration are not referred in the table below.

Not Addressed	Partially Addressed	Completely Addressed	Design Decisions 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-7	Modules across the layers and preliminary interfaces to support this use case have been identified.
	QA-1		The Availability of the system would be 24/7 as it is a website
	QA-2		The modifiability of the system should be flexible as many new functions will be added when the database is extended
	QA-3		The performance of the system is crucial functions must return a value whether it completes the operation or it does not
	QA-4		The security of the website is not a priority as the user is saving their information locally
QA-5			No Relevant decisions made
	QA-6		Data logging using the save tactic will aid in the usability attribute of the system.

CON-2	No Relevant decisions made
CON-3	No Relevant decisions made
CON-4	Use of the save tactic to manage data logging will aid in this constraint as it is required in the greater implementation.
CRN-1	
CRN-4	No Relevant decisions made