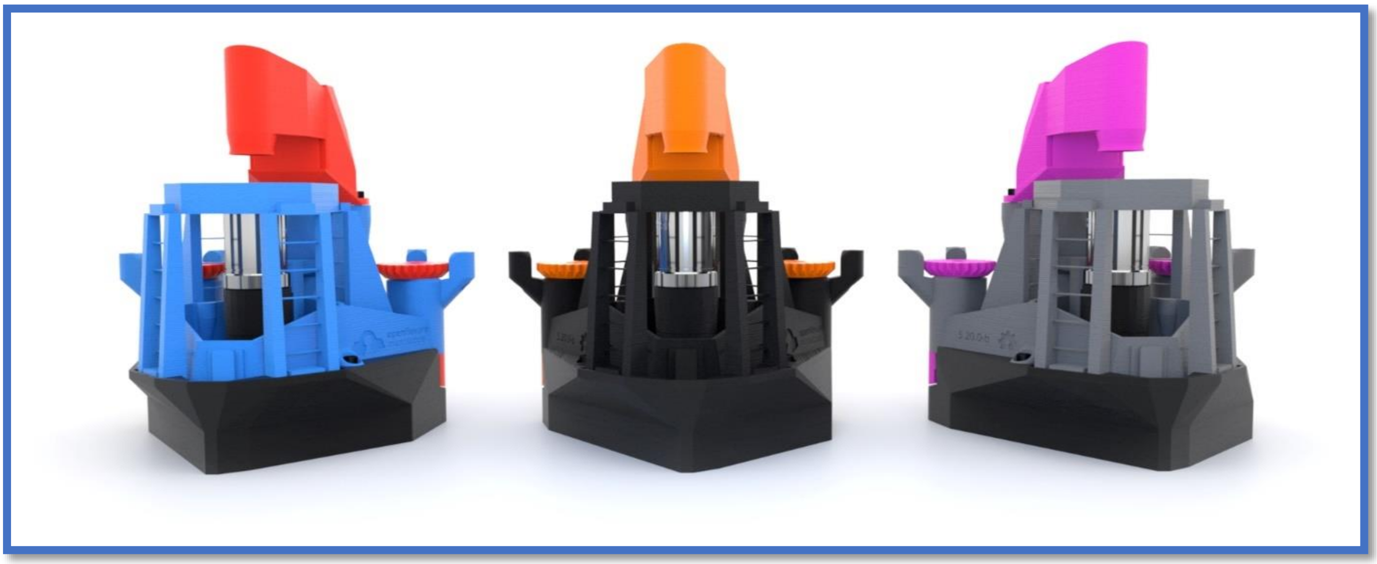
OpenFlexure Microscope



Software Design Description

CENK GURBET 2310050

BERKAY DEMİRÖREN 2309888

Table of Contents

List of Figures

List of Tables

Revision History

Version	Date	Explanation
1.0	28.04.2021	Purpose of the system, scope, Stakeholders and Their Concerns, References, Glossary, Context View.
1.1	11.06.2021	Fully Completed SDD

Table 1 Revision History

1 Introduction

This document is Software Design Description (SDD) of an open-source, customizable optical microscope project. OpenFlexure Microscope project supplies a material for printing microscope from a 3D printer and a software for using that microscope from another computer which can be a personal computer with modern processor or a small Raspberry Pi unit.

1.1 Purpose of the System

OpenFlexure Microscope Project aims to supply printable, customizable DIY microscopes designs with an open-source software to utilize it. The project offers a low-cost high-performance microscope product to people with low-budget constraints. Mainly targets personal amateur users, schools, and research labs in developing countries. With its open-source and low-cost nature it increases both availability and customizability in those areas therefore OFM project contributes to scientific advancement in developing countries.

The purpose of the Software Design Document (SDD) is to provide details about project's components. SSD is provided for groups which are related with software development of project.

- Developers (in our project Software Development Team)
- Plugin Developers
- Project Conductor Team

In addition, there are testers and software architects which are responsible for development of the project.

SDD is detailed guide since all stakeholders of the project could use it in development process and there should be no ambiguity. It includes technical details about software process, so users may not be concerned about SDD.

1.2 Scope

OpenFlexure Microscope project enables people to produce 3D-printed, low-cost, customizable, open-source and high-performance automated laboratory microscope with motorized sample positioning and focus control.

However, some drawbacks also exist. The range of mechanical motion is smaller than common mechanical stages. Furthermore, due to primarily plastic construction, loading large or heavy samples is not recommended.

- **3D-printable design:** Parts of the microscope will be printed and assembled except Raspberry Pi and its camera unit. All the design only uses around 200 grams plastic. To reduce assembling time and increase stability most of the parts will be printed as a single piece.
- **Customization:** Optics module (camera and lenses) can be easily changed to achieve different resolutions and magnification for various purposes. Furthermore, optional filter cubes can be printed, allowing for reflection (epi-) illumination, polarization-contrast microscopy, and even fluorescence imaging.
- **Open-source:** People with a programming experience are able to create new plugins specific to their needs and enable them on a per-microscope basis. Furthermore, there is a user-interface for the non-developers to get full control over the format and type of data stored.
- **High performance:** Focusing precisely and keeping sample steady is crucial when working at high magnification. Thanks to plastic flexures motion of the microscope is free from friction and vibration. Its miniature stepper motors can keep sample stable to a few microns over several days.

IEEE 29148-2011 standard is used for this document.

1.3 Stakeholders and Their Concerns

OFM is an open-source project so there is no commercial gain in the project. However, also, there are many groups which are responsible for development and maintenance of project.

Project includes so many software tools, so developers are responsible from development processes of them and maintenance of project. Some developers are responsible from creating new technologies (new features) for project. For example, Plugin Developers develop Web API extensions for project. Some parts of project need technical and scientific work. For instance, lenses of microscope (their technical properties; thickness etc.) should be developed regularly to meet new customer requirements. So, in project, Project Conductor Team is needed, and this team enhance microscope design.

These groups and their descriptions (relations with the project) :

Users:

Users are people who use project (project's software documents, software tools in website etc.). To use system, users need to complete some steps.

Basically:

- Downloading required softwares and documents
- Get required hardware tools
- Assembly hardware tools according to instructions which is provided by project
- Set up software in related hardware (such as Raspberry Pi)

Users have mainly 2 concerns:

- Users need to create their microscope with low prices, because not all users have high technology software or hardware devices.
- Creation (setup, assembly phases) of microscope should not take a long time and UI of system should address all kind of users. Simple and useful system for users.

Plugin Developers:

Plugin developers are responsible for developing Web API extensions for the project. They develop new attributes to project (such as new software properties) and, they fix them if there is a problem occur with developed part (for example, fixing problem when new attribute is not suitable for some operating systems).

Plugin developers have one main concern:

- User requests should be documented well and provided to developers. If there is an ambiguity in documents, developed software may not meet user requests or there may be a problem with users' previous requests (uncertainty in documents may cause contradiction with previous requests).

Project Conductor Teams:

Teams include members who are scientists and technicians. Teams are responsible from gathering information (statistical information) from users and experiments which are done with project tools and analyze them. For example, lenses' thickness is so crucial in project. Lenses should be well developed and should meet user requests. So, teams will develop new scientific features to project, or they improve existing project tools.
Tools can be both hardware (like lenses example) and software (for example control system of microscope, remotely-locally).

Teams have two main concerns:

- Gathered data (from experiments or users) should be well documented and analyzed. If there is a problem with gathered data, development process may cause problem for existing tools or new developed technology may be useless.

- Second main concern of project conductor teams is cost of development. OFM is open-source project and users do not pay any fee to use system. So, created (or developed) technology should be low-cost.

2. References

This document is written with respect to IEEE 29148-2011 standard:

IEEE. (2011, December 1). 29148-2011 - ISO/IEC/IEEE International Standard- Systems and software engineering-Life cycle processes-Requirements engineering. Retrieved from http://ieeexplore.ieee.org/document/6146379/ on March 12, 2018. doi:10.1109/IEEESTD.2011.6146379

Other Sources:

- https://openflexure.org/

- *Sharkey, J. P., Foo, D. C., Kabla, A., Baumberg, J. J., & Bowman, R. W. (2016).* ***A one-piece 3D PRINTED Flexure translation stage for Open-source microscopy. Review of Scientific Instruments****, 87(2), 025104. doi:10.1063/1.4941068*

- *Collins, J. T., Knapper, J., Stirling, J., Mduda, J., Mkindi, C., Mayagaya, V., . . . Bowman, R. (2019).* ***Robotic microscopy for everyone: THE Openflexure Microscope.*** *doi:10.1101/861856*

- *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

OFMP	OpenFlexure Microscope Project
HTTP	Hypertext Transfer Protocol
API	Application Programming Interface
GUI	Graphical User Interface
USB	Universal Serial Bus
HDMI	High-Definition Multimedia Interface
STL	Stereolithography

Table 2 Glossary

4. Architectural Views

4.1. Context View

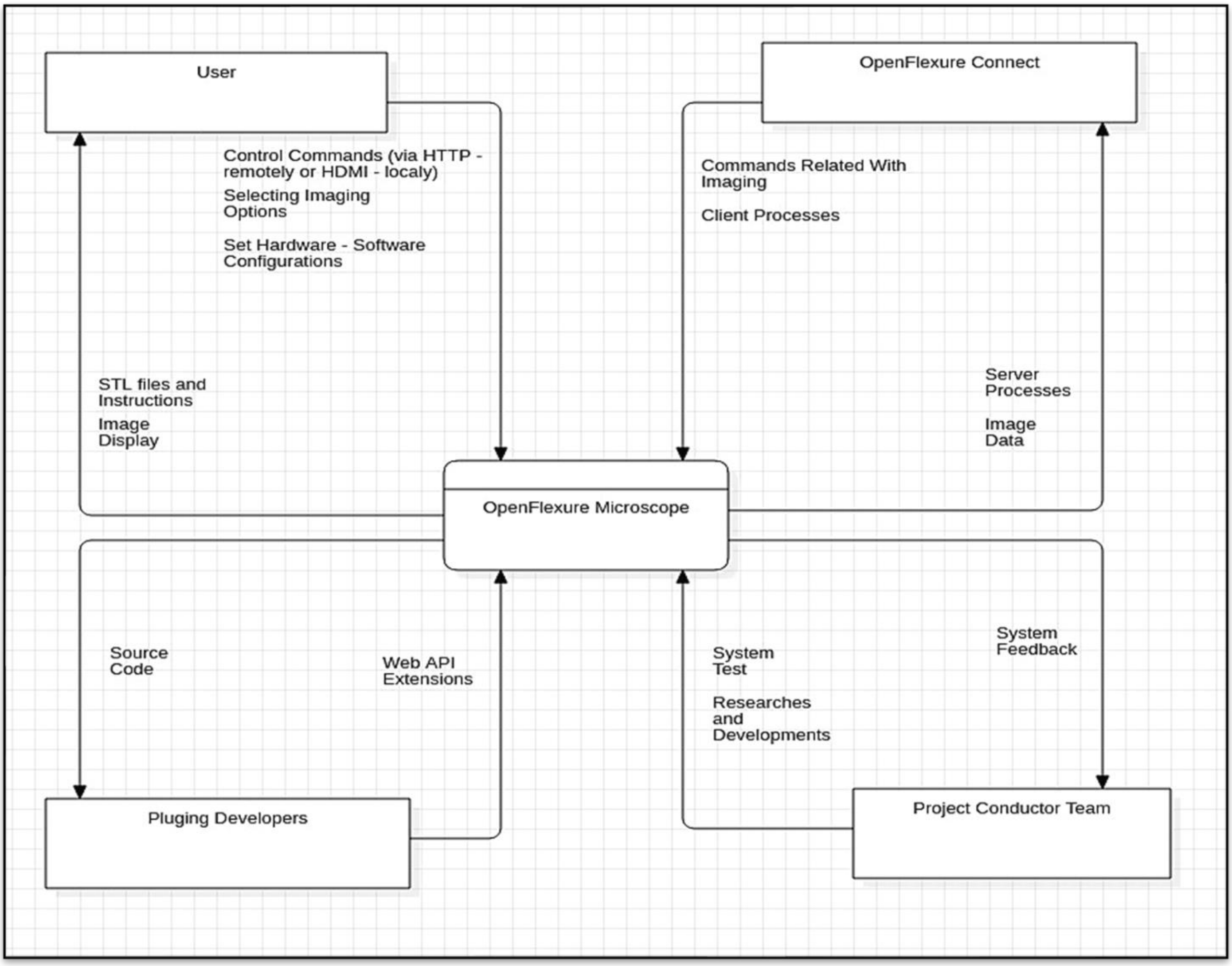


Figure 1: OpenFlexure Microscope Context Diagram

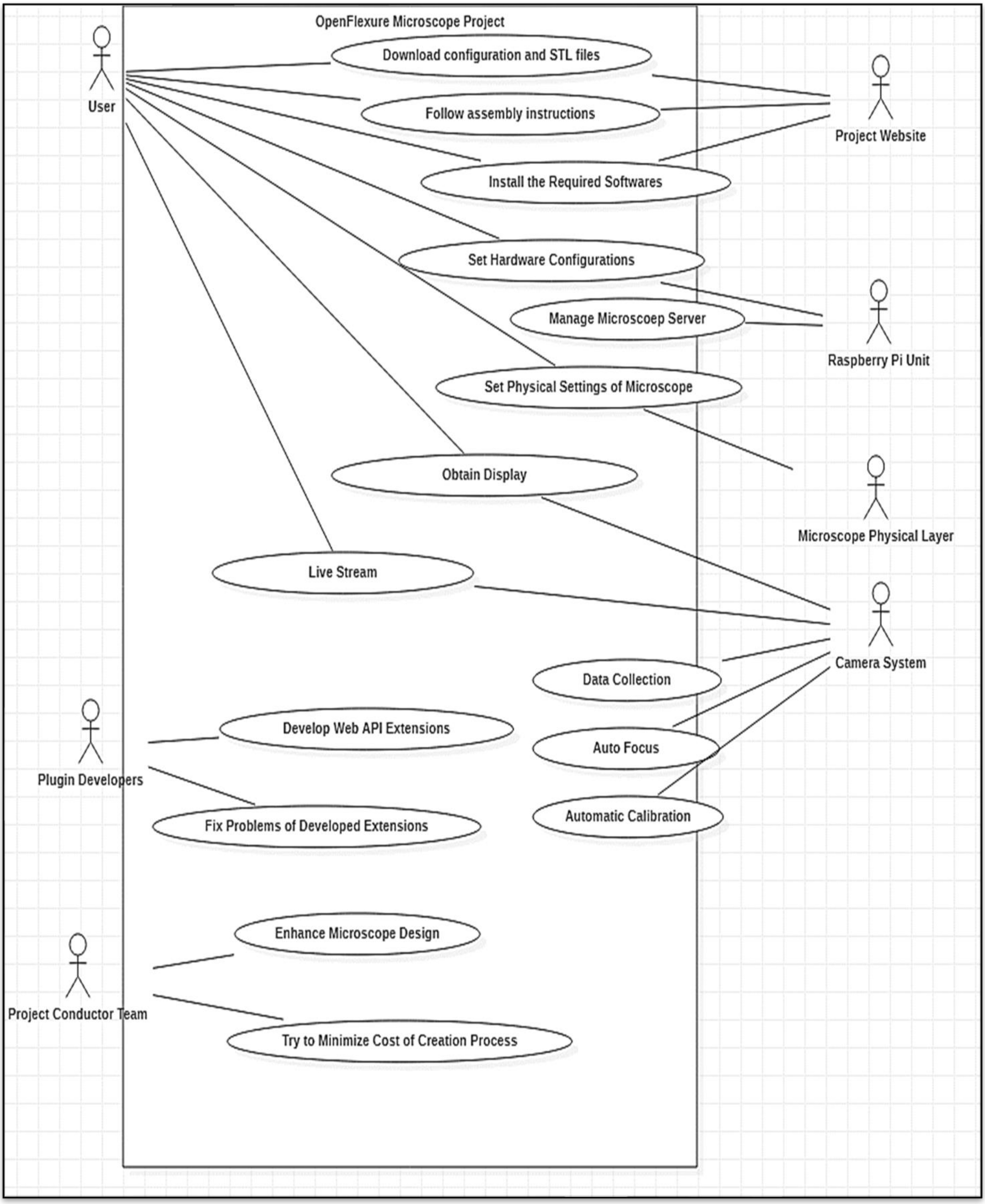


Figure 2 Use Case Diagram

Use case name	Download configuration and STL files
Actors	User, Project Website
Description	To use microscope (both locally and remotely) users need to install required files from website (or GitHub). Configure files are user special files which enable users to configure microscope technology and STL files are required to set up environment.
Data	User's device's features (OS, 64-32 bit options etc.) (these data come from user), other sources like codes in GitHub or files which will be installed.
Preconditions	Users need to be sure to that their system include enough hardware-software components which will be require in setup step. Also, users need to internet connection.
Stimulus	User enters the website and clicks the download button.
Basic Flow	1) Customers go to website 2) Users do not need to register 3) Click "Downloads" button 4) Download required files from "Downloads" section in page 5) At the end of the page, download STL files, clicking "Configure and Download" button.
Alternative Flow	-
Exception Flow	4) If there is an error during download, error message will be sent by user 's device to user. (internet connection error etc.) 5) Same exception flow as step 4
Postconditions	The files will be downloaded to user's machine.

Table 3: Download configuration and STL files

Use case name	Follow assembly instructions
Actors	User, Project Website
Description	Following instructions to set up required software and system.
Data	Instructions and steps which will be followed in setup phase.
Preconditions	Users need devices which software will be installed in and documents which include instructions of setup.
Stimulus	Reading instructions from the project's website.
Basic Flow	1) Users go to website of project. 2) Choose one of 3 options (Build microscope, Install the software, Use your microscope) 3) There are instructions about setup in each option. 4) Following instructions and set up environment.
Alternative Flow	-
Exception Flow	If there is an error during flow, error message will be sent to user by user's device. If the users cannot reach instructions pages this may happen due to two reasons: • Internet connection • Website problems **The first one:** users need to connect their devices to internet. **The second one:** Error messages are written in log files and system admins will fix them.
Postconditions	Microscope is assembled.

Table 4: Follow assembly instructions

<table>
<tr><td>Use case name</td><td>Install the required software</td></tr>
<tr><td>Actors</td><td>Users, Project Website (source codes and other required files may be in GitHub repositories.)</td></tr>
<tr><td>Description</td><td>Users need to install required software to use their microscopes.</td></tr>
<tr><td>Data</td><td>Files which are needed to install softwares. Main softwares which will be installed in website are:
Raspbian-OpenFlexure
OpenFlexure Connect (source codes are in link)</td></tr>
<tr><td>Preconditions</td><td>Users need devices which supports Windows or Linux to set up OpenFlexure Connect.
Raspberry Pi 2 Model B (or better) (for Lite version Raspberry Pi 1 Model B+)
Full desktop OS (For Lite, Minimal OS without desktop.)
Includes full graphical microscope control software. (For Lite, you will need another computer to control the microscope graphically.)</td></tr>
<tr><td>Stimulus</td><td>Clicking the download button in the project's website.</td></tr>
<tr><td>Basic Flow</td><td>1) Go to website
2) Click "Downloads"
3) There are two sections:
Raspbian-OpenFlexure
OpenFlexure Connect
Users need to install both of softwares from website.
4) After download, users need to set up both softwares. (Setup processes differs in different OS)
5) Set up downloaded softwares to devices. Raspbian-OpenFlexure must be set up in Raspberry Pi and OpenFlexure Connect must be set up in Windows or Linux devices.</td></tr>
<tr><td>Alternative Flow</td><td>Users need to follow two options to set up OpenFlexure Connect.
For Windows devices:
- Click "openflexure-connect-4.0.1-win.exe" file.
- Allow all permission messages.
- Software will be installed in very short time and users are ready to use it in few minutes.</td></tr>
</table>

	• For Linux devices: - Click "AppImage(x86-64)" button. - Make downloaded file executable - Run the file
Exception Flow	If any error occurs during installation this may happen due to two reasons: • Problems in softwares • Problems due to user-user's device **First one:** Users can report them to admins, and they will fix the bugs. **Second one:** Second may happen due to various reasons. Users need to allow permissions in setup phase and be sure their system meet minimum system requirements.
Postconditions	All required softwares are installed successfully

Table 5: Install the required softwares

Use case name	Set hardware configurations
Actors	User, Raspberry Pi Unit
Description	When the user sets up the microscope environment, he/she need to configure settings of microscope. Hardware configurations include optic settings, and manual mechanism's settings.
Data	Config settings on the project's Git-Hub repository.
Preconditions	Hardware configuration error or customization need.
Stimulus	On Raspberry Pi unit config settings are coded.
Basic Flow	1) User opens config files on the Raspberry Pi. 2) Then searches for the desired hardware config file. e.g. camera config, network config 3) After that user changes the file content to configure microscope parts.
Alternative Flow	-
Exception Flow	If new configurations cause microscope to not work properly. Then factory settings can be downloaded from the GitHub repositories of the project.
Postconditions	Configurations are set.

Table 6: Set Hardware Configurations

Use case name	Make physical settings of microscope
Actors	User,3D printer
Description	Users need to set their physical configurations to make their microscopes more personalized.
Data	There are several instructions about setup hardware of microscope. Some of them are related with: • Objective lens • Condenser mount • Actuator gear • Camera & Optics Module
Preconditions	Before making configurations user need to be sure that all required hardware components are printed by 3D printer. All other components (optic part etc.) must be ready for implementation.
Stimulus	Microscope is configuring physically.
Basic Flow	1) Download STL files from project's website 2) All basic components of microscope must be printed by user with 3D printer. 3) Other components must be provided by user. Some of main components in this category are: • Imaging Lenses • Electronic Housing • Raspberry Pi (Main operator unit of microscope) • Camera unit 4) For imaging components, users could choose different options (Imaging Modes). Some of them are: - Bright-field trans-illumination - Bright-field epi-illumination - Polarization-contrast imaging - Fluorescence imaging 5) Downloading assembly instructions. 6) Assembling both imaging components and other physical components with respect to instructions. (Raspberry Pi etc.)

Alternative Flow #1	6) If users want to use microscope locally, they need to HDMI connection with their devices.
Alternative Flow #2	6) If users want to use microscope remotely (with HTTP): They need other components: • Connection via Ethernet, WLAN, Internet
Exception Flow	Exceptions may occur due to lack of (or mistakes in getting components step) components. Users need to analyze all components and detect deficiencies. After that they need to get all components properly.
Postconditions	Microscope is set physically.

Table 7: Make Physical Setting of Microscope

Use case name	Managing microscope display remotely
Actors	Users
Description	Setting up and maintaining server and client machines for microscope's remote display.
Data	Commands for managing server. IP addresses.
Preconditions	Network connection is needed to be established. WLAN, LAN, or Ethernet directly.
Stimulus	Typing commands to Raspberry Pi's CL, and connecting to server via browser or Op
Basic Flow	1)Configure Raspberry Pi for network and interface settings. sudo raspi-config 2)Server machine connects to a LAN or WLAN. 3)Server machine user configures the server. Using below commands: ofm update ofm upgrade ofm start 4)Client user connects the same network with server machine. 5)Client machine types of IP address of the server machine to its browser's address bar. 6)Server shares its microscope images with the connected client machines. 7) Client user obtains display.
Alternative Flow	5)Client machine user launches OpenFlexure Connect. 5)Client machine connects via ethernet.
Exception Flow	If commands make server down or broke, server can be restarted with default settings.
Postconditions	Remote connection is set.

Table 8: Managing microscope display remotely

Use case name	Obtain display
Actors	User, Camera System
Description	Providing images by microscope to user
Data	Image data
Preconditions	User needs microscope connection and user ought to set all required configurations properly.
Stimulus	Plug HDMI cable to get display locally. Or use ethernet, WLAN, or LAN
Basic Flow	1) Users could choose any of imaging modes. (Imaging modes in 2.1) 2) Automatic focus is handled by OpenFlexure software.
Alternative Flow	-
Exception Flow	If basic display is not created check connection cables. If imaging modes are not working properly. Try reinstalling application.
Postconditions	Display is shown on the screen or screens.

Table 9: Obtain Display

Use case name	Developing Web API Extensions
Actors	Plugin Developers
Description	By using python libraries, a user develops plugins to advance microscope functionality.
Data	Source code of OpenFlexure Microscope.
Preconditions	Python knowledge and a plugin idea.
Stimulus	Implementing plugin code.
Basic Flow	1)To add a simple functionality, a single Python file which contains all the extensions can be included to extension directory of the project. 2)To add a more complex functionality, extensions need to be written as a package. That module must be a folder within the extensions folder and also includes a top-level __init__.py file.
Alternative Flow#1	Adding web API views 1)In order to observe and control functionality of a plugin user can develop a web API view.
Alternative Flow#2	Marshalling data 1)@marshal_with and @use_args decorators along with schemas can be used for serialization of API responses and parsing of request parameters respectively.
Alternative Flow#3	Thing description 1)OFM Server will generate a W3C Thing Description which makes microscope's features be compatible with other "Web of Things" objects.
Alternative Flow#4	Thing activations 1)OFM Server also supports Thing actions. Invoking functions of a thing which manipulates state or triggers a process
Exception Flow	1)Some extensions may perform tasks that takes a long time compared to the expected time of a web request. That may cause connection timeout or another client interference. To handle these issues background tasks and component locks can be used.
Postconditions	Plugins are ready for usage.

Table 10: Developing Web API Extensions

Use case name	Live stream
Actors	Camera System, User
Description	Live stream to client machines which connected to microscope system.
Data	Raspberry Pi camera image.
Preconditions	A connection to microscope server.
Stimulus	Connecting to microscope server. Then, stream starts automatically.
Basic Flow	1]Microscope server starts recording JPEG frames to its buffer then when a client connect to stream the server starts to send frames.
Alternative Flow	-
Exception Flow	1]If there is a problem with the connection or stream quality connection can be re-established.
Postconditions	Users can join to live stream.

Table 11: Live Stream

Use case name	Enhance microscope design
Actors	Project conductor team
Description	Advancing general outlook, performance, and design.
Data	User feedback and scientific research.
Preconditions	Needs to working scientists and designers.
Stimulus	Scientist and designer will.
Basic Flow	1)Scientists conduct some research and look for cutting-edge technologies. 2)After gathering information they convey it to engineers and designers. 3)Then designers and engineers improve the microscope.
Alternative Flow	-
Exception Flow	1) Any error will be reported and examined.
Postconditions	The microscope is improved.

Table 12: Enhance microscope design

Use case name	Print microscope parts
Actors	User,3D printer
Description	Printing required parts.
Data	Microscope plans.
Preconditions	Need to have a working 3D printer.
Stimulus	Starting print process.
Basic Flow	1)User downloads STL files from the project's website. 2)User inserts the plans to 3D printer. 3)3D printer creates assembly parts of microscope.
Alternative Flow	-
Exception Flow	1)If files are broken then redownload them.
Postconditions	The microscope is created.

Table 13: Print microscope parts.

Use case name	Fix problems of developed extensions
Actors	Plugin Developers
Description	Fixing problems about existing developed extensions.
Data	Detailed document about problem. Document must include this information (also other related information): - Problem occurs in which operating system - Problem definition - Problem's occurrence frequency
Preconditions	Devices which problem occur, required IDE or hardware devices which will be used in problem fixing step.
Stimulus	Occurrence of problem.
Basic Flow	1)User report that there is a problem with extension. Problem could occur due to both software and hardware devices. 2) Developers generate detailed document about problem. 3) Document is analyzed by developers and developer team solves it. 4) Fixed extension is released.
Alternative Flow	-
Exception Flow	3) Problem may not be fixed, in this situation related extension should be removed from Web
Postcondition	Problem with extension is solved

Table 14: Fix problems of developed extensions

Use case name	Try to Minimize Cost of Creation Process
Actors	Project Conductor Team
Description	Teams try to minimize cost of creation process of microscope. (This use case is valid for generally hardware devices)
Data	Documents about costs of production step.
Preconditions	Required hardware tools which are used in physical production step, 3D printer, cost analyze tools (software)
Stimulus	Analyzing cost documents.
Basic Flow	1) Project conductor team (mostly scientists and technicians) collect data from existing hardware and software tools. (Documents are related with cost effectiveness of tools) 2) Analyze collected documents and determine if there is a device which could be changed with another low-cost device. 3) If there is a device like that, team could choose one of 2 ways: - Change device completely with new low-cost device - Improve existing device and make it more cost effective 4) After changing or improving new device and it's documents are released in website of project
Alternative Flow	3) - Team could choose to develop new device and remove existing one from project. This step includes more scientific work and documentation. - Team could choose change existing device. This option cost less work (both scientific and technical) but, also, improved parts must be tested with existing software/hardware tools (Control that they work properly with existing environment.).
Exception Flow	After analyzing documents teams may not find any alternative production way to improve existing tool or develop new tool with more cost effective. In that situation, existing system may not be changed.
Postconditions	New cost-effective product will be generated.

Table 15: Try to Minimize Cost of Creation Process

4.2. Composition View

In this view, components, and interfaces between them are presented in a high-level view. Detailed explanations are given in the related sections.

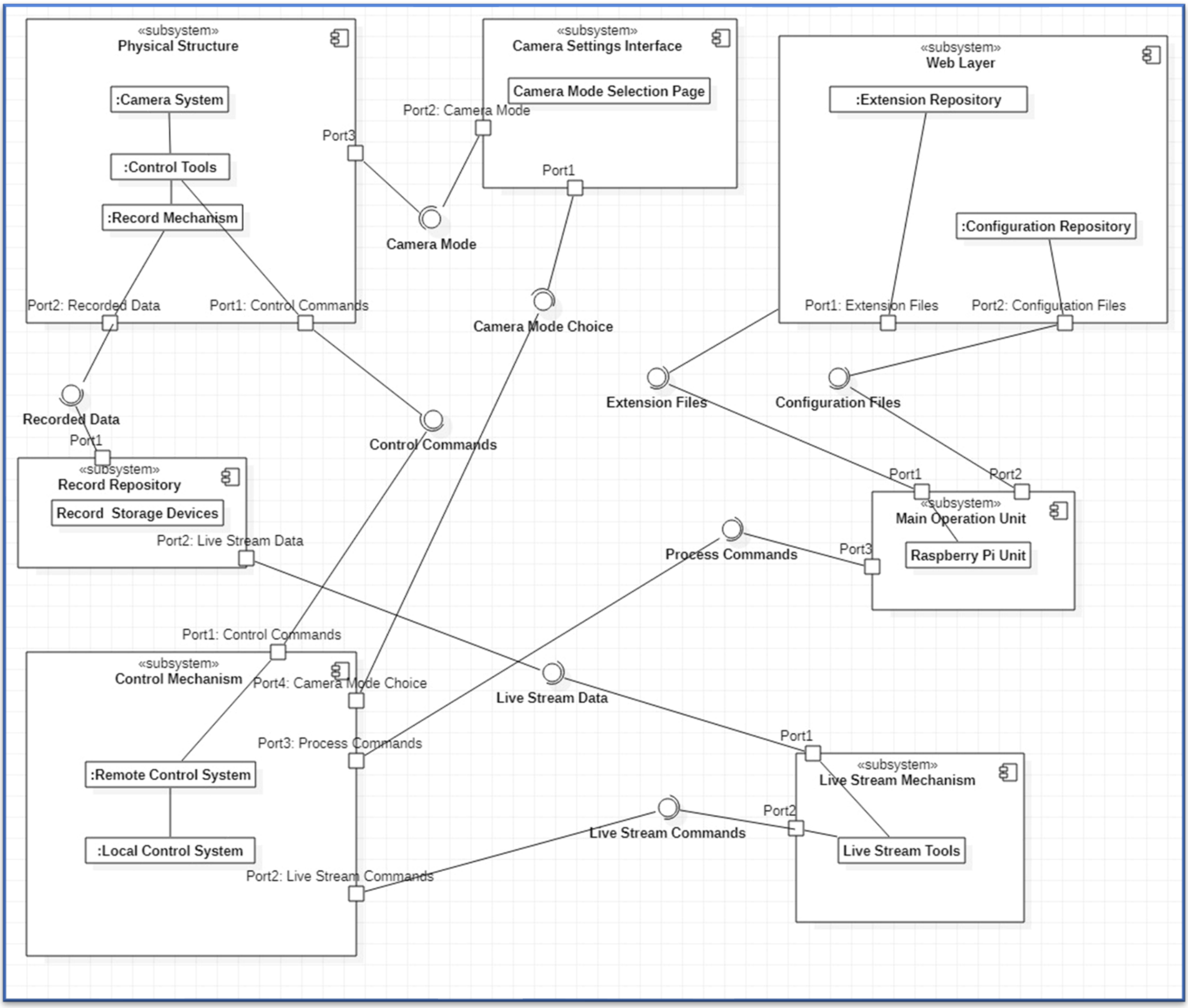


Figure 3: Composition Diagram

Design Rationale:

- Physical Structure of the project contains Camera System, Control Tools, and Record Mechanism. It is responsible for gathering and saving displays and controlling movements of the cameras of microscope. It sends ***recorded data*** to record repository in local server of microscope. And gets ***camera mode*** option from Camera Settings Interface subsystem. In addition to that, Control Mechanism subsystem sends ***control commands*** to its Control Tools component.
- Camera System Component is responsible unit for gathering images from external sources. With its lenses and regarding to other components' mode selection options, properties of display can be varying. Its optic modules and imaging capabilities may change with control tools and mode selection page.
- Control Tools Component in Physical Structure makes possible to control movements of the lenses and cameras. Utilization of the motors does not need any programming background. Mouse movements and keyboard strokes changes camera viewpoints and where does it focus.
- Record Mechanism Component in Physical Structure provides an option for the users to save their displays and images. A user is able to save it in various resolution options and various color modes to save storage space and to increase readability of displays.
- Camera Settings Interface Subsystem contains Camera Mode Selection Page component. It sends ***camera mode*** selection commands to Physical Structure subsystem and obtains ***camera mode choice*** commands from Control Mechanism subsystem.
- Camera Mode Selection Page enables users to choose which optic module will be used while obtaining display. Mainly there are 4 types of imaging modes namely, bright-field trans-illumination, bright-field epi-illumination, polarization-contrast imaging and fluorescence imaging.
- Web Layer Sub-system contains Extension Repository and Configuration Repository. It mainly responsible for website hosting and file transaction between web servers to users. It sends ***extension files and configuration files*** to main operation unit of the microscope.
- Extension Repository Component in Web Layer subsystem keeps extensions files which are developed by plugin developers to enhance microscope capabilities. All files are deposited with respect to GNU open-source license therefore anyone can access and modify them if theirs commits accepted by developer community.
- Configuration Repository Component in Web Layer subsystem keeps configuration files needed to achieve full features of the microscope. Those files are namely,

Raspbian-OpenFlexure SD Card Image, OpenFlexure Connect, and some STL files for 3D printing.

- Record Repository Subsystem contains Record Storage Devices Component. It retrieves ***recorded data*** from Physical Structure subsystem and supplies live stream data to Live Stream Mechanism subsystem.
- Record Storage Devices in Record Repository subsystem keeps saved displays in Record Mechanism Component. To decrease occupied storage space those data can be compressed with DCT and MC techniques.
- Control Mechanism Subsystem composed of Remote Control System Component and Local Control System Component. It is the main control unit of microscope system. It sends ***control commands*** to Physical Structure Subsystem, sends ***live stream commands*** to Live Stream Mechanism Subsystem, sends process commands to Main Operation Unit Subsystem and sends ***camera mode choice*** to Camera Settings Interface.
- Remote Control System Component in Control Mechanism Subsystem enables users to manage microscope remotely over the internet, WLAN, LAN or Ethernet. OpenFlexure Connect application is used for those purposes.
- Local Control System Component in Control Mechanism Subsystem enables users to manage microscope locally over the Raspberry Pi Unit.
- Main Operation Unit Subsystem composed of Raspberry Pi Unit. It mainly performs all the requests that users made. It gathers ***configuration and extension files*** from Web Layer Subsystem and gets ***process commands*** from Control Mechanism Subsystem.
- Raspberry Pi Unit Component in Main Operation Unit Subsystem execute all the operations that user and other components request. All the physical and logical components work on this component. It gets ***process commands*** from Control Mechanism Subsystem also gets ***extensions and config files*** from Web Layer and starts to perform with respect to those commands and settings.
- Live Stream Mechanism Subsystem consists of Live Stream Tools Component. It mainly manages live stream sessions. Fetches ***live stream data*** from Record Repository Subsystem and gets ***live stream commands*** from Control Mechanism Subsystem.
- Live Stream Tools Component enables users to stream their microscope displays lively. It gets ***live stream commands*** from Control Mechanism Subsystem and uses the visual data supplied from Record Repository Subsystem.

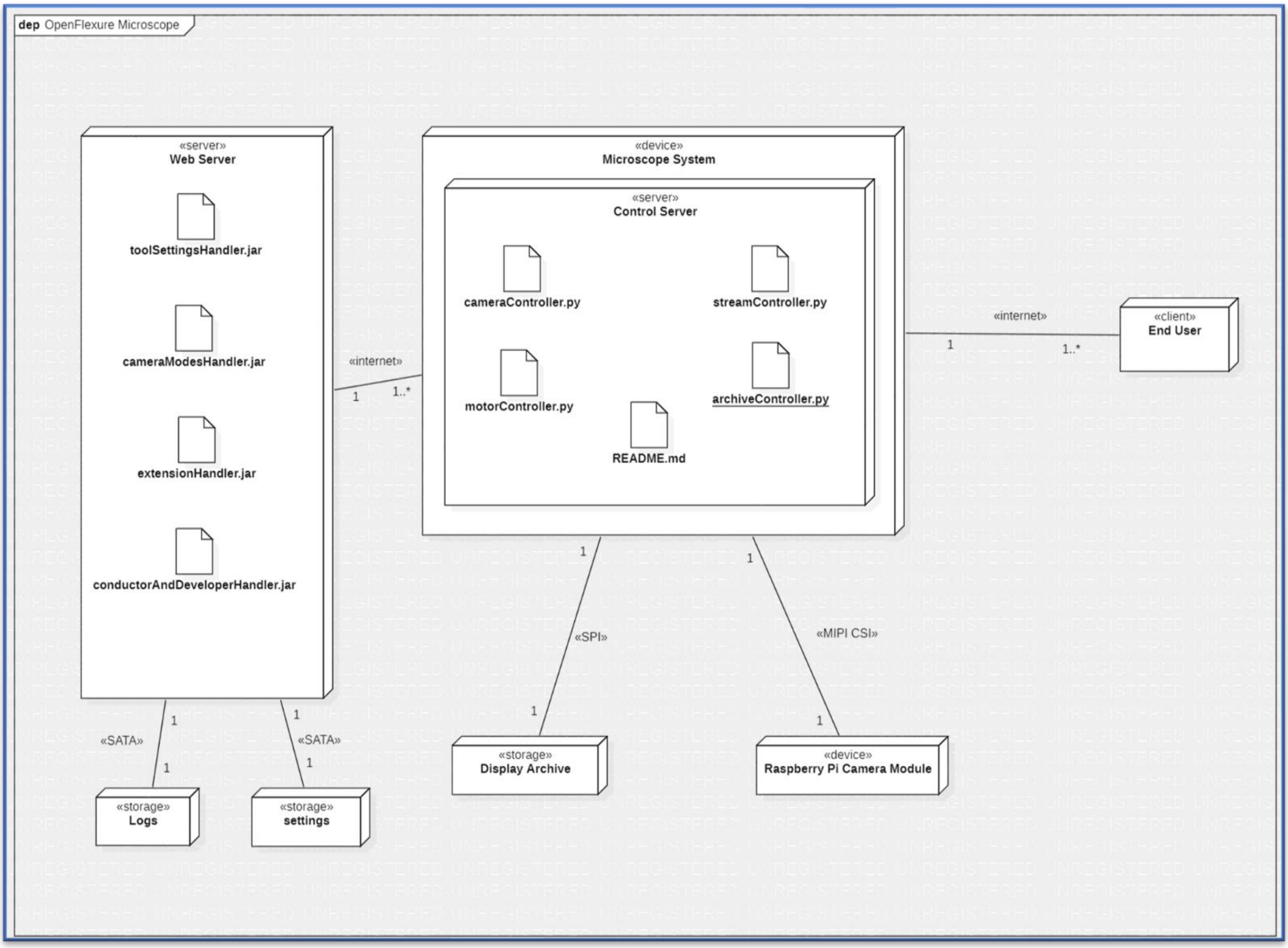


Figure 4 Deployment Diagram

Design Rationale:

- Web Server contains internal Java Run Time Environment to enable Java applications work. toolSettingsHandler, cameraModesHandler, extensionHandler and conductorAndDeveloperHandler programs all developed in Java. Since Java provides comprehensive object-oriented programming experience all handlers that perform web server operations in object-oriented manner Java is the most suitable programming language.
- Log files keeps events that web server recorded for instance GET, POST,CONNECT requests (since main protocol is HTTP for OpenFlexure's web site). Successful and susceptible request can also be monitored in these files. Therefore, web server admins can keep track of malicious activities it creates an extra security layer. Since one server keeps one log files structure one-to-one representation is suitable.
- Settings files contains configuration and operation options SATA is main protocol for storing data to Web Server. Since one settings files bundle configurate one web server one-to-one representation is used.

- Control Server programs developed in Java programming language. Since its object-oriented paradigm and multi-threading capabilities makes Java is the most appropriate programming languages for Control Server.
- Control Server and its artifacts are laid on microscope system. When remote connection is established microscope (Raspberry pi) unit behaves like a server. It provides camera and motor control access, live stream service and ability to keep captured displays.
- A camera module is connected to the Raspberry pi unit to obtain visual data. This device sends what it captured via MIPI Camera System Interface. Afterwards, Control Server decides whether to save them or send it via internet to End User. There is one control unit for each camera, therefore one-to-one is appropriate.
- archiveController program enables captured visual data to be stored in an archive namely, Display Archive unit in SD card. Since data is stored in SD card data transmission provided by Serial Peripheral Interface (SPI). There is one control unit for each display archive, therefore one-to-one is appropriate.
- End user who controls microscope remotely over the internet, sends requests over the mouse movements or keystrokes that enables user to control and observe microscope remotely.

4.3. Information View

In this view, components and their structures which will be stored is expressed in terms of data manipulation, data transmission and data collection. Database operations are examined in this section.

4.3.1 Interfaces

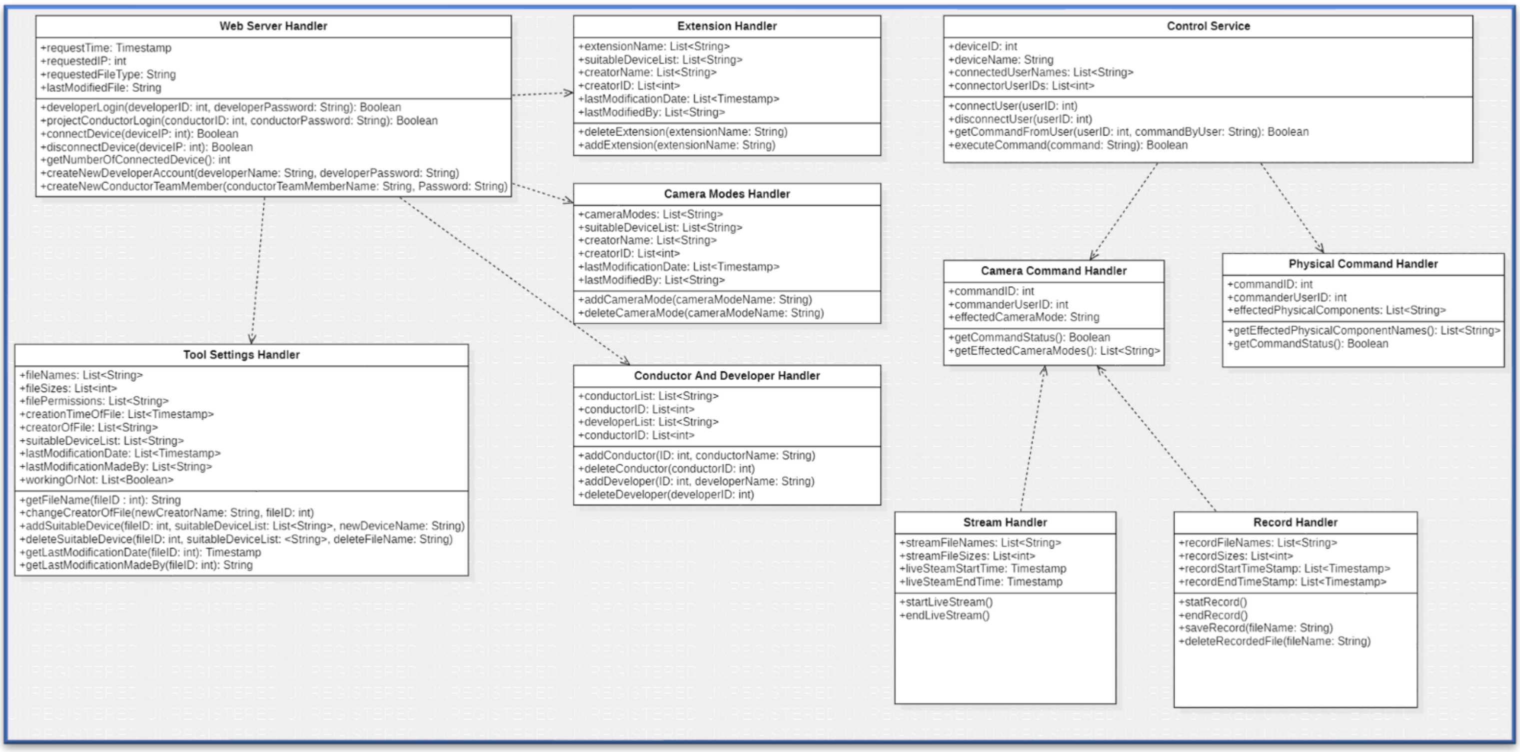


Figure 5 Interface Class Diagram

Operation	Description
developerLogin	Developer logs in with id and password successful login returns True else returns False.
projectConductorLogin	Project Conductor logs in with id and password successful login returns True else returns False.
connectDevice	Connects device with specified IP address successful connection returns True else returns False.
disconnectDevice	Disconnects device with specified IP address successful disconnection returns True else returns False.
getNumberOfConnectedDevice	Returns the number of successful connected devices.
createNewDevloperAccount	Creates new developer account with given name and password.
createNewConductorTeamMember	Creates new conductor team member account with given name and password.
getFileName	Returns the file name of a tool settings handler object as a String.
changeCreatorOfFile	Changes creatorOfFile variable to given newCreatorName
addSuitableDevice	Adds a new suitable device to suitableDeviceList array.
deleteSuitableDevice	Deletes an existing suitable device from suitableDeviceList array.
changeWorkingCondition	Changes workingOrNot Boolean value from its array to newWorkingCondition parameter.
getLastModification	Returns lastModificationDate variable for corresponding fileID parameter as a Timestamp object.
getLastModificationMadeBy	Returns a String object from lastModificationMadeBy String array which corresponds to fileID parameter.
deleteExtension	Deletes specified extension from extensionName String list.
addExtension	Adds specified extension to extensionName String list.

addCameraMode	Adds specified cameraModeName String to cameraModes array.
deleteCameraMode	Deletes specified cameraModeName String from cameraModes array.
addConductor	Adds specified conductor name and id to Conductor And Developer Handler object.
deleteConductor	Deletes specified conductor from Conductor And Developer Handler object.
addDeveloper	Adds specified developer with ID and name.
deleteDeveloper	Deletes specified developer from object.
connectUser	Connects user to Control Server object with specified userID.
disconnectUser	Disconnects user from Control Server object with specified userID.
getCommandFromUser	Fetches command from specified user successful attempts yields True else returns False.
executeCommand	Executes command successfully executed commands yields True else yields False.
getCommandStatus	Gets command status yields True if achieved otherwise returns False.
getEffectedCameraModes	Returns camera modes as a list of Strings.
stopCameraCommandExecution	It stops specified camera command's execution.
restartCameraCommandExecution	It restarts specified camera command's execution.
startCameraCommandExecution	It starts specified camera command's execution.
getEffectedPhysicalComponents	Returns effectedPhysicalComponenetNames variable of the object.
getCommandStatus	Returns command status as a Boolean value.
stopMotorCommandExecution	ops specified command's execution.
restartMotorCommandExecution	Restarts specified command's execution.
startMotorCommandExecution	Starts specified command's execution.
startLiveStream	Starts live stream.
endLiveStream	Ends live stream.

startRecord	Starts to record visual inputs.
endRecord	Ends record.
saveRecord	Saves record to SD Card with specified name.
deleteRecordedFile	Deletes specified record file.
changeCreatorOfArchiveFile	Changes creator of specified archive file to given newCreator name.
deleteArchiveFile	Deletes specified file from archive.
createArchiveFile	Creates a new archive file.

Table 16 Operation Descriptions Table

Design Rationale:

- Web server is responsible for extension storage management, permission control, developer and conductor logins, connected device tracking, STL files deposit and transportation.
- Tool settings handler maintains existing configuration files and stores information about their versions. There is a dependency relationship between Web Server.
- Extension handler operates extensions created by developers. It adds extensions to Web Server repositories also deletes from there. Thus there exists dependency relationship between them.
- Camera modes handler manages camera modes by adding to the server and deleting them from the server. Therefore, it dependent to Web Server.
- Conductor and developer handler be able to add and delete conductors and developers from Web Server. Dependency relationship exists between them.
- Control Server entity is independent from above objects it operates on Raspberry Pi unit. Furthermore, serves visual data and manages control command transaction between end-user and microscope module.
- Camera controller enables user to control camera modes by commanding manner. There are dependency relations between camera controller and control server, stream controller and archive controller.

- Motor controller enables users to move the motors. There is a dependency relation between control server and motor controllers.
- Stream controller manages live stream capabilities.
- Archive controller handles archive files.

4.3.2 Database Operations

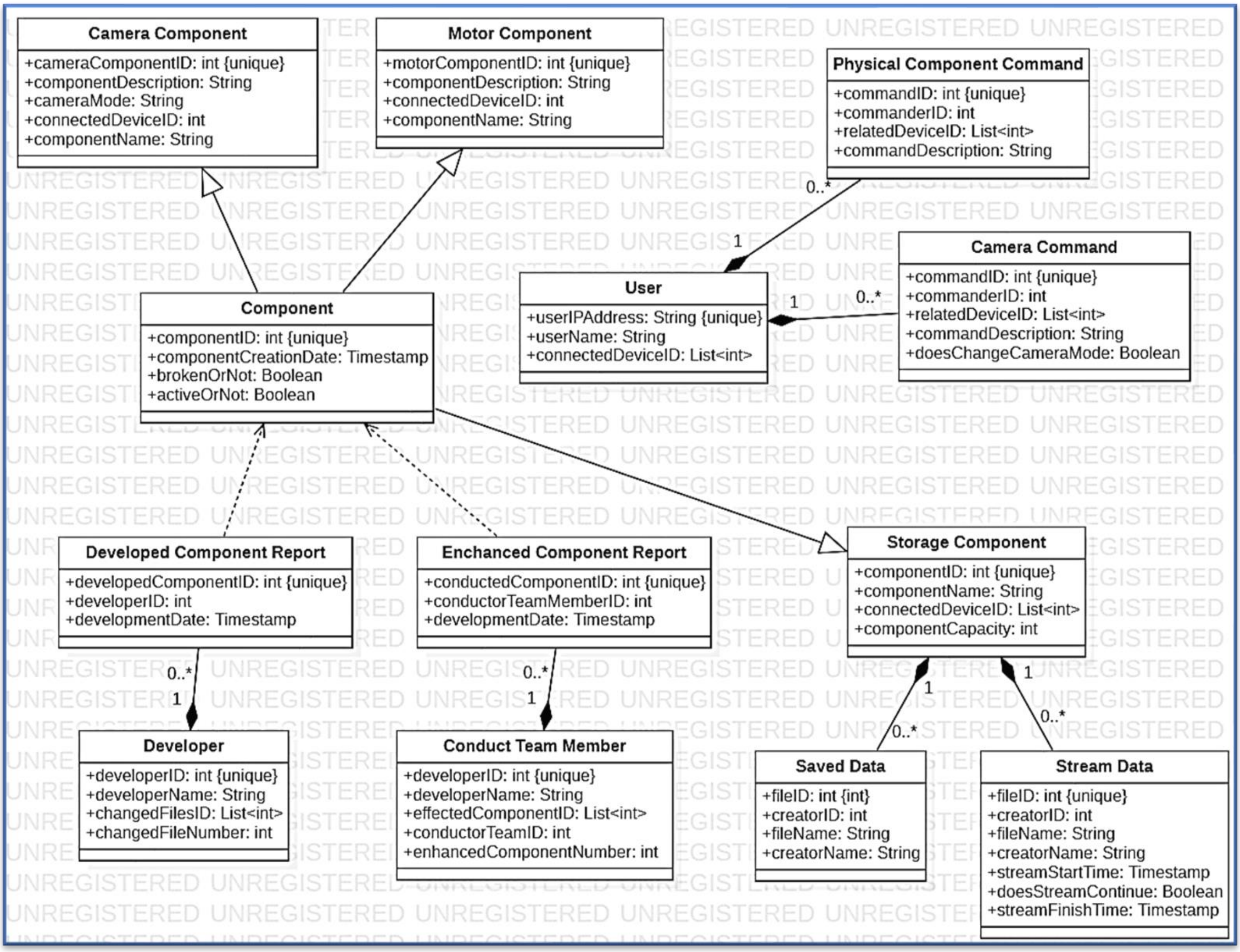


Figure 6 Database Class Diagram

Operation	CRUD Operations
developerLogin	Create: Read: Developer Update: Delete:
projectConductorLogin	Create: Read: Conduct Team Member Update: Delete:
connectDevice	Create: Read: ComponentID Update: Component Delete:
disconnectDevice	Create: Read: ComponentID Update: Delete: Component
getNumberOfConnectedDevice	Create: Read: ConnectedDeviceID Update: Delete:
createNewDevloperAccount	Create: Developer Read: Update: Delete:
createNewConductorTeamMember	Create: Conduct Team Member Read: Update: Delete:
getFileName	Create: Read: fileName Update: Delete:

changeCreatorOfFile	Create: Read: Update: CreatorID,CreatorName Delete:
addSuitableDevice	Create: Component Read: Update: Delete:
deleteSuitableDevice	Create: Read: Update: Delete: Component
changeWorkingCondition	Create: Read: componentID Update: Component Delete:
getLastModification	Create: Read: lastModification Update: Delete:
getLastModificationMadeBy	Create: Read: lastModificationMadeBy Update: Delete:
deleteExtension	Create: Read: developerID Update: Delete: Extension
addExtension	Create: Extension Read: developerID Update: Delete:

addCameraMode	Create: cameraMode Read: Update: Delete:
deleteCameraMode	Create: Read: Update: Delete: cameraMode
addConductor	Create: ConductTeamMember Read: Update: Delete:
deleteConductor	Create: Read: Update: Delete:ConductTeamMember
addDeveloper	Create: Developer Read: Update: Delete:
deleteDeveloper	Create: Read: Update: Delete: Developer
connectUser	Create: Read: UserIPAddress,userName Update: connectedDeviceID Delete:
disconnectUser	Create: Read: UserIPAddress,userName Update: Delete: connectedDeviceID

getCommandFromUser	Create: Read: PhysicalComponentCommand or CameraCommand Update: Delete:
executeCommand	Create: Read: PhysicalComponentCommand or CameraCommand Update: Delete:
getCommandStatus	Create: Read: PhysicalComponentCommand or CameraCommand Update: Delete:
getEffectedCameraModes	Create: Read: cameraMode Update: Delete:
stopCameraCommandExecution	Create: Read: commandID Update: Delete: Camera Command
restartCameraCommandExecution	Create: Read: CommandID Update: Camera Command Delete:
startCameraCommandExecution	Create: Camera Command Read: CommandID Update: Delete:
getEffectedPhysicalComponents	Create: Read: relatedDeviceID Update: Delete:
getCommandStatus	Create: Read: PhysicalComponentCommand or CameraCommand Update: Delete:

stopMotorCommandExecution	Create: Read: commandID Update: Delete: Physical Component Command
restartMotorCommandExecution	Create: Read: commandID Update: Physical Component Command Delete:
startMotorCommandExecution	Create: Physical Component Comamnd Read: commandID Update: Delete:
startLiveStream	Create: Stream Data Read: Update: Delete:
endLiveStream	Create: Read: componentID Update: Delete: Stream Data
startRecord	Create: Saved Data Read: componentID Update: Delete:
endRecord	Create: Read: componentID Update: Saved Data Delete:
saveRecord	Create: Saved Data Read: componentID Update: Delete:
deleteRecordedFile	Create: Read: componentID Update: Delete: Saved Data

changeCreatorOfArchiveFile	Create: Read: fileID,componentID Update: creatorName Delete:
deleteArchiveFile	Create: Read: fileID,componentID Update: Delete: savedData
createArchiveFile	Create: savedData Read: fileID,componentID Update: Delete:

Table 17 CRUD Operations Table

Design Rationale:

- Derived components connected logically and physically with main Component table. All queries will pass over it.
- A user can send more than one Physical and Camera commands therefore one-to-many representation is used.
- In a storage component there might be more than one saved and streamed data there fore one-to-many representation is chosen.
- A developer or a conduct team member can create zero or more reports therefore one-to-many representation is used.
- MySQL database is deployed.

4.4 Interface View

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

4.4.1 Internal Interfaces

Interface between Camera Command Handler and Stream Handler

When the user starts streaming Camera Command Handler and Stream Handler begin to communicate over the established internal connection. Stream Handler sends commands which affects camera modes or camera status. If Stream Handler sends a command which changes camera mode, then it will also change effected camera mode variable in Camera Command Handler object. That communication is crucial since it enables live stream to start and operate smoothly and without any visual distortions. If any problem occurs between two steps there is a possibility to encounter with unexpected and indeterministic behavior.

Design Rationale:

- Camera Command Handler checks whether incoming commands are valid or not that's way incorrect messages can demand correct ones.
- Most of the time Stream Handler does not send any command packages since Camera Modes does not change rapidly.

Interface between Camera Command Handler and Record Handler

When the user starts recording, Camera Command Handler and Record Handler begin to communicate over the established internal connection. Like interface operations between Camera Command Handler and Stream Handler, Record Handler also sends commands to Camera Command Handler. Those commands mostly determine which camera modes will be used while recording visual data. In order to achieve well-operated record command packets, need to be conveyed correctly.

Design Rationale:

- Camera Command Handler checks whether incoming commands are valid or not that's way incorrect messages can demand correct ones.
- Most of the time Record Handler does not send any command packages since Camera Modes does not change rapidly.

Interface between Control Service and Camera Command Handler

Control Service receives keystrokes or mouse movements and clicks from external user or users. Another interface is managing that process will be covered in external interfaces section. Afterwards, it transforms those inputs to command packets which Camera Command Handler can understand and execute them. Control service solely conveys corresponding commands in other words when user wants to make changes on physical components of the microscope it does not conveys those messages to Camera Command Handler.

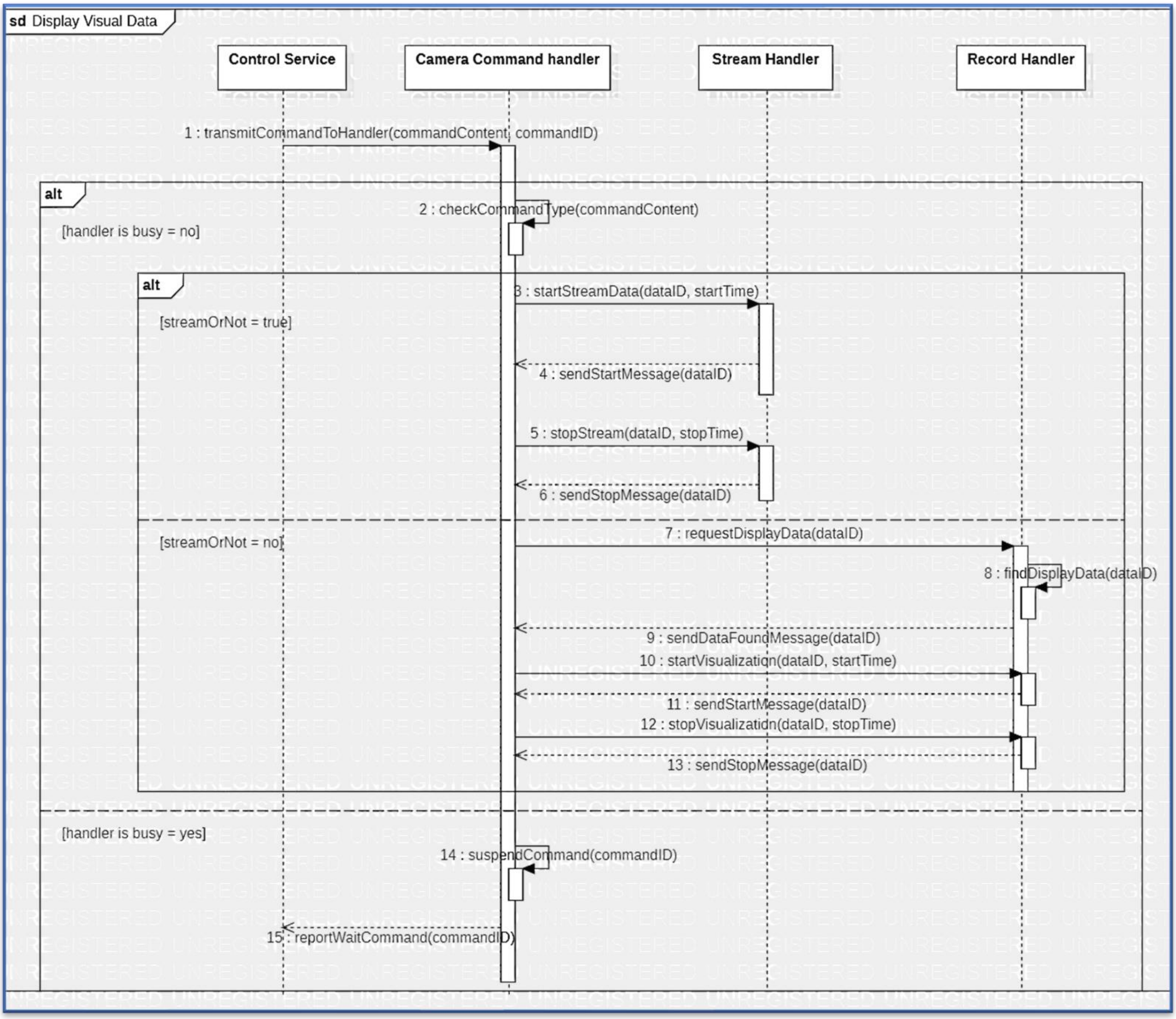


Figure 7 Interface between Control Service and Camera Command Handler Sequence Diagram

Design Rationale:

- Most of the time connection between Camera Command Handler and Control Service stays idle.
- Camera Command Handler and Control Service Interface not only uses mentioned handler and service but also uses Stream Handler and Record Handler since interface mostly operates on these processes.

Interface between Control Service and Physical Command Handler

Control Service receives keystrokes or mouse movements and clicks from external user or users. Another interface is managing that process will be covered in external interfaces section. Afterwards, it transforms those inputs to command packets which Physical Command Handler is able to understand and execute them. Control service solely conveys corresponding commands in other words when user wants to make changes on physical components of the microscope it does not conveys those messages to Physical Command Handler.

Design Rationale

- Physical Command Handler gets effected component names from Control Service therefore there is a dependency relation.
- Command status will be determined by Control Service invalid statuses will be dropped.

Interface between Web Server Handler and Tool Settings Handler

Web Server Handler manages web server operations mentioned server will be reachable from https://openflexure.org/projects/microscope/ address. It contains STL files to build microscope also some guides and setting tools. Those tools operated by Tool Settings Handler. This handler maintains configuration files for setting up microscope and enables users to achieve extra microscope functionalities.

Design Rationale

- Tools Settings Handler is dependent to Web Server Handler therefore dependency relation is suitable.
- Tool Settings are not a trivial web file since it continuously updated and new configurations are located quickly. Therefore, a handling mechanism needed to deploy.

Interface between Web Server Handler and Extension Handler

Web Server Handler like mentioned above operates not only web hosting also responsible for other components and handlers to work properly. It can be said that Web Server Handler main management unit for all the operations on the internet side. Therefore, other web side handlers are in a dependency relation with it

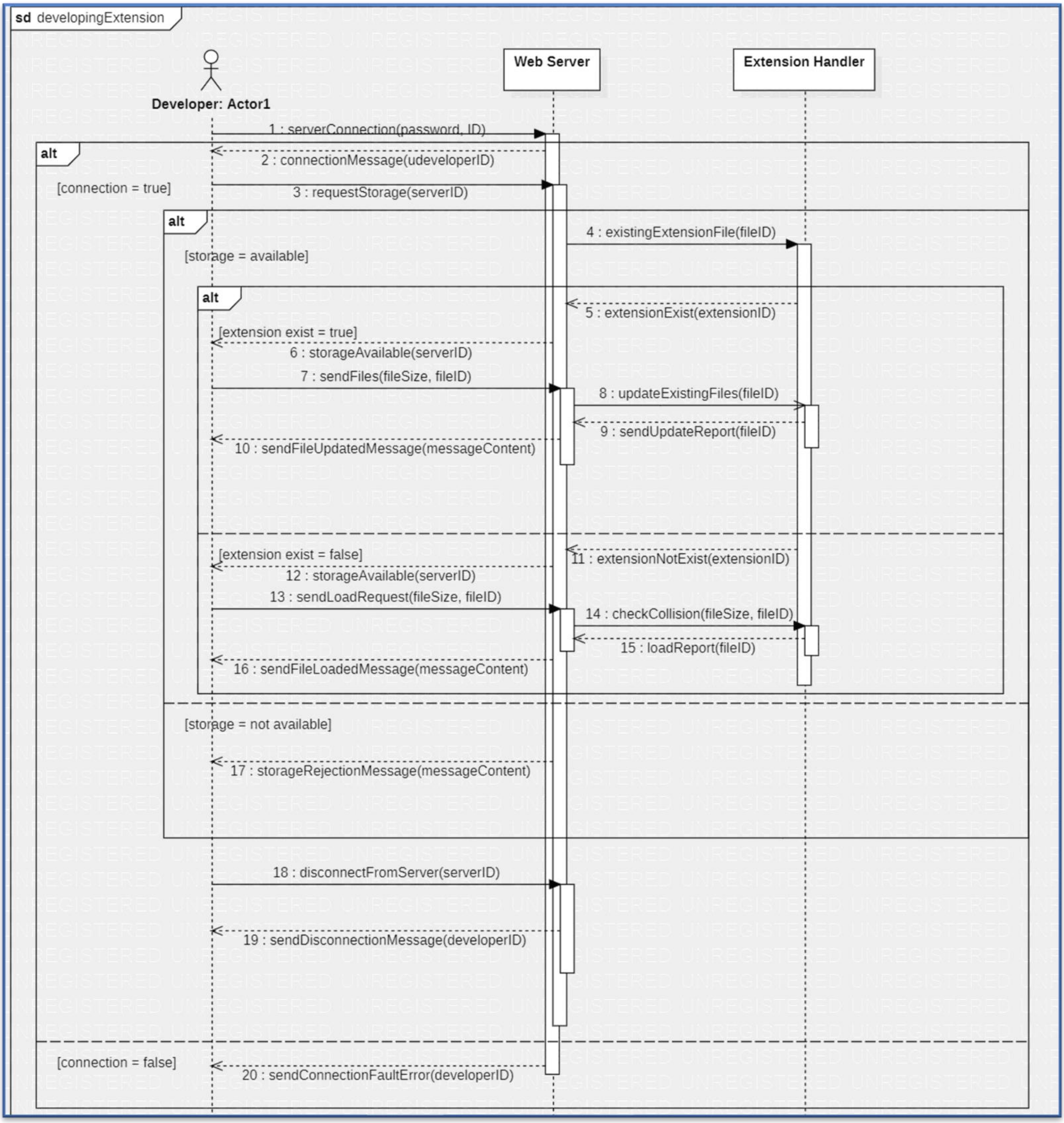


Figure 8 Interface between Web Server Handler and Extension Handler

Design Rationale:

- Web Server Handler and Extension Handler communication is crucial for maintaining overall user experience therefore it needs a private channel away from interventions.
- Web server keeps all tool settings inside of itself. Therefore Web Server Handler in a higher abstraction level also operates Extension Handler.

Interface between Web Server Handler and Camera Modes Handler

Web Server Handler is responsible unit for all web hosting operations and dependent handler's operation. Camera Modes Handler is also a web side service which responsible for storage, maintain and update of Camera Modes users interact. An example flow shows how Camera Modes Handler and Web Server Handler are operate; when a new camera mode needed to be added and published. First, Camera Mode Handler registers it with addCameraMode method afterwards Web Server Handler gets that mode and publishes it to corresponding website.

Design Rationale:

- An explicit handler which is Camera Mode Handler needed since process of addition and deletion of camera modes needed to be done explicitly.
- Camera Modes Handler dependent to Web Server Handler since publishing added modes can not be done without supervision of Web Server Handler.

Interface between Web Server Handler and Conductor and Developer Handler

Web Server's last operated sub handler is Conductor and Developer Handler. An example flow of how Web Server Handler and Conductor and Developer Handler interact is; when a conductor or a developer wanted to include in a OpenFlexure Microscope Project, it registers with a name and id. That registration operation will be done with addConductor or addDeveloper method. After registration completed Web Server Handler publishes it in corresponding website.

Design Rationale:

- Conductor and Developer Handler is a sub-handler for Web Server Handler. The reason why there is an explicit handler is keeping system tidier and more stable.
- Conductor And Developer Handler is dependent to Web Server Handler since when a developer or a conductor added it need to be published on a web site.

4.4.2 External Interfaces

User Interfaces

OpenFlexure Microscope Project supplies user interfaces to not only users but also project stakeholders. Those are namely; Developer Interface, Conductor Team Member Interface, Remote Connection Interface, Local Connection Interface and Connection API Preferences Interface.

Developer Interface

OpenFlexure Microscope project enables people to not only use existing extensions but also develops their extensions for OpenFlexure Connect Web API. Most of the job done via Python programming language. A website https://openflexure-microscope-software.readthedocs.io/en/master/ is dedicated solely for this purpose. A developer installs required libraries and starts to manage development server with real-time debug logging. Some json files also needed to be created and manipulated.

Design Rationale:

- Developer interface guide created on GitLab since its an open project and it mostly lay on GitHub and GitLab.
- Python is chosen for main programming language to create extensions since it is easy to use and highly widespread among developers.

Conductor Team Member Interface

OpenFlexure Project provides an interface for who conducts and maintains the project. In spite of the fact that it is an open-source project still there exist some responsible people mostly on hardware and new optic technologies deployment to microscope. For this purpose the project needs an interface to satisfy those needs. When a conductor team member gets involved in the project first, it is registered by project admins and a user panel website assigns for him/her. All the jobs will be done through this panel. When a conductor discharged deleteConductor method will be used. It removes all data and privileges related with him/her.

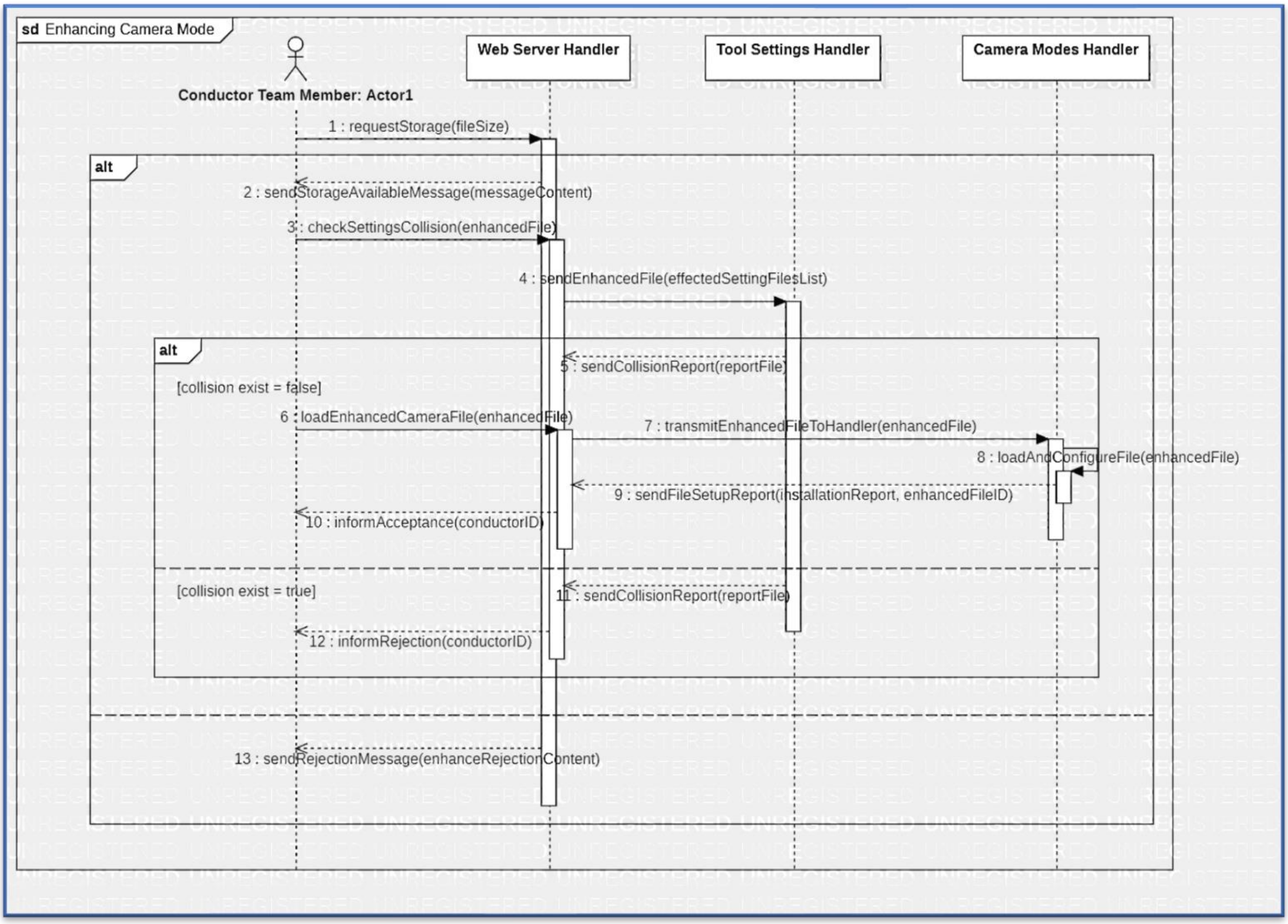


Figure 9 Conductor Team Member Interface Sequence Diagram

Design Rationale

- Conductor Team Members needed to a distinct web interface since extra functionalities will be implemented through that.
- Deleted conductor team members requires more extra attention since if a removed member still be able to access restricted materials the project can be in the danger.

Local Connection Interface

A user can connect to OpenFlexure Microscope in two ways namely, Local Connection Interface or Remote Connection Interface. Local Connection Interface enables users to connect directly. Since the microscope is an external device which connected to Raspberry Pi Unit, as we know Raspberry Pi is a computer with an operating system which peripheral devices can be connected through serial connections. Local connection means that to display what microscope examines and control its camera and motors with connected mouse and keyboard. To use that user runs OpenFlexure Connect app then it will mark “Connect locally” option below shown.

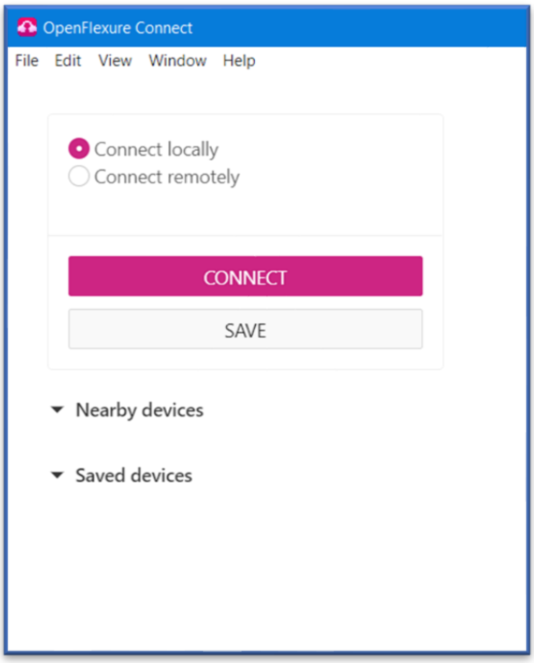


Figure 10 OpenFlexure Connect Local Connection

Design Rationale:

- Local access is implemented since not every time an external computer can connect over the internet or ethernet.
- Local Connection Interface is developed for Raspbian operating system since it will work on this distribution.

Remote Connection Interface

OpenFlexure Microscope provides an interface to users for remote connection. It is not the case that users demand to connect locally sometimes user and the microscope does not be in the same environment. Therefore, Remote Connection Interface is developed. Mainly it enables users to connect in higher abstraction application namely, SSH, Telnet and HTTP. In lower layer they will connect over the LAN, WLAN, Ethernet and Internet. First user connects to the same network it can be internet or a local area network without internet. Since the machines are in the same broadcast domain can communicate directly without using any gateway or network address translation protocols. After establishing a connection, a user will mark “Connect remotely” option like shown below. Need to specify hostname and port to start communication. After successfully managing initial configurations, user is able to use microscope remotely.

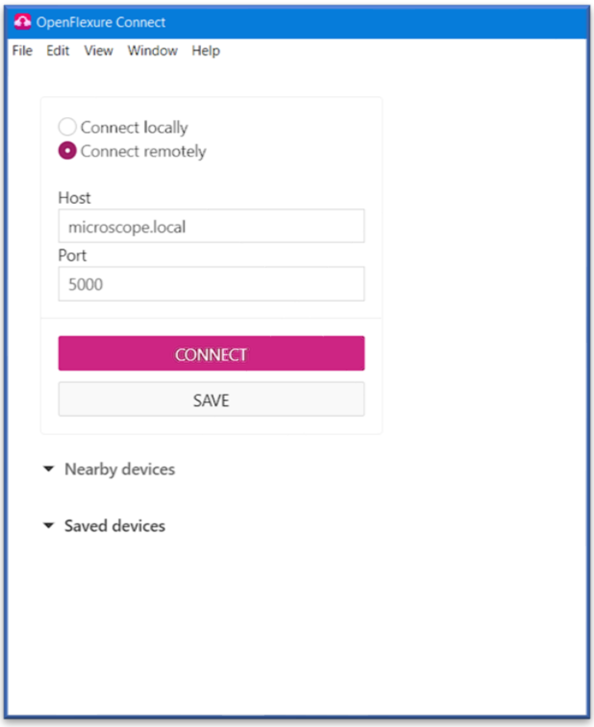


Figure 11 OpenFlexure Connect Remote Connection

Design Rationale:

- Remote Connection Interface is developed since some users demand remote functionalities. For instance, microscope and examined material stays in the laboratory and user locates outside of the lab.
- OpenFlexure Connect works with TCP/IP fashion therefore hostname and port needed to be typed correctly.

Connection API Preferences Interface

OpenFlexure Connect App provides some extra functionalities that users can benefit. In the upper part of OpenFlexure Connect App there is some tabs manages this interface. Namely, File, Edit, View, Window and Help tabs operates Connection API Preferences Interface. A user can exit, edit text that inserted, show developer tools similar to browser's developer tools section some console to run JavaScript codes and some extra utilization tools for debugging and observing OpenFlexure Connect App's behaviors.

Design Rationale:

- Connection API Preferences Interface developed to provide extra functionalities.
- Debugging tools implemented since if there any problem emerges a user can detect the problem easily.

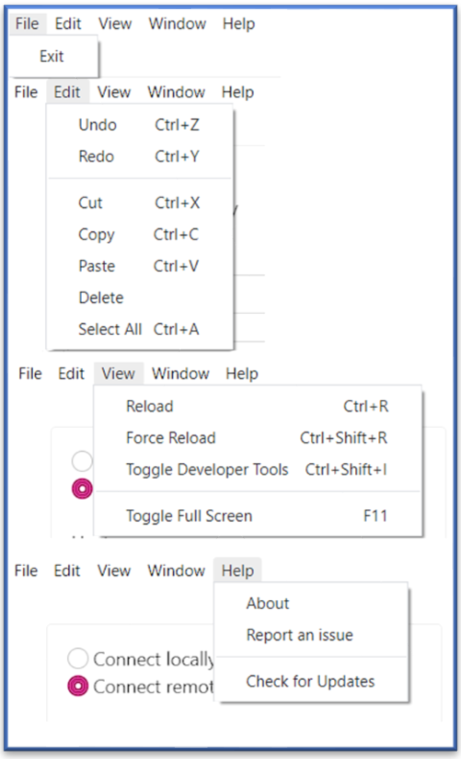


Figure 12 OpenFlexure Connet Preferences Tabs

System Interfaces

Interface between User Control Mechanism and Camera Movement Mechanism

Interface between User Control Mechanism and Camera Movement Mechanism mainly operates the interactions and functions between User Control Mechanism and Camera Movement Mechanism. When a user regardless of connected locally or remotely tried to control camera it will manages it over the User Control Mechanism. Then User Control Mechanism sends appropriate movement requests to Camera Movement Mechanism. In this way both communication and operation will be successfully done.

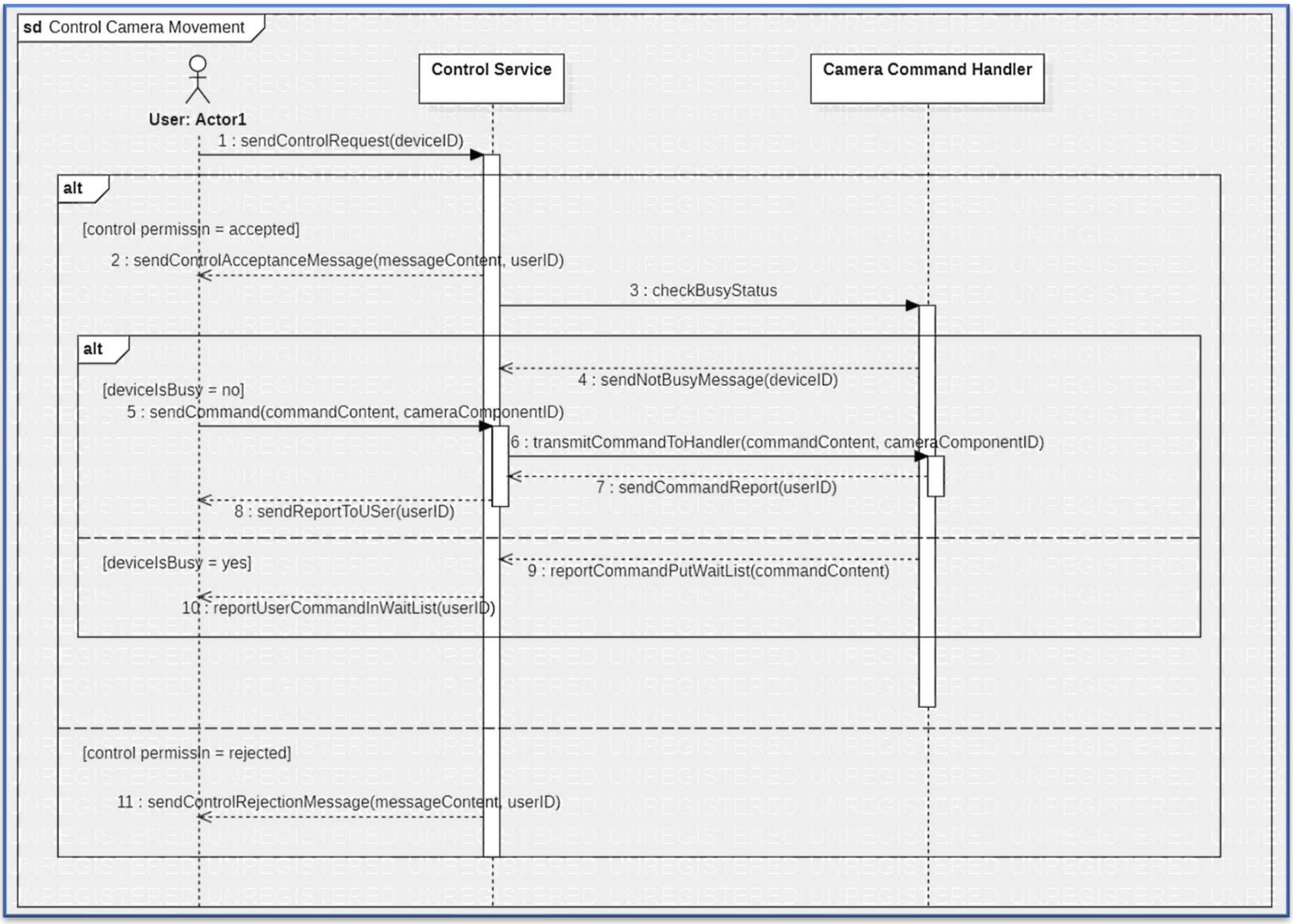


Figure 13 Interface between User Control Mechanism and Camera Movement Mechanism Sequence Diagram

Design Rationale

- Interface between User Control Mechanism and Camera Movement Mechanism created since when a user tries to control camera it needs to an interface to manage this process.
- To make user to move camera more smoothly communication between User Control Mechanism and Camera Movement Mechanism is crucial therefore some extra functionalities added.

Interface between User Control Mechanism and Camera Modes Handler

Interface between User Control Mechanism and Camera Modes Handler enables users to manage and switch between camera modes smoothly. When a user demands a camera mode change he/she will sends mouse movement signals and keystrokes to User Control Mechanism. Then User Control Mechanism will convert those signals as a command which Camera Modes Handler can understand. In that movement the interface will involve and deal with the communication operation. The interface solely contains available camera modes and when a new mode is inserted or downloaded from the website then it will display that option.

Design Rationale

- This interface needed since it enables user to change camera modes in microscope system.
- A user needs to add or download new camera modes to achieve more functionality

Interface between User Camera Mode Setter and Camera Modes Handler

Interface between User Camera Mode Setter and Camera Modes Handler enables setting camera modes autonomously. When a user in this example a scientist wants to observe the microscopic creatures in different camera modes with respect to time variable this interface operates this process. Scientist schedules which modes will be used when. Then this interface makes this work be done.

Design Rationale

- User needs to schedule Camera Mode Setter at first.
- Camera Mode Setter and Camera Modes Handler need to communicate well.

Interface between User Camera Display System and Record Handler

Interface between User Camera Display System and Record Handler enables record requests to be operated smoothly. Since when a user sends a display command through User Camera Display System it required to be sent to Record Handler. After receiving record command from User Camera Display System Record Handler starts to record displays. Not only record command is sent by this interface but also some extra functionalities and options added or discarded. Interface can decide on when the record will end or whether the storage capacity exceeded or not.

Design Rationale

- User Camera Display system mainly operates how display transferred to the user however the interface between Record Handler solely focusses on how the record process managed.
- Record Handler cannot operate without receiving signals from User Camera Display System.

Interface between User Camera Display System and Stream Handler

Interface between User Camera Display System and Stream Handler enables stream requests to be operated smoothly. Since when a user sends a display command through User Camera Display System it required to be sent to Stream Handler. After receiving stream command from User Camera Display System Stream Handler starts to stream displays. Not only stream command is sent by this interface but also some extra functionalities and options added or discarded. Interface can decide on when the stream will end or whether the stream limit exceeded or not.

Design Rationale

- User Camera Display system mainly operates how display transferred to the user however the interface between Stream Handler solely focusses on how the stream process is managed.
- Stream Handler cannot operate without receiving any signals from User Camera Display System. It always waits stream commands.

Interface between Physical Command Handler and Control Mechanism

Physical Command Handler is the responsible unit for controlling physical peripherals that are included to system. Those peripherals motors which moves camera and zoom tools that operates zooming. Control Mechanism is a responsible unit for all the microscope operations not only physical ones also virtual ones will be operated by this component. The interface between that enables Control Mechanism to achieve Physical Commands to be executed by Control Mechanism. An example flow: a user first sends signals over the mouse clicks or keystrokes then Control Mechanism converts those signals to commands which Physical Command Handler can understand. Then Physical Command Handler sends appropriate signals to motors or zooming tools. Finally, all the process is ends and camera completes its move or zoom.

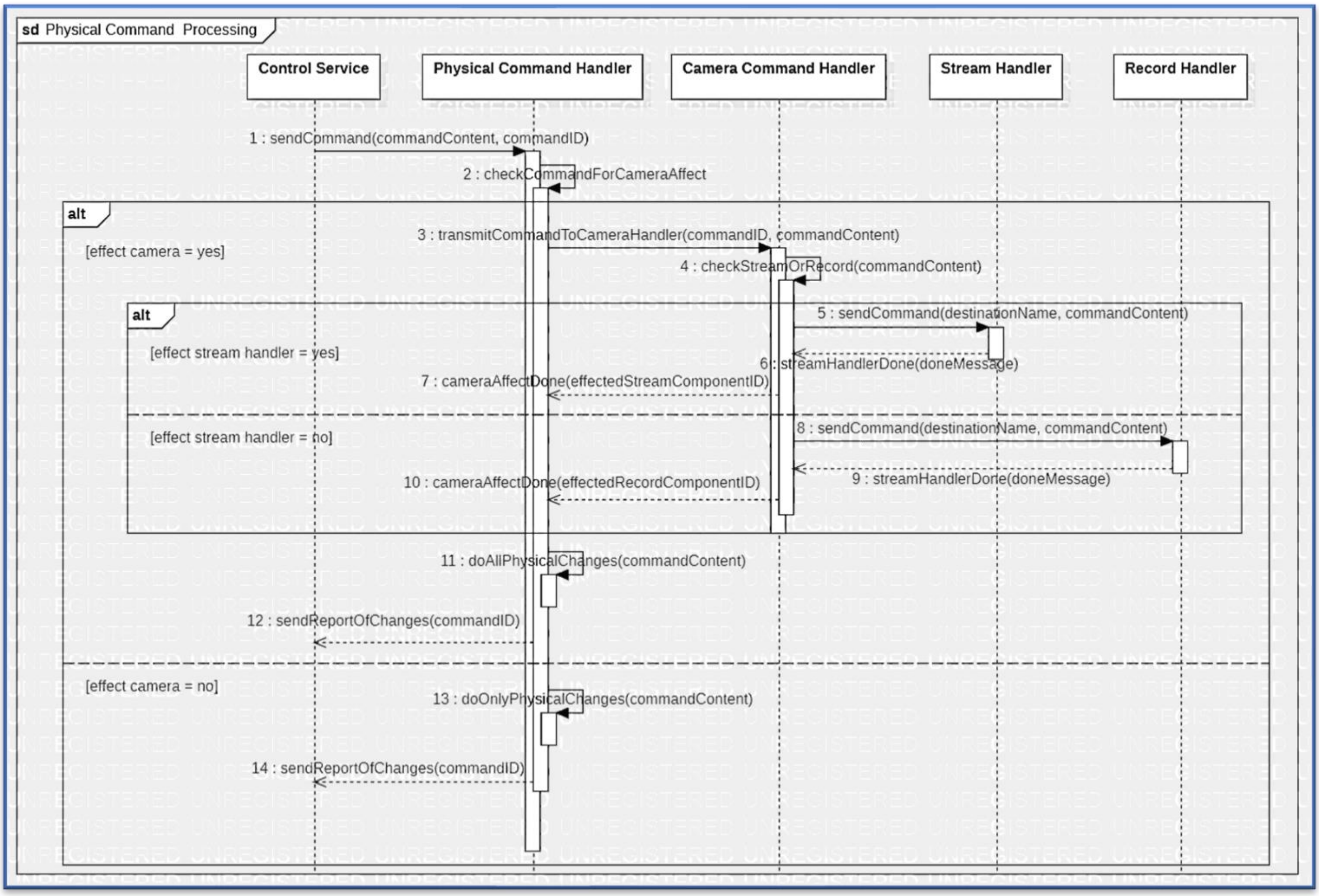


Figure 14 Interface between Physical Command Handler and Control Mechanism Sequence Diagram

Design Rationale

- This interface implemented since Control Mechanism can not directly reach physical components.
- Physical Command Handler always waits requests from Control Mechanism it can not operate with its own.