Iteration 2:

Step 2:

Primary use cases:

1. UC-1
2. UC-3

Step 4:

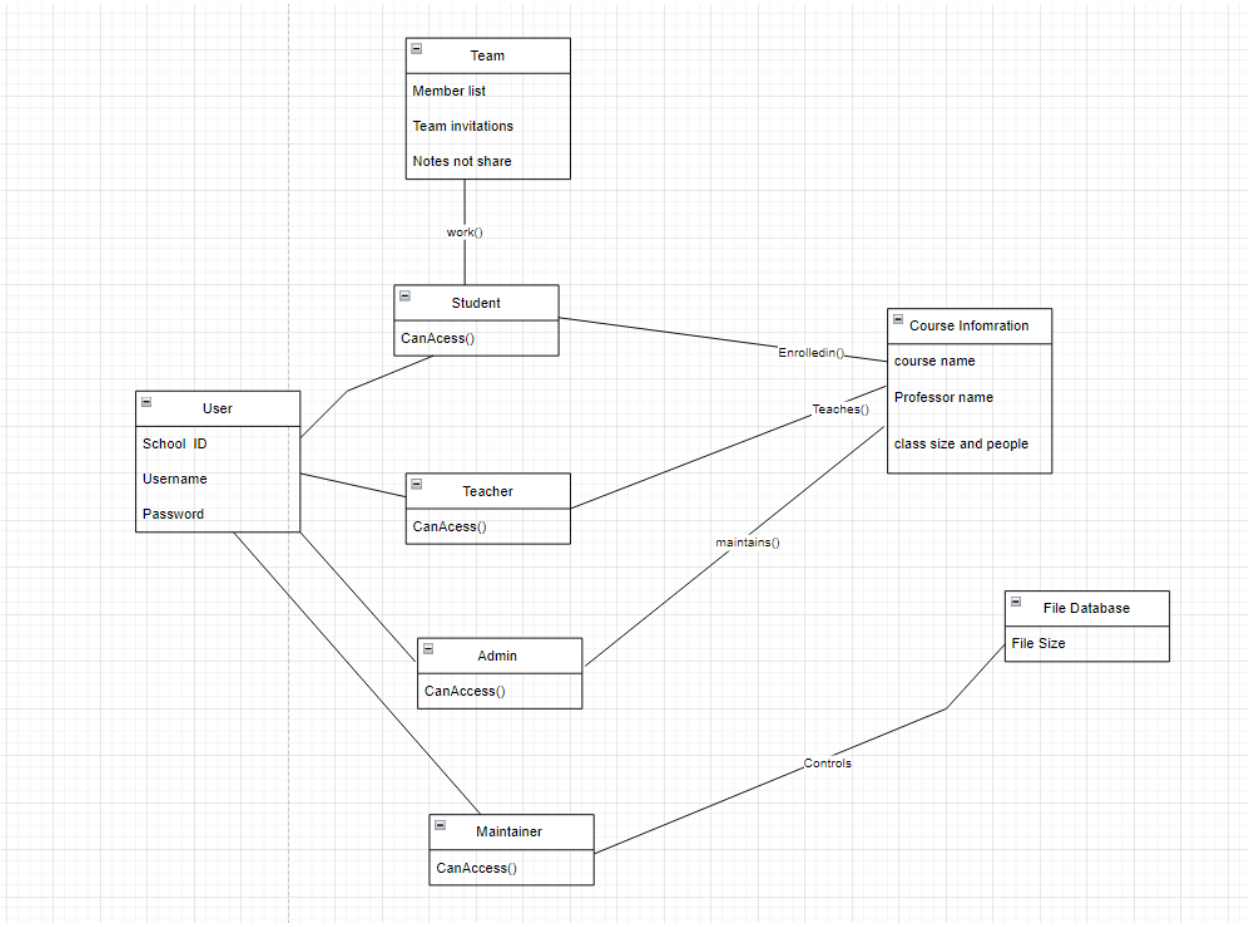
Design decisions and location	Rationale and Assumptions
Create a Domain model	We decided to create a domain model to know the major entities, relationships and how the system could interact.
Identify Domain objects that map to functional requirements.	This is used to find the relation between the entities, so that we can find the relationships b for our CMS system.
Use of Java Framework	We decided to use Java language to implement our system, because it has packages that we can use and would be helpful and our team members are more comfortable with Java.

Step 5:

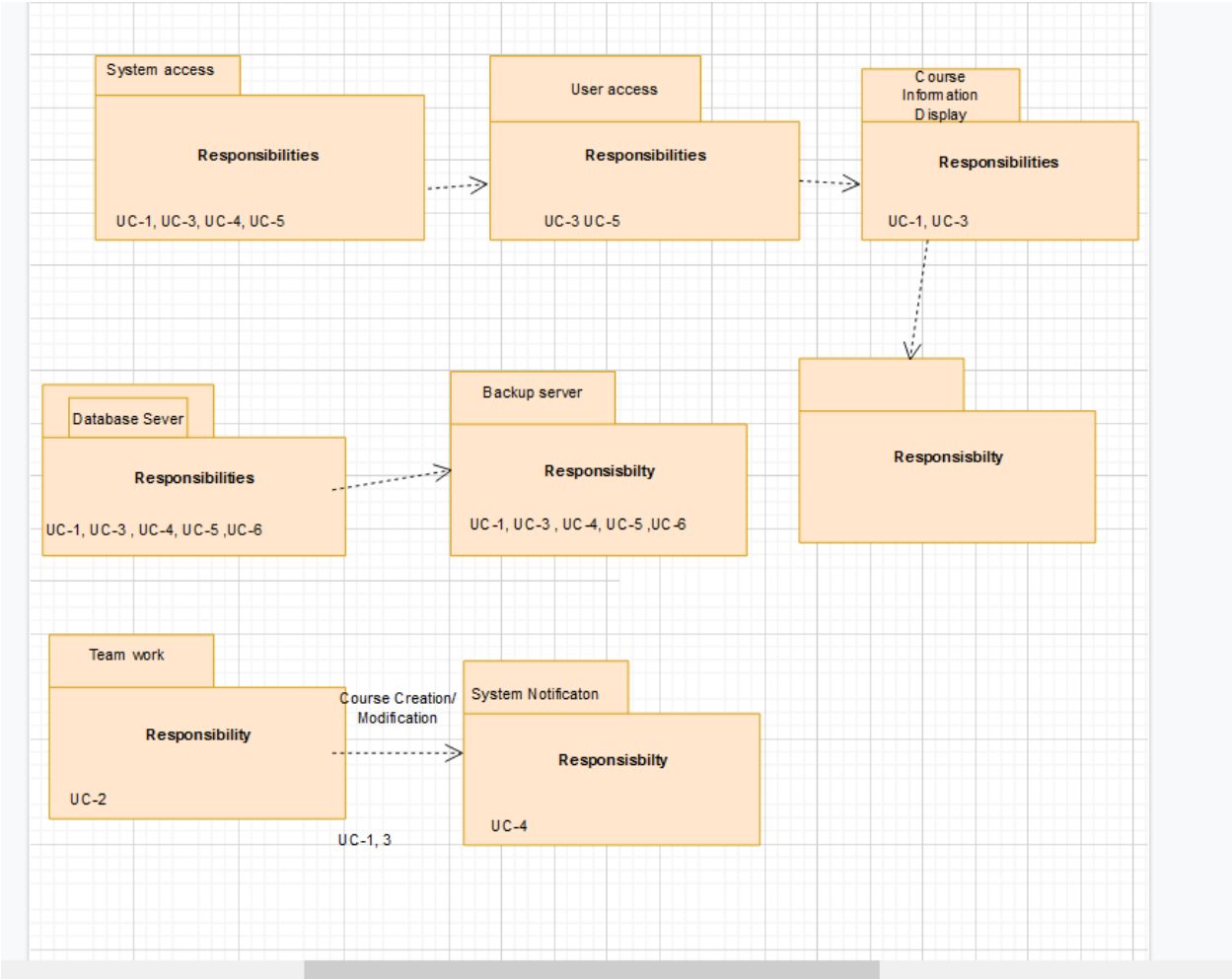
Design decisions and location	Rationale

Create a Domain model	We decided to create a domain model to know the major entities, relationships and how the system could interact.
Map the system use cases to the domain object.	We decide to analyze the use cases to identify the domain objects that can be made based on those use cases.
Decompose the domain objects across the layers to identify layer specific modules with an explicit interface.	These identities are supporting modules. This is only applied to the primary use cases. Once modules are identified, work can be split up.

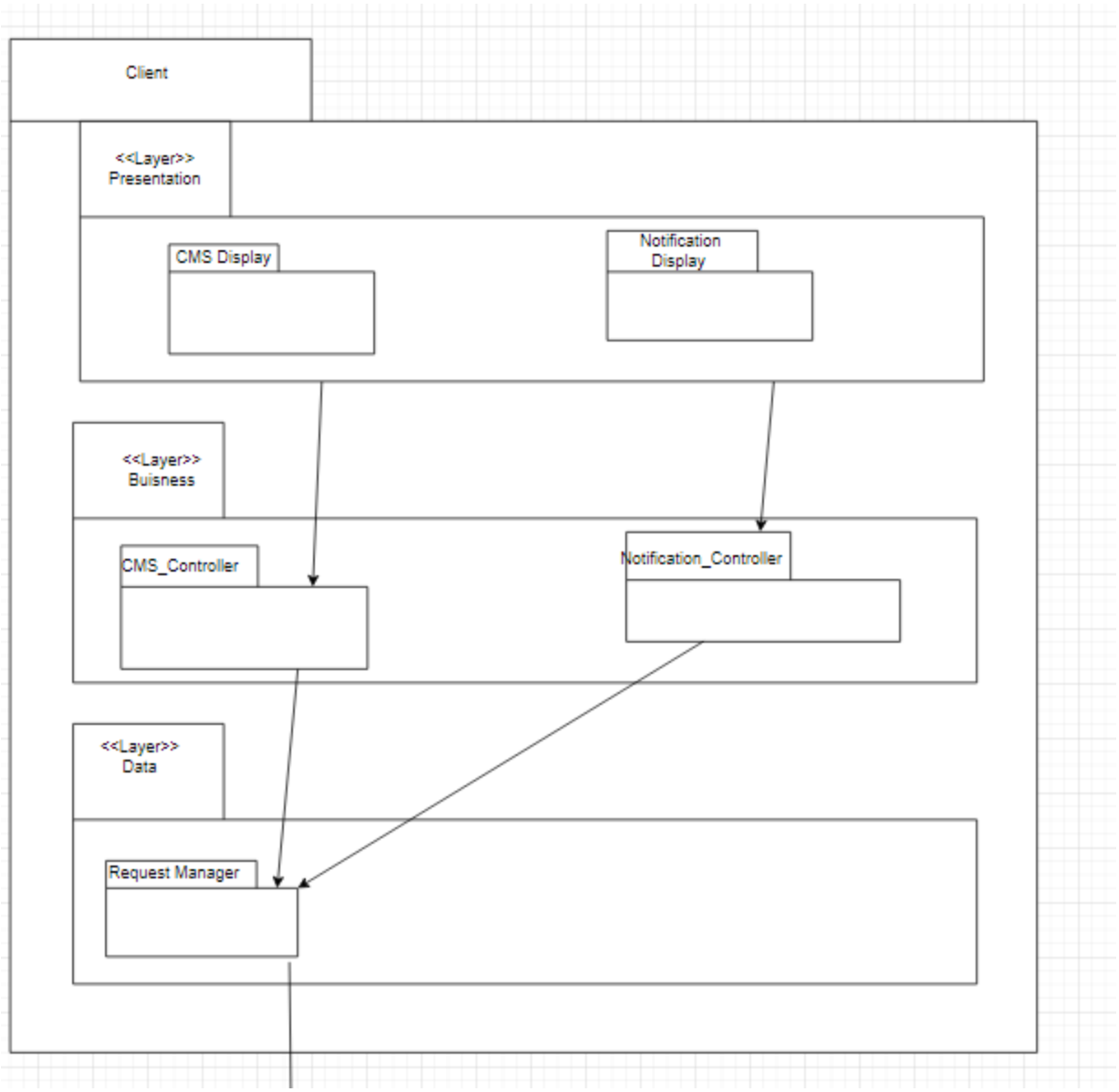
Step 6:

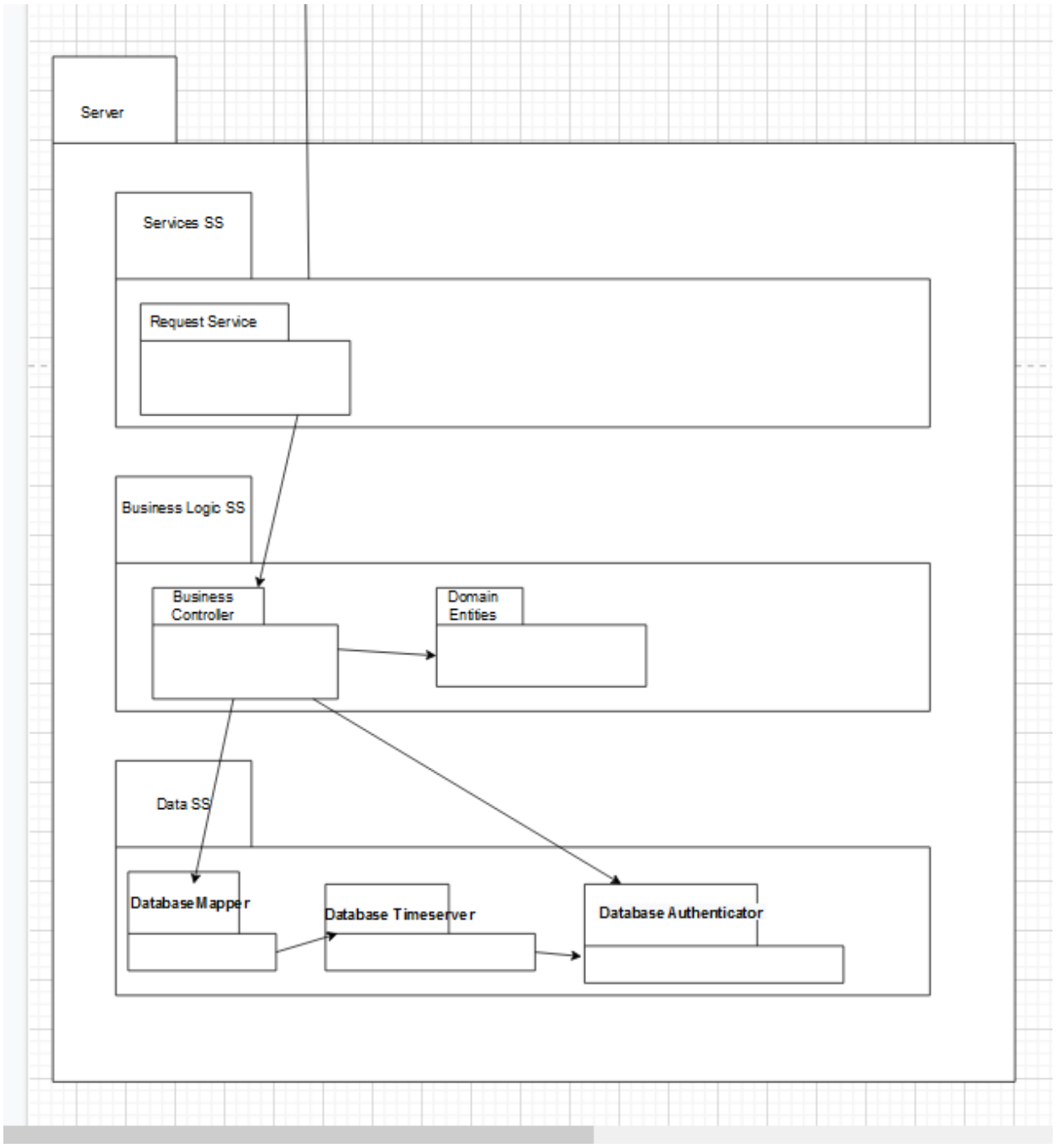


Initial Domain Model



Domain Objects associated with the use case model


Client
<<Layer>>
Presentation
CMS Display
Notification
Display
<<Layer>>
Buisness
CMS_Controller
Notification_Controller
<<Layer>>
Data
Request Manager

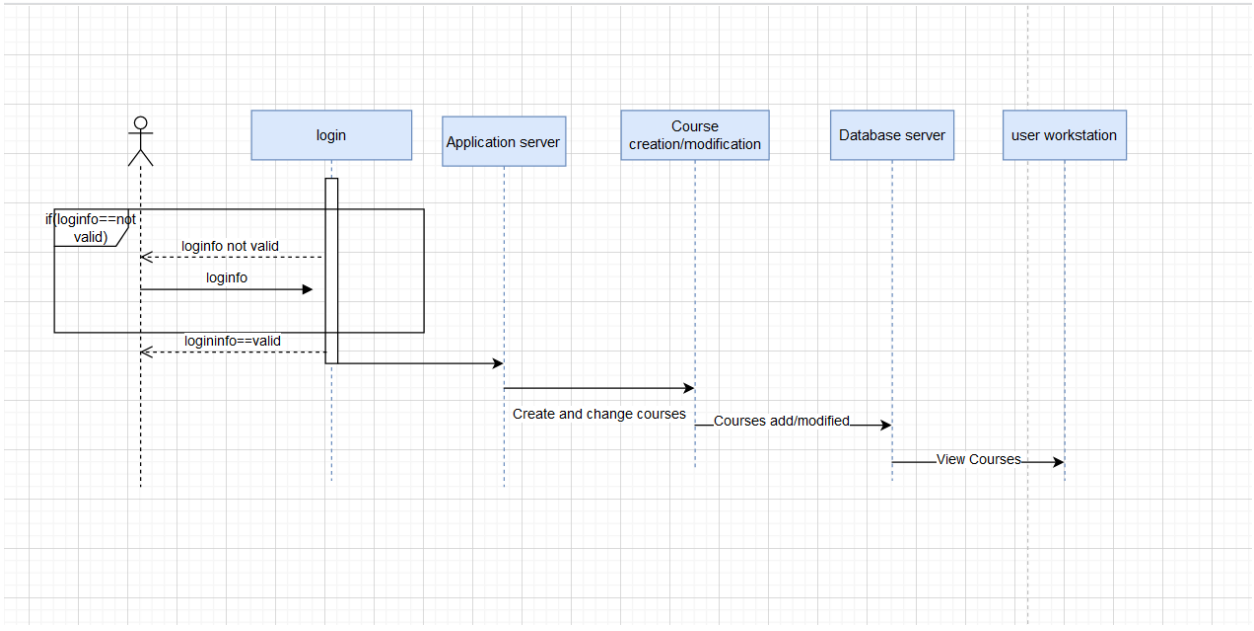


Modules that support the primary use cases

Element	Responsibility
CMS Display	Displays all the information related to the courses and all the events and all the information that the user wants to know.
Notification Display	Displays all the notifications from the system or if there is a downtime due to maintenance
CMS controller	Use to provide information to CMS display to display
Notification Controller	Used to provide information to Notification displays to display notifications.
Request Manager	Use to communicate with server side
Request service	Used to receive services from the client.
Business controller	It is used to provide Business logics
Domain entities	It contains entities from the domain model
Database Mapper	Responsible for the persistent operations related to Database

Database Time server	automatically Logouts out the user and saves the information.
Database authenticator	Authenticates the user information like login information.

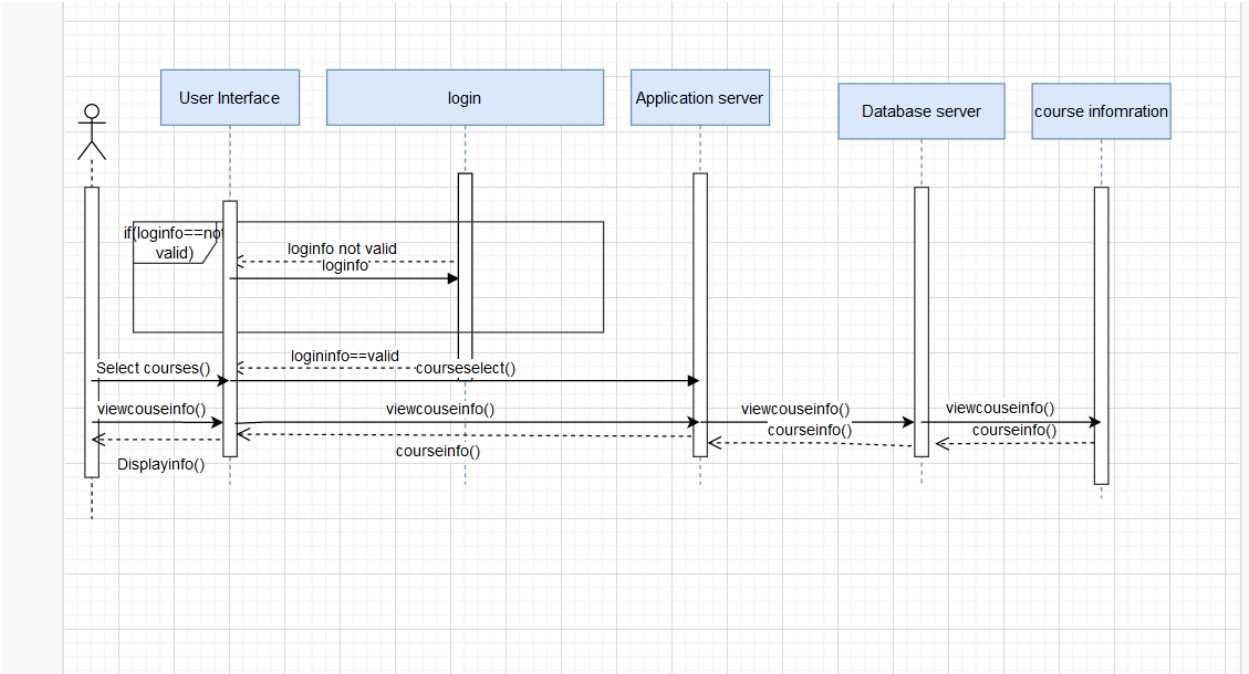
The following sequence diagram shows the steps that the lecturer takes whenever they want to do some course creation or modification. They first have to log into the application server, where it will look if they get their login information wrong. Once they log in, they are able to access the pages that handle course creation and modification. When a course is created or altered, it is changed in the database server. Once it is changed in the database, the user is able to access it from the user workstation.



Sequence Diagram for use case UC-1

Method Name	Description
logininfo()	**Just send the login information from the user to the login system to authenticate.**
Element: Login	
	It just verifies the login information of the userand if the user login information is not correct it asks the user again for the information.
Element: Application Server	
	Allows the user to have access to the website and all the features
Element: Course Creation/Modification	
Create and change courses	It's just to show the courses and lecture can modify those and also create.
Element: Database Server	
Course add/modified	Contains all the informations about students and courses
Element: User Workstation	
View Courses	Acts as the user interface and Device

The following sequence diagram is for use case 3. It shows how all users could access course information. The user would log into the user interface and validate his login information. If his information is wrong, it would loop until it is correct. Once the user logs in, they can access the application server. The application server would get the course information from the database and display it to the user.



Sequence Diagram for use case UC-3

Method Name	Description
SelectCourse()	**Just notify system what courses user had select**
Element: User Interface	
courseselect() Viewcourseinfo()	It controls how user interacts with the system
Element: Login	
	It just verifies the login information of the userand if the user login information is not correct it asks the user again for the information.
Element: Application Server	
	Allows the user to have access to the website and all the features.
Element: Database Server	
ViewCourseInfo() courseInfo()	Contains all the informations about students and courses

Element: Course Information	
ViewCourseInfo() courseInfo()	Send the course information to display.

Step 7:

Not Addressed	Partially Addressed	Completely Addressed	Design Decisions made during the iteration
		UC-1	The modules and architecture pattern to support this use case has been identified.
		UC-3	The modules and architecture pattern to support this use case has been identified.
		UC-5	The modules and architecture pattern to support this use case has been identified.
		QA-1	The website has high accessibility and is able to be accessed by blind users,so this decision has been met.
	QA-3		The website is user friendly and already convenient to access. Currently in the process of implementing RAID

			and we are still deciding whether to use that or not.
		QA-5	Allowing information to be uploaded such as lectures allows for a large amount of resources. A fully complete database is implemented and this QA is implemented.
		CON-1	Students are only able to change study information. Not personal information.
		CON-2	Access to private on the website is already limited and only the administration has the power to manage courses.
	CRN-2		A structure to have a second login has been identified and is being implemented
		CRN-3	The personal information of the students and lecturers has been hidden. The decision for this has been taken and implemented