Py Classroom

Nattaphol Srisa : 59090014

Panupong Viriyasuebpong : 59090018

Tharita Tipdecho : 59090035

Project Description

PyClassroom is a non-profit desktop classroom application which separates into two software; for teachers and students. The project allows anyone to attend the classes created by the teachers. Additionally, students can search for classes of their own interests, and can select to attend any of them. In each classroom, the teacher shares the knowledge by live streaming a public lecture to all the students participating in the classroom. Students can also send messages to the teacher via public chat during the live stream.

Requirements

User Requirements

Functional:

1) The teacher can create a room which will be included in the list of rooms.
2) The teacher can edit room information in the waiting lobby.
3) The teacher can live stream a lecture directly via their computers
4) The student can enter any room that is available in the list of rooms
5) The teacher can remove any participating student from the room
6) The student can ask questions to the teachers.
7) The student can search for courses by entering the course id, course name, or lecturer's name.
8) The teacher can publish course material's URLs to online storage.
9) The teacher can see and manage a list of student attending their lecture.
10) The student can leave the room anytime after the student has joined the room

Non-functional:

1) There is a GUI (Graphic User Interface) using PyQt5 so it will be easy to understand and navigate by the users.
2) The system is able to response the users action without delaying or lacking.
3) The system is up to date. The system's information is regularly maintained by the system's administrator.
4) The system is extensible. New functions can be added to the system in the future.
5) Any action in a lecture room will not be visible to the users outside.

6) The system must be connected to the internet constantly.
7) The system will deallocate unused resource from user’s device.
8) The system is non-profit.
9) The system is compatible with many Operating Systems.
10) The system is provided in English language.

System requirements

Functional:

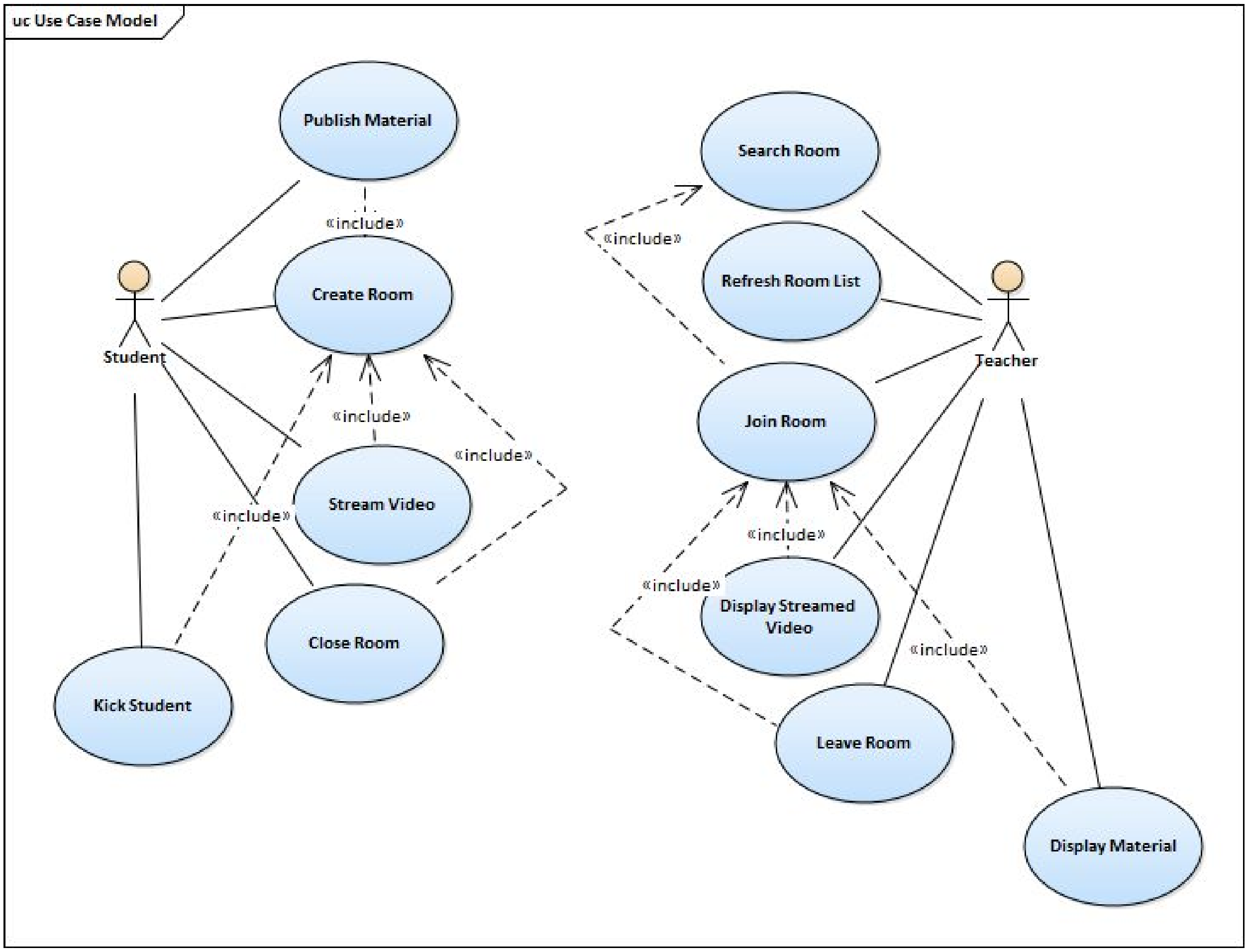
1) The software is able to capture a video from the teacher’s camera using OpenCV2 and broadcast it to every students in the same room by sending the video to the main server and then server will send it to all students in the same room via a socket and IP address.
2) The software is able to retrieve the room information from the socket and show it as a list of rooms with course ID, course name, course lecturer, course description, and the current viewers out of the maximum student capacity to all online students.
3) The list of public lecture rooms are shown to student using QListView along with QStandardItemModel from PyQt5.
4) The student can refresh the list of the public lecture rooms with QPushButton from PyQt5, and the list of updated rooms will be retrieved from the server using socket.
5) The student is able to ask questions to the teacher or discuss with other students during the live-stream by filling in their queries in QLineEdit from PyQt5 and clicking the send button, in which the questions will be sent via socket.
6) In student’s lobby, there would be QLineEdit for students to search for their required courses. The matching courses will be displayed whereas the unmatched courses would be hidden from the QListView
7) In teacher’s lobby, the teacher can post their course material’s URLs on Material Tab via socket. Those URLs would lead to the material on the online storage websites (in this case, Google Drive).
8) The teachers can see the list of students that are attending their courses by using QListView from PyQt5.
9) The teacher can remove any student that is participating in the room from the list displayed by QListView from PyQt5, by disconnecting the connection of that student, using socket.

10) The student can leave the room by disconnecting the connection with the server, using socket.

Non-functional:

1) The software is able to handle multiple users at the same time by keeping the information (name and IP address) of each online users on the server database and separate each user operations by using multithreading design.
2) The software can maintain a room's availability until the teacher choose to close the room by saving its state in the main server since the time of the creation of a room.
3) It is possible for the software to be added new features or fixed bugs easily by using the OOP design.
4) The software will use Python Garbage Collector interface to make sure that unused objects will be deleted, so the software would not take up too much resources from device.
5) There will always be one Daemon Thread running in the background to check the stability of connection and will try to reconnect. If the connect was lost for more than 20 seconds the program will be forced to terminal.
6) By being developed with PyQt5, the software will be able to run on all Microsoft Windows and other Linux based OS.
7) The software is free and may be redistributed, but is not open for modification to avoid hacking and ensure the community's peacefulness.
8) The software includes the software design pattern called MVC (Model / View / Controller). The model is responsible for maintaining data, the view is responsible for displaying all, or a portion of the data to the user, and the controller is the software code that controls the interactions between the model and the view.
9) The software includes the observer pattern, in order to allow sending data to other objects effectively without any change in the Observer classes
10) In case of an occurrence of error, the software provides exception handling in order to maintain the normal flow of the application as exception normally disrupts the normal flow of the application.

Use Case Diagram



Use Case Description

Teacher:

Publish Educational Material

Basic:

1) Teacher performs Create Room use case
2) Teacher selects Material Page
3) Teacher enters information to be published
4) System has published the material

Exceptional:

1) Teacher performs Create Room use case
2) Teacher selects Material Page
3) Teacher enters an empty string
4) System does not allow teacher to publish an empty string
5) System does not publish the material

Live Stream Video

Basic**:**

1) Teacher performs Create Room use case
2) Teacher selects a button to start Live Streaming
3) System live streams video

Exceptional:

1) Teacher performs Create Room use case
2) Teacher selects a button to start Live Streaming
3) System is interrupted by internet connection problems
4) System stops live streaming

Close Room

Basic:

1) Teacher performs Create Room use case
2) Teacher selects button to close room
3) Server disconnects students in the room one by one, followed by disconnection of the teacher
4) Server cleans up
5) System has closed the room

Create Room

Basic:

1) Teacher fills in sufficient information in order to create a room
2) Information gathered is translated into a server request
3) Server request is sent
4) System has created a room

Alternative (Server is full):

1) Teacher fills in sufficient information in order to create a room
2) Information gathered is translated into a server request
3) Server request is sent
4) Server is full
5) The request to create a room is reject
6) System does not create a room

Exceptional (Unstable Connection):

1) Teacher fills in sufficient information in order to create a room
2) Information gathered is translated into a server request
3) Connection is unavailable, request cannot be sent
4) Requests will not be queued during this period to prevent spamming, end of use case

Kick Student

Basic:

1) Teacher performs Create Room use case
2) Teacher selects Viewer page
3) Teacher selects a student to be kicked
4) The selected student has been kicked

Student

Search Room

Basic:

1) Student enters course id, course name, or lecturer’s name in the search box
2) Student clicks search button
3) System displays filtered rooms

Alternative:

1) Student enters course id, course name, or lecturer’s name in the search box
2) Student clicks search button
3) No matched courses, end of use case

Exceptional:

1) Student enters an empty string in the search box
2) Student clicks search button
3) System displays all unfiltered rooms available

Refresh Room List

Basic:

1) Student clicks refresh button
2) System sends a request asking for a new room list to server
3) System receives a new room list from server
4) New room list is updated and displayed
5) System has refreshed the room list

Join Room

Basic:

1) Student selects a room to join
2) Student clicks join button
3) Student has joined the selected room

Alternative (Maximum number of students already joined):

1) Student selects a room to join
2) Student clicks join button
3) Maximum number of students already joined, no more position for the student, end of use case.

Display Streamed Video

Basic:

1) Student performs the Join Room use case
2) Student selects the Stream Tab
3) System receives a request sent from the Teacher's server
4) System displays a live stream video

Exceptional:

1) Student performs the Join Room use case
2) Student selects the Stream Tab
3) System is interrupted by internet connection problems
4) System does not display the live stream video

Leave Room

Basic:

1) Student performs the Join Room use case
2) Student selects the Leave Room button
3) Student is disconnected from the lecture room

Display Educational Material

Basic:

1) Student performs the Join Room use case
2) Student selects the Material Tab
3) System displays material

Alternative (No material):

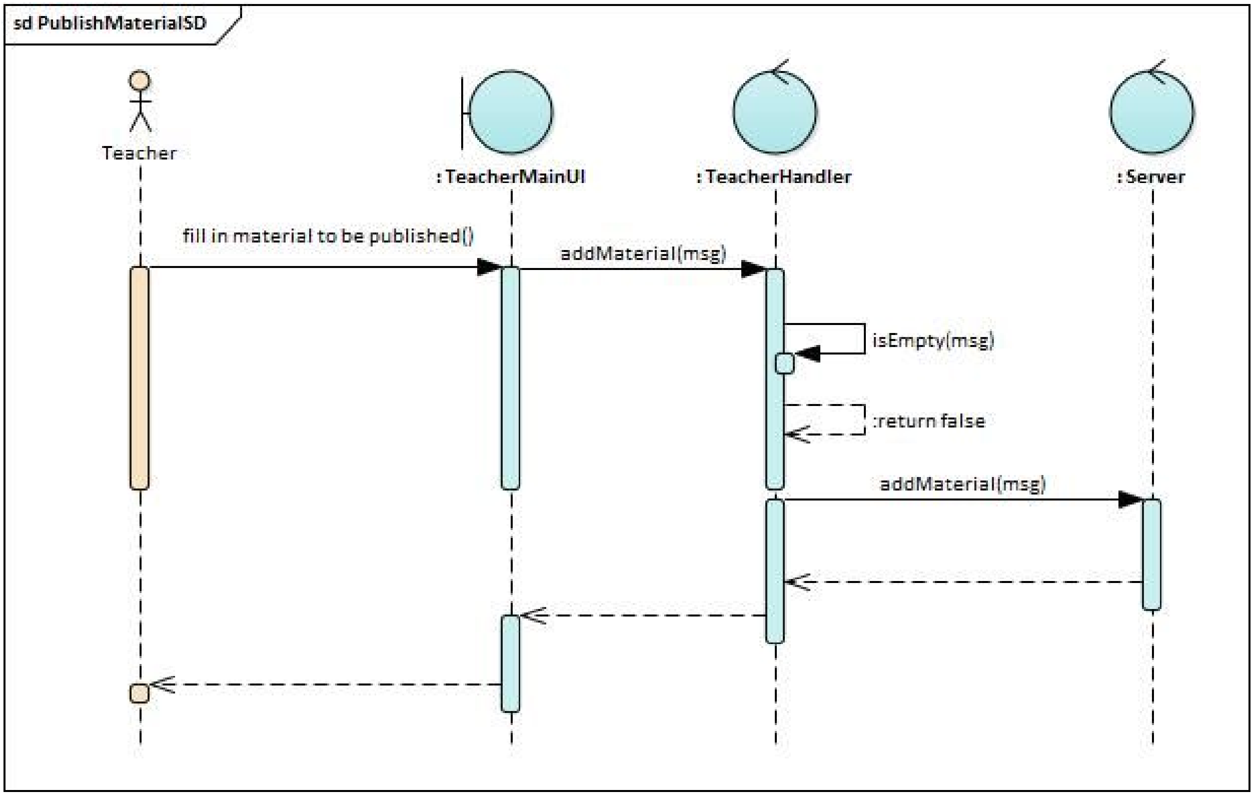
1) Student performs the Join Room use case
2) Student selects the Material Tab
3) No material is displayed, end of use case

Exceptional (Server offline):

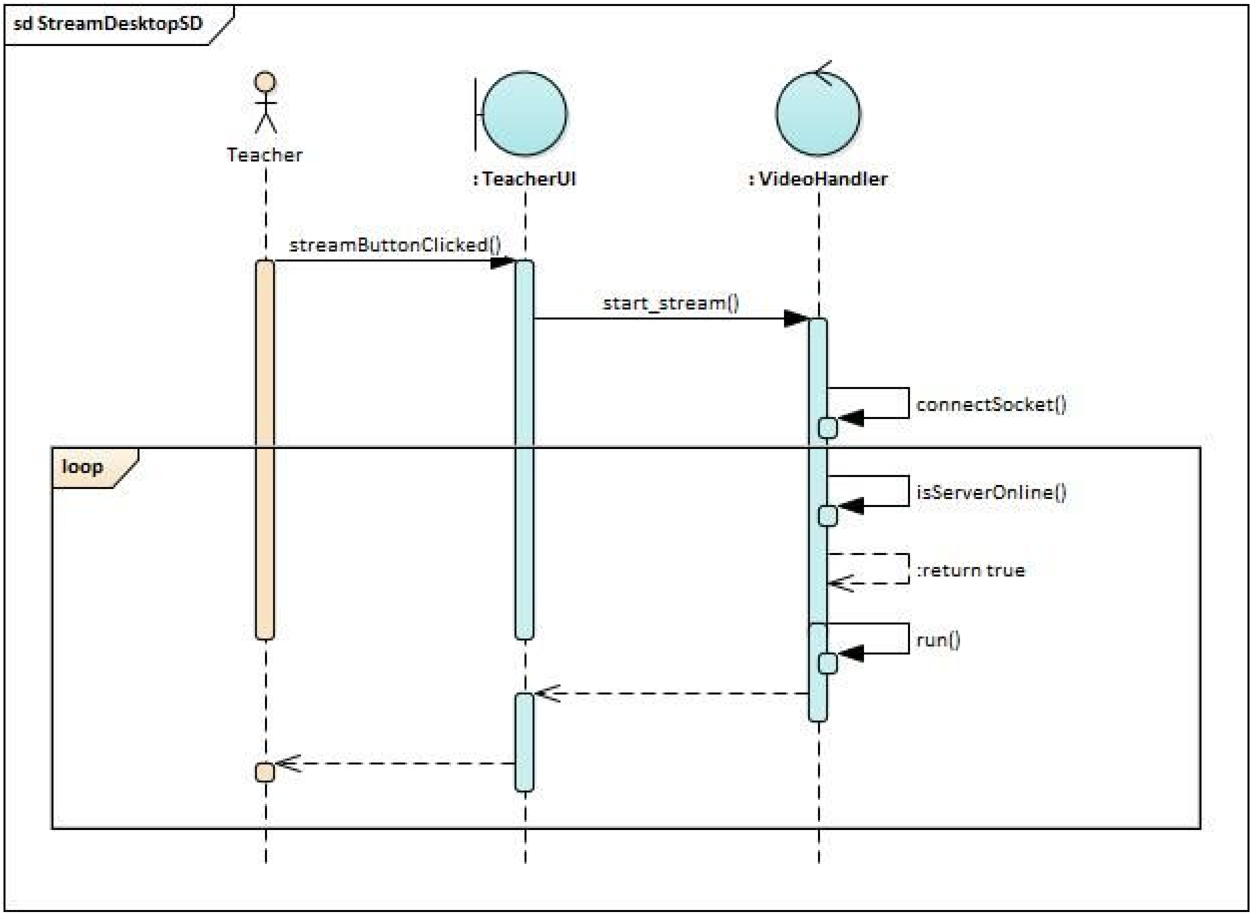
1) Student performs the Join Room use case
2) Student selects the Material Tab
3) Server is offline
4) System cannot display educational material

Sequence Diagram

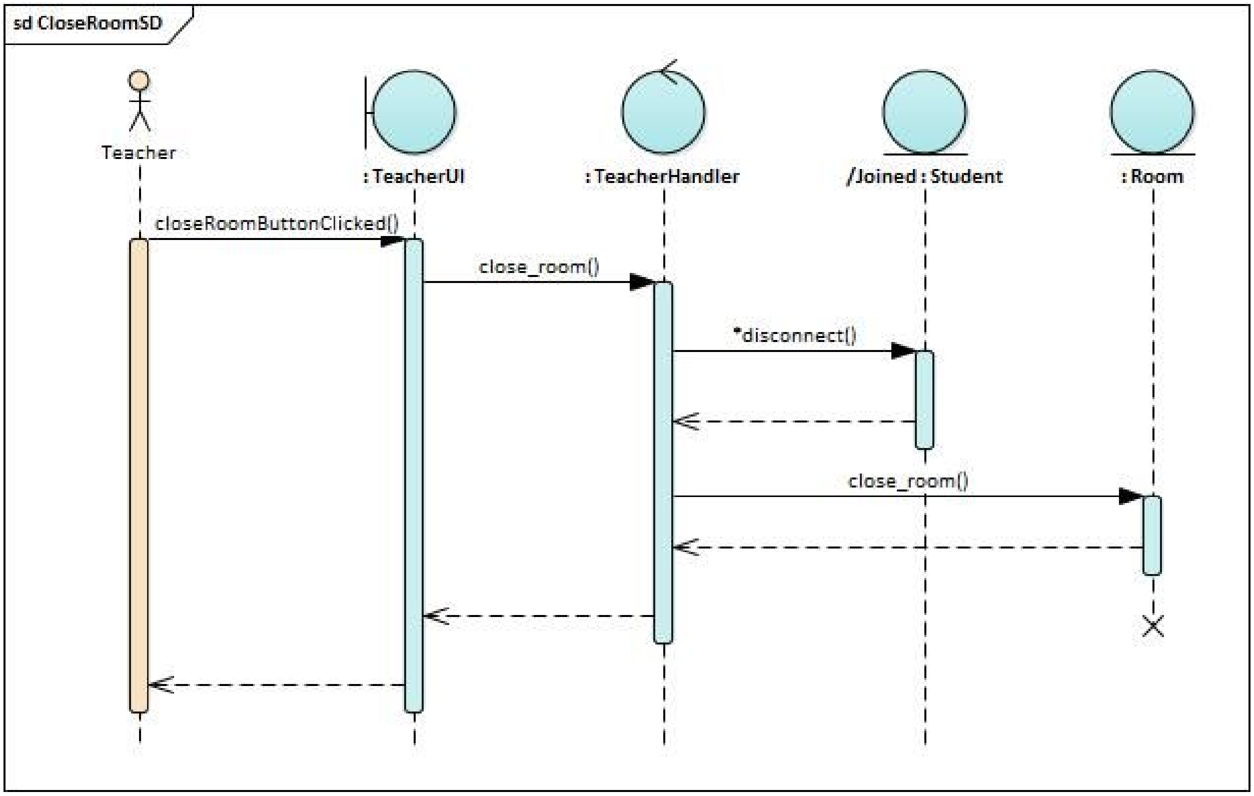
Publish Educational Material : BASIC



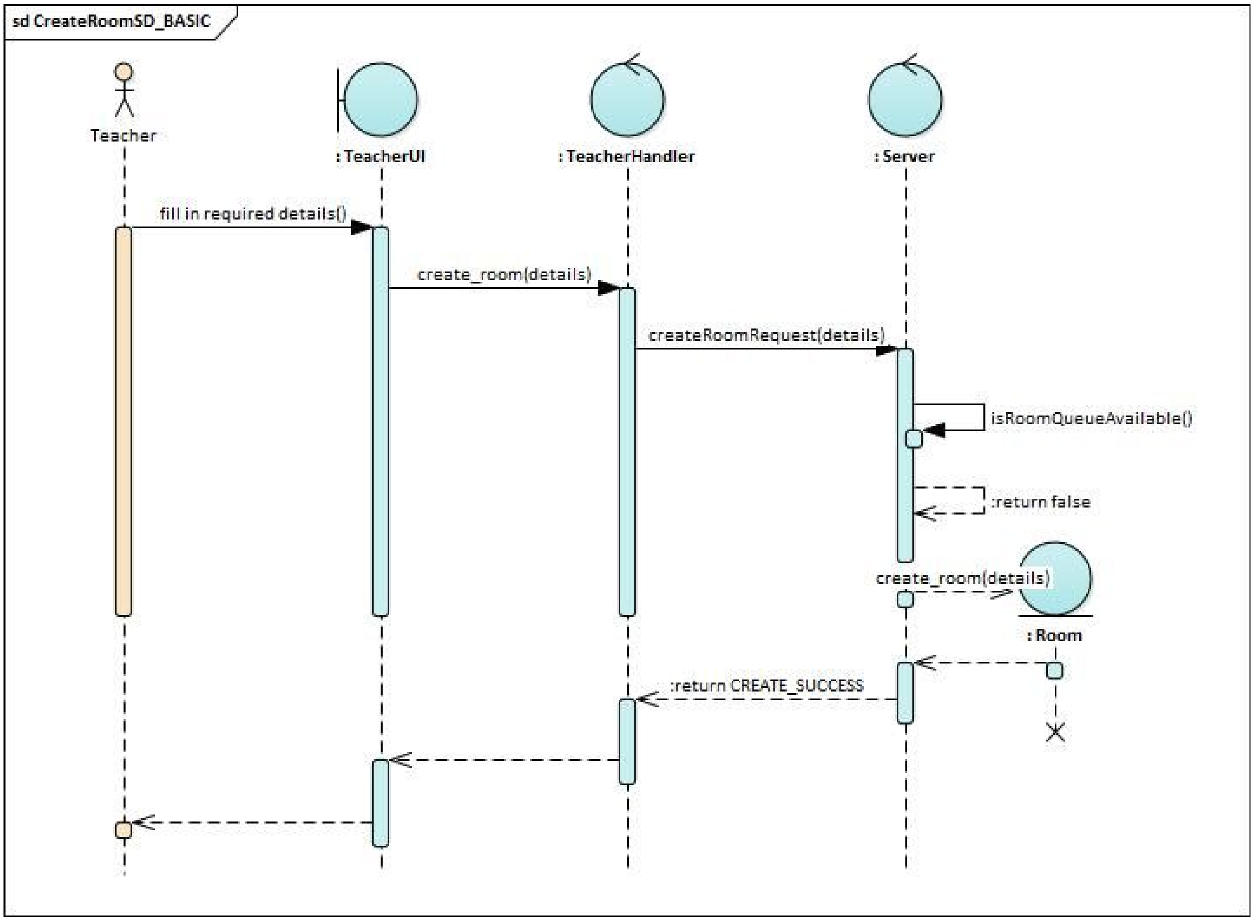
Live Stream Video : BASIC



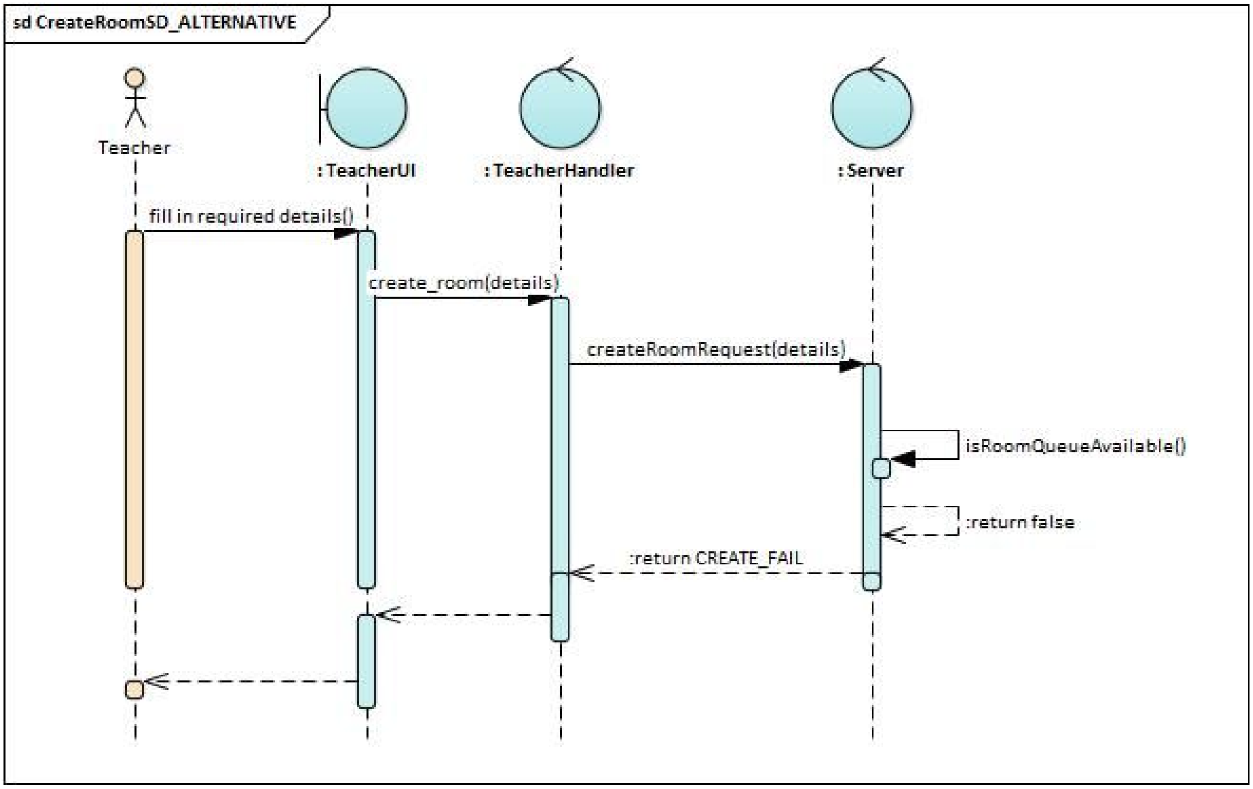
Close Room : BASIC



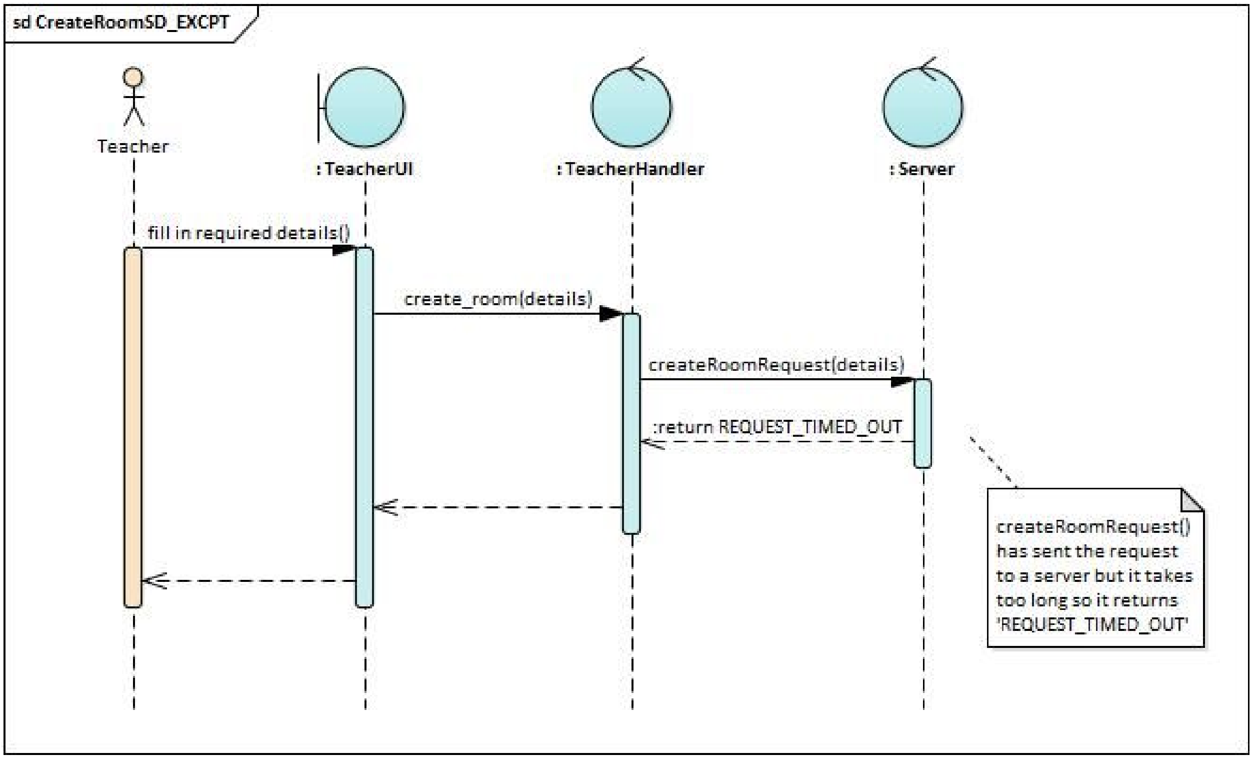
Create Room : BASIC



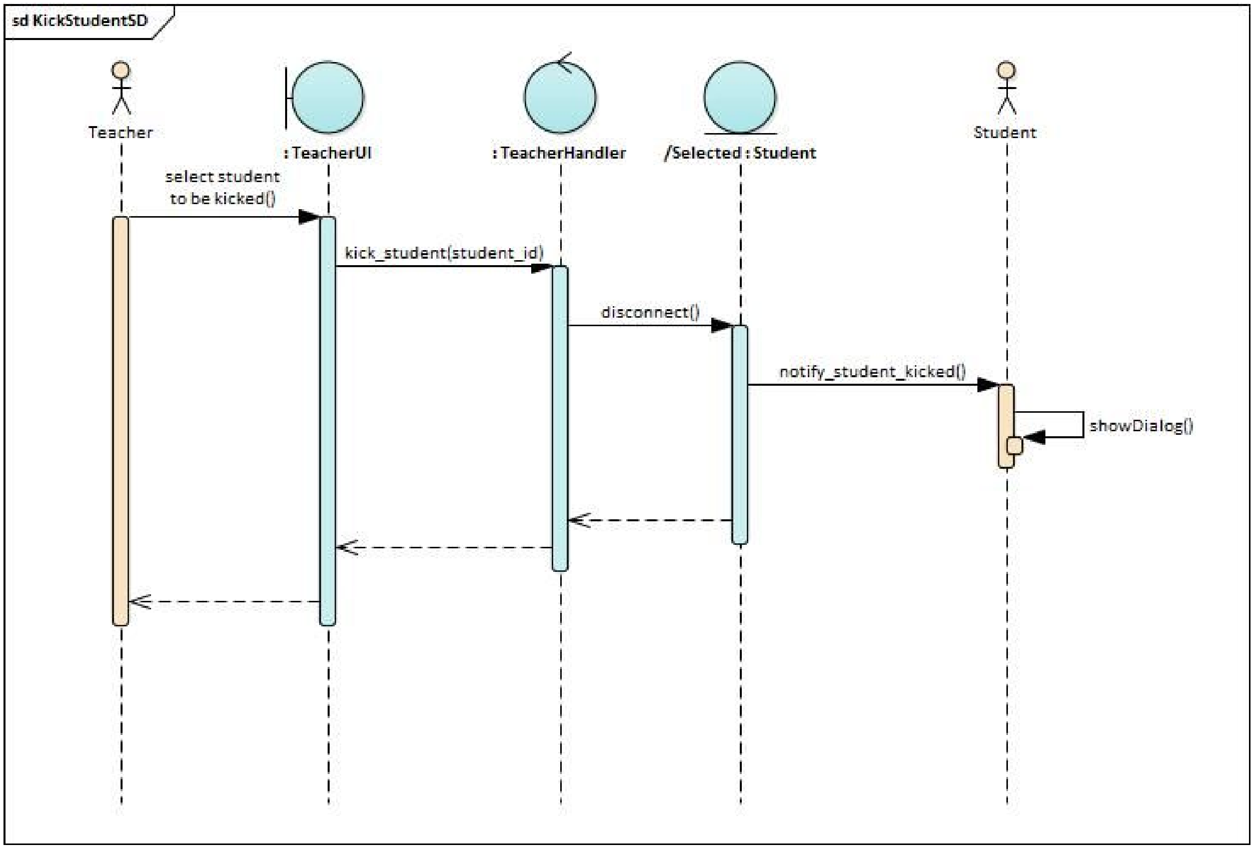
Create Room : ALTERNATIVE



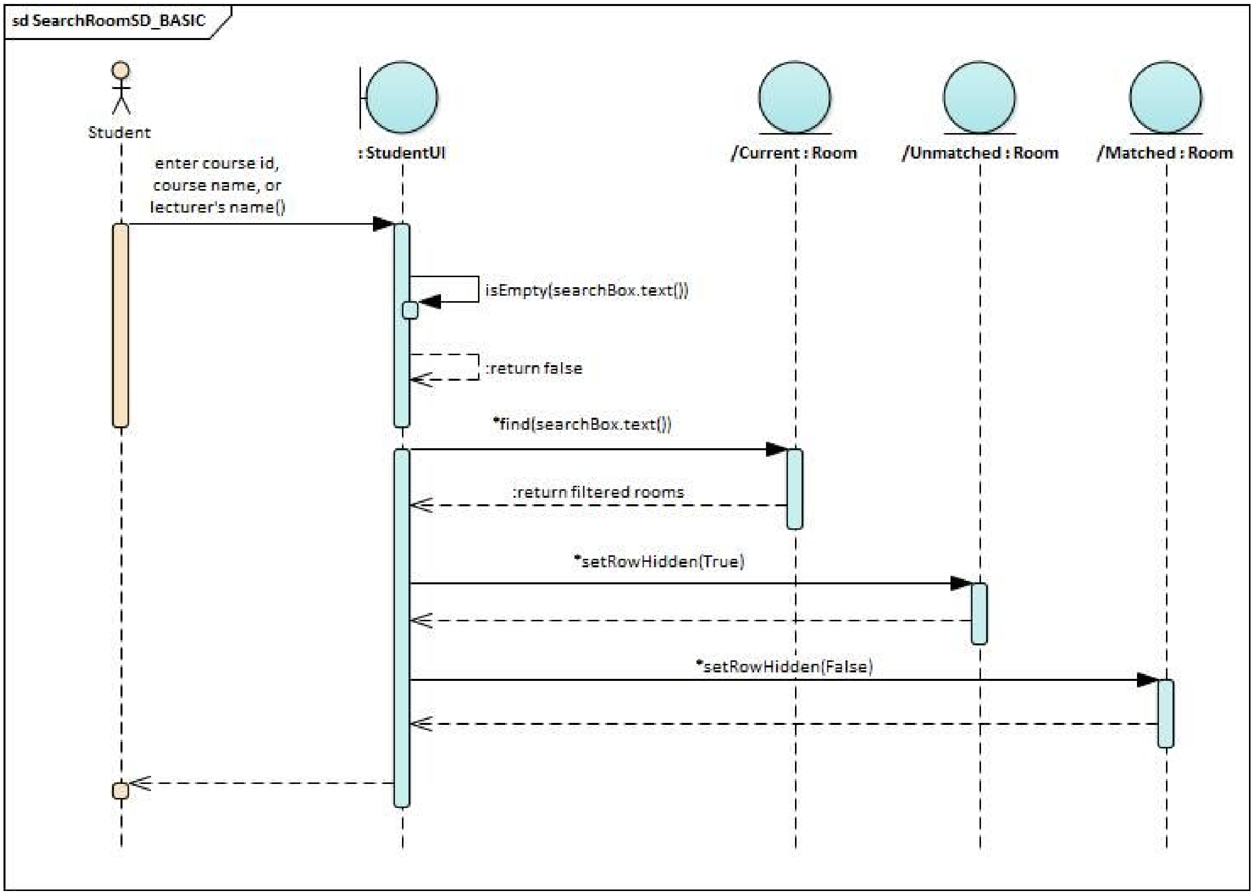
Create Room : EXCEPTIONAL


sd CreateRoomSD_EXCPT
Teacher
: TeacherUI
: TeacherHandler
: Server
fill in required details()
create_room(details)
createRoomRequest(details)
:return REQUEST_TIMED_OUT
createRoomRequest() has sent the request to a server but it takes too long so it returns 'REQUEST_TIMED_OUT'

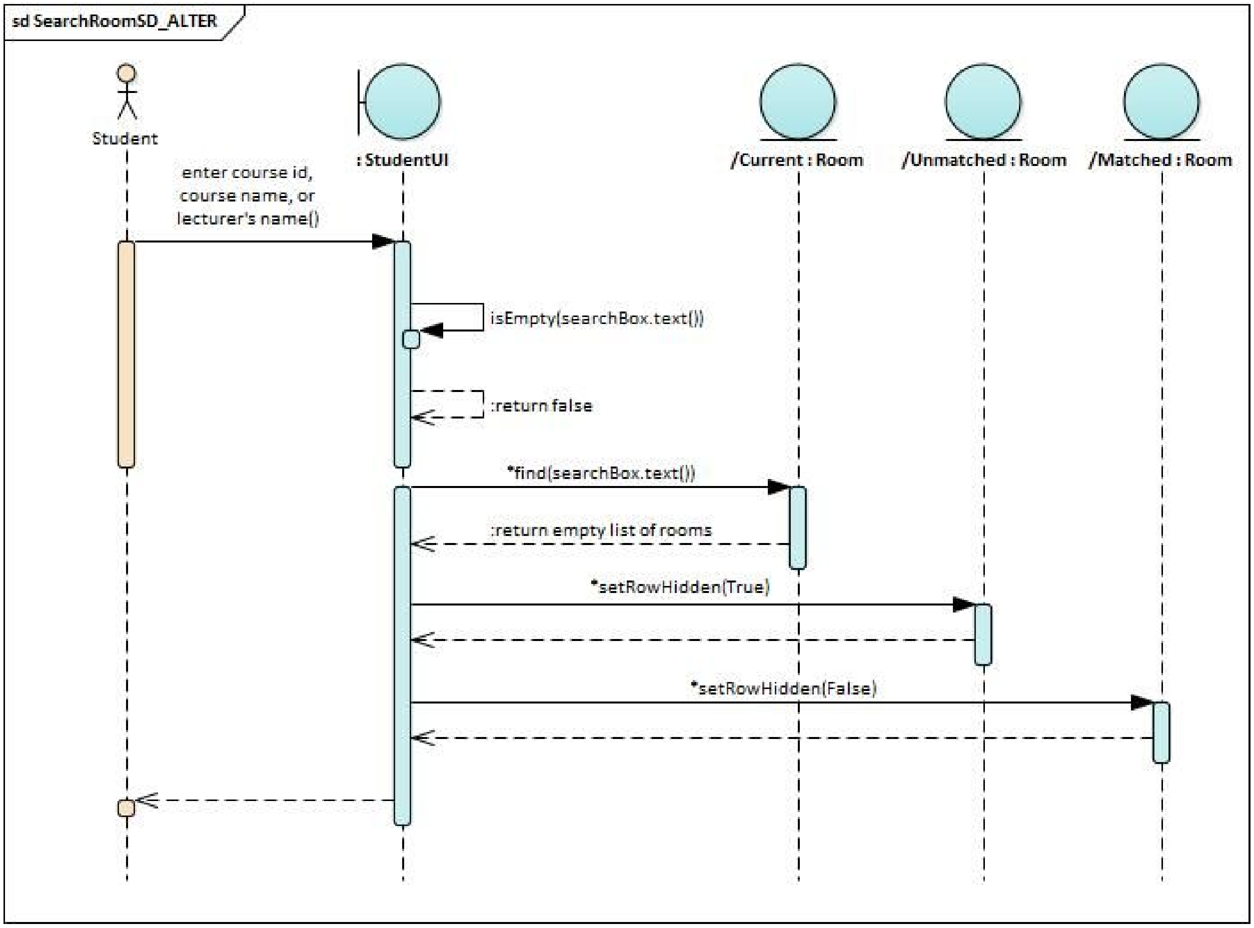
Kick Student : BASIC



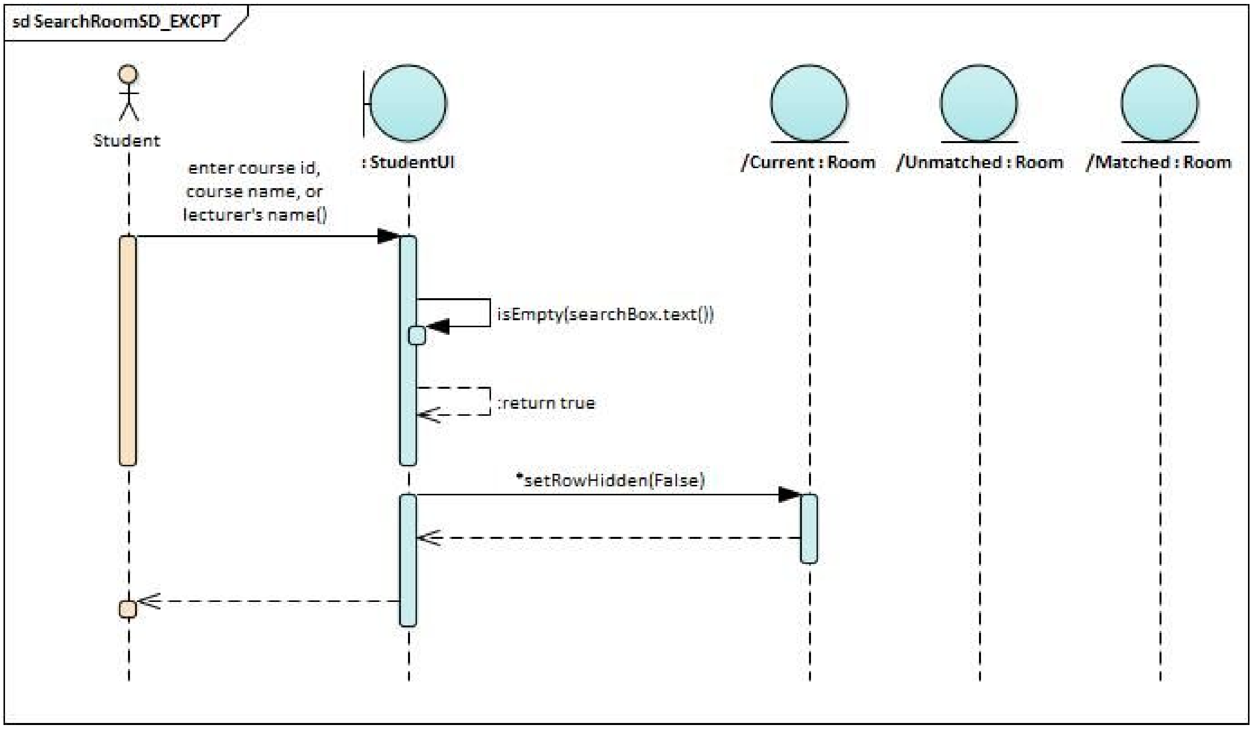
Search Room : BASIC



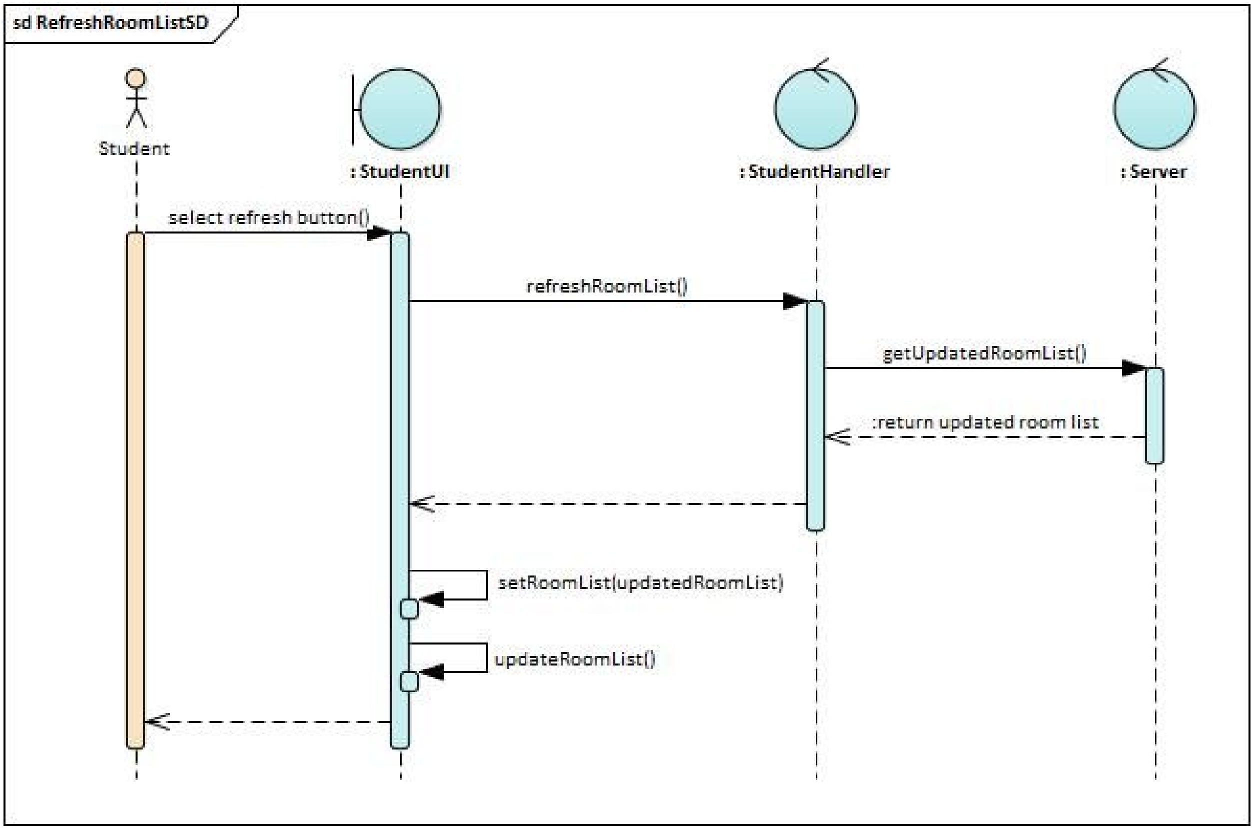
Search Room : ALTERNATIVE



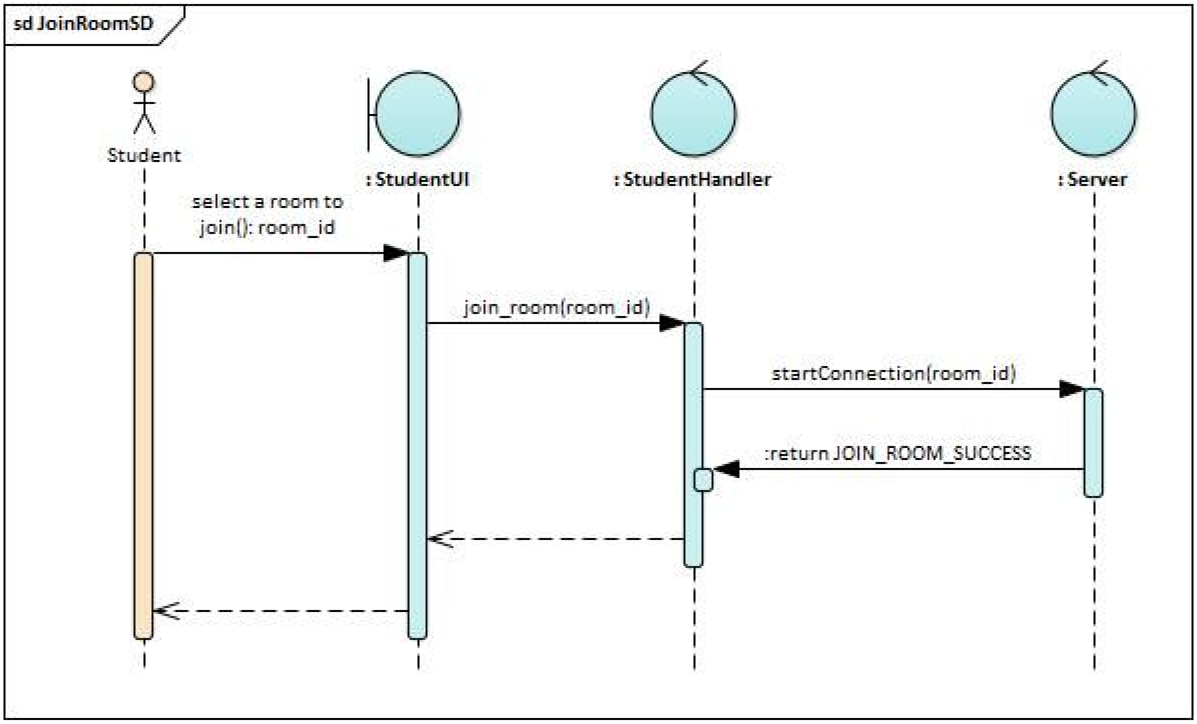
Search Room : EXCEPTIONAL



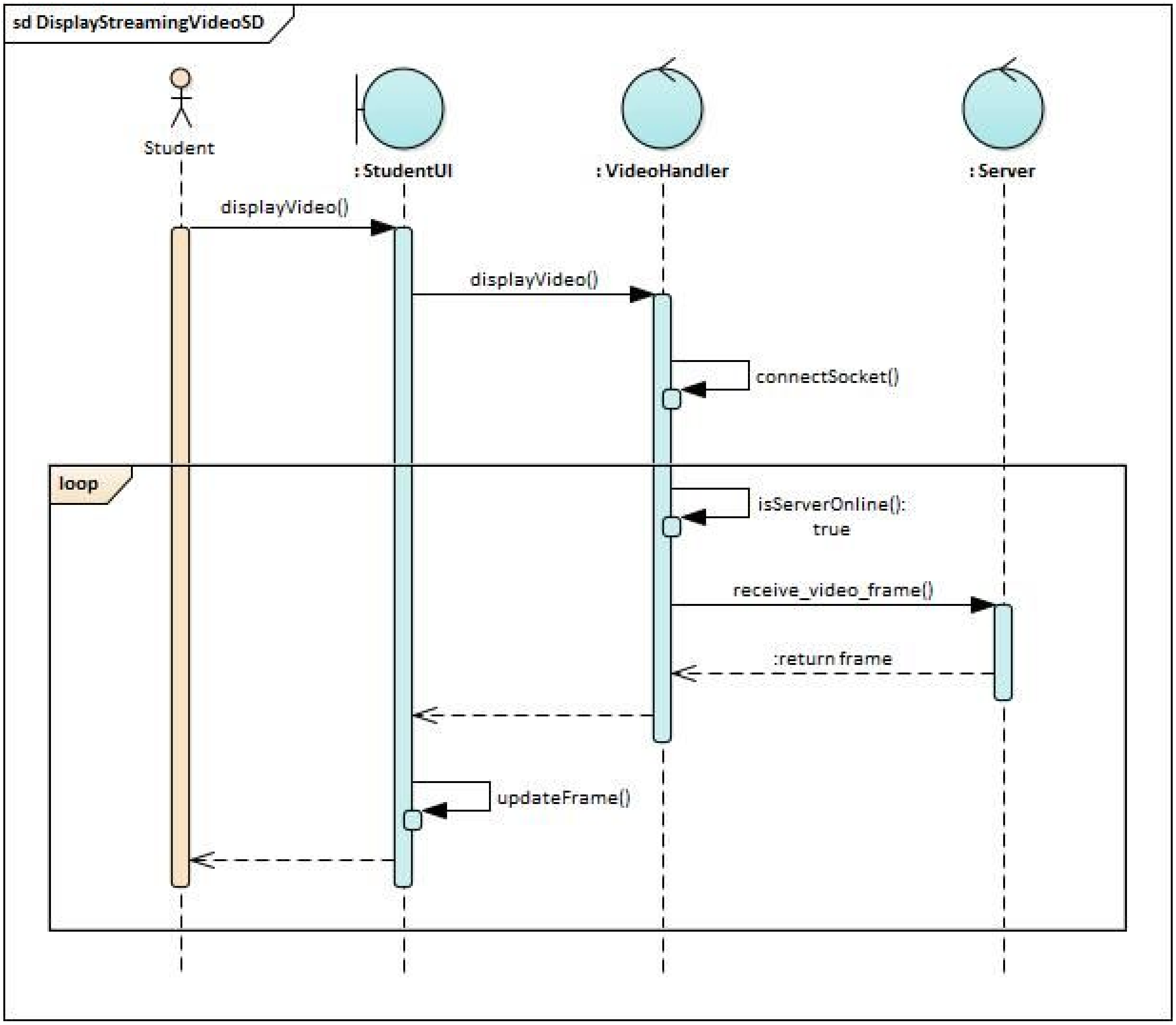
Refresh Room List : BASIC



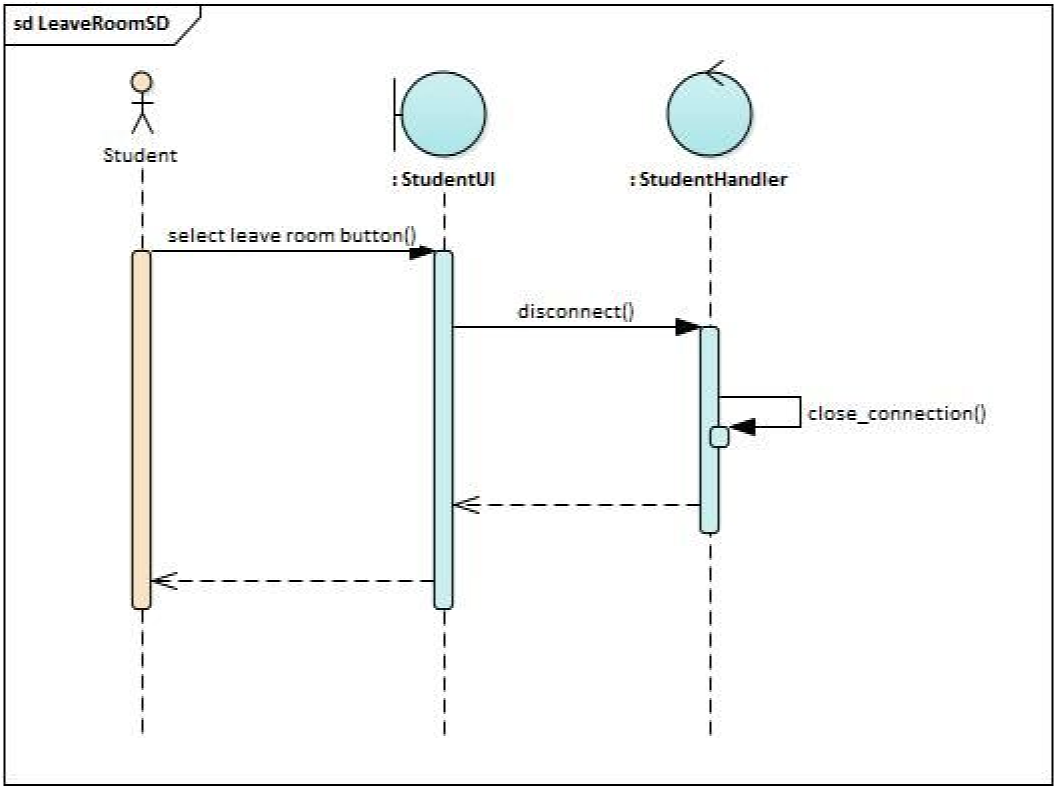
Join Room : BASIC



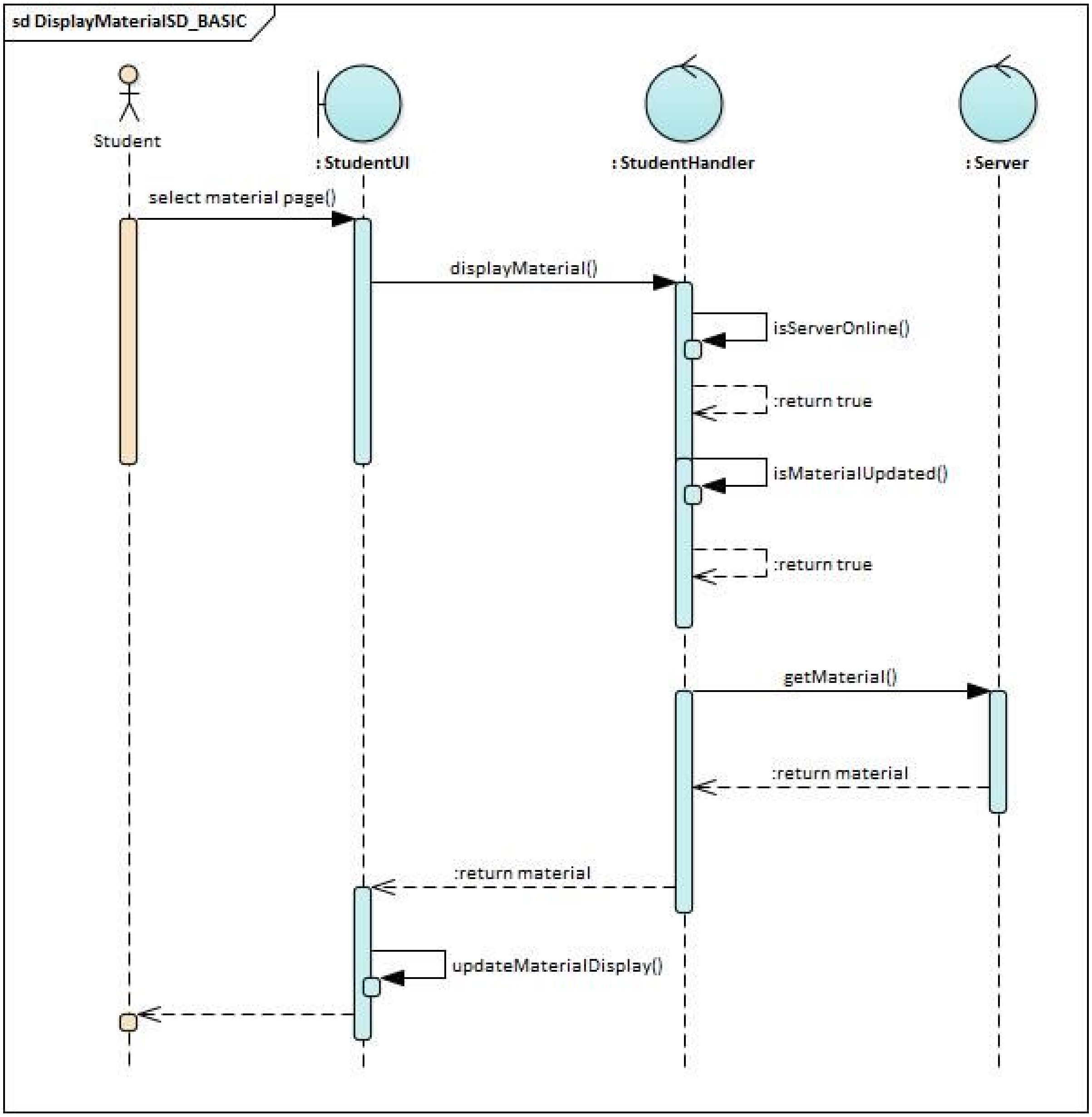
Display Streamed Video : BASIC



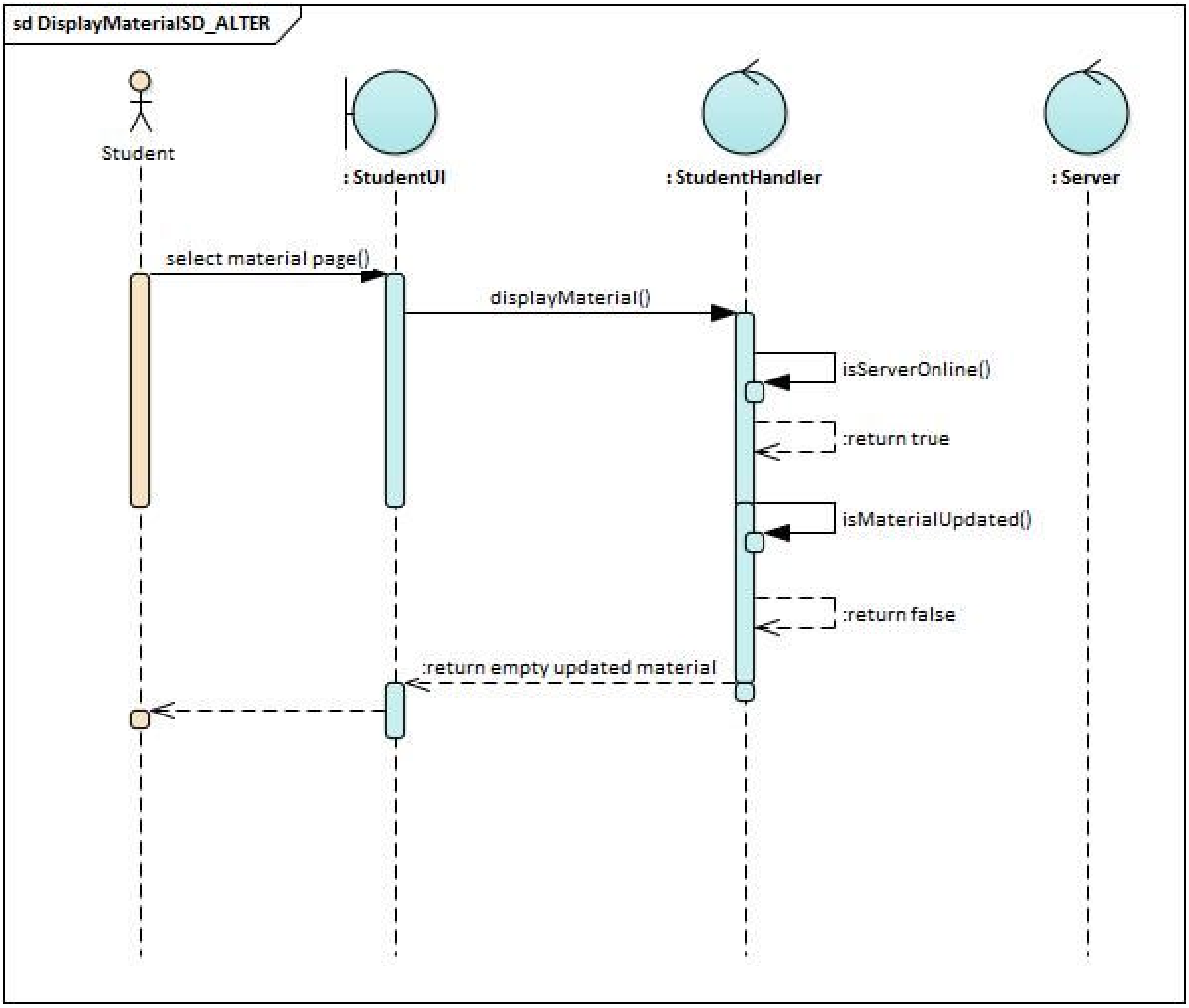
Leave Room : BASIC



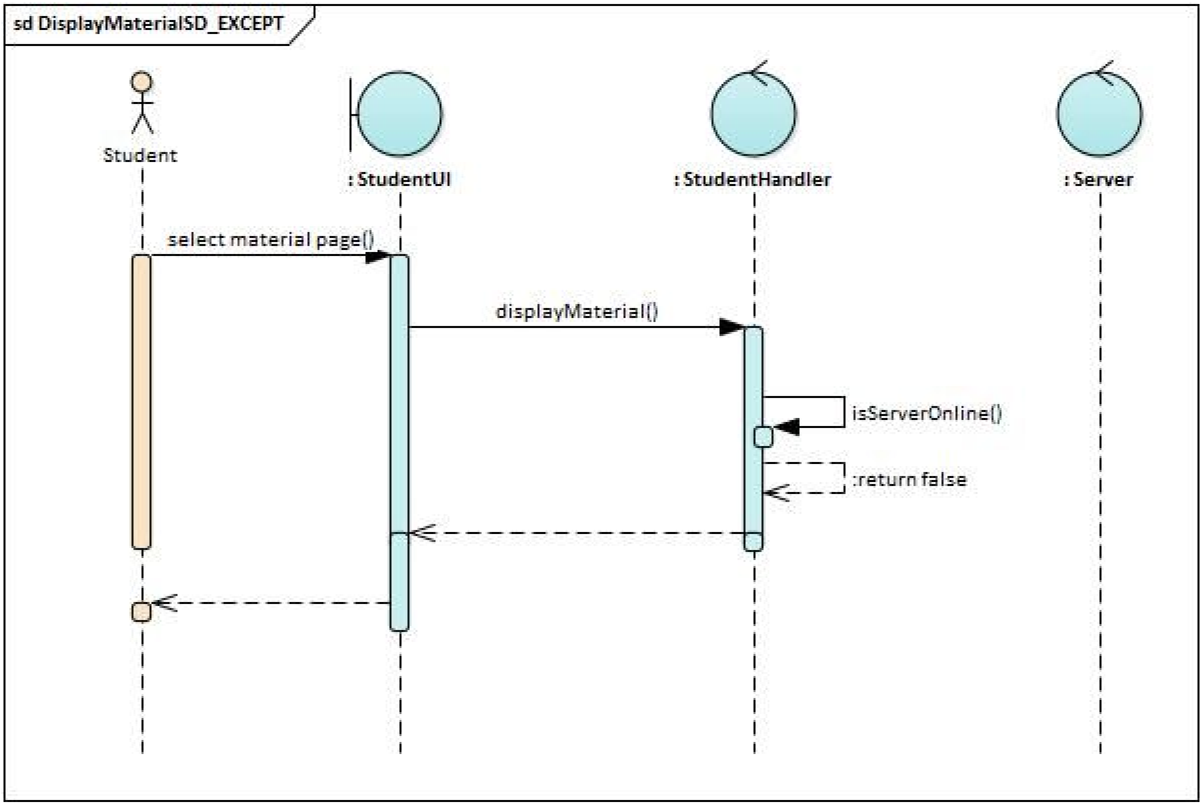
Display Educational Material : BASIC



Display Educational Material : ALTERNATIVE

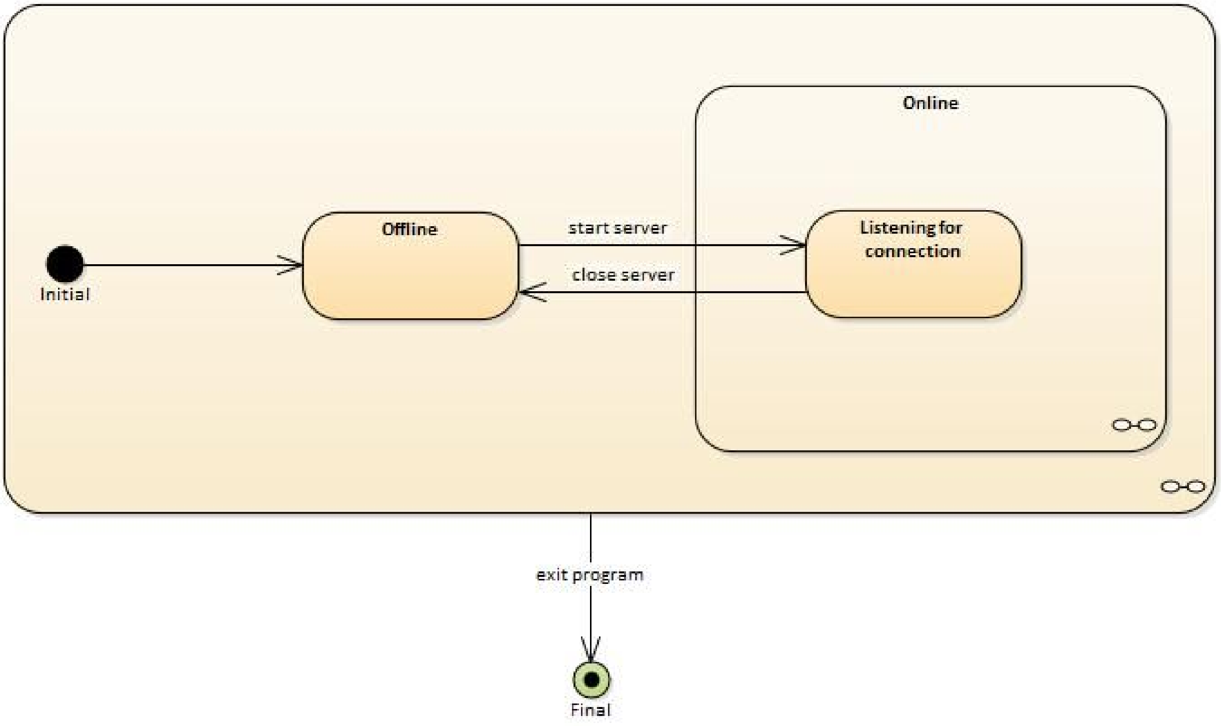


Display Educational Material : EXCEPTIONAL

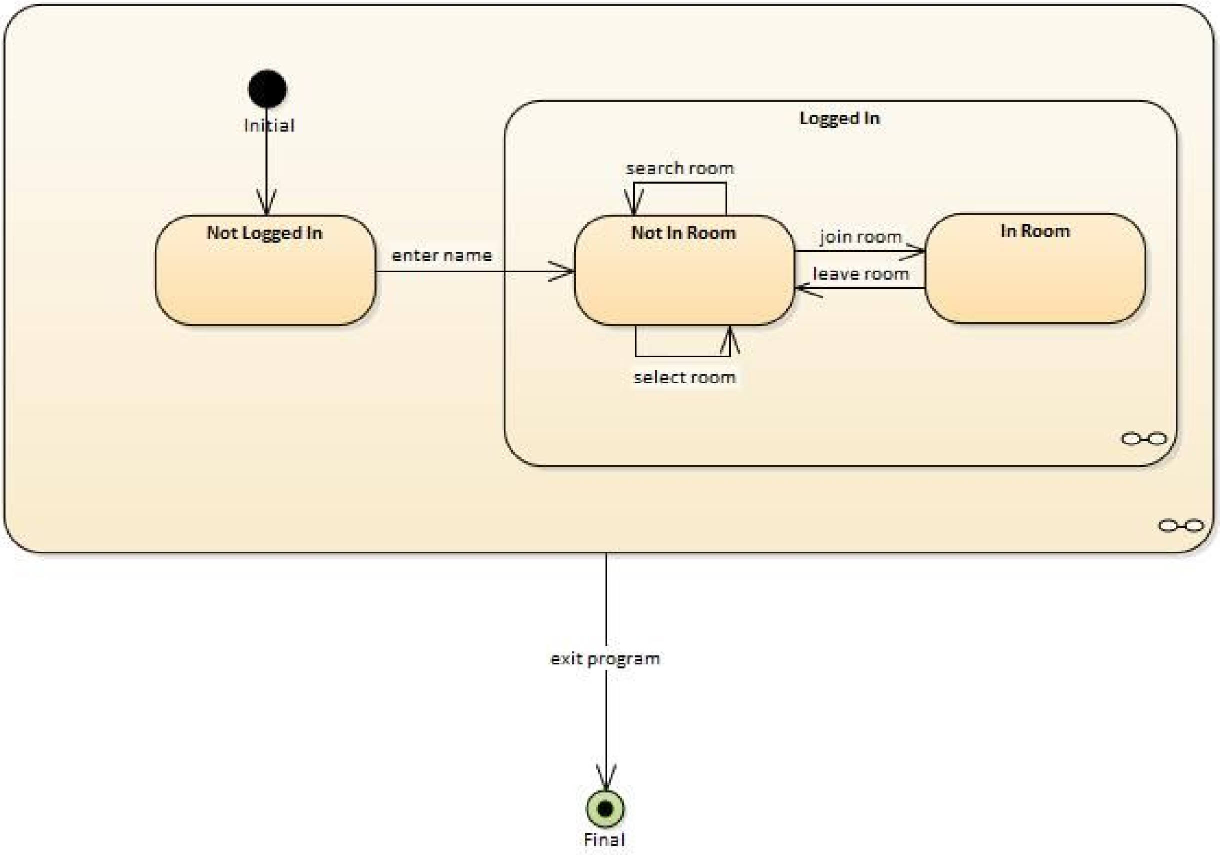


State Chart Diagram

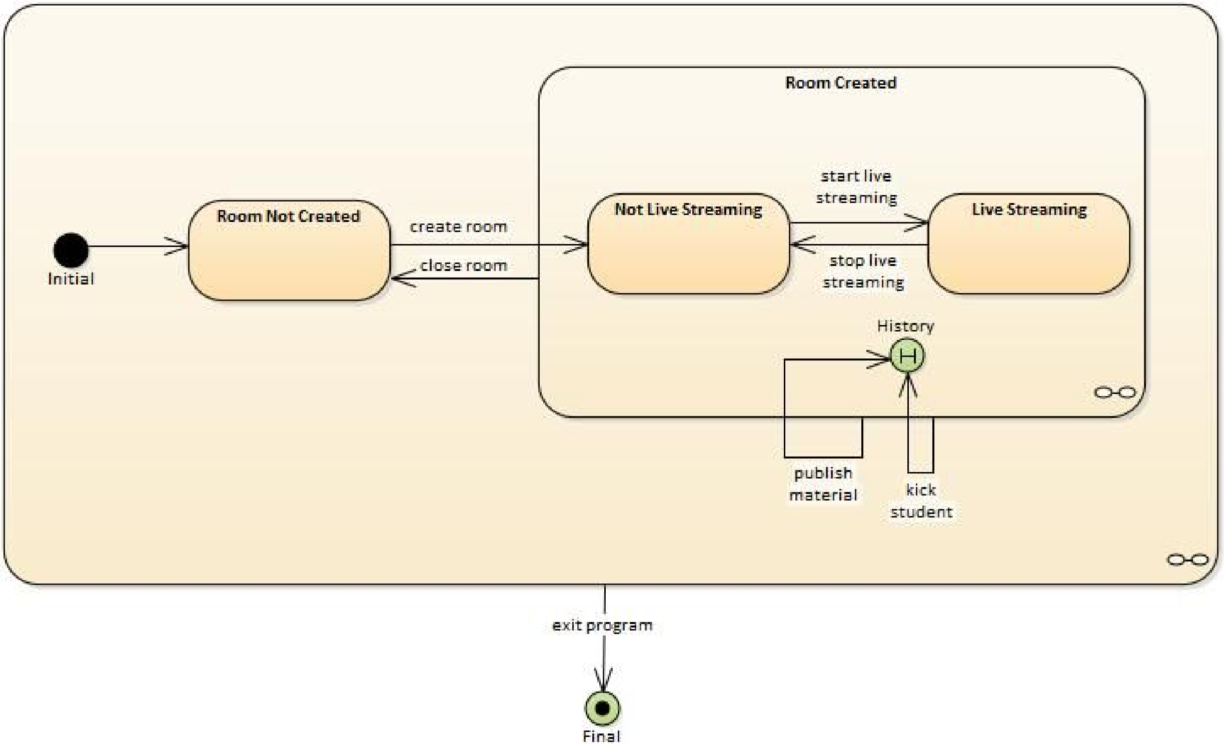
Server



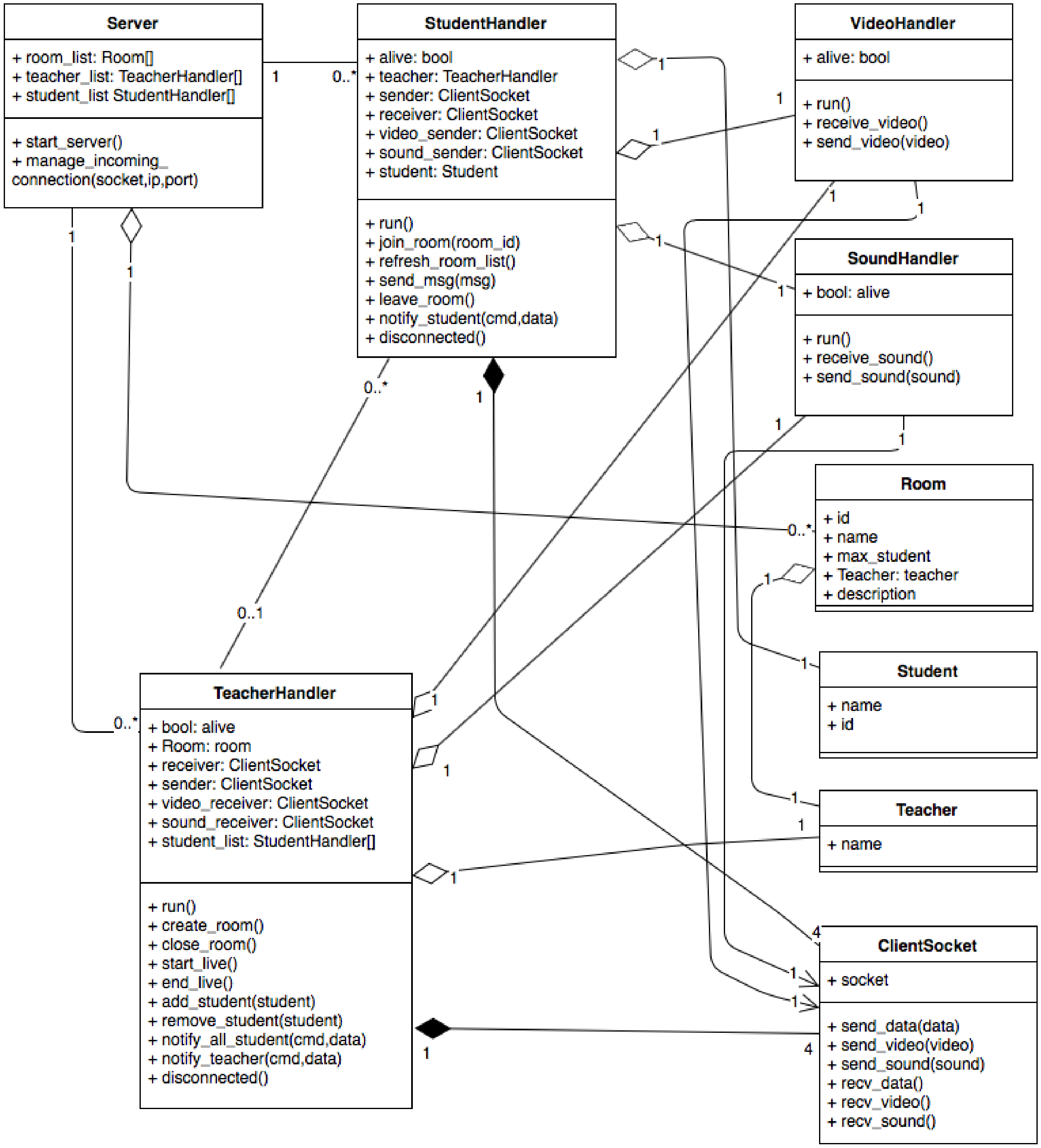
Student

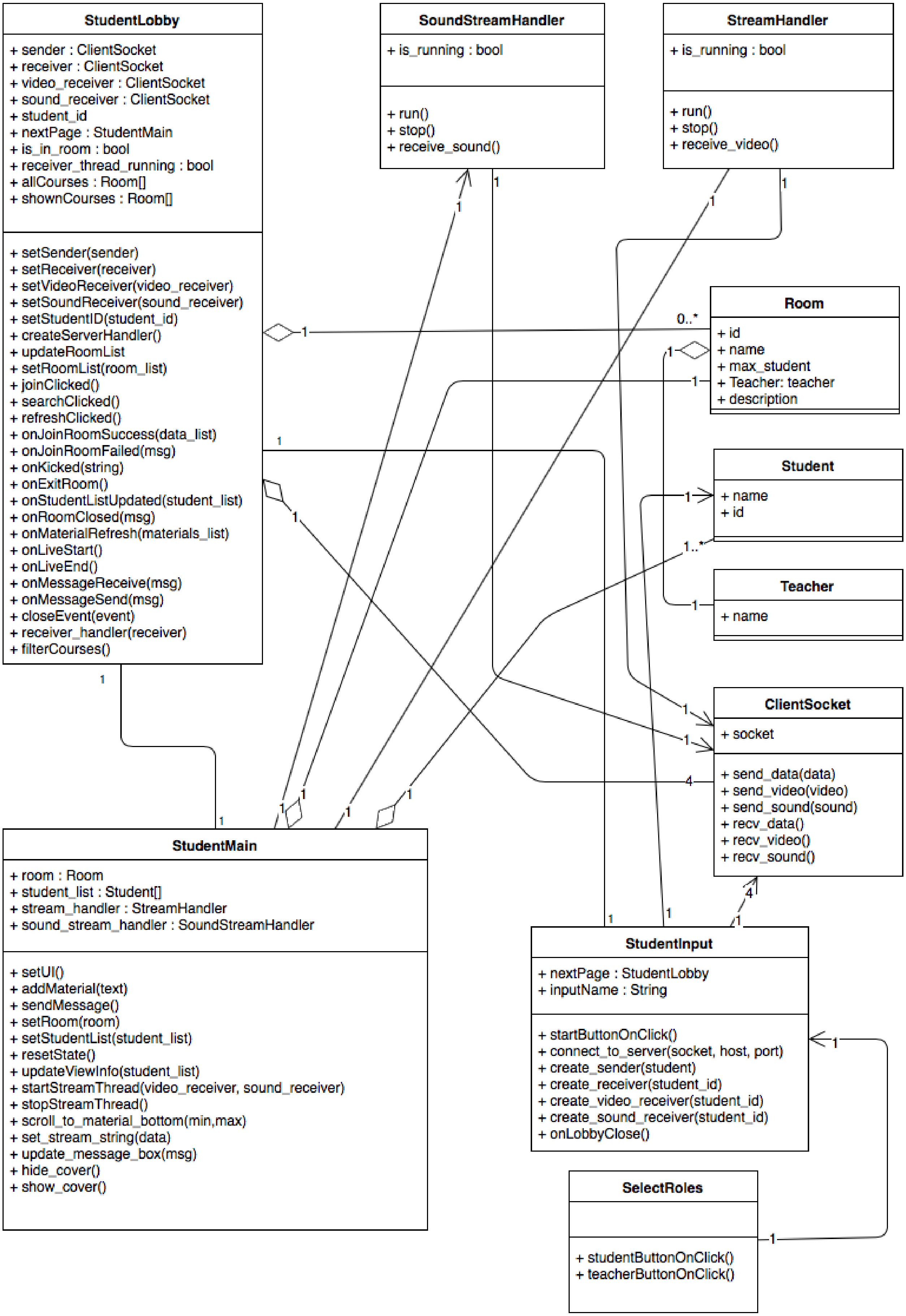

Initial
Logged In
search room
Not Logged In
enter name
Not In Room
join room
leave room
In Room
select room
exit program
Final

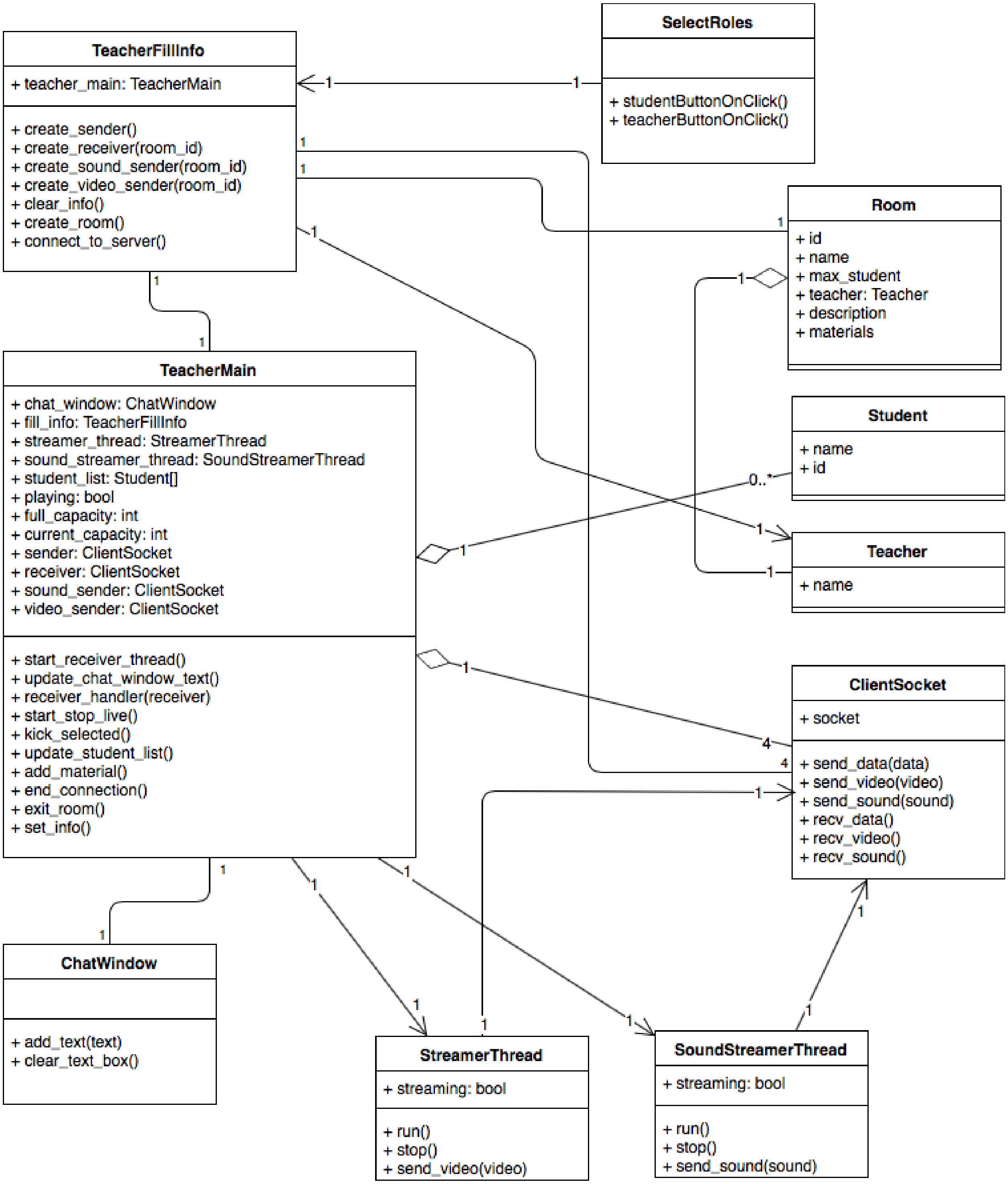
Teacher


Room Created
Initial
Room Not Created
create room
close room
Not Live Streaming
start live streaming
stop live streaming
Live Streaming
History
H
publish material
kick student
exit program
Final

Class Diagram




StudentLobby
+ sender : ClientSocket
+ receiver : ClientSocket
+ video_receiver : ClientSocket
+ sound_receiver : ClientSocket
+ student_id
+ nextPage : StudentMain
+ is_in_room : bool
+ receiver_thread_running : bool
+ allCourses : Room[]
+ shownCourses : Room[]
+ setSender(sender)
+ setReceiver(receiver)
+ setVideoReceiver(video_receiver)
+ setSoundReceiver(sound_receiver)
+ setStudentID(student_id)
+ createServerHandler()
+ updateRoomList
+ setRoomList(room_list)
+ joinClicked()
+ searchClicked()
+ refreshClicked()
+ onJoinRoomSuccess(data_list)
+ onJoinRoomFailed(msg)
+ onKicked(string)
+ onExitRoom()
+ onStudentListUpdated(student_list)
+ onRoomClosed(msg)
+ onMaterialRefresh(materials_list)
+ onLiveStart()
+ onLiveEnd()
+ onMessageReceive(msg)
+ onMessageSend(msg)
+ closeEvent(event)
+ receiver_handler(receiver)
+ filterCourses()
SoundStreamHandler
+ is_running : bool
+ run()
+ stop()
+ receive_sound()
StreamHandler
+ is_running : bool
+ run()
+ stop()
+ receive_video()
Room
+ id
+ name
+ max_student
+ Teacher: teacher
+ description
Student
+ name
+ id
Teacher
+ name
ClientSocket
+ socket
+ send_data(data)
+ send_video(video)
+ send_sound(sound)
+ recv_data()
+ recv_video()
+ recv_sound()
StudentMain
+ room : Room
+ student_list : Student[]
+ stream_handler : StreamHandler
+ sound_stream_handler : SoundStreamHandler
+ setUI()
+ addMaterial(text)
+ sendMessage()
+ setRoom(room)
+ setStudentList(student_list)
+ resetState()
+ updateViewInfo(student_list)
+ startStreamThread(video_receiver, sound_receiver)
+ stopStreamThread()
+ scroll_to_material_bottom(min,max)
+ set_stream_string(data)
+ update_message_box(msg)
+ hide_cover()
+ show_cover()
StudentInput
+ nextPage : StudentLobby
+ inputName : String
+ startButtonOnClick()
+ connect_to_server(socket, host, port)
+ create_sender(student)
+ create_receiver(student_id)
+ create_video_receiver(student_id)
+ create_sound_receiver(student_id)
+ onLobbyClose()
SelectRoles
+ studentButtonOnClick()
+ teacherButtonOnClick()
1
0..*
1
1
1
1..*
1
1
1
4
4
1
1
1
1
1
1
1
1
1
1
1
1
1
1
1


TeacherFillInfo
+ teacher_main: TeacherMain
+ create_sender()
+ create_receiver(room_id)
+ create_sound_sender(room_id)
+ create_video_sender(room_id)
+ clear_info()
+ create_room()
+ connect_to_server()
SelectRoles
+ studentButtonOnClick()
+ teacherButtonOnClick()
Room
+ id
+ name
+ max_student
+ teacher: Teacher
+ description
+ materials
TeacherMain
+ chat_window: ChatWindow
+ fill_info: TeacherFillInfo
+ streamer_thread: StreamerThread
+ sound_streamer_thread: SoundStreamerThread
+ student_list: Student[]
+ playing: bool
+ full_capacity: int
+ current_capacity: int
+ sender: ClientSocket
+ receiver: ClientSocket
+ sound_sender: ClientSocket
+ video_sender: ClientSocket
+ start_receiver_thread()
+ update_chat_window_text()
+ receiver_handler(receiver)
+ start_stop_live()
+ kick_selected()
+ update_student_list()
+ add_material()
+ end_connection()
+ exit_room()
+ set_info()
Student
+ name
+ id
Teacher
+ name
ClientSocket
+ socket
+ send_data(data)
+ send_video(video)
+ send_sound(sound)
+ recv_data()
+ recv_video()
+ recv_sound()
ChatWindow
+ add_text(text)
+ clear_text_box()
StreamerThread
+ streaming: bool
+ run()
+ stop()
+ send_video(video)
SoundStreamerThread
+ streaming: bool
+ run()
+ stop()
+ send_sound(sound)
1
0..*
4