Use Case Narratives

Use Case 1.1

System Name:		
Author: Your Name	**Date:**	Version: 1.0.0
Use case name:	Login/Register	**USE CASE TYPE**
Use case id:	1.1	Business Requirements: ☐
Priority:	High	System Analysis: ☒
Source:	Case Study	System Design: ☐
Primary business actor	User	
Primary system actor	N/A	
Other participating actors:	• None	
Other interested stakeholders:	• None	
Description:	With a Chat Room website, it is important for people to be able to identify the users who will be sending messages. This is why a Person must register as a user who will be uniquely identified by the system and reduce confusion. The process starts off by the user entering his information such as Name, Gender, MemberType (Teacher or Student) and Profile Description. These fields are captures onto the system as attributes which will be used to call various methods/facilitate certain functionality within the application.	
Pre-condition:	User must have opened the application and be present on Enter Rooms View. User must also be a person (of course) and must have credentials set aside/in mind.	
Trigger:	User loads Enter Rooms view and begins to enter required fields.	

Use Case 1.2

System Name:		
Author: Your Name	**Date:**	Version: 1.0.0
Use case name:	Send Message	**USE CASE TYPE**
Use case id:	1.2	Business Requirements: ☐
Priority:	High	System Analysis: ☒
Source:	Case Study	System Design: ☐
Primary business actor	User	
Primary system actor	N/A	
Other participating actors:	• Online User (ERA/ESA)	
Other interested stakeholders:	• None	
Description:	Once you have registered as a user, you will have the ability to send messages to all different users who are currently active on the server. Your messages will be sent via web sockets (SignalR) which will update the chat in real time to whoever it is you are sending the data to. The send method from the ChatHub class will do the processing to facilitate instantaneous data flow. All the user has to do is enter their message in the textbox then proceed to click send.	
Pre-condition:	User must have logged in as a student or teacher on the website.	
Trigger:	User presses the textbox to begin typing message.	

Use Case 1.3

System Name:		
Author: Your Name	**Date:**	Version: 1.0.0
Use case name:	Block Student	**USE CASE TYPE**
Use case id:	1.3	Business Requirements: ☐
Priority:	Medium	System Analysis: ☒
Source:	Case Study	System Design: ☐
Primary business actor	Teacher (inherited from User class)	
Primary system actor	N/A	
Other participating actors:	• Student	
Other interested stakeholders:	• None	
Description:	In an eLearning platform full of young students, some may step the line and send inappropriate messages. In order to prevent that from happening, the teacher is given system rights to block a student who is not behaving accordingly. This is done by loading the Block View (via clicking Blocked Students button), then you may select the student that you may want to block, and the system will update the status of the student. This will mean that the student will have the ability to send messages in the chat unless the teachers decide to unblock the student (which is a very similar process to unblocking).	
Pre-condition:	Must be a teacher to initiate and student blocked status should be false.	
Trigger:	User presses “Block Student” button from Chat View.	

Use Case 1.4

System Name:		
Author: Your Name	**Date:**	Version: 1.0.0
Use case name:	Refresh	**USE CASE TYPE**
Use case id:	1.4	Business Requirements: ☐
Priority:	High	System Analysis: ☒
Source:	Case Study	System Design: ☐
Primary business actor	User	
Primary system actor	N/A	
Other participating actors:	• Online Student/Teacher (ERA)	
Other interested stakeholders:	• None	
Description:	This could be considered an abstract use case where it will be used a lot to update the various data for students/teachers. When someone is Blocked or Unblock, the system will only register their new status once the data is sent form client side to server side. Unlike web sockets that provide one channel for bidirectional flow of data between client and server, Razer pages only has the standard HttpGet and HttpPost which means data must sent and requested via two channels. This means in order to update the view; the page needs to be refreshed and this is done by the user proceeding to the Chat View where they will see a “Refresh” button that will initiate the update. This action, like the others, will update all client’s sides that are active on the application, effecting other teachers/students.	
Pre-condition:	User must have requested updated data from server and be on Chat View.	
Trigger:	User clicks “Refresh” button.	