ChatMaps Deliverable 5

Sequence Diagram

Stephen Goodridge, Clark LaChance, Nicholas Pease, Joseph Gallant, Aidan Bradley

COS420

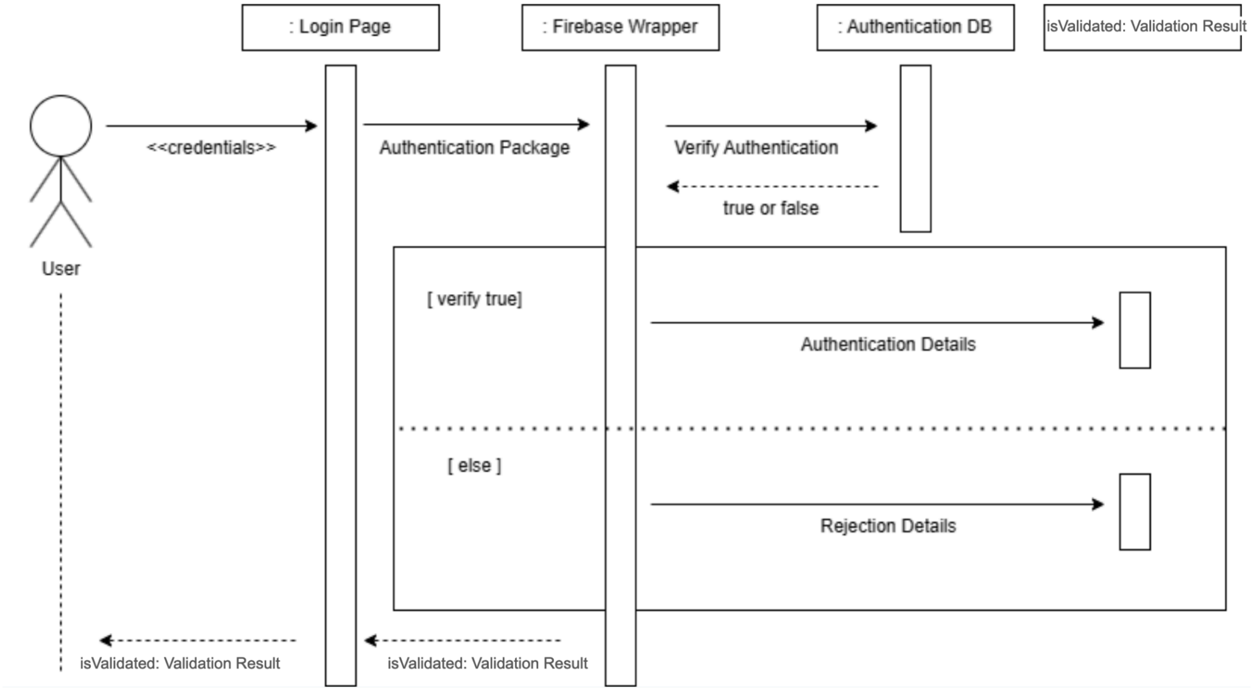
21 April 2024

Github: https://github.com/ChatMaps/ChatMaps

Kanban: https://trello.com/b/TaygvBv7/chatmaps

Sequence Diagram

Login

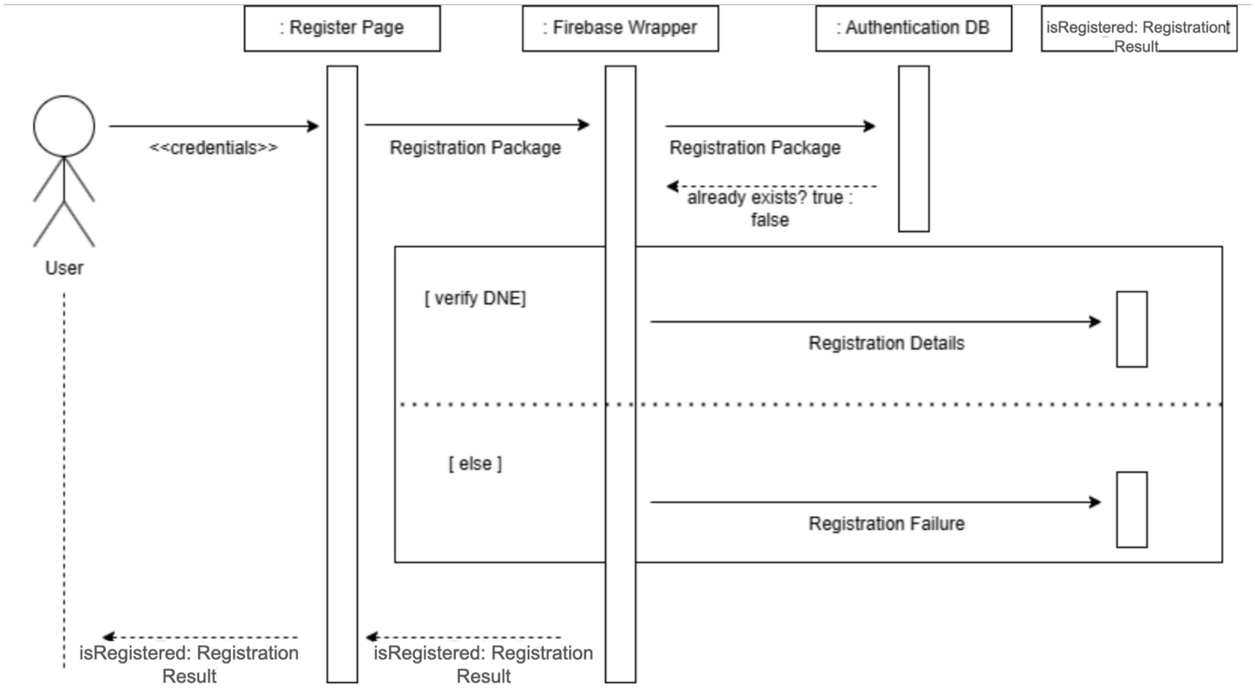


#	Subject	Subject Action	Parameters	Object Acted Upon
1	User	enters	Credentials	Login Page
1.1	Login Page	dispenses	Authentication Package	Firebase Wrapper
1.2	Firebase Wrapper	verifies	Authentication Package	Authentication DB
1.3	Authentication DB	returns	True or false to	Firebase Wrapper
1.4	If true is returned			
1.4.1	Firebase Wrapper	dispenses	Authentication Details	Validation Result
1.5	else			
1.5.1	Firebase Wrapper	dispenses	Rejection Details	Validation Result

1.6	Firebase Wrapper	returns	Validation Result	Login Page
1.7	Login Page	returns	Validation Result	User

```
1 User enters login details on the login page
1.1 The login page packages those details and sends the package to the Firebase Wrapper
1.2 The Firebase wrapper then verifies the login details against the authentication database
1.3 The authentication database returns a Firebase auth object, along with a true/false determination of login success
1.4 If the authentication returns a true
        1.4.1 The Firebase wrapper returns the authenticated user object
1.5 else
        1.5.1 The firebase wrapper returns the rejection message object
1.6 The Firebase wrapper returns the given object to the login page
1.7 The login page returns the object to the user with a message
```

Register



#	Subject	Subject Action	Parameters	Object Acted Upon
1	User	enters	Credentials	Register Page
1.1	Register Page	dispenses	Registration Package	Firebase Wrapper
1.2	Firebase Wrapper	verifies	Registration Package	Authentication DB
1.3	Authentication DB	returns	Account already exists true or false	Firebase Wrapper
1.4	If true is returned			
1.4.1	Firebase Wrapper	dispenses	Registration Details	Registration Result
1.5	else			
1.5.1	Firebase Wrapper	dispenses	Registration Failure	Registration Result
1.6	Firebase Wrapper	returns	Registration Result	Register Page

1.7	Register Page	returns	Registration Result	User

1 User enters registration details on the register page

1.1 The register page packages those details and sends the package to the Firebase Wrapper

1.2 The Firebase wrapper then verifies the account does not exist against the Authentication DB

1.3 The authentication database returns a true or false result depending on if the account exists

1.4 If the account does not exist

1.4.1 The Firebase wrapper returns the registration result

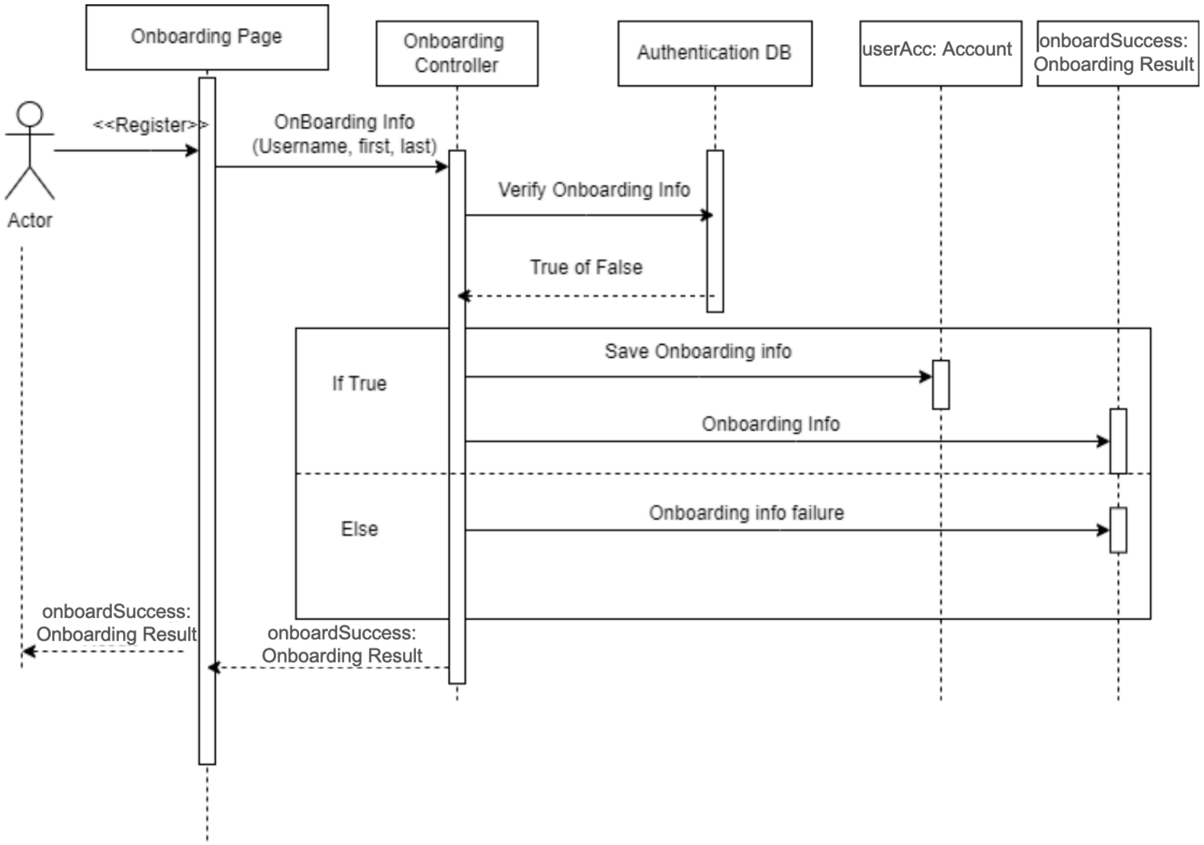
1.5 else

1.5.1 The firebase wrapper returns registration failure

1.6 The Firebase wrapper returns the given object to register page

1.7 The login page returns the object to the user with a message

Onboarding

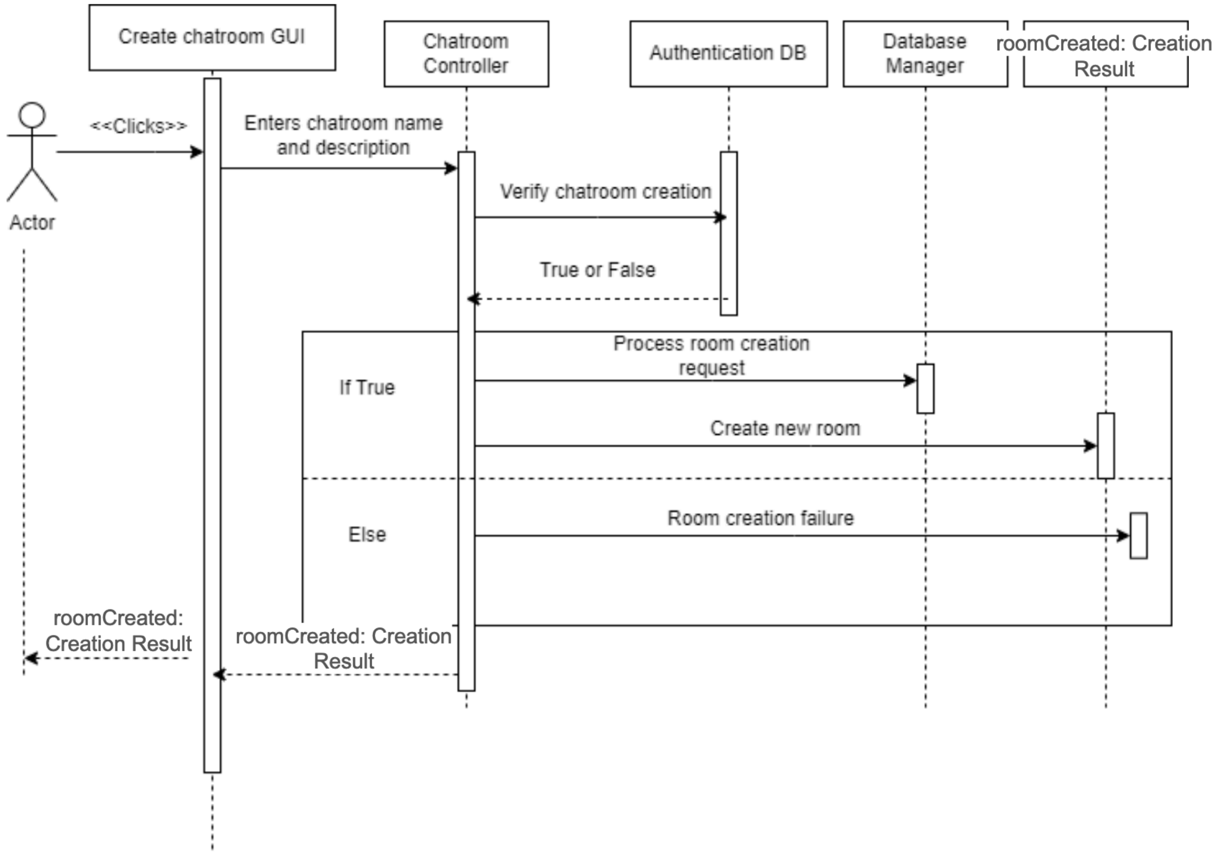


#	Subject	Subject Action	Parameters	Object acted upon
1	User	Register	credentials	Onboarding Page
1.1	Onboarding Page	dispenses	Onboarding info	Onboarding controller
1.2	Onboarding controller	verify	Onboarding info	Authentication database
1.3	If True is returned			
1.3.1	Onboarding controller	saves	Onboarding info	account
1.3.2	Onboarding controller	dispenses	Onboarding	Validation

			info	Result
1.4	If False is returned			
1.4.1	Onboarding controller	dispenses	Onboarding info failure	Validation result
1.5	Onboarding controller	Returns	Onboarding result	Onboarding Page
1.6	Onboarding page	Returns	onBoarding result	User

1: User registers an account and is brought to the Onboarding page
1.1: Onboarding information is taken to onboarding controller
1.2: Verify Username, first, and last name with authentication database
1.3: If true is returned
1.3.1: Save onboarding information to account
1.3.2: Return valid validation result
1.4 : If false is returned
1.4.1: Return invalid validation result
1.5: Returns validation result to Onboarding page
1.6 Returns validation result to User

Creating a Room

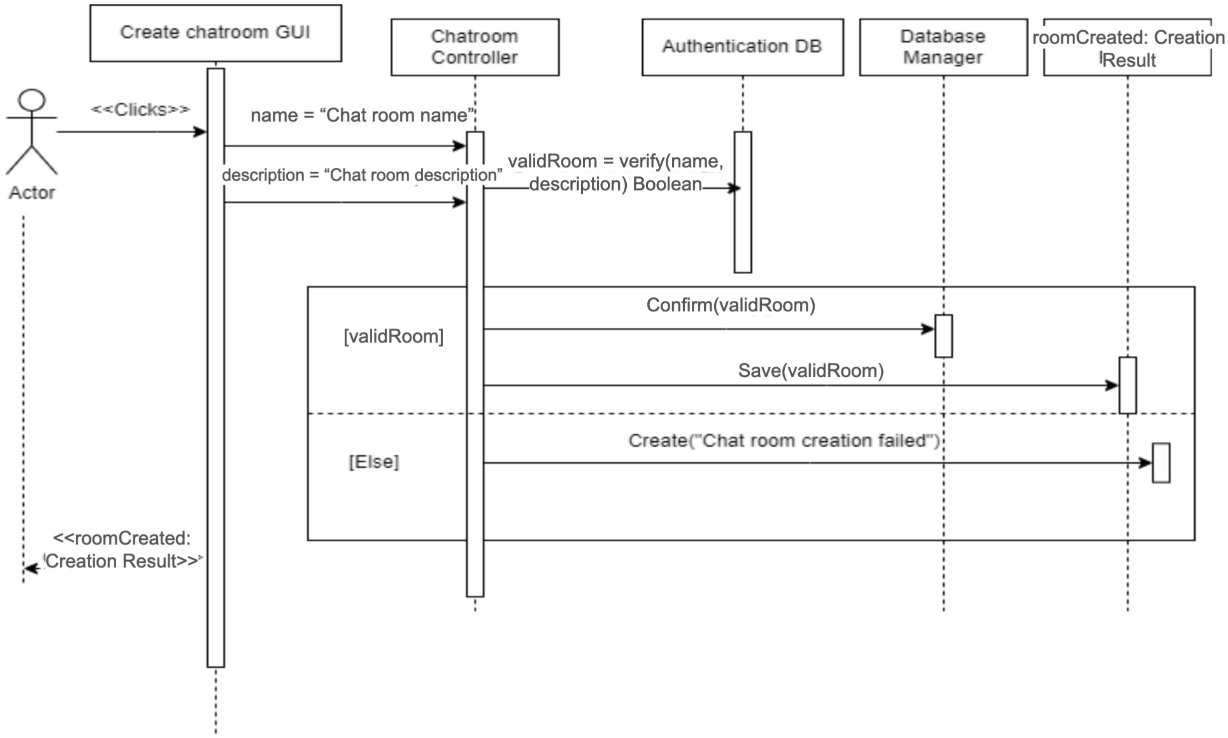


#	Subject	Subject Action	Parameters	Object Acted Upon
1	User	clicks	Create chat room	Create chat room GUI
1.1	Create Chat Room GUI	dispenses	Chat room name/description	Chat room controller
1.2	Chat room controller	verify	Chat room creation	Authentication database
1.3	If True is returned			
1.3.1	Chat room controller	dispenses	Room Creation	Database Manager
1.3.2	Chat room controller	creates	Chat room	Room creation result

1.4	If False is returned			
1.4.1	Chat room controller	dispenses	Room Creation Failure	Room creation result
1.5	Chat room controller	Returns	Room Creation Result	Create chat room GUI
1.6	Create chat room GUI	Returns	Room Creation Result	User

1: User clicks create chat room button
1.1: User enters chat room details such as room name/description
1.2: Authentication database verifies the name and description of the chat room
1.3: If True is returned
1.3.1: Database manager process the new room being created
1.3.2: Validate room creation
1.4: If False is returned
1.4.1: Reject room creation
1.5: Return room validation to Chat room GUI
1.6: Return room validation to User

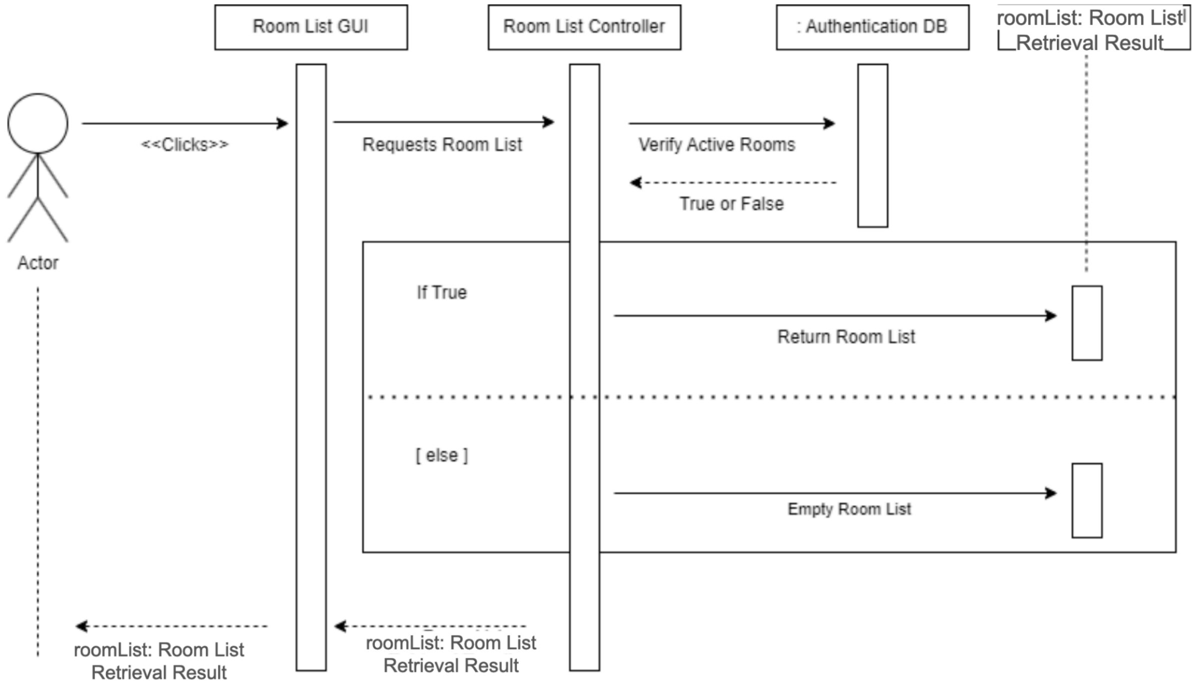
Design Sequence Diagram:


Create chatroom GUI
Chatroom Controller
Authentication DB
Database Manager
roomCreated: Creation Result
Actor
<<Clicks>>
name = “Chat room name”
description = “Chat room description”
validRoom = verify(name, description) Boolean
Confirm(validRoom)
[validRoom]
Save(validRoom)
Create("Chat room creation failed")
[Else]
<<roomCreated: Creation Result>>

#	Subject	Subject Action	Parameters	Object Acted Upon
1	User	clicks	Create chat room	Create chat room GUI
1.1	Create Chat Room GUI	dispenses	name,description	Chat room controller
1.2	Chat room controller	verify()	name,description	Authentication database
1.3	If True is returned			
1.3.1	Chat room controller	confirm()	validRoom	Database Manager
1.3.2	Chat room controller	roomManager = save()	validRoom	Room creation result
1.4	If False is returned			
1.4.1	Chat room controller	roomManager = create()	Room Creation Failure	Room creation result

1.5	Chat room controller	Returns	roomManager	Create chat room GUI
1.6	Create chat room GUI	Returns	roomManager	User

Room List Retrieval

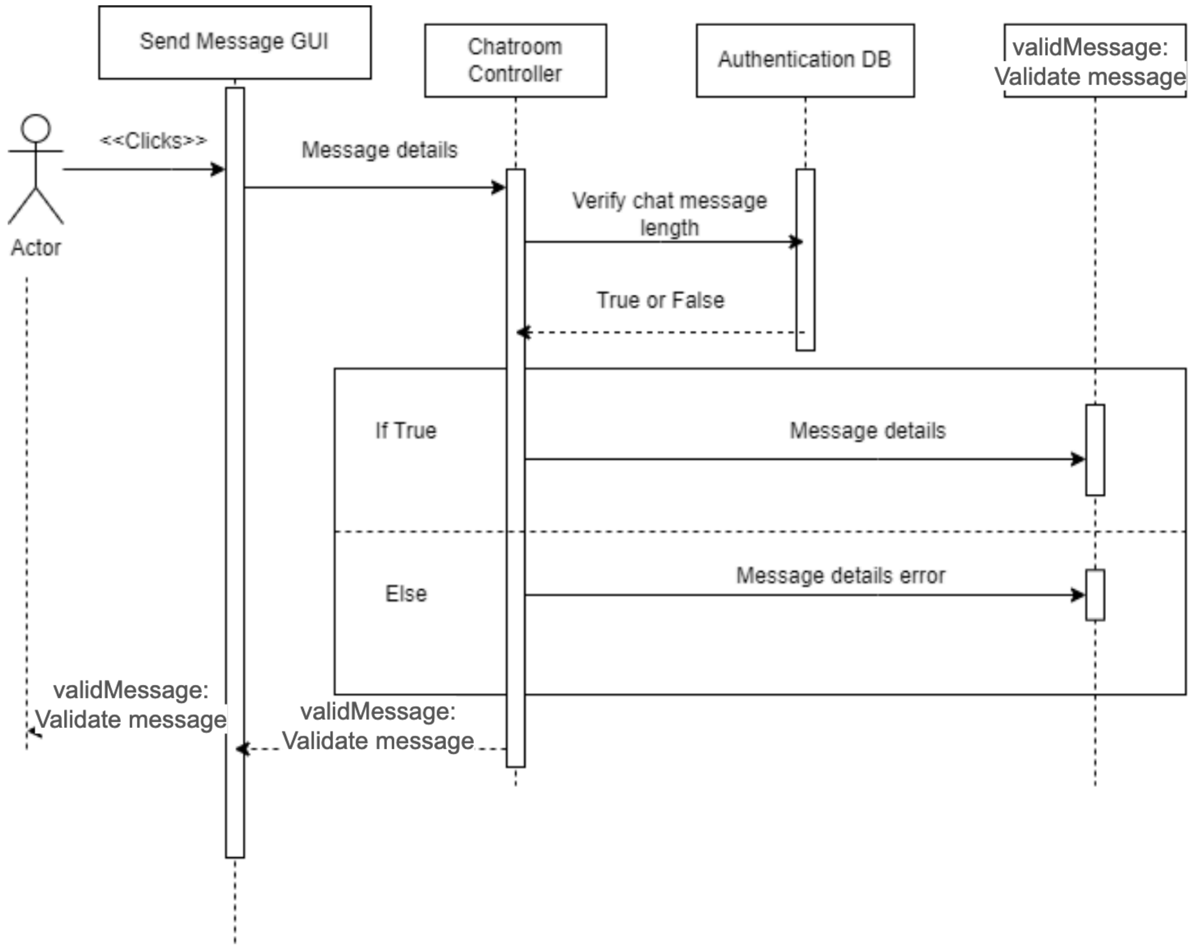


#	Subject	Subject Action	Parameters	Object Acted Upon
1	User	Clicks	Request Room List	Room List GUI
1.1	Room List GUI	request	Room List	Room List Controller
1.2	Room List Controller	verify	Active Rooms	Authentication DB
1.3	If True is returned			
1.3.1	Room List Controller	return	Room List	User
1.4	If False is returned			
1.4.1	Room List	return	Empty Room	User

	Controller		List	

1: User clicks “Rooms” to get the list of rooms.
1.1: The request gets sent to the Room List Controller for processing.
1.2: The Room List Controller verifies that there’s Active Rooms by checking the database.
1.3: If True
1.3.1: Return a list that contains active rooms.
1.4: If False
1.4.1: Return an empty list that contains no rooms.

Sending a message

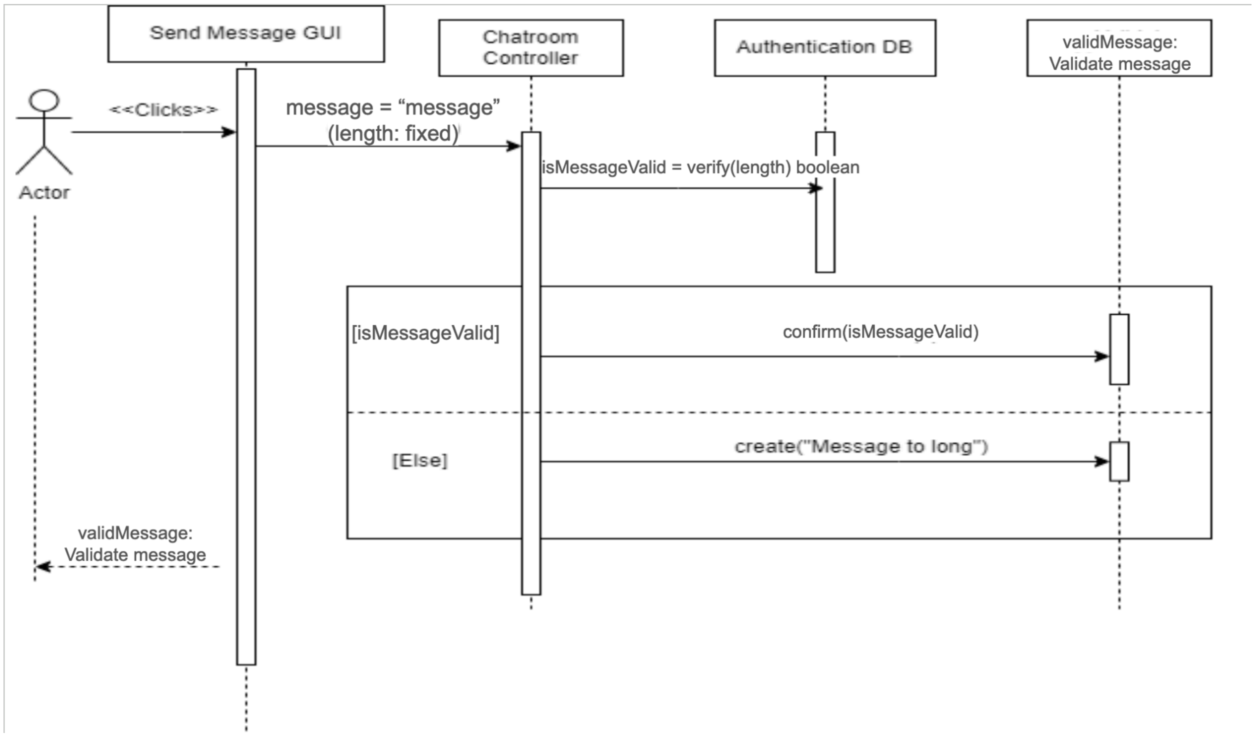


#	Subject	Subject Action	Parameters	Object Acted Upon
1	User	Clicks	Send	Send Message GUI
1.1	Send Message GUI	dispenses	Message details	Chatroom controller
1.2	Chat room controller	verify	Chat length	Authentication DB
1.3	If True is returned			
1.3.1	Chat room controller	dispenses	Message	Validate

			details	Message
1.4	If False is returned			
1.4.1	Chat room controller	dispenses	Message details error	Validate Message
1.5	Chat room controller	Return	Validate message	Send Message GUI
1.6	Send Message GUI	Return	Validate message	User

1: User clicks “send” to send message
1.1: The users typed message is processed to the chat room controller
1.2: Chatroom controller verifies the message isn’t to long
1.3: If True
1.3.1: Validate the message
1.4: If False
1.4.1: Message will fail and won’t send
1.5: Return validation that the message sent to the chat room controller
1.6: Return validation that the message sent to the user

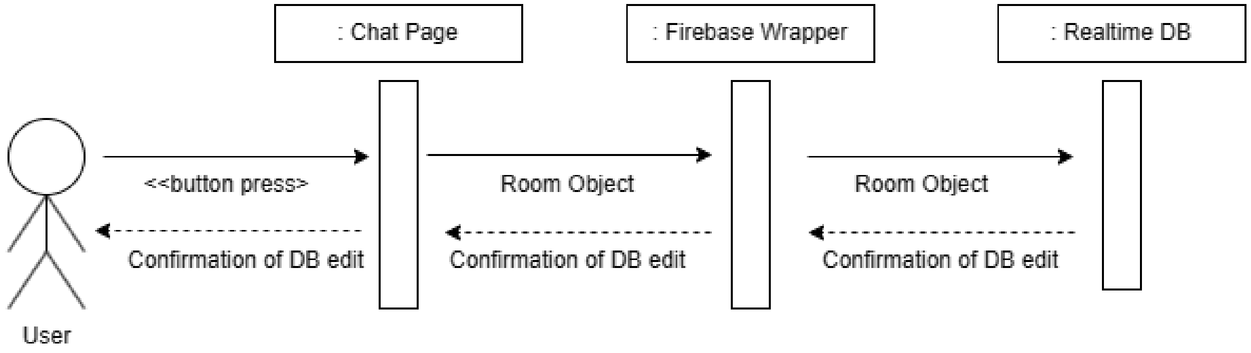
Design Sequence Diagram:


Send Message GUI
Chatroom Controller
Authentication DB
validMessage: Validate message
Actor
<<Clicks>>
message = "message" (length: fixed)
isMessageValid = verify(length) boolean
[isMessageValid]
confirm(isMessageValid)
[Else]
create("Message to long")
validMessage: Validate message

#	Subject	Subject Action	Parameters	Object Acted Upon
1	User	Clicks	Send	Send Message GUI
1.1	Send Message GUI	dispenses	message	Chatroom controller
1.2	Chat room controller	isMessageValid = verify()	message	Authentication DB
1.3	If True is returned			
1.3.1	Chat room controller	confirm()	isMessageValid	Validate Message
1.4	If False is returned			
1.4.1	Chat room controller	create()	Message details error	Validate Message
1.5	Chat room controller	Return	validMessage	Send Message GUI

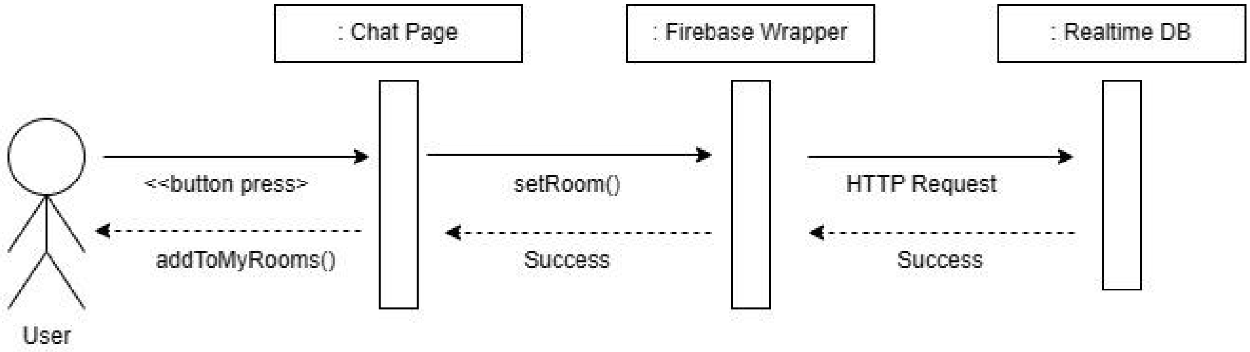
1.6	Send Message GUI	Return	validMessage	User

Saving a Room



#	Subject	Subject Action	Parameters	Object Acted Upon
1	User	presses	Button	Chat Page
1.1	Chat Page	sends	Room Object	Firebase Wrapper
1.2	Firebase Wrapper	sends	Room Object	Realtime DB
1.3	Realtime DB	sends	Confirmation	Firebase Wrapper
1.4	Firebase Wrapper	sends	Confirmation	Chat Page
1.5	Chat Page	sends	confirmation	User

1 User presses the save room button
1.1 The chat page sends the room object to the Firebase wrapper for packaging
1.2 The firebase wrapper sends the room object to the real-time DB
1.3 The real-time DB sends a confirmation to the Firebase wrapper
1.4 The firebase wrapper sends confirmation to the chat page
1.5 The chat page sends a confirmation to the user via a message



#	Subject	Subject Action	Parameters	Object Acted Upon
1	User	presses	Button	Chat Page
1.1	Chat Page	sends	Room Object via setRoom()	Firebase Wrapper
1.2	Firebase Wrapper	sends	Room Object via HTTP	Realtime DB
1.3	Realtime DB	sends	Success	Firebase Wrapper
1.4	Firebase Wrapper	sends	Success	Chat Page
1.5	Chat Page	sends	Success via addToMyRooms()	User

1 User presses the save room button
1.1 The chat page sends the room object to the Firebase wrapper for packaging
1.2 The firebase wrapper sends the room object to the real-time DB
1.3 The real-time DB sends a confirmation to the Firebase wrapper
1.4 The firebase wrapper sends confirmation to the chat page
1.5 The chat page sends a confirmation to the user via a message