SOFTWARE DESIGN DESCRIPTION

OpenFlexure Microscope

Baybars AYDIN – 2309672

M. Ali BÜYÜKADAM – 2309771

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

1. Introduction

1.1. Purpose of the System

In today's digital world, everyone can become a lab-scientist, since OpenFlexure Microscope makes physical microscopes affordable and accessible for everyone! OFM provides the necessary toolkit for anyone so that one can simply build her/his own microscope and start experimenting with it right away. Furthermore, it supports multiple users so that they can work on the same microscope simultaneously. Users can also automate the process to work easily with big amount of data.

1.2. Scope

In the OpenFlexure Microscope system, users can experiment with the microscope they built; they can run Python scripts on it to automate the periodic and deterministic tasks that they want the microscope to do, and they monitor the results and get large amount of data to work with. Multiple users can work on the same microscope while one user can manage several microscope, which is good for collaboration and examining different microscope environments.

The microscope can only be used for collecting data, processing of the retrieved data is out of the scope of the OpenFlexure Microscope system.

1.3. Stakeholders and Their Concerns

There are three main users for the OpenFlexure system, which are end users, the R&D team and the producers.

End users: End users can be anyone who buys the product and use it for their interest. Students and teachers can use the microscope for educational purposes, while a hobbyist can buy the microscope and examine the microscopic organism he/she found in a lake to satisfy their curiosity. They are also involved in the development of the OFM; they give feedbacks about the software bugs they encounter while using the OFM, or they can simply request a new feature. Finally, these feedbacks goes directly to the R&D team. Note that these feedbacks are related merely with the development of OFM. In case

of customer support or problems related with the product usage, end users should get in touch with Producers instead of the R&D team.

Researchers / Developers (R&D team): Since researchers are also involved in the development process and vice versa, we cannot separate those two group of people. These are the people who does research and development for OpenFlexure microscopes. Research of OFM includes improvement the current state of the microscope or adding new features to it. Development includes the management of the project code base of OFM. The R&D team consists of seven engineers; three of them has PhD in related field and they are mostly responsible for the research related tasks. The R&D team uses **agile methodologies** because customer feedback is one of the main priorities and they need to be able to add new features or fix bugs according to those feedbacks as soon as possible.

Producers: Producers of the OpenFlexure microscopes are responsible for manufacturing of the microscopes. They work closely to the R&D team and they produce the new series of OFM when new version is out. They are also the maintainers of the products since they provide a customer service that end users can make contact with in case of a problem with the product.

2. References

This document is written with respect to IEEE 1016-1998 standard:

IEEE standard for information technology–systems design–software design descriptions. (2009). New York, NY: Institute of Electrical and Electronics Engineers.

Other sources:

Collins, Joel & Knapper, Joe & Stirling, Julian & McDermott, Samuel & Bowman, Richard. (2021). Modern Microscopy with the Web of Things: The OpenFlexure Microscope Software Stack.

3. Glossary

Table 1: Glossary

Term	Definition
OFM	Abbreviation for OpenFlexure Microscope
R & D	Research & Development
Researcher	A person who carries out academic or scientific research
Developer	a person or company that creates new products, especially computer products such as software
Automation	The use or introduction of automatic equipment in a manufacturing or other process or facility.
Microscope	An optical instrument used for viewing very small objects, such as mineral samples or animal or plant cells, typically magnified several hundred times.
LED	a light-emitting diode
DBMS	Database Management Systems
AI	Artificial Intelligence
CRUD	Abbreviation for Create/Read/Update/Delete database operations.

4. Architectural Views

4.1. Context View

In this viewpoint, context of the system with all actors are defined in general and detailed viewpoints. In the context diagram, actors and their interaction with the OpenFlexure Microscope will be explained in general terms. Use case diagrams and the detailed explanations of every possible use case of the system will be specified below the context diagram.

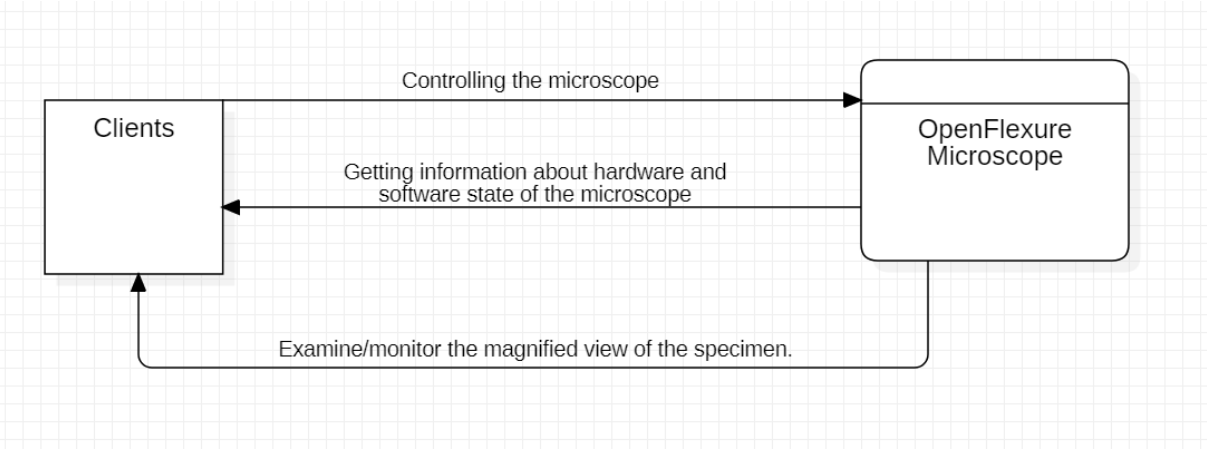

Figure 1: Context Diagram

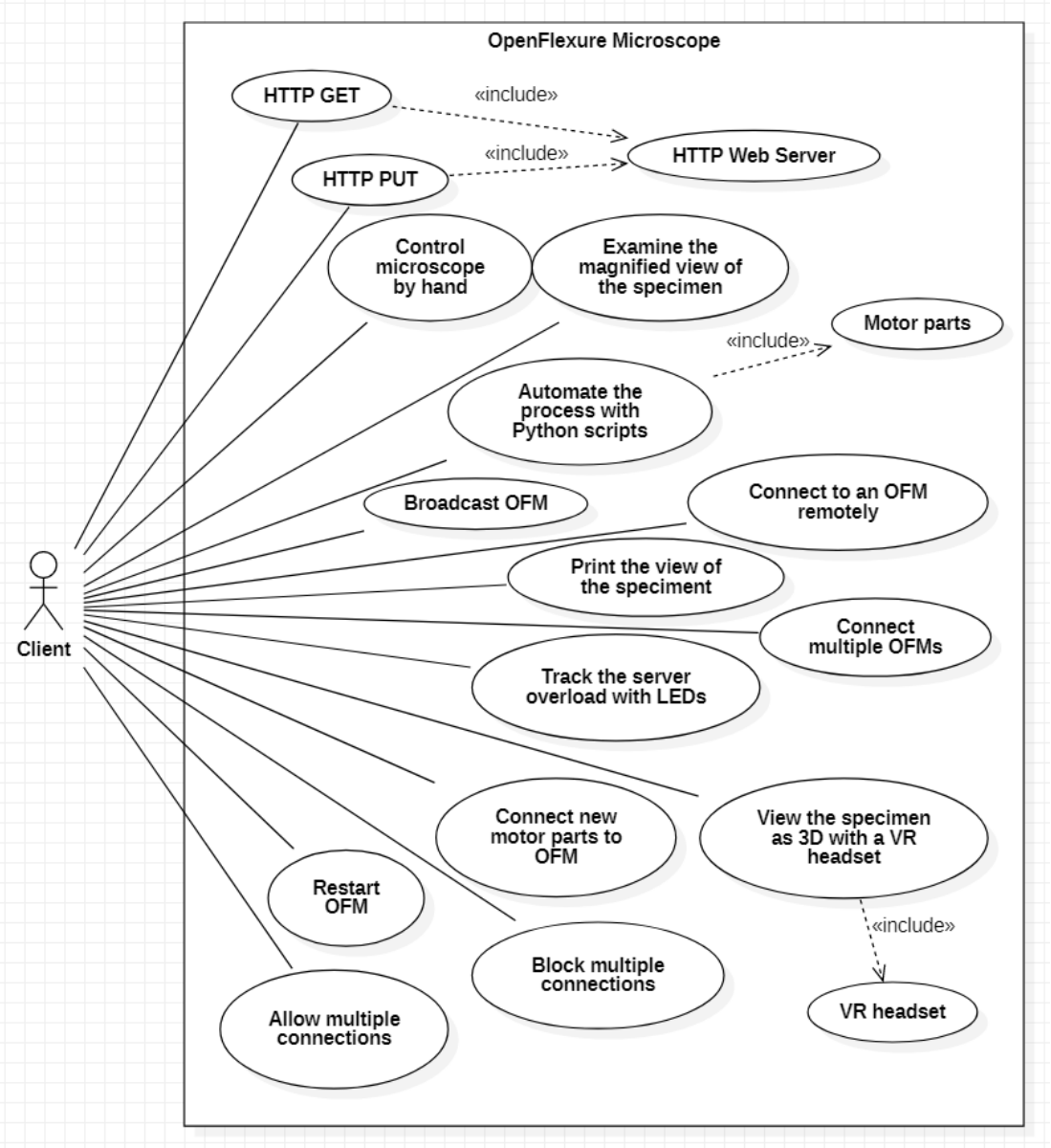

Figure 2: Use Case Diagram

Table 2: Use Case 1

Use Case ID	1
Use Case Name	HTTP GET
Actors	Client
Description	Clients send HTTP GET request over an IP network to get information about components of microscope or the magnified view of the specimen.
Preconditions	1. Clients must be in the same network as the microscope. 2. Clients must know the IP address or hostname of the microscope. 3. Clients must know on which port the microscope interface runs.
Post condition	Clients get the information they requested.
Normal Flow	1. Client sends GET request on one of the components of the microscope or the view of the specimen. 2. HTTP Web server on the microscope handles the request by retrieving the corresponding information from the physical components and sending it back to the client. 3. Client gets the information.
Exceptions	If the request client sends is not a valid HTTP GET request, the server should not bother processing it.

Table 3: Use Case 2

Use Case ID	2
Use Case Name	HTTP PUT
Actors	Client
Description	Clients send HTTP PUT request over an IP network to change information about components of microscope.
Preconditions	1. Clients must be in the same network as the microscope. 2. Clients must know the IP address or hostname of the microscope. 3. Clients must know on which port the microscope interface runs.
Post condition	Specified property of the microscope is being updated according to the PUT request sent by the client.
Normal Flow	1. Client sends PUT request on one of the components of the microscope. 2. HTTP Web server on the microscope handles the request by updating the corresponding value of the physical component. 3. Client can observe the change in the next run.
Exceptions	If the request client sends is not a valid HTTP PUT request, the server should not bother processing it.

Table 4: Use Case 3

Use Case ID	3
Use Case Name	Control microscope by hand
Actors	Client
Description	Clients control components on the microscope by hand. For example, they can zoom in the view or change the degree of the optic lens.
Preconditions	1. Physical components must work properly in order clients to control them by hand.
Post condition	Working conditions of the physical components should not change after clients use them.
Normal Flow	1. Client changes the state or the use case of components of the microscope by hand. 2. The microscope processes the change. 3. Client can observe the change by retrieving information from the microscope.
Exceptions	The client might damage the component which might result in a malfunction.

Table 5: Use Case 4

Use Case ID	4
Use Case Name	Examine the magnified view of the specimen.
Actors	Client
Description	Clients can get the view of the specimen by connecting the microscope camera to a display-unit via HDMI cable, or they can get it over the network.
Preconditions	1. There should be a specimen, so the client can monitor it. 2. There should be a camera connected to the microscope. 3. HDMI cable should work properly in the case of a HDMI cable connection. 4. There should be a working network connection in the case of a network medium.
Post condition	The display-unit must display the monitoring data coming from the microscope properly.
Normal Flow	1. Client controls the microscope by hand or by sending PUT requests to the microscope. 2. The microscope processes the data coming from the client and sends the updated view of the specimen back the client. 3. Client gets the view of the specimen over HDMI cable or IP network. 4. Client observes the view of the specimen.
Exceptions	Connection problem related with HDMI cable or IP network can interrupt the monitoring of the specimen.

Table 6: Use Case 5

Use Case ID	5
Use Case Name	Automate the process with Python scripts.
Actors	Client
Description	Clients can automate the various tasks using Python scripts. These tasks can be zooming in and out in order with a given time, or following a move pattern with the optic lens using motor parts etc.
Preconditions	1. Clients must write Python scripts that will automate the process using motor parts. 2. External motor parts should be added in proper locations of the microscope.
Post condition	Scripts should not cause malfunction or physical damage on the components or motor parts.
Normal Flow	1. Clients manually add the motor parts near the components that they want to automate, and they integrate these parts to the microscope. 2. Clients write the Python script that will make the newly added motor parts meaningful. 3. Target components of the microscope will be automated as predicted with the help of motor parts and Python scripts.
Exceptions	A bad written script can physically damage the components or cause malfunction. For example, a script that turns the lens controller over and over might burn the motor parts and make them unusable.

Table 7: Use Case 6

Use Case ID	6
Use Case Name	Connect to an OFM remotely
Actors	Client
Description	Clients can connect to an OFM remotely and they can start experimenting with it.
Preconditions	1. The OFM that client wants to connect remotely must turn on "allow external connections" option. 2. The client should know the public IP address of the OFM.
Post condition	The OFM should cut the connection if the remote connected client goes idle for more than one hours.
Normal Flow	1. Client sends a connection request to a remote OFM using the public IP address of the OFM. 2. OFM accepts the request and connects the client to itself. 3. After being connected, the client can control the microscope using HTTP request. 4. Client gets HTTP responses from the OFM. 5. Client can also watch the public broadcasting of the OFM to observe the changes.
Exceptions	In a scenario where multiple clients are connected to an OFM and they are sending simultaneous requests, the OFM can also send multiple requests simultaneously. However, the requests should be independent from each other. For example, if OFM receives GET and PUT request for a motor part X at the same time, it should use a lock mechanism and process the requests separately, since PUT request can change the value of the response for the GET request.

Table 8: Use Case 7

Use Case ID	7
Use Case Name	Broadcast OFM
Actors	Client
Description	The client owning the OFM can broadcast the current state of the microscope and the specimen over public network of the OFM server, in case multiple clients are connected.
Preconditions	1. The client can activate broadcasting only when there are more than one client connected.
Post condition	OFM should stop broadcasting automatically when there is only one or zero user is connected.
Normal Flow	1. Client activates “broadcasting” from the OFM software. 2. OFM checks the connected client count and if there is only one (the owner), it does nothing. 3. If connected client count is greater than one, it starts broadcasting its current state and the view of the specimen over its public network. 4. Connected clients can view the microscope status from broadcasting instead of sending GET requests for every individual part.
Exceptions	The OFM server is free to stop broadcasting in case of excessive load in the server. After stopping the broadcasting, it individually sends a HTTP response to all the connected clients to inform that the broadcasting has ended due to server overload.

Table 9: Use Case 8

Use Case ID	8
Use Case Name	Connect multiple OFMs
Actors	Client
Description	Clients can connect multiple OFMs and process data coming from different OFMs in real time.
Preconditions	1. All the OFMs that client wants to connect remotely must turn on "allow external connections" option. 2. The client should know the public IP address of all of the OFMs.
Post condition	OFMs should cut the connection if the remote connected client goes idle for more than one hours.
Normal Flow	1. Client connects to multiple OFMs. 2. Client starts retrieving data from multiple OFMs. 3. Client can use those data coming from different OFMs simultaneously. For example, the client can get temperature of motor part X from different OFMs that have motor part X, and calculate the average temperature of motor part X by feeding the data into the calculator.
Exceptions	Data flow from OFMs can be cut anytime, so it is client's responsibility to handle such cases in the program he/she wrote that processes multiple data.

Table 10: Use Case 9

Use Case ID	9
Use Case Name	Print the view of the specimen
Actors	Client
Description	The current view of the specimen is printed when client sends a print request to OFM.
Preconditions	1. There should be a printer recognized by the OFM software.
Post condition	None
Normal Flow	1. Client selects “print the current view of specimen” option in the OFM software. 2. OFM software takes a photo of the current view of specimen using its camera. 3. Client can modify how the image looks like in the internal image editor of OFM. Then confirms the print operation. 4. After confirmation, OFM software sends print request to the connected printer. 5. Printer prints the photo of the current view of the specimen.
Exceptions	To overcome issues related with printer connection, OFM does not wait for a response whether the printing was successful or not. Therefore, it is client’s responsibility to send the print request again in case of a connection problem.

Table 11: Use Case 10

Use Case ID	10
Use Case Name	Track the server overload with LEDs.
Actors	Client
Description	Client can track of the server overload using different colors of LEDs externally connected to the OFM.
Preconditions	1. Client should buy LEDs separately.
Post condition	1. It is client's responsibility to remove and repair in case of a LED burn out.
Normal Flow	1. Client externally connects LEDs to the OFM. At least three different colors of LEDs should be used in order to reflect the OFM overload correctly. 2. Client writes a Python script and sets different thresholds for different overload values. For example, if the client has three LEDs (green, yellow and red), he/she may choose the OFM server overload thresholds as %0-40, %41-80, and %80+. He/she may select green LED for the first interval, and when the OFM server overload exceeds %40, the yellow LEDs starts to glow. Similarly, red LEDs starts to glow after overload exceeds %80.
Exceptions	OFM server should handle the cases when a LED burns out and it should stop trying to active the LED.

Table 12: Use Case 11

Use Case ID	11
Use Case Name	View the specimen as 3D with a VR headset
Actors	Client
Description	Client can view of specimen as 3D using VR headsets. The 3D view is created using AI.
Preconditions	1. A VR headset is required. 2. A software that uses AI to create 3D view is required.
Post condition	None
Normal Flow	1. Client connects VR headset to OFM and wears it. 2. 2D view of the specimen is fed into a software that uses AI to create 3D views from the existing 2D views. 3. Client views the specimen as 3D .
Exceptions	It is responsibility of the AI software to deal with corrupted images and internally handle them to give the best possible result.

Table 13: Use Case 12

Use Case ID	12
Use Case Name	Connect new motor parts to OFM
Actors	Client
Description	Client physically connect variety of different motor parts to OFM and use them with different purposes.
Preconditions	1. The motor part that client wants to add to OFM should be compatible with the OFM.
Post condition	Motor parts should not affect the workability of other parts of the microscope or other motor parts after they are activated via a Python script.
Normal Flow	1. Client connects a new motor part to the microscope. 2. Client writes a Python script to use that motor part for some purpose. 3. After accomplishing its duty, motor stops peacefully (does not affect the other parts) and goes idle until it is activated from a script.
Exceptions	In case of physical defect in a motor part, it should be recognized as defected by OFM and OFM should stop communicating with it. Client should remove the motor part from the OFM afterwards.

Table 14: Use Case 13

Use Case ID	13
Use Case Name	Block multiple connections
Actors	Client
Description	Clients can block multiple connections of his/her OFM, if he/she does not want other clients to connect to the OFM.
Preconditions	1. Clients should be the owner of the OFM in order to block multiple connections.
Post condition	Clients should be the owner of the OFM in order to enable multiple connections.
Normal Flow	1. Client selects “block multiple connections” from the OFM software. (Client needs to be the root user of the OFM) 2. OFM server stops listening requests from other clients.
Exceptions	A non-root user cannot disable multiple connections on an OFM, in case of a multiple wrong attempts; the OFM blocks the IP address of the user for couple of minutes.

Table 15: Use Case 14

Use Case ID	14
Use Case Name	Allow multiple connections
Actors	Client
Description	Clients can allow multiple connections on an OFM they own.
Preconditions	2. Clients should be the owner of the OFM in order to allow for multiple connections.
Post condition	Clients should be the owner of the OFM in order to disable multiple connections.
Normal Flow	3. Client selects “allow multiple connections” from the OFM software. (Client needs to be the root user of the OFM) 4. OFM server starts listening requests from other clients.
Exceptions	A non-root user cannot enable multiple connections on an OFM, in case of a multiple wrong attempts; the OFM blocks the IP address of the user for couple of minutes.

Table 16: Use Case 15

Use Case ID	15
Use Case Name	Restart OFM
Actors	Client
Description	Clients can restart the OFM.
Preconditions	3. Clients should be the owner of the OFM in order to be able to restart it.
Post condition	None
Normal Flow	5. Client selects “restart” from the OFM software. (Client needs to be the root user of the OFM) 6. OFM server and all components restart.
Exceptions	A non-root user cannot restart an OFM, in case of a multiple wrong attempts; the OFM blocks the IP address of the user for couple of minutes.

4.2. Composition View

In this view, components and interfaces between them are shown in a high-level view, and they are explained in detail in the related sections.

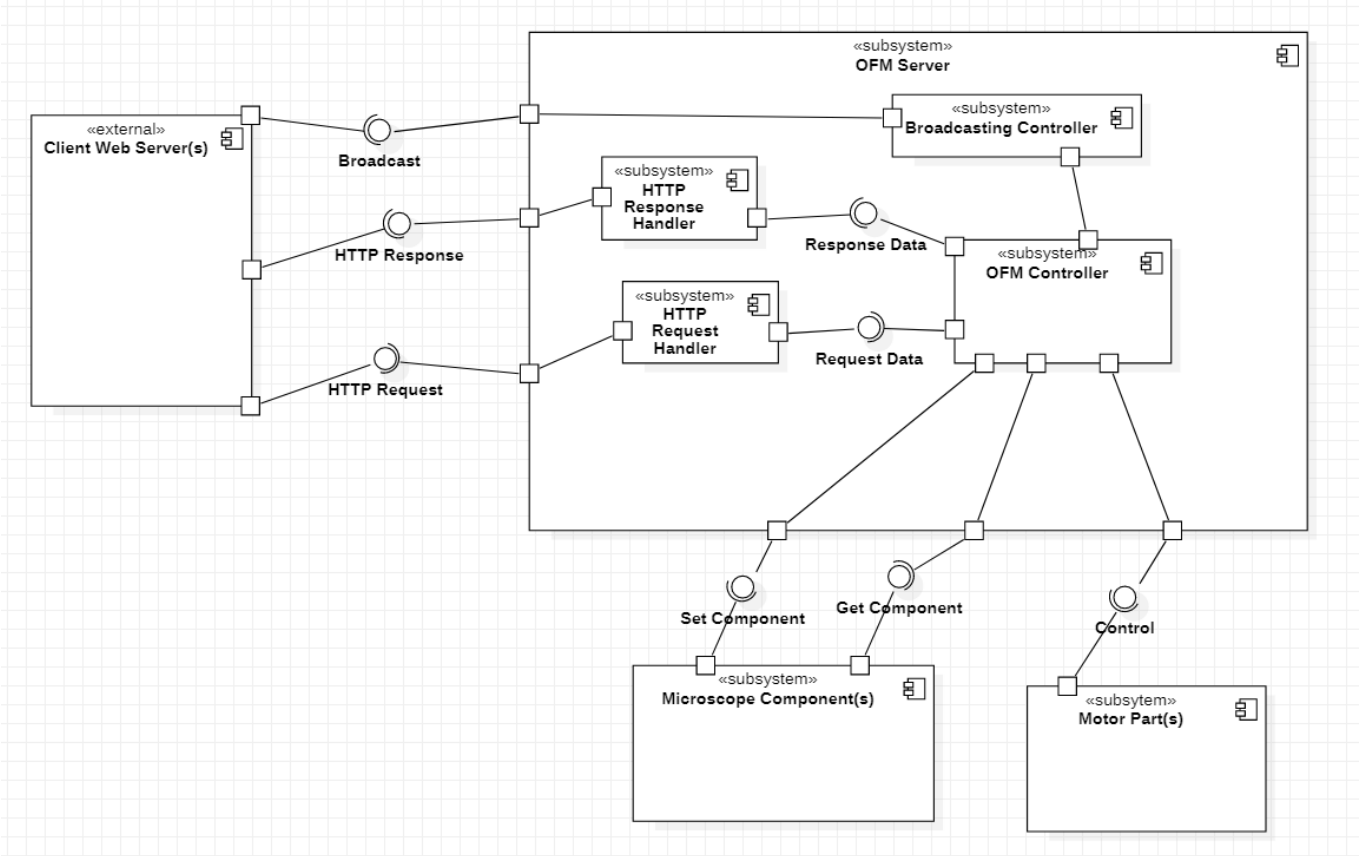

Figure 3: Component Diagram

Design Rationale:

- OFM Server is responsible for communicating with clients and controlling the microscope.
- OFM Server has dedicated HTTP Response and Request handlers, Broadcasting controller and OFM controller.
- HTTP Request handler is responsible for handling requests coming from client web servers. It is capable of handling multiple request simultaneously, since multiple clients might connect to the same OFM at once. It gets requests from clients, classifies them, and sends them to OFM Controller.
- HTTP Response handler is responsible for sending responses to the relevant clients. It is capable of sending multiple responses simultaneously, since multiple clients might connect to the same OFM at once. It gets required information from OFM Controller and uses it to prepare responses. Then, it sends those requests to the corresponding clients.

- Broadcasting controller is responsible for managing broadcasting operation of OFM. After owner of the OFM enables broadcasting, the OFM starts broadcasting over its public IP address. Then the remotely connected clients can start monitoring the broadcast using the same IP address.
- OFM controller is the key element of the OFM server. It interacts with both clients and the microscope (components and motor parts). For example, if HTTP requests handler gets a GET request for a temperature of a component, it sends this request to OFM Controller. OFM Controller then signals microscope using Get Component interface with the component name and property, which is temperature in this case, and waits for response from the microscope. After receiving the response, which includes the current temperature of the component for this case, OFM controller sends it to HTTP Response handler.
- Client Web Server is the clients' computers over which they send HTTP request to the OFM server. Sending HTTP requests and getting HTTP responses, clients can control the microscope or get the specimen data.
- Microscope components do not include a server installed in them; however, they can interact with OFM Controller using hardware signals. OFM Controller can generate a signal on the microscope components. All the finite number of operations that can be performed by the components are mapped into a signal. For example, OFM Controller can generate a "get_temperature" signal on the component, and the component gets and sends its temperature to OFM Controller over an Analog to Digital channel.
- Motor Parts are manipulated via the Python Scripts, which OFM Controller has access to. Data transfer between Motor parts and OFM controller is provided with an Ethernet cable.

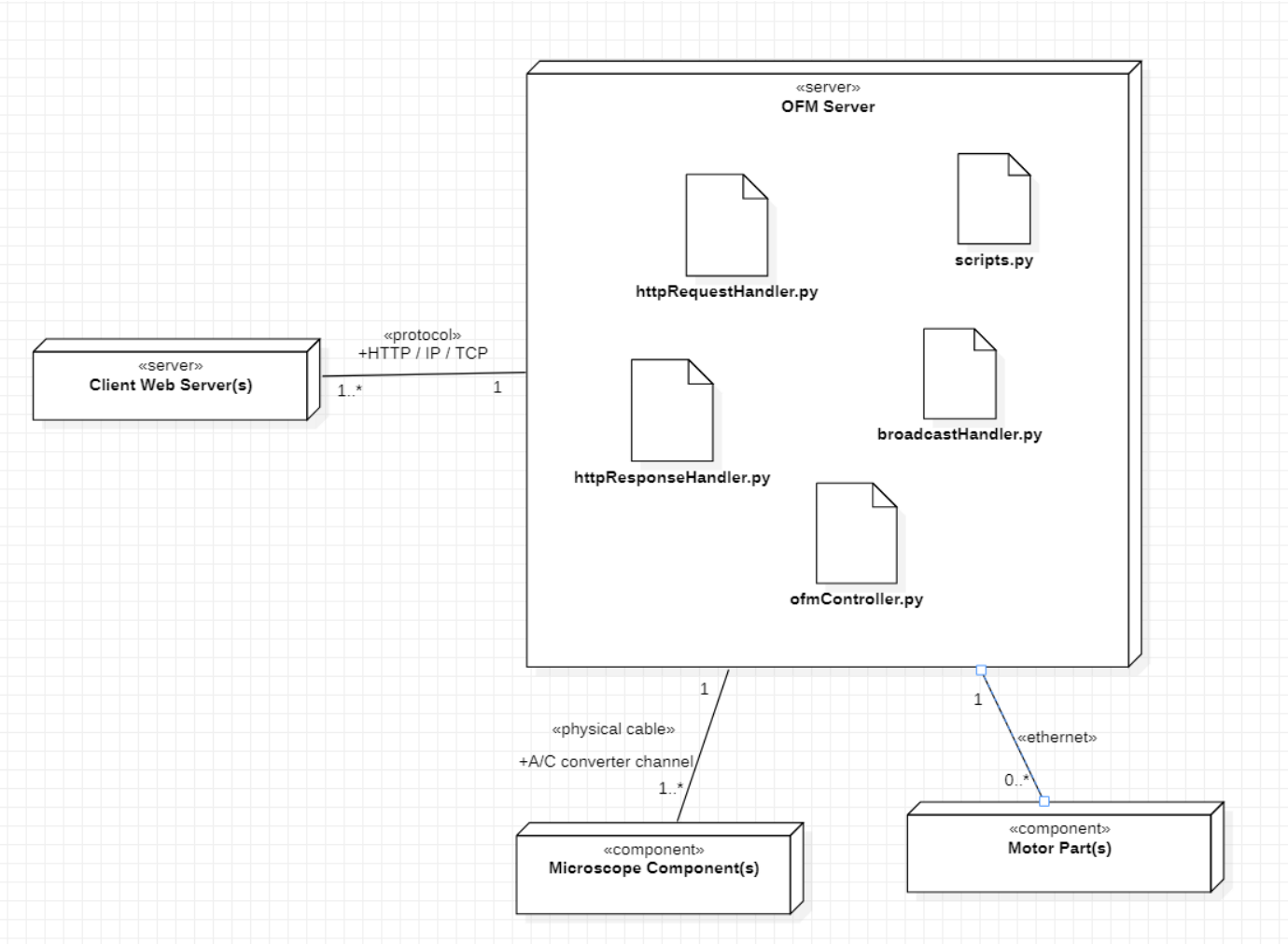

Figure 4: Deployment Diagram

Design Rationale:

- Python is used in the OFM server: Handlers, OFM controller and the scripts for motor parts are all written in Python.
- Connection between Client Web Server and OFM Server is provided via HTTP, IP and TCP protocols. HTTP is the protocol for request and responses. IP is used for broadcasting of OFM, and it is also the main interface of OFM. For the HTTP requests and responses, the data is transferred with TCP packages in order to prevent possible data losses.
- Microscope is connected to OFM controller via a physical cable which carries analog electrical signals. This analog digital signal is then converted into a digital signal in OFM controller. For example, the digital signal that OFM controller received will be the current temperature of the component, if the controller asked for the current temperature.

- Connection between Motor Parts and OFM Controller is provided with Ethernet cables. Motor parts do not directly communicate with microscope components, they rather communicate with OFM controller. For example, after updating or creating a Python script, we sent this to the corresponding Motor part using the Ethernet cable.

4.3. Information View

In this view, the organization and the relations of the data that will be stored with the operations of the system that create, use, modify and delete the data will be specified.

In addition, the effects of the system operations on the data will be examined in terms of CRUD operations.

Interfaces

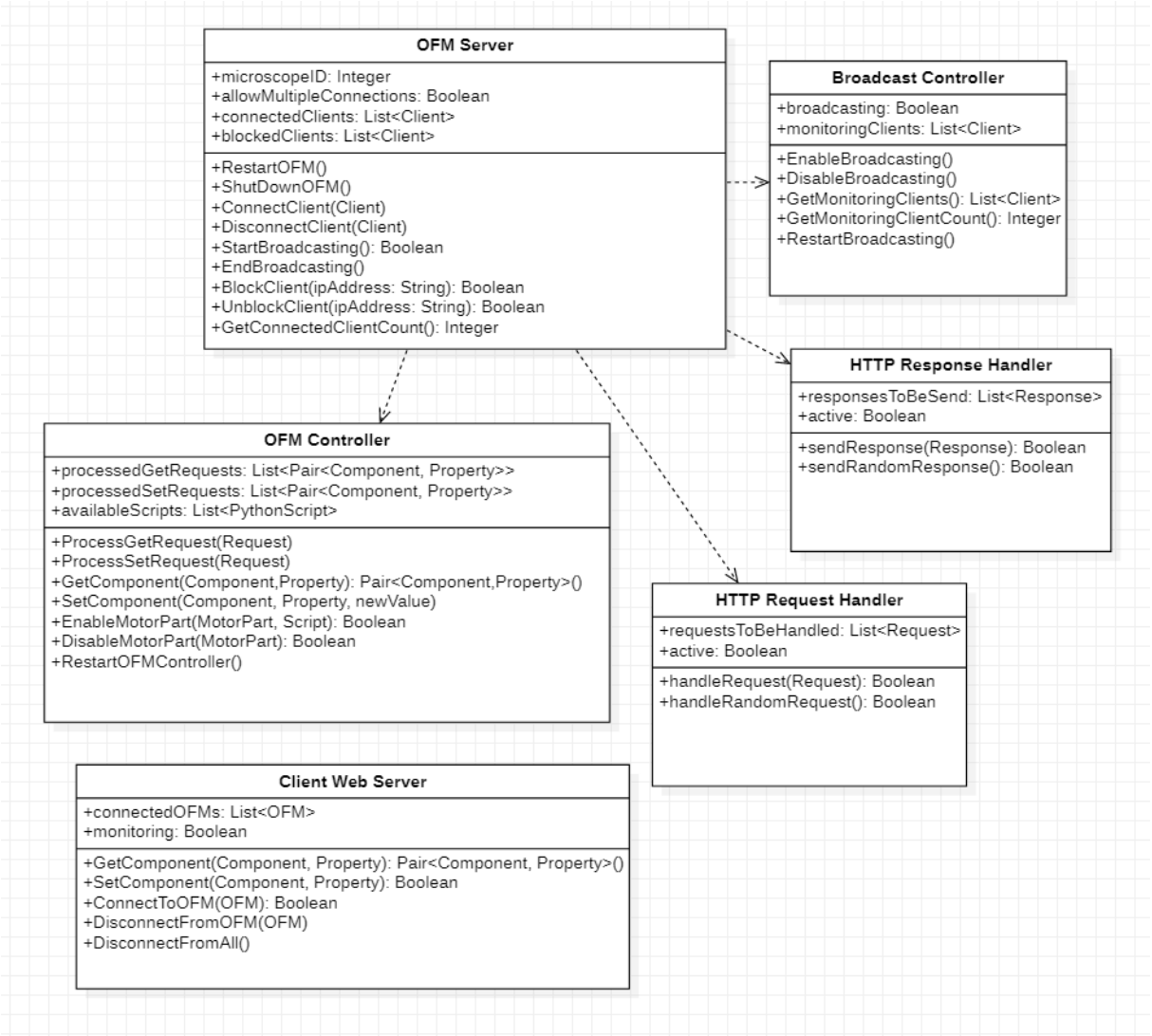

Figure 5: Interface Class Diagram

Table 17: Operation Descriptions

OPERATION	DESCRIPTION
RestartOFM	Restarts the whole OFM.
ShutDownOFM	Shuts down the whole OFM.
ConnectClient	Connects client to the OFM.
DisconnectClient	Disconnects client from the OFM.
StartBroadcasting	OFM server tries to start broadcasting. If there are more than one user connected, it calls EnableBroadcasting on from the Broadcast Controller and returns True. Otherwise, it does not start broadcasting indicating that there is no enough clients connected, and returns False.
EndBroadCasting	Ends broadcasting if there the OFM is already broadcasting by calling DisableBroadcasting from Broadcast Controller. If the OFM currently does not broadcasting, calling EndBroadCasting has no affect.
BlockClient	Blocks client IP address and does not accept further requests from that client until it is unblocked.
UnblockClient	Unblocks client IP address and starts accepting requests from that client.
GetConnectedClientCount	Returns the length of connectedClients List.
EnableBroadcasting	OFM starts broadcasting at its public interface. Connected clients can monitor this action.
DisableBroadcasting	OFM stops broadcasting and informs connected clients that is stopped broadcasting. If connected user count drops to one or less, the OFM automatically disables broadcasting.
GetMonitoringClients	Gets list of clients who are currently connected to the OFM and monitoring the broadcasting. Returns empty list if broadcasting is disabled.
GetMonitoringClientCount	Returns the length of monitoringClients List.
RestartBroadcasting	OFM stops and starts broadcasting. While starting, connected users are informed about the restart of the broadcasting.
ProcessGetRequest	Get request is processed, and Component and Property data is added to processedGetRequests List.
ProcessSetRequest	Set request is processed, and Component and Property data is added to processedSetRequests List.
GetComponent	The current value of (Component, Property) pair is returned.
SetComponent	Specified Property of the component is set to newValue.
EnableMotorPart	Motor part is enabled to run with the provided Python Script.
DisableMotorPart	Motor part stops functioning.
RestartOFMController	Restarts OFM Controller unit.
ConnectToOFM	Client tries to establish a connection with the OFM. Returns True if the connection is successfully established. If connection cannot be established, due to OFM shutting down or the client being blocked, returns False.
DisconnectFromOFM	Client disconnects from the OFM.
DisconnectFromAll	Clients disconnects from all the OFMs that it is connected to.

handleRequest	HTTP Request Handler gets request from client and transfers it to OFM Controller after some arrangement.
handleRandomRequest	HTTP Request Handler calls handleRequest on a random request in requestsToBeHandled List.
sendResponse	HTTP Response Handler gets the meaningful response data from OFM Controller and creates a HTTP response using that data, and sends it back to the client.
sendRandomResponse	HTTP Response Handler calls sendResponse on a random response in responsesToBeHandled List.

Table 18: Operation Design

OPERATION	**INPUT**	**OUTPUT**	**EXCEPTIONS**
RestartOFM	-	-	Communication error between components and OFM server
ShutDownOFM	-	-	Unexpected shutdown causing physical damage on components
ConnectClient	Client	-	Invalid client id
DisconnectClient	Client	-	Disconnecting client with invalid id
StartBroadcasting	-	Boolean	Server overload and unexpected shutdown
EndBroadCasting	-	-	Errors occur when informing clients
BlockClient	ipAddress	Boolean	Block client with invalid id
UnblockClient	ipAddress	Boolean	Unblock client with invalid id
GetConnectedClientCount	-	Integer	Change in count while processing the count we get from the last call
EnableBroadcasting	-	-	No checking involved, might start when there is no connected user
DisableBroadcasting	-	-	Data loss
GetMonitoringClients	-	List<Client>	Invalid client id

GetMonitoringClientCount	-	Integer	Change in count while processing the count we get from the last call
RestartBroadcasting	-	-	Unexpectedly disconnects clients
ProcessGetRequest	Request	-	Corrupted HTTP request
ProcessSetRequest	Request	-	Corrupted HTTP request
GetComponent	Component, Property	{Component, Property}	Non-existing component or property
SetComponent	Component, Property, newValue	-	Unsuitable newValue, non-existing component or property
EnableMotorPart	MotorPart, Script	Boolean	Motor part – script incompatibility
DisableMotorPart	MotorPart	Boolean	Non-existing motor part
RestartOFMController	-	-	Communication error between software and hardware components
ConnectToOFM	OFM	Boolean	Invalid OFM id
DisconnectFromOFM	OFM	-	Disconnect from OFM with invalid id
DisconnectFromAll	-	-	Disconnect from OFMs with invalid ids
handleRequest	Request	Boolean	Corrupted HTTP request
handleRandomRequest	-	Boolean	Non-existing or corrupted HTTP request
sendResponse	Response	Boolean	Corrupted HTTP response
sendRandomResponse	-	Boolean	Non-existing or corrupted HTTP response

Design Rationale:

- OFM Server is responsible for processing HTTP request and creating meaningful response data from microscope components and motor parts and making a HTTP response using that data.
- Broadcasting related tasks are also handled in OFM server, such as starting or ending broadcasting.
- Broadcast Controller is responsible for enabling and disabling broadcasting. Checking the feasibility of broadcasting, such as checking if the connected user count is sufficient or there is no server overload, is handled by OFM Controller.
- Client Web Servers communicates with OFM server through HTTP request/response pairs and public IP address of the OFM.

Database Operations

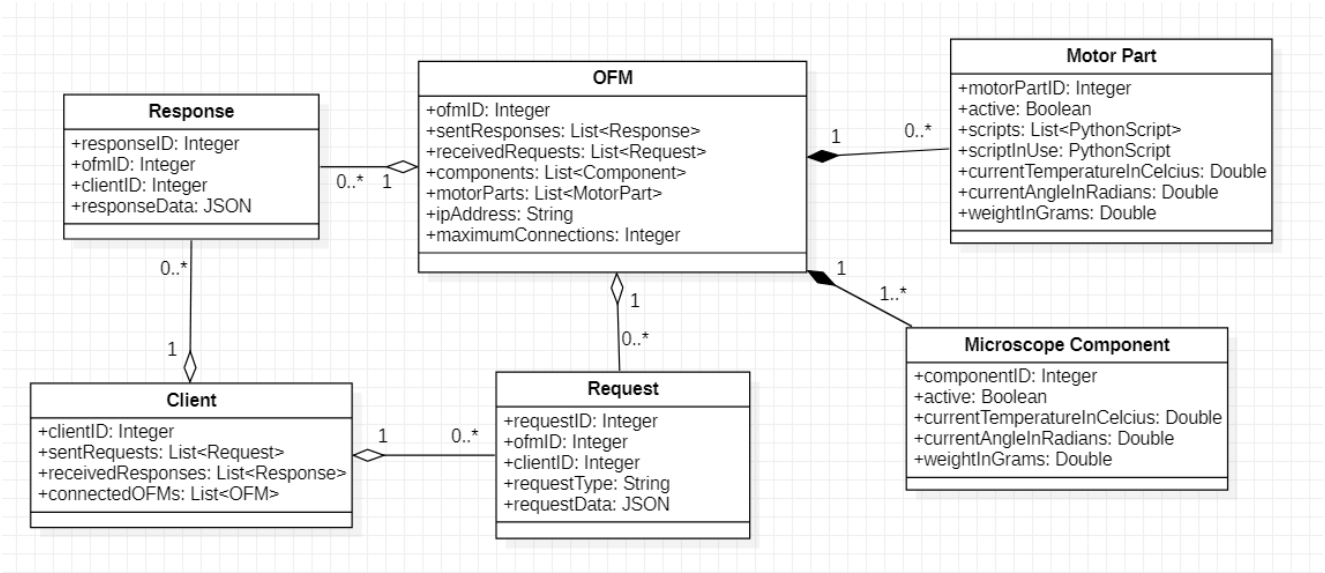

Figure 6: Database Class Diagram

Table 19: CRUD Operations

OPERATION	CRUD OPERATIONS
RestartOFM	Create: Read: OFM Update: OFM, Client Delete:
ShutDownOFM	Create: Read: Update: OFM, Client Delete: OFM, Client
ConnectClient	Create: Client Read: OFM Update: OFM Delete:
DisconnectClient	Create: Read: OFM Update: OFM Delete: Client
StartBroadcasting	Create: Read: OFM, Client Update: OFM Delete:
EndBroadCasting	Create: Read: OFM, Client Update: OFM Delete:
BlockClient	Create: Read: OFM, Client Update: OFM Delete:
UnblockClient	Create: Read: OFM Update: OFM Delete:
GetConnectedClientCount	Create: Read: OFM Update: Delete:
EnableBroadcasting	Create: Read: OFM Update: OFM Delete:
DisableBroadcasting	Create: Read: OFM Update: OFM Delete:

GetMonitoringClients	Create: Read: OFM Update: Delete:
GetMonitoringClientCount	Create: Read: OFM Update: Delete:
RestartBroadcasting	Create: Read: OFM Update: OFM Delete:
ProcessGetRequest	Create: Read: Request, OFM Update: OFM Delete:
ProcessSetRequest	Create: Read: Request, OFM Update: OFM Delete:
GetComponent	Create: Read: Microscope Component, OFM Update: OFM Delete:
SetComponent	Create: Read: Microscope Component, OFM Update: Microscope Component, OFM Delete:
EnableMotorPart	Create: Read: Motor Part, OFM Update: Motor Part, OFM Delete:
DisableMotorPart	Create: Read: Motor Part, OFM Update: Motor Part, OFM Delete:
RestartOFMController	Create: Read: OFM Update: OFM Delete:
ConnectToOFM	Create: Read: OFM, Client Update: OFM, Client Delete:
DisconnectFromOFM	Create: Read: OFM, Client Update: OFM, Client

	Delete:
DisconnectFromAll	Create: Read: OFM, Client Update: OFM, Client Delete:
handleRequest	Create: Request Read: Client, Request Update: OFM, Client Delete:
handleRandomRequest	Create: Request Read: Client, Request Update: OFM, Client Delete:
sendResponse	Create: Response Read: OFM Update: OFM, Client Delete:
sendRandomResponse	Create: Response Read: OFM Update: OFM, Client Delete:

Design Rationale:

- PostgreSQL database will be used.
- Database entries will be kept for future investigation, i.e. research.
- Corresponding Request database entry will be created when the request reaches to OFM.
- Sending and handling response/requests affect OFM and Client database tables at the same time.

4.4. Interface View

In this view, the internal interfaces between the components of the system and the external interfaces between the OpenFlexure Microscope and the other systems will be shown in detail.

Internal Interfaces

The Interface between OFM Controller and Microscope Component (Set Component)

OFM Controller receives HTTP requests from HTTP Request Handler and extracts meaningful information and uses that to communicate with Microscope Components. For example, it decides whether it is GET or PUT, request and communicates with components according to that (setting or getting property value).

Design Rationale:

- OFM Controller provides this interface.
- OFM Controller is responsible for directly communicating with components and mediate between Client and Microscope.
- In case of a PUT request, OFM Controller sets the corresponding component with newly provided value.
- The corresponding PostgreSQL database tables will be storing all the OFM and Microscope Component information.
- PostgreSQL table entries will not be deleted in order to be used for research purposes later.

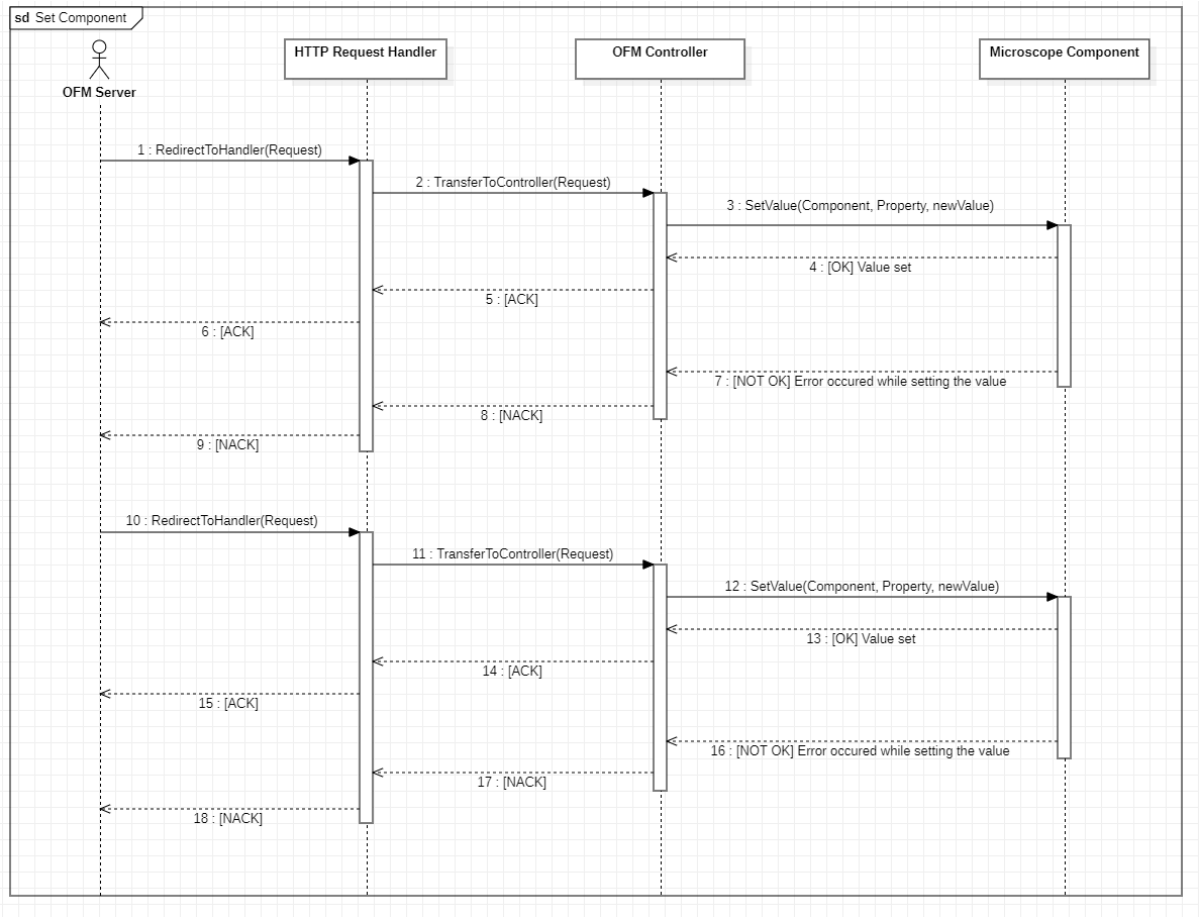

Figure 7: Sequence Diagram showing the interface between OFM Controller and Microscope Component (SET Component)

The Interface between OFM Controller and Microscope Component

(Get Component)

OFM Controller gets meaningful information from OFM Controller and uses that to create a HTTP Response and sends it to the client. For example, it decides whether it is GET or PUT, request and communicates with components according to that (setting or getting property value).

Design Rationale:

- Microscope Component provides this interface.
- OFM Controller is responsible for directly communicating with components and mediate between Client and Microscope.
- In case of a GET request, OFM Controller gets the property value from the corresponding component.
- The corresponding PostgreSQL database tables will be storing all the OFM and Microscope Component information.
- PostgreSQL table entries will not be deleted in order to be used for research purposes later.

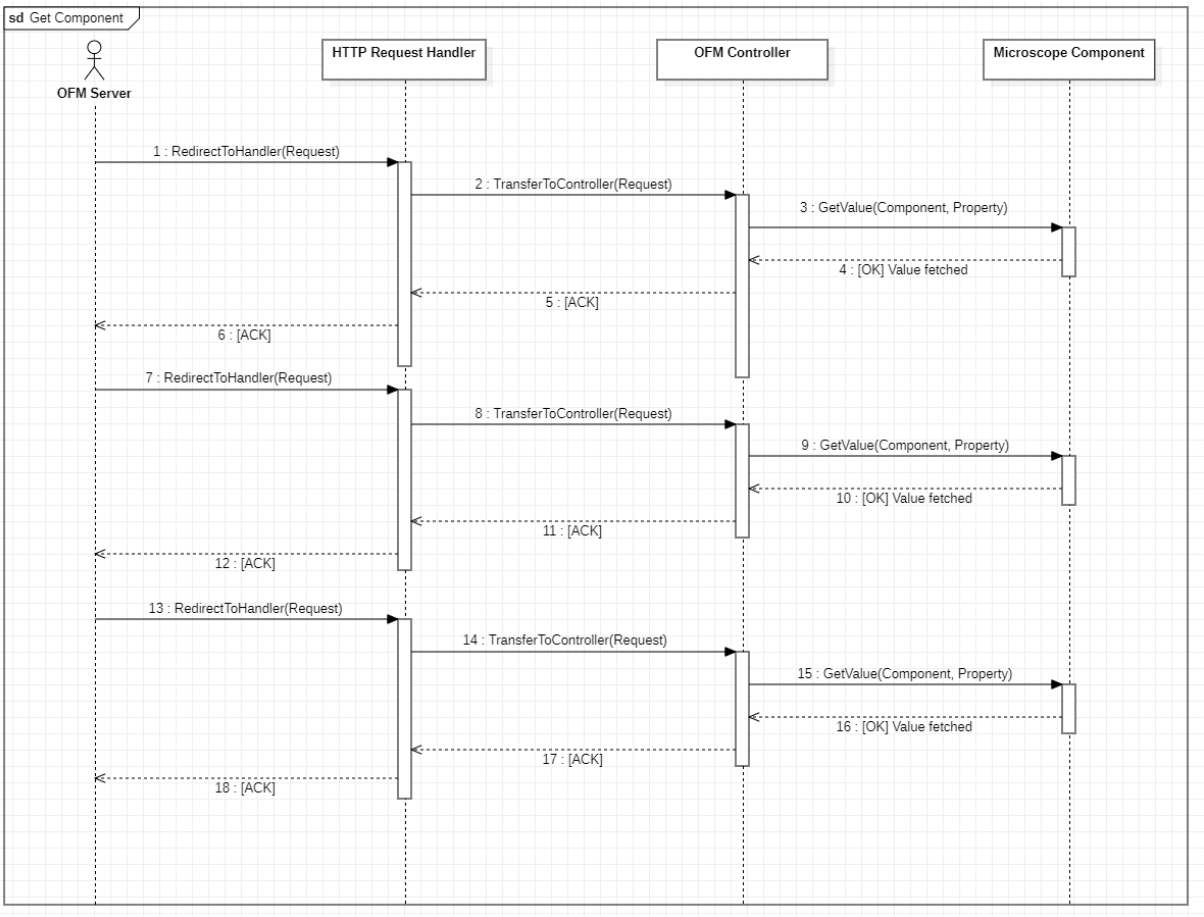

Figure 8: Sequence Diagram showing the interface between OFM Controller and Microscope Component (GET Component)

The Interface between OFM Controller and Motor Parts

OFM Controller communicates with existing Motor Parts on the microscope using Ethernet cable. OFM Controller updates script list of Motor Part when needed. Furthermore, OFM Controller updates the script currently in use field of Motor Parts.

Design Rationale:

- OFM Controller provides this interface.
- Motor Parts are controlled by OFM Controller. There is no required interface for OFM Controller when communicating with Motor Parts. However, Motor Parts need the Control interface provided by OFM Controller.
- OFM Controller is responsible for directly communicating with components and mediate between Client and Microscope.
- The corresponding PostgreSQL database tables will be storing all the OFM and Motor Part information.
- PostgreSQL table entries will not be deleted in order to be used for research purposes later.

The Interface between OFM Controller and HTTP Request Handler

OFM Controller internally communicates with HTTP Request Handler. The request coming from a client first redirected to HTTP Request Handler by OFM Server where some initial arrangements are made on the request. Then, HTTP Request Handler sends the request to OFM Controller where it is processed to determine whether it is for getting or setting some component value.

Design Rationale:

- HTTP Request Handler provides this interface.
- There is no required interface for HTTP Request Handler when communicating with OFM Controller. However, OFM Controller needs the Request Data interface provided by HTTP Request Handler.
- The corresponding PostgreSQL database tables will be storing all the OFM and Request information.
- PostgreSQL table entries will not be deleted in order to be used for research purposes later.

The Interface between OFM Controller and HTTP Response Handler

OFM Controller internally communicates with HTTP Response Handler. The response data coming from OFM Controller is sent arranged into a useful form to be used in response object, and sent to HTTP Response Handler where the response object is created using this response data.

Design Rationale:

- OFM Controller provides this interface.
- There is no required interface for OFM Controller when communicating with HTTP Response Handler. However, HTTP Response Handler needs the Response Data interface provided by OFM Controller.
- The corresponding PostgreSQL database tables will be storing all the OFM and Response information.
- PostgreSQL table entries will not be deleted in order to be used for research purposes later.

External Interfaces

User Interfaces

Student Interface:

Students use OpenFlexure Microscope to enhance their school education. The elementary or undergraduate level courses they take related in fundamental sciences such as Physics, Biology and Chemistry require lab work such as homework and exams; and students can use OFM in those tasks to get easily involved in observing, experiencing and questioning what they study.

Design Rationale:

- Different colors of OFM are good to attract students in elementary schools.
- DIY property of OFMs opens door for students to use their imagination and build the microscope in their dream.
- OFMs provide support for group work and collaboration for students to experiment together.

Teacher Interface:

Teachers use OpenFlexure Microscope to create an easy to access educational environment for their students. Since OFMs come with an affordable price, students can buy an OFM for their (say) Biology class and interactively learn the biology subjects.

Design Rationale:

- OFMs provide an easy-to-use software and hardware interface, so the teachers can easily start using it.
- Teachers can use broadcasting option in distant learning to let students monitor the current view of the experiment live.

- Teachers can also make exams by connecting themselves to all the OFMs of students and recording their experimenting process during a time interval.

Hobbyist Interface:

Beside from students and teachers, hobbyists can also make use OFMs to experiment as they wish with the microscope and the specimen of their choice.

Design Rationale:

- They can start experimenting with it right away, as setting up and using OFMs are quite easy.
- They automate the process using Motor Parts and Python Scripts. So, a group of hobbyists can work together to, for example, observe the water samples and try to improve the quality of it.

Researcher Interface:

Researchers can use OFMs to find meaningful data for their research. Furthermore, OFMs can also be used to speed up and automate their research.

Design Rationale:

- OFMs are compatible with common programming languages and libraries. So, a researcher working on a Data Science related task can use programming languages such as Python, Matlab or libraries such as Pandas.
- Cooperation has a very important role in research activities. OFMs support multiple connections, which enables group work and thus, enhances the usefulness of the research.
- Researchers can give worldwide conferences using the broadcasting feature of the OpenFlexure Microscopes.

Developer Interface:

Developers of OFMs use agile methodologies and priorities customer feedbacks. Customers contribute the development of the OpenFlexure Microscope.

Design Rationale:

- Developers apply stress testing before the new release of OFM.
- They try to connect to multiple OFMs as a single client at once.
- They try to connect multiple clients to the same OFM at once.
- The easy to use software and hardware interfaces of OFMs make those testing related tasks easier to manage.

Producer Interface:

Producers of OpenFlexure Microscopes are mostly interested in physical specifications of OFMs. The fact that they can be assembled and reassembled easily, options for different colors, comparably small size etc.

Design Rationale:

- Physical design of the OFMs allow for color preference.
- The plastic material used for DIY parts are approved to be safe for children usage.
- OFMS are quite small, and does not require a lab environment to start working on it.
- Different OFM parts can be bought separately and attach to the microscope without an additional setup.

Support Interface:

Support Team of OpenFlexure are consist of developers and producers. Because the customer may need support for a hardware related problem, which most of the time requires a producer hand; or they may need a software related problem, which is most of the time solved with the help of a developer.

Design Rationale:

- Using OFM software, customers may get in touch with the support team in no time and start telling their problem about the product.
- Customers that contribute to the development of the OFM use a different communication channel, which is also provided with the OFM software, to inform producers about bugs, errors, or they tell them the feature they want to be added on OFMs.

System Interfaces

The Interface between Client Web Server and Broadcasting Controller

OFM server has a dedicated Broadcasting Controller to handle broadcasting related tasks. OFM Controller is responsible for checking whether the conditions required for broadcasting is met, and enable broadcasting if so.

Design Rationale:

- Broadcasting Controller provides this interface.
- Broadcasting Controller is responsible for maintaining the broadcasting and getting information from clients who are monitoring the broadcasting.
- The corresponding PostgreSQL database tables will be storing all the OFM and Client information.
- PostgreSQL table entries will not be deleted in order to be used for research purposes later.

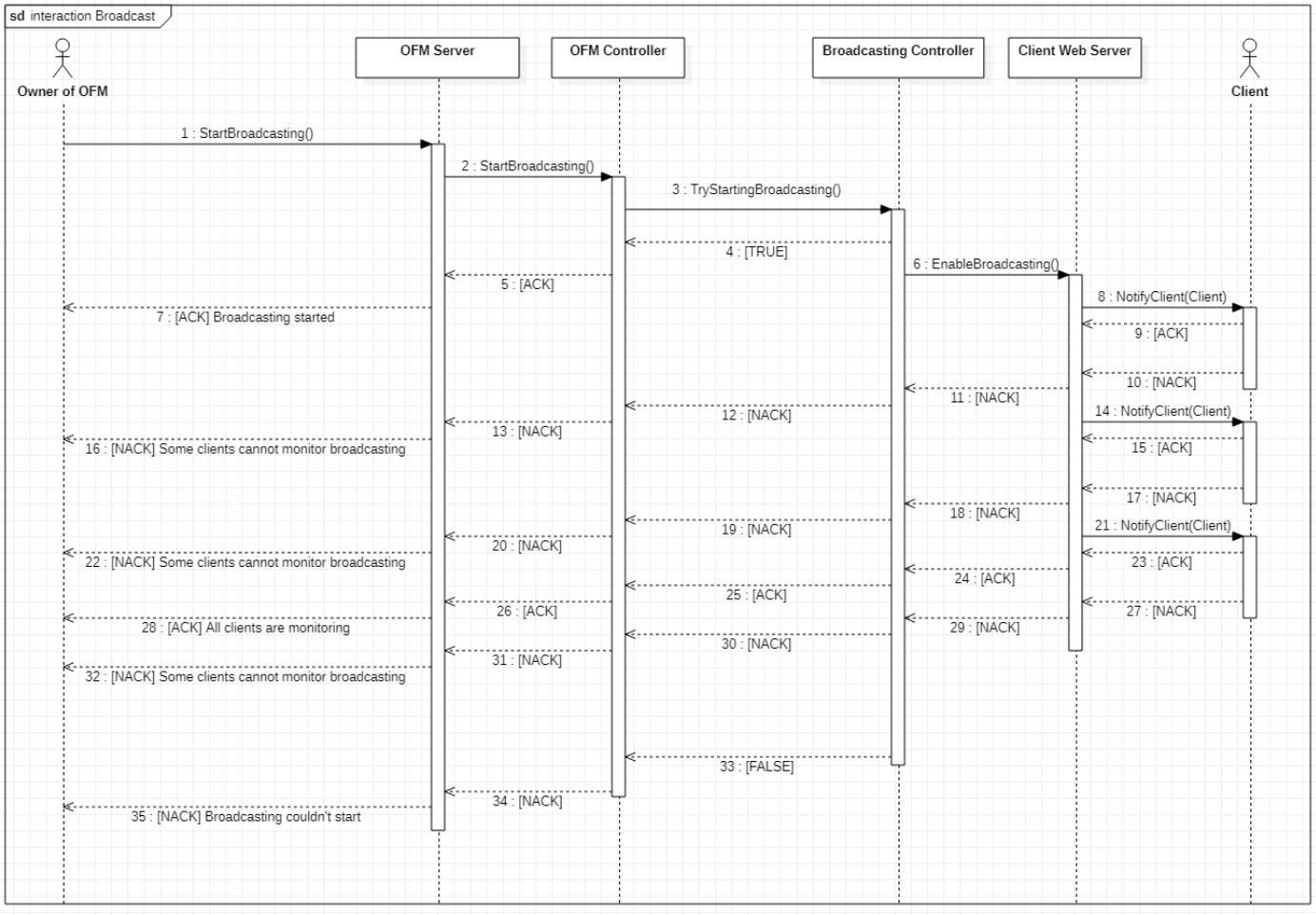

Figure 9: Sequence Diagram showing the interface between Client Web Server and Broadcasting Controller

The Interface between Client Web Server and HTTP Request Handler

In order to fetch and manipulate microscope data, clients need a web server to send their HTTP requests from. HTTP Request Handler in the OFM Server receives requests send from Client Web Server. HTTP Request handler then creates a request entry in the Request table in PostgreSQL database, which triggers update operation in both Client and OFM tables in the database.

Design Rationale

- Client Web Server provides this interface.
- HTTP Request Handler will be responsible for creating Request entries in the database.
- The corresponding PostgreSQL database tables will be storing all the OFM, Client and Request information.
- PostgreSQL table entries will not be deleted in order to be used for research purposes later.

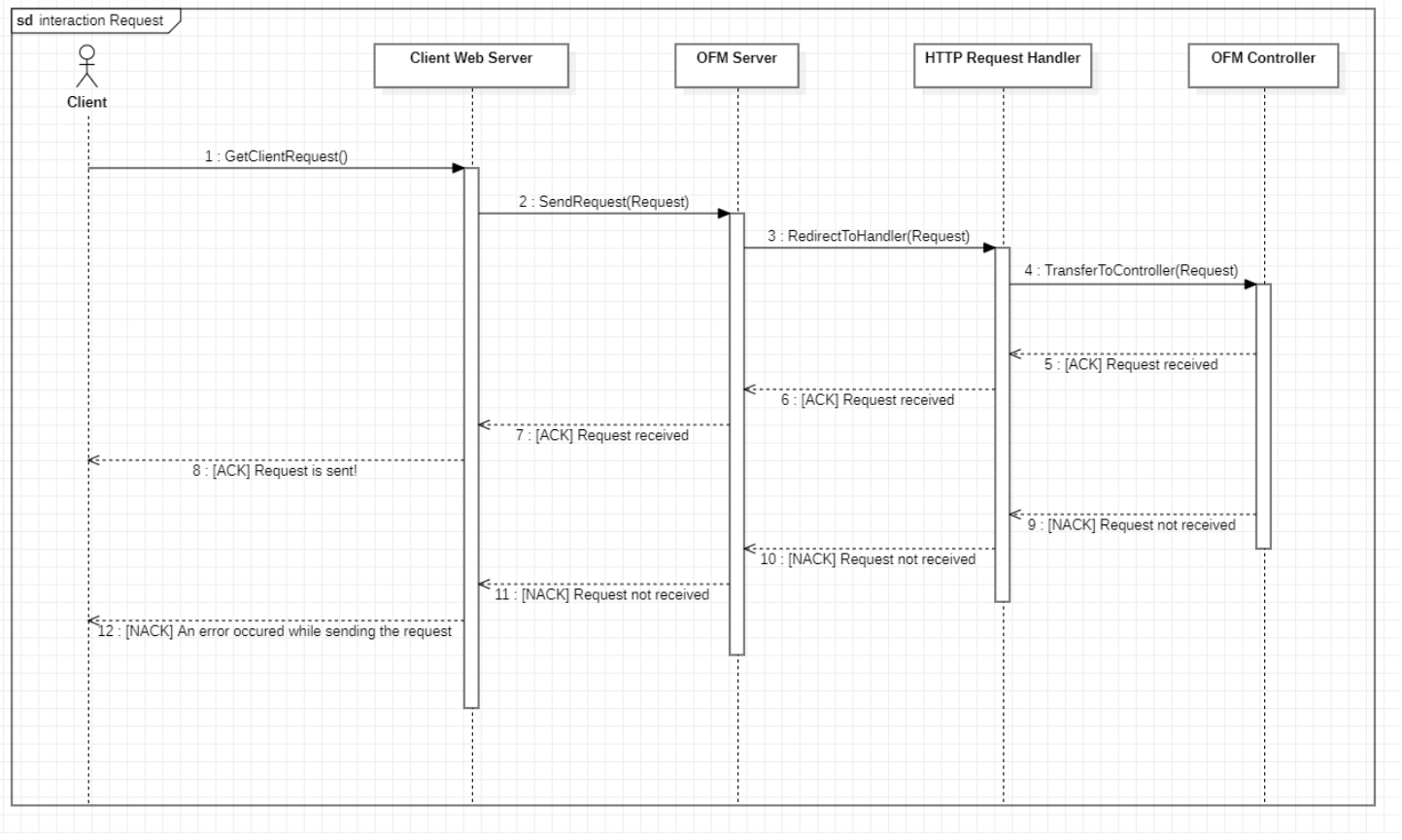

Figure 10: Sequence Diagram showing the interface between Client Web Server and HTTP Request Handler

The Interface between Client Web Server and HTTP Response Handler

OFM Server uses a HTTP Response Handler, in order to send microscope data and other related information to clients. After HTTP response is received by client, Response, Client and OFM entries of the corresponding tables in the PostgreSQL database are updated accordingly.

Design Rationale:

- HTTP Response Handler provides this interface.
- HTTP Response Handler is responsible for retrieving meaningful data from OFM Controller and use that data to create a HTTP Response and send it to the relevant client.
- The corresponding PostgreSQL database tables will be storing all the OFM, Client and Response information.
- PostgreSQL table entries will not be deleted in order to be used for research purposes later.

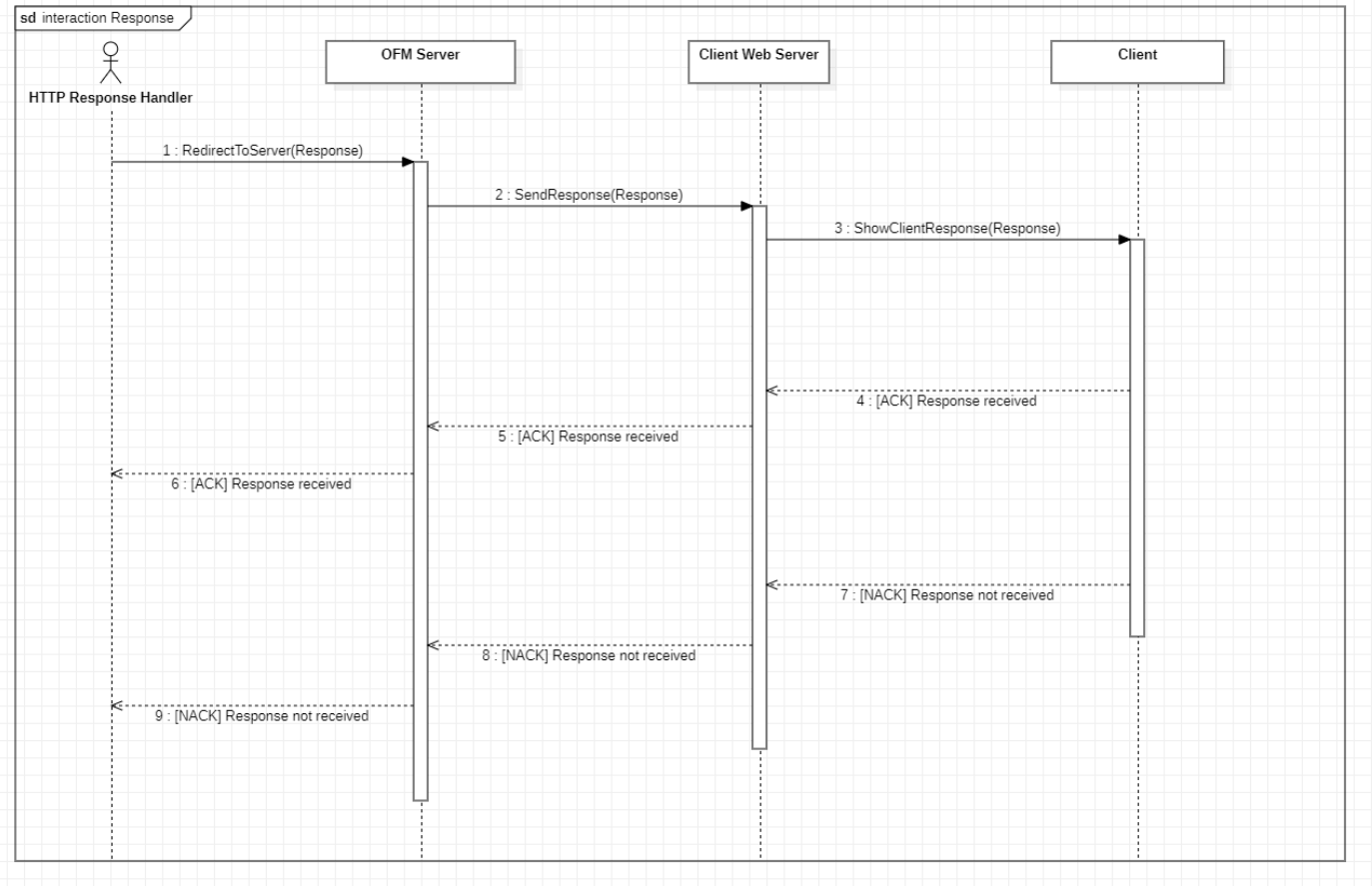

Figure 11: Sequence Diagram showing the interface between Client Web Server and HTTP Response Handler