ADD Step 1: Review Inputs

<table>
<tr><th>Category</th><th>Details</th></tr>
<tr><td>Design Process</td><td>This is a greenfield system from a mature domain. The purpose is to produce a sufficiently detailed design to support the construction of the system.</td></tr>
<tr><td>Primary Functional Requirements</td><td>From the use cases presented from the first deliverable, the primary ones were determined to be:
UC-1 Monitor System Status: Essential Component of CMS
UC-5 Course Portal Display: Essential Component of CMS
UC-6 User Profile Utilities: Essential Component of CMS
UC-7 Manage Database: Essential Component of CMS
UC-10 User Login: Essential Component of CMS
UC-11 User Management: Essential Component of CMS</td></tr>
<tr><td>Quality Attribute Scenarios</td><td><table><tr><th>Scenario ID</th><th>Importance to Stakeholders</th><th>Difficulty of Implementation according to architect</th></tr><tr><td>QA-1</td><td>High</td><td>High</td></tr><tr><td>QA-2</td><td>Medium</td><td>Medium</td></tr><tr><td>QA-3</td><td>High</td><td>High</td></tr><tr><td>QA-4</td><td>High</td><td>High</td></tr><tr><td>QA-5</td><td>High</td><td>Medium</td></tr><tr><td>QA-6</td><td>High</td><td>Medium</td></tr></table>Only QA-1, QA-3, QA-4, QA-5, QA-6 are selected as drivers</td></tr>
<tr><td>Constraints</td><td>All the constraints discussed in the first deliverable are</td></tr>
</table>

	included as drivers.
Architectural Concerns	All the architectural concerns discussed in first deliverable are included as drivers.

Iteration 2: Identifying Structures to Support Primary Functionality

In the second iteration of the CMS System, the functionality of iteration 1 is revised to make more detailed decisions that will drive its implementation.

Step 2: Establish Iteration Goal by Selecting Drivers

The goal of iteration 2 is to address the general architectural concern of *identifying structures to support primary functionality*. The architect considers the systems primary use cases:

- UC-1: Monitor System Status
- UC-3: Detect Faults
- UC-9: Performance Data Collection

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

The elements that will be refined in this iteration are modules located in different layers of the three tier web application, especially in the database and business logic layer. In general, the support of functionality in this system requires the collaboration of components associated with modules that are located in the different layers.

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

In this iteration, several design concepts—in this case, architectural design patterns—are selected from the book Pattern Oriented Software Architecture, Volume 4. The following table summarizes the design decisions.

Design Decisions and Location	**Rationale and Assumption**
Create a **Domain Model** for the application	Before starting a functional decomposition, it

	is imperative to create an initial domain model for the system. This identifies major entities in domain, along with their relationships. There are no good alternatives. A domain model must eventually be created, or it will appear in sub/non-optimal fashion, leading to an ad hoc architecture that is hard to understand and maintain.
Identify **Domain Objects** that map to functional requirements	Each distinct functional element of application needs to be encapsulated in self-contained building block which is a domain object. Another possibility is to not consider domain objects but directly decompose the layers into modules. However this will increase the risk of not considering a requirement.
Decompose **Domain Objects** into general and specialized **Components**	Domain objects represent complete sets of functionality. This functionality however is supported by finer-grain elements located in within the layers. The “components” in this pattern are called modules.

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 entities that participate in the primary uses cases need to be identified and modeled but only an initial domain model is created. This is to accelerate this phase of the design.
Map the system use cases to domain objects	An initial identification of domain objects can be made by analyzing the system’s use cases. To address CRN-3, we identify domain objects for all use cases presented in

	Deliverable 1.
Decompose the domain objects across the layers to identify layer-specific modules with an explicit interface	This technique ensures that modules that support all the functionalities are identified. The architect will perform this task just for the primary use cases. This will allow another team member to identify the rest of the modules, thereby allocation work among team members. After establishing a set of modules, the architect realizes the need to test these modules. So a new architectural concern is identified. ● CRN-4: Need to test a majority of the modules Only a majority of the modules are covered by CRN-4 because the modules that implement the UI is difficult to test independently.

Step 6: Sketch views and record design decisions

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

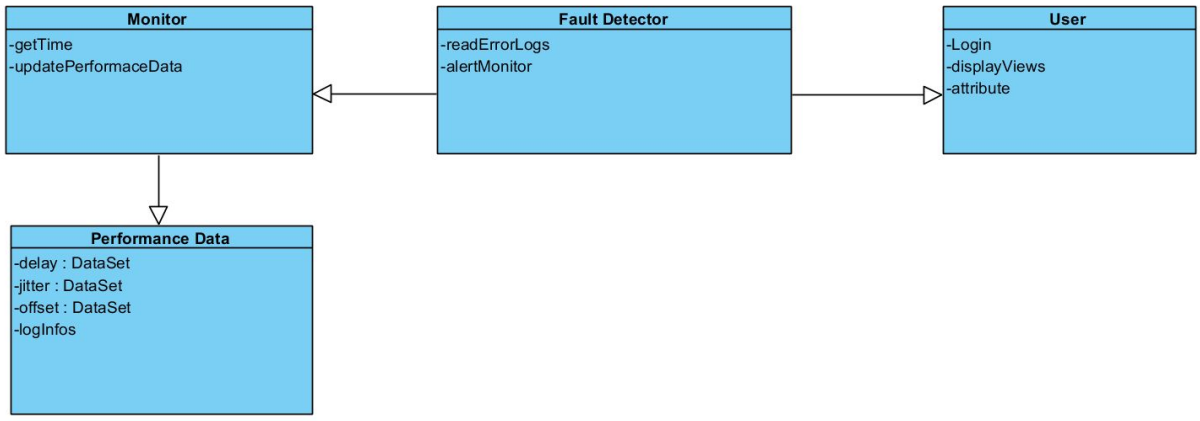

Figure - Initial Domain Model

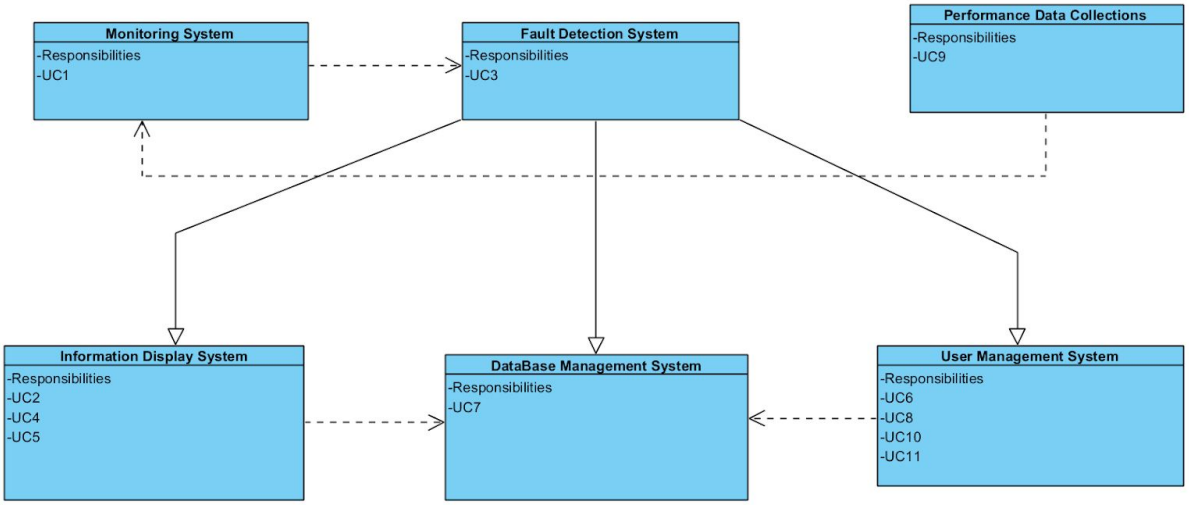

Figure - Domain objects associated with the use cases

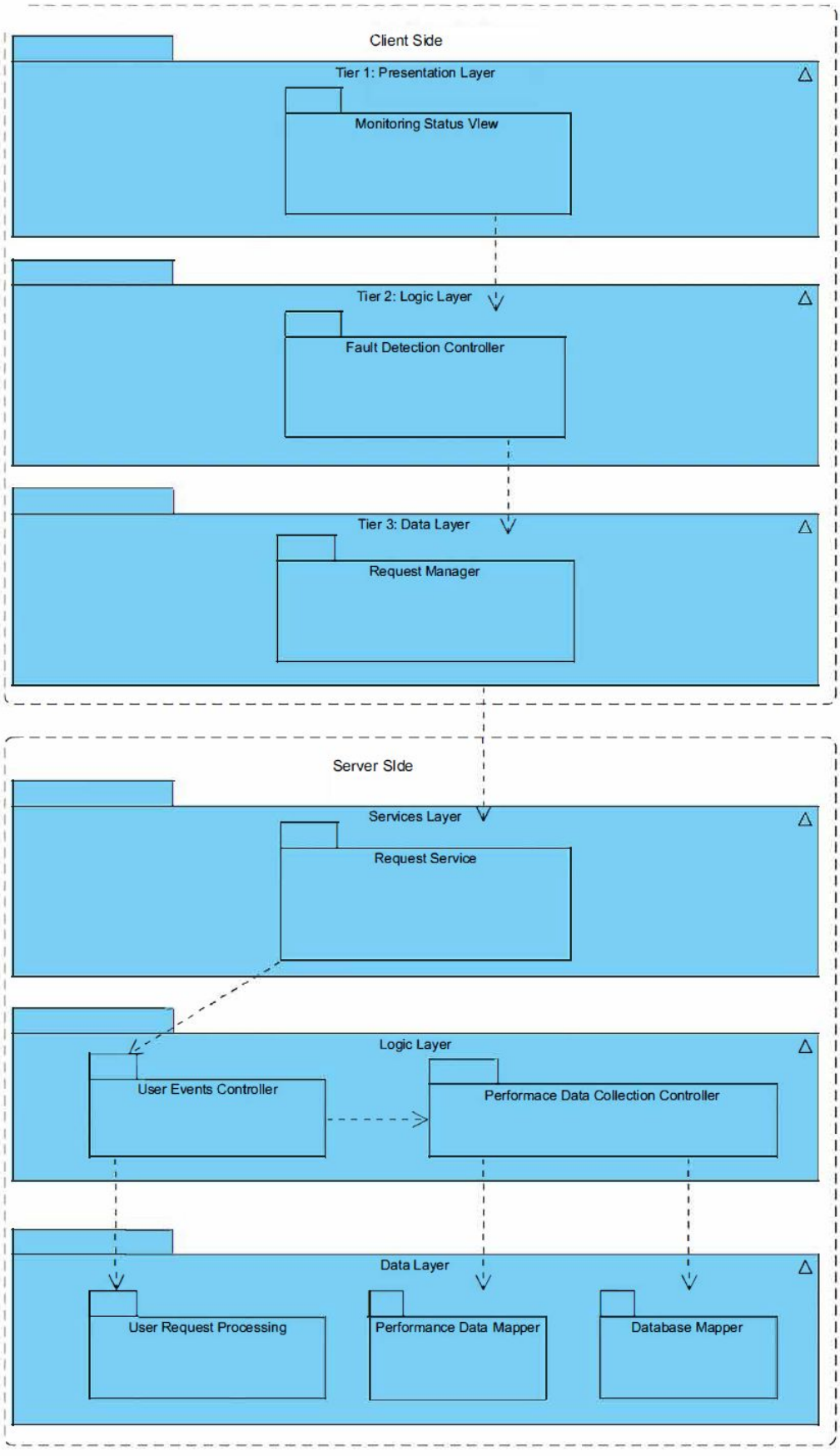
Client Side
Tier 1: Presentation Layer
Monitoring Status View
Tier 2: Logic Layer
Fault Detection Controller
Tier 3: Data Layer
Request Manager
Server SIde
Services Layer
Request Service
Logic Layer
User Events Controller
Performace Data Collection Controller
Data Layer
User Request Processing
Performance Data Mapper
Database Mapper

Figure - Modules that support the primary use cases

Element	**Responsibility**
Monitoring Status View	Responsible for letting users know all systems are properly functional.
Fault Detection Controller	All errors are collected and analyzed to see whether the fault can lead to a total system failure.
Request Manager	Responsible for communication with the server side logic
Request Service	Receives requests from the users
User Events Controller	Responsible for actions taken by the user inside the system whether it is checking grades or registering course etc.
Performance Data Collection Controller	Responsibility includes controls which data is crucial for logging and is logged.
User Request Processing	Responsible for processing user requests made in the course management system.
Performance Data Mapper	Responsible for the persistence operations related to the events.
Database Mapper	Responsible for the persistence operations related to the events.

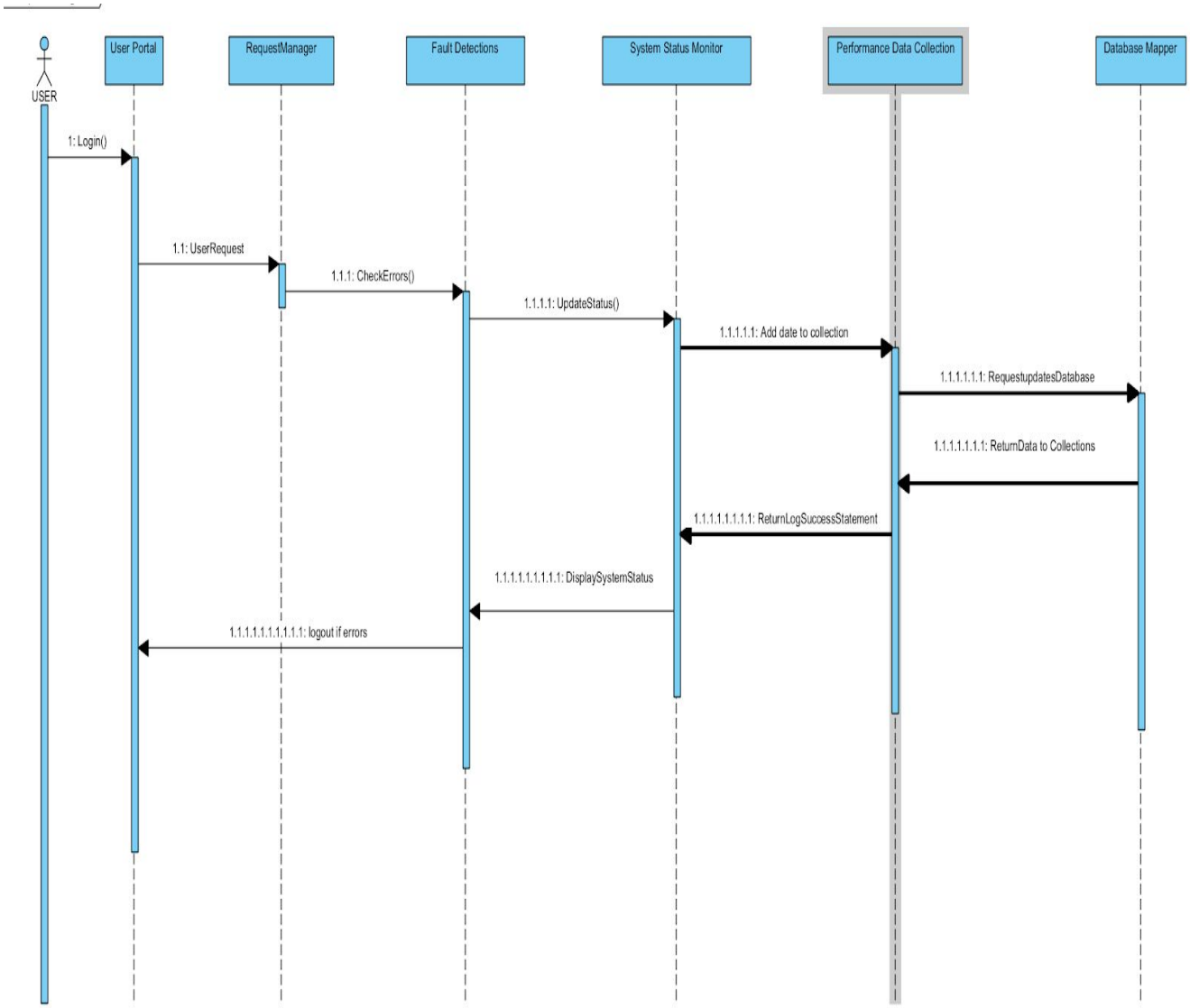

Figure - Sequence Diagram for USE CASE 1,3 & 9

The figure above shows the sequence diagram for the USE CASE 1,3 & 9. When any user is logged into the system and tries to perform any kind of request, the system checks if there are any faults associated with that request or the system in general. The faults detector detects if the faults are minor or major to cause a total system failure. The data is collected by the performance data collector and the user requests are pulled/updated in the database. If there is a fault in the system, the user is forced to logout in order to not create errors in the system.

Name	**Description**
User Portal	The place where the user can login to view the necessary pages.

Request Manager	Manages the request received by the client in order to process the request for client to see.
Fault Detectors	Checks for errors in the system. Checks for errors from the start of the system where the client has to login in.
login()	Method the client uses to login to the CMS
CheckErrors()	Method the Fault Detectors use to check for errors in order to avoid a total system failure.
updateStatus()	Used by the monitoring system to update the system status based on the errors found.
System Status Monitor	CMS System status monitor determines whether the systems remain up or down.
Database Mapper	Allows for communication between the requests made by the user and the database.

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

Established	Not Established	Partly Established	Design Decisions made during Iteration
UC- 1			Interfaces and modules have been identified in relation to this use case
UC - 3			Interfaces and modules have been identified in relation to this use case
UC - 9			Interfaces and modules have been identified in relation to this use case
		QA - 1	The performance of the system is constantly scanned

			and secured to prevent any malfunctions, but may need improvement
QA- 3			If a failure of malfunction does occur, the system has been designed to shutdown and restart.
		QA- 4	Data is logged at certain intervals to prevent data loss if performance is suffering due to load.
	QA- 5		Design decisions have not been made in this iteration
	QA- 6		Design decisions have not been made in this iteration
	CON -1		Design decisions have not been made in this iteration
CON -2			Servers have been established in order to use system databases for the campus
		CON -3	Design decisions have been made, ,but need expansion
CON -4			Modules for privacy of grades in classes have been established.
CON-5			Modules for static and dynamic view of the course have been

			established.
	CON-6		Design decisions have not been made in this iteration
	CON-7		Design decisions have not been made in this iteration
CON-8			Final grade statistics are given to the administrators through the established module.
CON-9			System has been established with an administration database
CON-10			System has been set in place to verify qualifications of students
		CRN -1	Architecture has been established and needs more implementation
CRN -2			Programming language is chosen by the average familiarity with the language in accordance of the team
CRN -3			Jobs have been divided.