Iteration 2: Identifying Structures to Support Primary Functionality

Goal

To identify the elements that support the primary functionality of the design. The functionality is typically supported by elements that are spread throughout the layers of the system, the elements are different layers that were identified in the first iteration. (1st - interfaces are not defined yet, draw sketches, keep records...)

In this Iteration we will present the results of the activities performed in each step of ADD in 2nd iteration of design process for CMS System. Translate vague concepts to specific descriptions of functionality used in 1st iteration to be converted to more detailed decisions that will drive implementation. Goal is to reason about the units of implementation, which affect team formation, interfaces, and means of which development tasks maybe distributed/outsourced.

Step 2: Establish Iteration Goal by Selecting Drivers

To accomplish our iteration of identifying structures that support our primary functions, we will consider the system's primary use cases

- ➢ Use Case 1
- ➢ Use Case 2
- ➢ Use Case 5

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

Based on the our usage of greenfield system iteration, elements to be refined in this iteration are the elements that support of functionality in our systems requires mixture of different components that are connected with the modules that are in different layers of our design.

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

Design Decisions and Location	**Rationale and Assumptions**
Create Domain Model	In order to do our decomposition based on iteration 1, it is vital to make an initial domain model for our CMS. We must then identify the major entities that we will use in this domain and how they relate to one another. This is the most straightforward solution and it is also the most optimal. This domain model will hold our conceptual model of the domain that will incorporate both the functionality and data. We will incline towards our

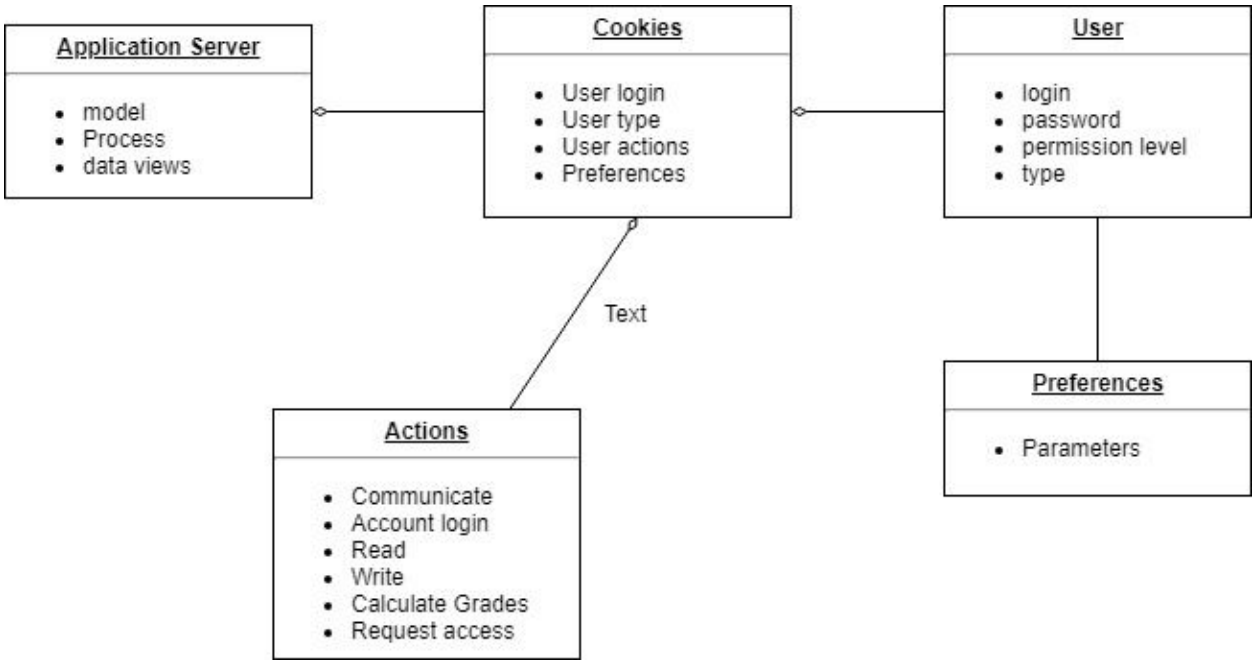
	selected drivers above. As well as QA-5 to focus on our security system.
Identify Domain Objects that are mappable	The functional elements of our application requires to be preserved in a domain object. This is the most optimal solution. Our domain objects will be composed of the application entities and the workflow of the system.
Decompose into general and specialized Components	The Domain Objects that we selected will represent their each individual sets of functionality. These objects is supported by more specific elements located in the sub layers. Modules are our components in our selected pattern. Specialized modules or components are all related with their corresponding layers.
Use Hibernate	We are using Hibernate,, which maps our objects that will support our functionality.

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

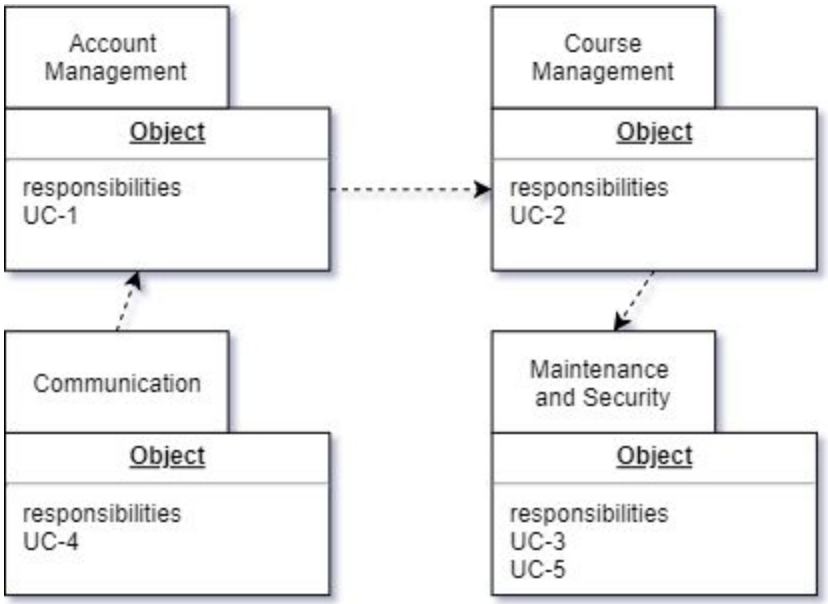
Design Decisions and Location	**Rationale and Assumptions**
Create an initial domain model	In order to accelerate the implementation of our design, we must create an initial domain model that has our selected drivers of use cases located in our Iteration 0.
Mapping	We will utilize our use of the Use Case Diagram model to map our system.
Decompose Objects into specific interfaces	CRN-1 will need to be decomposed into interfaces as the architecture will be our basis on the rest of the upcoming design.
Connect components	After decomposing CRN-1, the broken down components will allow us to use different aspects that will support our structure.

Step 6: Sketch Views and Record Design Decisions

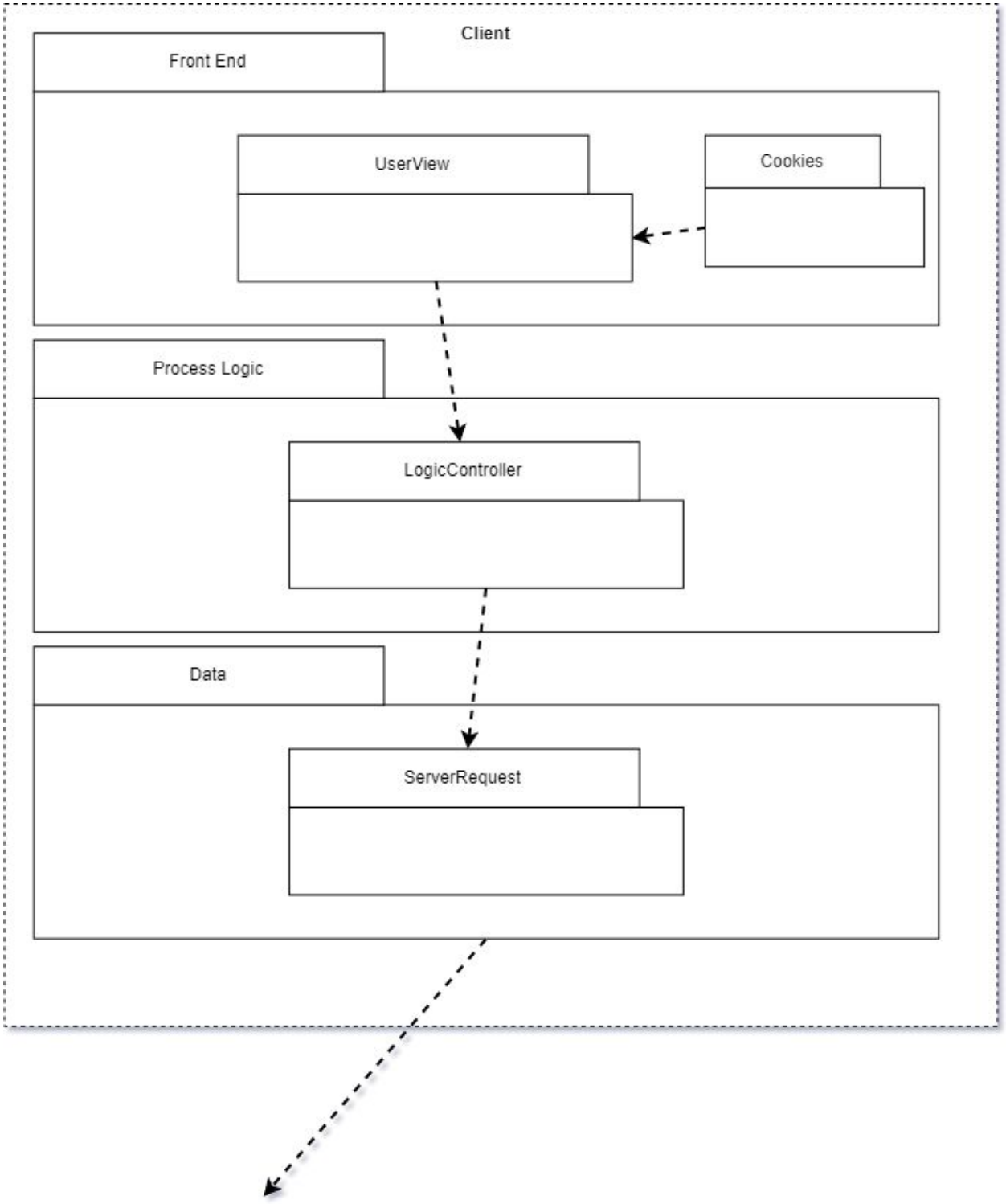
Initial Domain Model

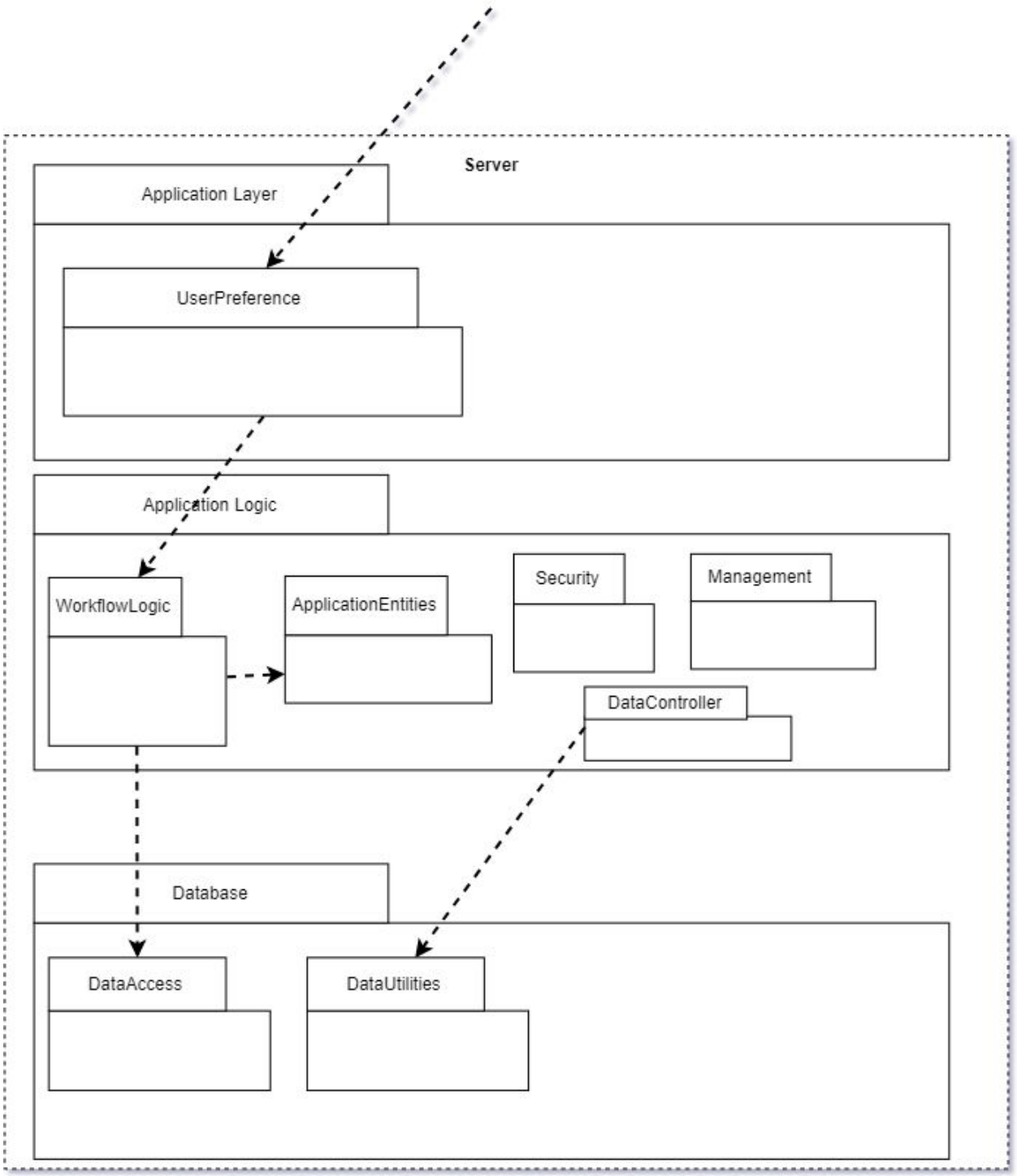


Domain Objects for Use Cases



Module View


Client
Front End
UserView
Cookies
Process Logic
LogicController
Data
ServerRequest


Server
Application Layer
UserPreference
Application Logic
WorkflowLogic
ApplicationEntities
Security
Management
DataController
Database
DataAccess
DataUtilities

Element	Responsibility
UserView	UI Process Logic including UI interface and Database Views HTML and CSS designed UI
Cookies	Save page presets and user details for statistical analysis.
LogicController	Query handling, user accounts management, course management, generates views
ServerRequest	Simple logic and querying
UserPreference	Logic that will set the User Preferences based on what ServerRequest has sent
WorkflowLogic	Works with Application Logic
ApplicationEntities	Database views, objects, users, etc.
Security	Cross-cutting security layer checking access permissions and privileges, Firewall
Management	Administrator and Maintenance-level user tools and features
DataController	Contains Data Logic that enables DataAccess
DataAccess	Contains data collection and storage
DataUtilities	Utilities that database can use to assist in different database functions

Step 7: Perform Analysis of Current Design and Review Iteration

Not Addressed	Partially Addressed	Addressed	Designs Decisions Made During the Iteration
		UC-1	Account Management features are facilitated by our system structure, and we also specified the stated implementation.
		UC-2	Course Management features are facilitated by our system structure, and we also specified the stated implementation.
	UC-5		User Security and Permissions requirements were identified in step 6
		QA-2	Availability was a central focus for this iteration, with Initial domain model reference architecture being leveraged to better exhibit and application with high availability.
		QA-5	The domain model allows security at every tier of the system. Specific implementation were also conceptualized in step 6.
		QA-6	Extensibility and interoperability have been primarily focused on for this iteration, and the domain model have been picked specifically to allow better future modification and upgrading of the system. Further improvement was done in this Iteration.
	CON-1		No relevant decisions made
		CON-2	No relevant decisions made
	CON-3		Some function of relational database were mapped
		CRN-1	No relevant decisions made
CRN-2			No relevant decisions made
		CRN-3	EJS (Embedded Javascript)-complimented HTML and CSS were chosen for our front-end presentation, both of which are highly compatible with our team's familiarity with Javascript. Further improvements were done.