Software Requirements Specification

for

OpenFlexure Microscopy

Version 1.0

Prepared by
Adnan Harun DOGAN
& Mert SAADET

April 23, 2021

Table of Contents

List of Figures

List of Tables

Revision History

Name	Date	Reason For Changes	Version
SRS First Draft	Apr 15, 2021	Initial commit	1.0.0
SRS Full Draft	Apr 22, 2021	Final	1.0.1

Chapter 1

Introduction

1.1 System Purpose

This project, developed by a group of researchers from Tanzania, Cambridge and Bath Universities, aims to provide a 3D-printed, customisable microscope as a cheap and locally-producible optical solution for especially the regions where the conventional microscopes are not easily accessible. The researchers and microscopists around the world will be able to reach and maintain this open-source project to make it available for everyone.

1.2 System Scope

- Web Application Interface includes STL files and SD card images of Raspbian OpenFlexure OS, besides documentation and notes regarding usage of the product.
- OpenFlexure Connect allows client to offers local and remote connection establishment to the microscope over the app. Users will enter IP address or select a local device over the network.
- OpenFlexure Software Application contains view interface in which a live stream from the RaspberryPi camera is available.
- OpenFlexure Software Application's Capture interface enable users to change filename and resolution as well as configure data type of the frame and taking notes about frame. Moreover users can add tags to the images and scan a wide range of area.
- Navigation interface of the OpenFlexure Software Application includes the settings for devices(joystick, keyboard) to move microscope camera. In addition to that a user can enter desired coordinates as an input or click on the live stream frame to relocate the camera.
- Storage interface of the OpenFlexure Software Application manages the necessary space images have been captured. A user can change storage device to an external storage device.
- Gallery interface of the OpenFlexure Software Application lists the captured and scanned images. A user can delete and download images as well as export them from gallery.

1.3 System Overview

1.3.1 System Perspective

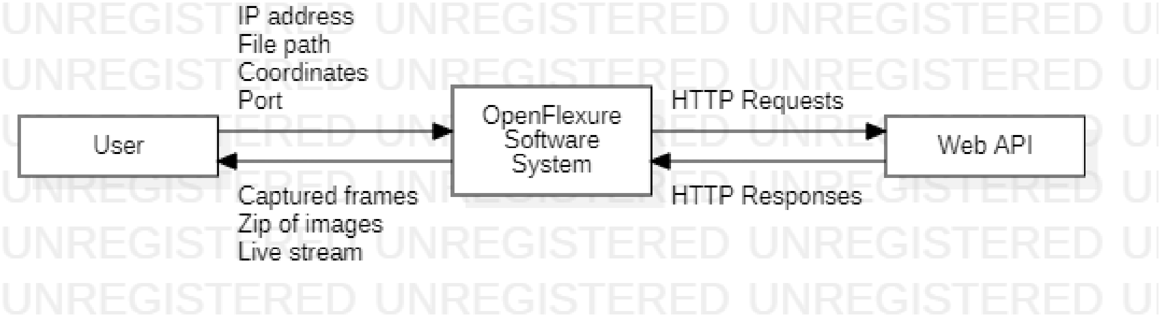


Figure 1: Context Model

The OpenFlexure microscope product is not an element of a larger system. However OpenFlexure control code is splitted into two parts using client-server architecture. Client and server applications are interfaced via a web API that conforms W3C[1] Web of Things Standard [1]. A graphical user interface is used for user interaction of the OpenFlexure Microscope system. User has to either download and install client application on the remote machine or can make use of the RaspberryPi in local for user interface. The client application will run on devices connected via Ethernet or WiFi interface to enable remote control through internet protocol(IP) networks. Designing software stack around W3C WoT Architecture paved the way for comprehensive integration with existing microscopy software solutions and enabled networked interaction using W3C WoT API model. Therefore the graphical user interface can be configured or modified by those who want to make modifications on the OpenFlexure Microscope. Written extensions in any modern language that support web requests such as Python, will have a direct access to microscope and those extensions can provide HTML API endpoints and HTML interfaces.

[1] Word Wide Web Consortium, https://www.w3.org/WoT/

1.3.1.1 System interfaces

This microscope is formed by the combination of small physical and software parts. These parts intercommunicate among themselves and with the user through interfaces. Those interfaces are:

- User interfaces
- Hardware interfaces
- Software interfaces
- Communication interfaces

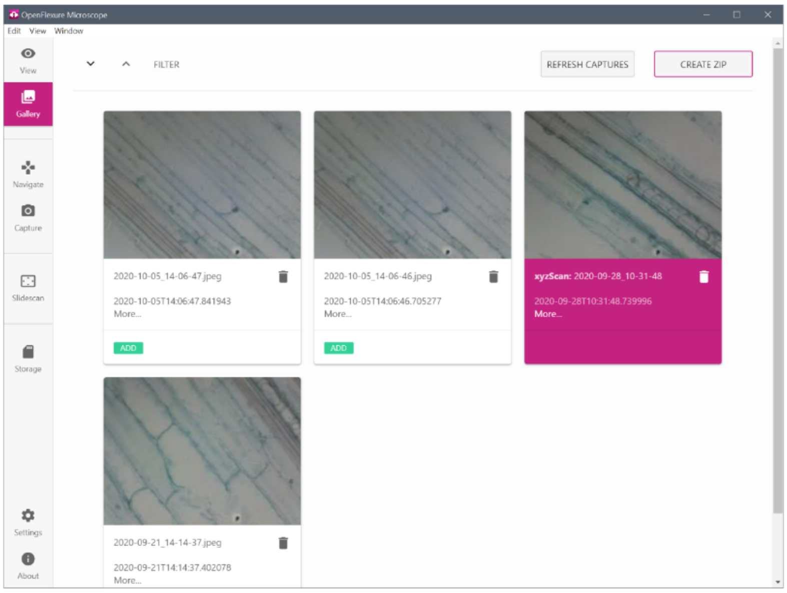
Users interacts with the client-side of the OpenFlexure Microscope Application program using the modular designed graphical user interface. The server-side of the OpenFlexure Microscope Application program communicates with the physical tools such as RaspberryPi camera, using the hardware interface.The server and client side of the OpenFlexure Microscope Application communicate with each other using HTTP Web API.

1.3.1.2 User interfaces

Microscopists who want to control and manage OpenFlexure Microscope, shall download and install the suitable version of OpenFlexure Connect application according to operating system type of their computers. OpenFlexure Connect is currently available for Windows and Linux operating systems. In addition to setting up local computer, RaspberryPi on the microscope must have a custom RaspbianOS image which is designed specifically for OpenFlexure Microscope Project and currently available at Open Flexure website[2]. Automatic detection of the server's IP address via multicast Domain Name System (mDNS) requires both OpenFlexure Connect and RaspberryPi to be connected.

The first interface welcomes user and helps establishing connection, then redirects user to other interfaces such as capture interface and navigation interface, which can be seen in Figure 2 and Figure 3. Gallery Interface displays images in the server's SD Card. Users may also create and add tags to images. Navigate Interface enables users to move the camera to any location on the 3D coordinate plane. They may also configure navigation's step size for all directions. Moreover, the autofocus feature helps the user zoom in or out the camera to get the maximum resolution automatically. Capture Interface takes a photo from the Server's Camera. Users shall configure the photos' path and resolution, as well as create tags and notes. Slide-Scan Interface takes a sequence of images instead of one per click. Finally, Storage Interface helps users to change the storage path for captured images.

[2] https://openflexure.org/projects/microscope/



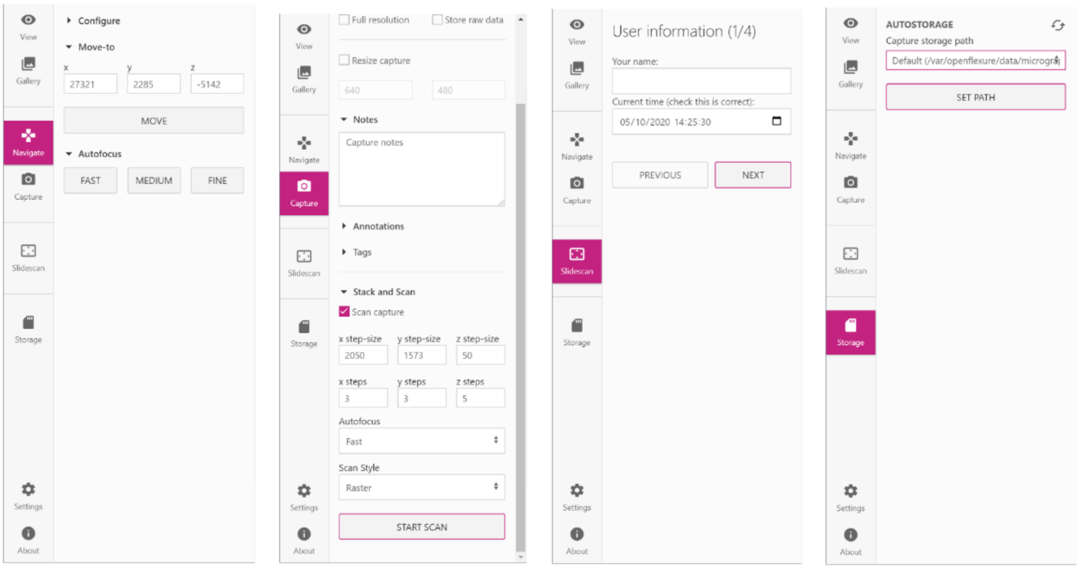
2020-10-05_14-06-47.jpeg
2020-10-05T14:06:47.841943
More...

2020-10-05_14-06-46.jpeg
2020-10-05T14:06:46.705277
More...

xyzScan: 2020-09-28_10-31-48
2020-09-28T10:31:48.739996
More...

2020-09-21_14-14-37.jpeg
2020-09-21T14:14:37.402078
More...

Figure 2: Gallery Interface



x 27321 y 2285 z -5142

□ Full resolution □ Store raw data

□ Resize capture

640 480

Capture notes

☑ Scan capture

x step-size 2050 y step-size 1573 z step-size 50

x steps 3 y steps 3 z steps 5

Autofocus: Fast

Scan Style: Raster

User information (1/4)

Your name:

Current time (check this is correct): 05/10/2020 14:25:30

Capture storage path: Default (/var/openflexure/data/microgra

Figure 3: Navigate, Capture, Slide-Scan, and Storage Interfaces

1.3.1.3 Hardware interfaces

A server application running on a Raspberry Pi handles the interaction with the physical hardware. This server application takes communication with the sample translation stage, imaging camera, and any additional hardware and logic for data management and different functions such as tiled scans and auto-focus. RaspberryPi on OpenFlexure Microscope is connected via Serial Communication with Arduino. Access to physical hardware throughout the Arduino is carefully managed to avoid conflicting instructions by use of re-entrant blocks(Rlocks). The lock prevents requests from sending instruction to a device which is already in use. picameraX a forked version of the picamera library is used for accessing lens shading table in the OpenFlexure Microscope's RaspberryPi camera's GPU-based image processing pipeline. The software is configured such that an external joystick I/O can control OpenFlexure Microscope's motors.

1.3.1.4 Software interfaces

OpenFlextture Server executes, on Rasberry Pi OS, python scripts for server applications and controlling physical hardware on the A Raspberry Pi. Main server applications use the "Flask"[3], a web application framework, and includes various utilities to simplify thread-based concurrency, mDNS discovery, hardware synchronization, and documentation generation. Additionally, python scripts runnin on the server makes use of the programming interfaces of numpy, matplotlib, opencv, and scipy libraries to get better performance and portability.

1.3.1.5 Communication interfaces

IP networking is the leading way of communication between clients and servers. Users may choose either Ethernet or WiFi to connect large numbers of microscopes simultaneously and without the need for bespoke or proprietary hardware. Users can control the microscope directly from another device (without any external router or switch) using either ad-hoc wireless or wired network. Moreover, standard local area networks (LANs) enable one-to-many or many-to-one communication with OpenFlexure Microscopes. Finally, Secure Shell Protocol (SSH) port forwarding and connections through Virtual Private Networking (VPN) also available to control OpenFlextture Microscope remotely via well-established secure protocols.

1.3.1.6 Operations

The operations provided by OpenFlexure Microscope Software are just one type of an operation:

User-initiated Operations:

- Delete images from gallery
- Save current device connection
- View live stream
- Export images from gallery
- Create zip of images
- View images from gallery
- Set storage path
- Capture image to gallery

Periods of interactive operations and periods of unattended operations:

[3] https://flask.palletsprojects.com/en/1.1.x/

- Scan images to gallery
- Move camera to a given coordinate
- Discover and Connect to a device
- Click-to-move
- Focus camera

Data processing support functions:

- Filter captures

Backup and recovery operations:

- Refresh captures

All details of these operations will be covered deeply in the 3.2 Functions section.

1.3.2 System Functions

Function	**Summary**
Discover and connect to a device	User launches the app, OpenFlexure Software System(OpenFlexure Connect) asks for IP address of the RaspberryPi on the microscope or waits for a nearby device to be chosen. After selection, OpenFlexure Software System is responsible for connecting provided IP by user.
Save current device's connection	Application lets user saves current IP address and port for future usage
Set Storage Path	Application lets user input a path as a string. Application saves given string as a path for storing further data.
Refresh Captures	Application requests most updated version of images in the gallery and displays it.
Create Zip of Captures	Application lets user select number of images and creating a zip file with them.
Filter Captures	Application let user filter captured images in the gallery using tags and annotations(metadata).
View live stream	OpenFlexure Software Application displays the live feed from microscope on the user interface.
View images in the gallery	Application lists all images that has been captured or scanned. If user chooses an image, OpenFlexure Software Application displays it in the full resolution. If user right clicks on the image, OpenFlexure Software Application displays metadata of the image.
Delete images from the gallery	Application lets user delete an image from gallery
Capture images to the gallery	Application lets user to capture an image from live stream
Scan images to the gallery	Application lets user scan images to the gallery
Move camera to a coordinate	Application lets user to migrate the camera to a point in 3D coordinate space, after configuration of x,y and z coordinates, camera moves to given point.
Click to move	Lets a user to migrate the camera to a point in 2D coordinate space,if (s)he clicks on the point from camera screen in the user interface

Table 1: System Functions

1.3.3 User characteristics

There are two types of users; Microscopists and Developers. Microscopists are end-users who are expected to have prior knowledge at properly using of microscopes. Developers are supposed to be good at programming with Python or Matlab. They shall develop patches as an extensions to OpenFlexture Microscope.

1.3.4 Limitations

a. Regulatory policies: The OpenFlexure Microscope is an open source hardware and software project. Therefore the project files and codes are accessible for everyone. [?] The OpenFlexure Microscope project uses GNU General Public License version 3.

b. Hardware limitations: The OpenFlexure Microscope captures images with camera, stores them on the SD card and shares them over network. Therefore, hardware devices shall send and receive data from the system without any delay. For microscopists, a computer in which OpenFlexure Connect app is installed in is enough. In addition to that stable network connection is required for RaspberryPi.

c. Interfaces to other applications: The OpenFlexure Microscope system shall be compatible with web browsers, physical hardware and operating systems.

d. Parallel operation: Parallelization takes an important role in OpenFlexure Microscope project because not only multiple users can connect to single microscope but also a single user can connect to multiple microscopes for scanning functionality. Therefore multiple OpenFlexure Microscopes must be able to function concurrently.

e. Audit Functions: OpenFlexure Microscope project does not have any audit functions.

f. Control Functions: OpenFlexure Microscope project is controlled not by a center but by a community therefore control functions does not exist.

g. Higher-order language requirements: System shall be coded in modern programming languages that supports web API. In addition to that software stack designed such that writing and integrating extensions especially in Python and MATLAB are encouraged.

h. Signal handshake protocols: HTTP protocol is required for communication between client and server applications. SSH, VPN, mDNS and IP networking protocols are applied for OpenFlexure Microscope remotely.

i. Quality requirements: Stability, maintainability and ease-of-usage are important priorities for OpenFlexure Microscope. The microscope shall be easily installed and used globally. Therefore stable network connection and community support are crucial for this project.

j. Criticality of the application: The OpenFlexure Microscope is not a critical system. It won't have huge effects if it fails.

k. Safety and security considerations: The app is open source therefore anyone who knows how to read code can detect vulnerabilities however it does not harm anybody since every connection is establishments are not centralized.

k. Physical/Mental considerations: Screen reader extensions for web browsers can help Physically/Mentally disabled people.

1.4 Definitions

- SSH, secure shell protocol
- VPN, virtual private networking
- mDNS, multicast Domain Name System
- W3C, word wide web consorsium
- WoT, Web of Thighs
- IP, internet protocol
- HTML, Hyper Text Markup Language
- API, Application Programming Interface

Chapter 2

References

[1] Joel T. Collins, Joe Knapper, Julian Stirling, Samuel McDermott, and Richard Bowman. Modern microscopy with the web of things: The openflexure microscope software stack, 2021.

Chapter 3

Specific requirements

3.1 External interfaces

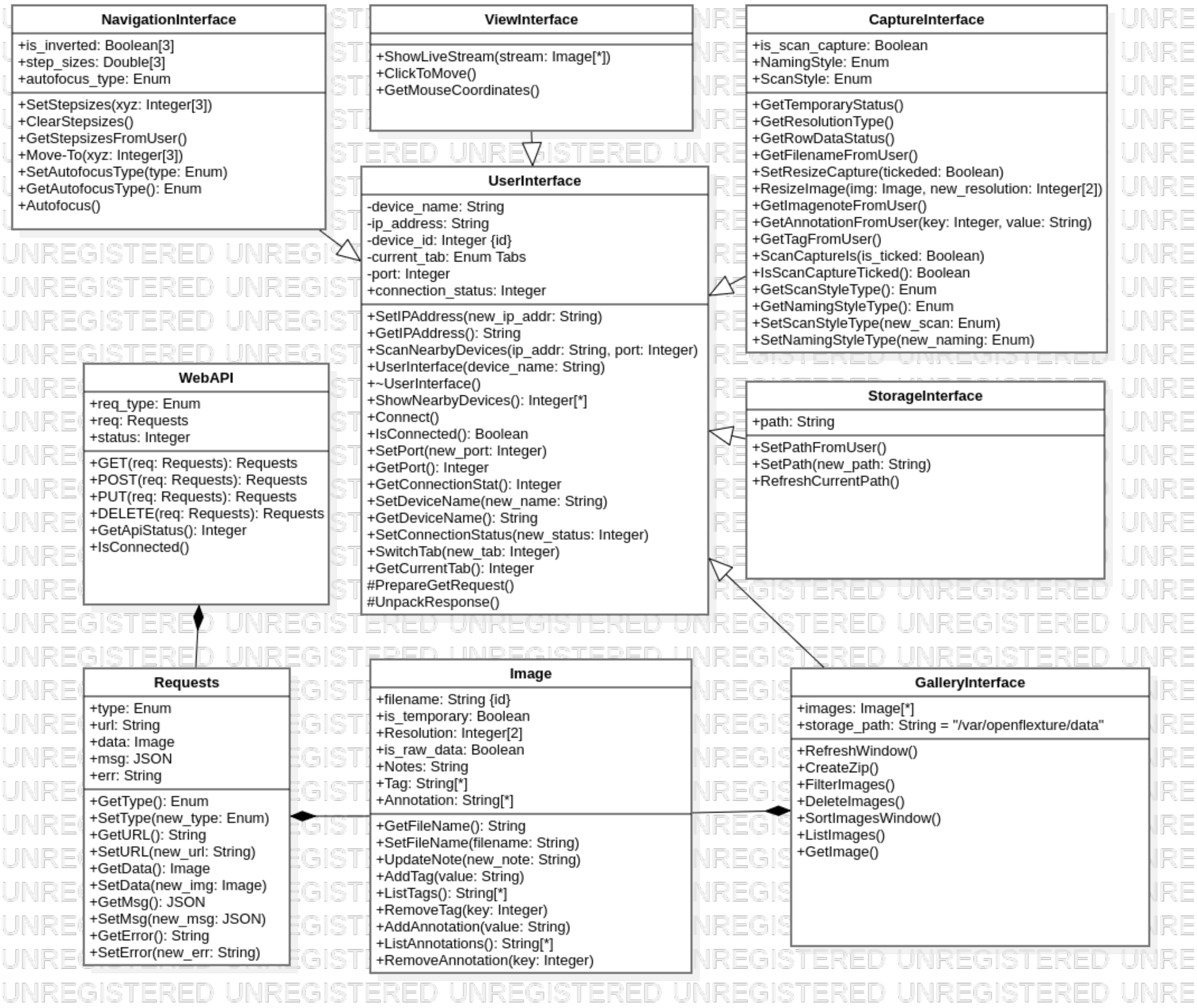


Figure 4: Use Case Diagram

3.2 Functions

3.2.1 Product Perspective

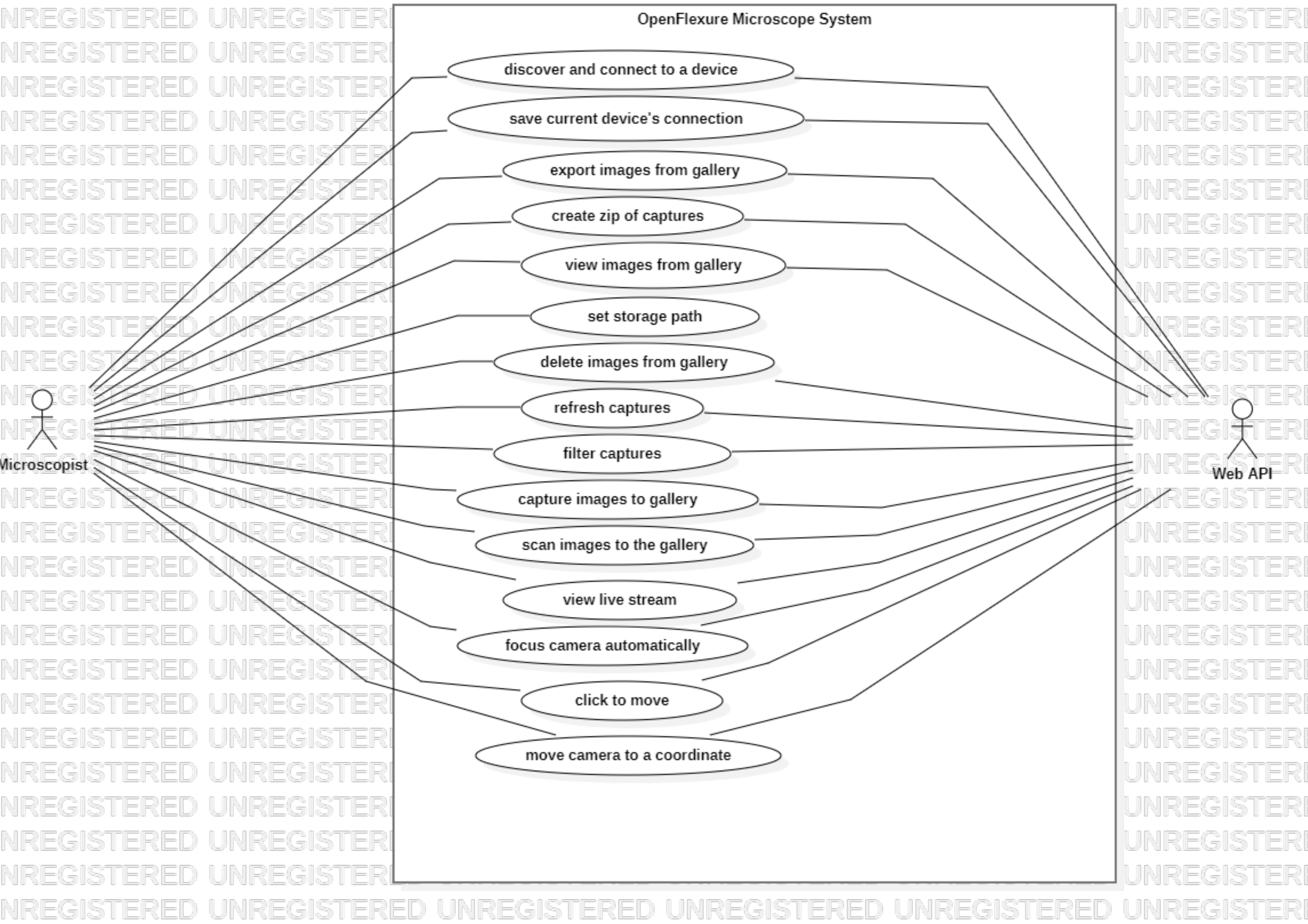

OpenFlexure Microscope System
discover and connect to a device
save current device's connection
export images from gallery
create zip of captures
view images from gallery
set storage path
delete images from gallery
refresh captures
filter captures
capture images to gallery
scan images to the gallery
view live stream
focus camera automatically
click to move
move camera to a coordinate
Microscopist
Web API
UNREGISTERED

Figure 5: Use Case Diagram

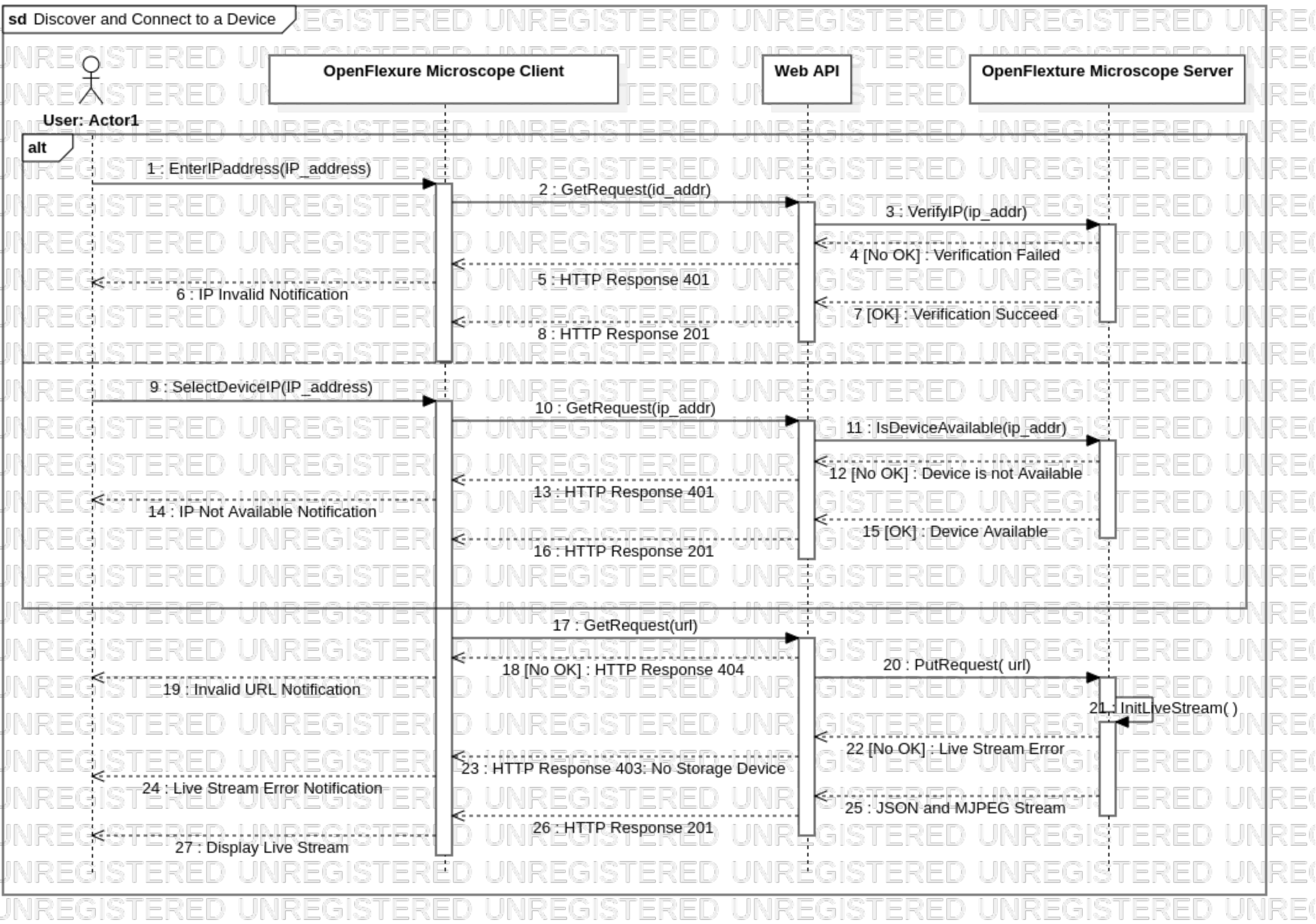

sd Discover and Connect to a Device
User: Actor1
OpenFlexure Microscope Client
Web API
OpenFlexture Microscope Server
alt
1 : EnterIPaddress(IP_address)
2 : GetRequest(id_addr)
3 : VerifyIP(ip_addr)
4 [No OK] : Verification Failed
5 : HTTP Response 401
6 : IP Invalid Notification
7 [OK] : Verification Succeed
8 : HTTP Response 201
9 : SelectDeviceIP(IP_address)
10 : GetRequest(ip_addr)
11 : IsDeviceAvailable(ip_addr)
12 [No OK] : Device is not Available
13 : HTTP Response 401
14 : IP Not Available Notification
15 [OK] : Device Available
16 : HTTP Response 201
17 : GetRequest(url)
18 [No OK] : HTTP Response 404
19 : Invalid URL Notification
20 : PutRequest(url)
21 : InitLiveStream( )
22 [No OK] : Live Stream Error
23 : HTTP Response 403: No Storage Device
24 : Live Stream Error Notification
25 : JSON and MJPEG Stream
26 : HTTP Response 201
27 : Display Live Stream

Figure 6: Discover and Connect to a Device Sequence Diagram

Use-Case Name	Discover and connect to a device
Actors	Microscopist
Description	When a user launches the app, OpenFlexure Software System(OpenFlexure Connect) will asks for IP address of the RaspberryPi on the microscope. OpenFlexure Software System is responsible for connecting provided IP by user.
Data	IP address of the RaspberryPi on the microscope, local IP address of RaspberryPi in the network
Preconditions	User must either connect RaspberryPi to the network OpenFlexure Connect uses or know the IP address of the RaspberryPi device. RaspberryPi on the network must been flashed by RaspbianOpenFlexure OS. RaspberryPi must be opened.
Stimulus	User clicks on the connect button.
Basic Flow	Step 1 – A user opens up the OpenFlexure Connect Application Step 2 – User enters IP address of the RaspberryPi Step 3 – User clicks connect button Step 4 – OpenFlexure Connect establishes connection to the host machine. Step 5 – OpenFlexure Connect opens up the main menu of the application.
Alternative Flow#1	Step 2 – User chooses a local microscope from the list of scanned devices over network Step 3 – User clicks on the connect button Step 4 – OpenFlexure Connect establishes connection to the host machine. Step 5 – OpenFlexure Connect opens up the main menu of the application.
Alternative Flow#2	–
Exception Flow	Step 3 – User enters an invalid IP address or there is no available device on the local network Step 4 – OpenFlexure Connect is not able to establish a valid connection to the host. Step 5 – OpenFlexure Connect shows an error and waits for a valid IP address.
Post Conditions	OpenFlexure redirects user to the main user interface of the app. The server starts a background thread that records MJPEG frames from the camera into a buffer.

Table 2: Discover and connect to a device Function

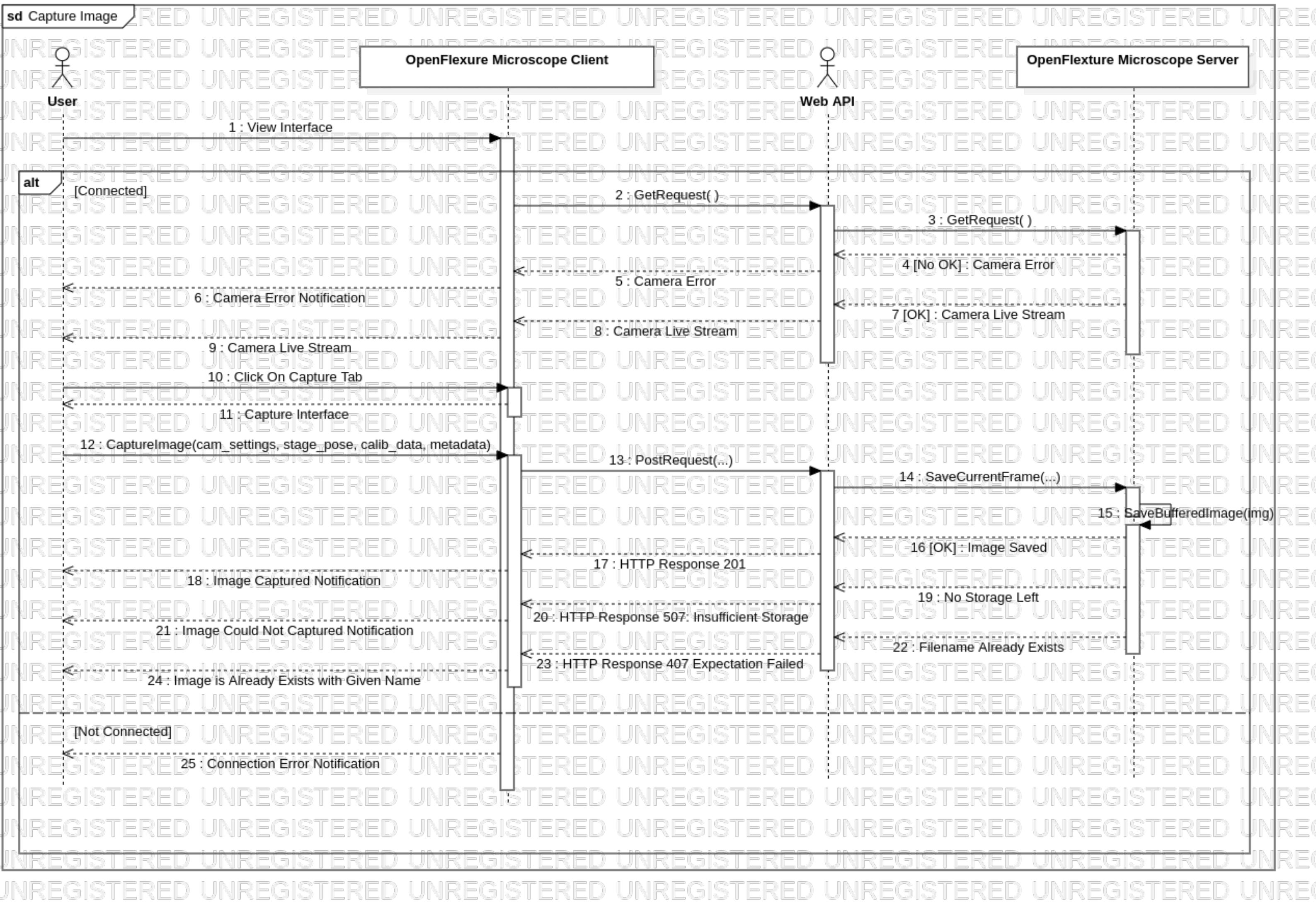

sd Capture Image
User
OpenFlexure Microscope Client
Web API
OpenFlextures Microscope Server
1 : View Interface
alt
[Connected]
2 : GetRequest( )
3 : GetRequest( )
4 [No OK] : Camera Error
5 : Camera Error
6 : Camera Error Notification
7 [OK] : Camera Live Stream
8 : Camera Live Stream
9 : Camera Live Stream
10 : Click On Capture Tab
11 : Capture Interface
12 : CaptureImage(cam_settings, stage_pose, calib_data, metadata)
13 : PostRequest(...)
14 : SaveCurrentFrame(...)
15 : SaveBufferedImage(img)
16 [OK] : Image Saved
17 : HTTP Response 201
18 : Image Captured Notification
19 : No Storage Left
20 : HTTP Response 507: Insufficient Storage
21 : Image Could Not Captured Notification
22 : Filename Already Exists
23 : HTTP Response 407 Expectation Failed
24 : Image is Already Exists with Given Name
[Not Connected]
25 : Connection Error Notification

Figure 7: Capture Image Sequence Diagram

Use-Case Name	Save current device's connection
Actors	Microscopist
Description	In the OpenFlexure Connect application, user writes down an IP address and port or selects from nearby microscopes, saves it for future usage
Data	IP address, Port
Preconditions	–
Stimulus	User clicks on the Save Current button.
Basic Flow	Step 1 – User opens up the OpenFlexure Connect Application Step 2 – User enters IP address of the RaspberryPi Step 3 – User clicks save current button
Alternative Flow#1	Step 2 – User selects a microscope from nearby devices Step 3 – User clicks save current button
Alternative Flow#2	–
Exception Flow	Step 3 – User enters an invalid IP address or port. There is no available device on the local network Step 4 – OpenFlexure Connect shows an error and waits for a IP/port input.
Post Conditions	Microscope appears in the Saved Devices menu

Table 3: Saving Current Device Function

Use-Case Name	Set Storage Path
Actors	Microscopist
Description	User can set storage path for saved images
Data	Path Variable
Preconditions	–
Stimulus	User clicks on the set current path button.
Basic Flow	Step 1 – User opens up the OpenFlexure Connect Application Step 2 – User clicks on the storage interface Step 3 – User enters a Path Step 4 – User clicks on the Set Path button
Alternative Flow#1	Step 3 – User clicks on the auto-choose button Step 4 – User clicks on the Set Path button
Alternative Flow#2	–
Exception Flow	Step 3 – OpenFlexure Connect shows an error about invalid path variable
Post Conditions	Capture storage path changes

Table 4: Set Storage Path Function

Use-Case Name	Refresh Captures
Actors	Microscopist
Description	User refreshes and updates the page of saved images.
Data	–
Preconditions	–
Stimulus	User clicks on the refresh button
Basic Flow	Step 1 – User opens up the OpenFlexure Connect Application Step 2 – User clicks on the gallery interface Step 3 – User clicks on refresh button
Alternative Flow#1	–
Alternative Flow#2	–
Exception Flow	–
Post Conditions	Gallery page updated

Table 5: Refresh Captures Function

Use-Case Name	Create Zip of Captures
Actors	Microscopist
Description	User selects and creates a zip of images from gallery interface.
Data	Captured Images
Preconditions	–
Stimulus	User clicks on the Create Zip button
Basic Flow	Step 1 – User opens up the OpenFlexure Connect Application Step 2 – User clicks on the gallery interface Step 3 – User selects images Step 4 – User clicks on create zip button
Alternative Flow#1	–
Alternative Flow#2	–
Exception Flow	–
Post Conditions	A zip of selected images has been created

Table 6: Create Zip of Captures Function

Use-Case Name	Filter Captures
Actors	Microscopist
Description	User filters captured images from gallery interface.
Data	Captured Images
Preconditions	–
Stimulus	User clicks on the filter button
Basic Flow	Step 1 – User opens up the OpenFlexure Connect Application Step 2 – User clicks on the gallery interface Step 3 – User chooses filter
Alternative Flow#1	–
Alternative Flow#2	–
Exception Flow	–
Post Conditions	Captured images shown using desired filters

Table 7: Filter Captures Function

Use-Case Name	View live stream
Actors	Microscopist, Web API
Description	When a user switches to the view interface, OpenFlexure Software Application sends an HTTP request for live stream data from the server. Server responses with the buffered frames to the OpenFlexure Software Application. OpenFlexure Software Application displays the response data on the user interface.
Data	Buffered image frames from Raspberry-Pi camera
Preconditions	OpenFlexure Connect must be already connected to the host server.
Stimulus	OpenFlexure Software Application displays the live stream on the graphical user interface.
Basic Flow	Step 1 – User opens up the OpenFlexure Software Application Step 2 – User clicks on the view interface
Alternative Flow#1	–
Alternative Flow#2	–
Exception Flow	Step 2 – OpenFlexure Software Application shows a message about camera connection error.
Post Conditions	Live stream stays at the background of the application.

Table 8: View Live Stream Function

Use-Case Name	Export images from gallery
Actors	Microscopist
Description	If a user wants to download image, switches to the gallery interface. OpenFlexure Application lists all images, user chooses stack of images or a single image then saves to the client machine.
Data	Saved or scanned images on server
Preconditions	OpenFlexure Connect must be already connected to the host server. There must be enough disk space in the local system. There must be at least one image in the gallery.
Stimulus	OpenFlexure Connect clones desired image/images from the server to the client machine.
Basic Flow	Step 1 – User switches to the gallery interface. Step 2 – User selects an image Step 3 – User right clicks on the selected image Step 4 – User clicks on the download as image option
Alternative Flow#1	Step 2 – User selects an image Step 3 – User right clicks on the selected image Step 4 – User clicks on the download as a different format option
Alternative Flow#2	Step 2 – User selects a stack of images Step 3 – User right clicks on the selected stack Step 4 – User clicks on the download option
Exception Flow	Step 2 – OpenFlexure Application shows a message due to connection error.
Post Conditions	Image/images are saved on the selected path on the client computer.

Table 9: Export Images From Gallery Function

Use-Case Name	View images in the gallery
Actors	Microscopist
Description	If a user wants to view an image, switches to the gallery interface. OpenFlexure Software Application lists all images that has been captured or scanned. If user chooses an image, OpenFlexure Software Application displays it in the full resolution. If user right clicks on the image, OpenFlexure Software Application displays metadata of the image.
Data	Saved or scanned images on server
Preconditions	OpenFlexure Connect must be already connected to the host server.
Stimulus	OpenFlexure Software Application displays an image in the graphical user interface.
Basic Flow	Step 1 – User switches to the gallery tab. Step 2 – User selects an image
Alternative Flow#1	Step 2 – User selects a stack of images Step 3 – User chooses one of stacked images
Alternative Flow#2	–
Exception Flow	–
Post Conditions	Gallery images are viewed by the user

Table 10: View Images In the Gallery Function

Use-Case Name	Delete images from the gallery
Actors	Microscopist
Description	If a user wants to delete an image, switches to the gallery interface. Selects image and deletes it from the server
Data	Saved or scanned images on server
Preconditions	OpenFlexure Connect must be already connected to the host server.There must be at least one image in the gallery.
Stimulus	User clicks on the delete button
Basic Flow	Step 1 – User switches to the gallery interface. Step 2 – User selects an image Step 3 – User clicks on the delete button
Alternative Flow#1	Step 2 – User selects a stack of images Step 3 – User clicks on the delete button
Alternative Flow#2	Step 2 – User selects a stack of images Step 3 – User selects an image Step 4 – User clicks on the delete button
Exception Flow	OpenFlexure Connect shows a message due to connection error.
Post Conditions	Gallery images are deleted by the user

Table 11: Delete Image Function

Use-Case Name	Capture images to the gallery
Actors	Microscopist, Web API
Description	If a user wants to capture an image, clicks on the capture button to save a frame to the gallery.
Data	Buffered image frames from Raspberry-Pi camera system
Preconditions	OpenFlexure Software Application must be already connected to the host server. Displaying live stream must be triggered before.
Stimulus	User clicks on the capture button
Basic Flow	Step 1 – User switches to the capture interface. Step 2 – User clicks capture image
Alternative Flow#1	Step 2 – User enters a filename Step 3 – User resizes frame using custom x-y inputs Step 4 – User takes notes about the frame Step 5 – User adds annotation and tags Step 6 – User clicks on the capture button
Alternative Flow#2	Step 2 – User writes a filename Step 3 – User ticks ‘Store raw data’ option Step 4 – User chooses scan option Step 5 – User clicks on the capture button
Exception Flow	OpenFlexure Software Application shows an error if there exists an image with the same filename.
Post Conditions	Frames are captured, preprocessed and saved to the gallery

Table 12: Capture Images the Gallery Function

Use-Case Name	Scan images to the gallery
Actors	Microscopist,Web API
Description	If a user wants to scan over an area, configures options and clicks on the scan button to save stack of frames to the gallery.
Data	Buffered image frames from Raspberry-Pi camera
Preconditions	OpenFlexure Connect must be already connected to the host server. Displaying live stream must be triggered before.
Stimulus	User clicks on the scan button
Basic Flow	Step 1 – User switches to the capture interface. Step 2 – User configures step-sizes, steps, auto-focus type and scan style Step 3 – User clicks on the start scan button
Alternative Flow#1	Step 2 – User enters a Filename Step 3 – User resizes frames using custom x-y inputs Step 4 – User takes notes about the frames Step 5 – User adds annotations and tags Step 6 – User configures step-sizes, steps, auto-focus type and scan style Step 7 – User clicks on the start scan button
Alternative Flow#2	–
Exception Flow	
Post Conditions	Area is scanned, preprocessed and saved to the gallery

Table 13: Scan Images to the Gallery Function

Use-Case Name	Move camera to a coordinate
Actors	Microscopist, Web API
Description	If a user wants to migrate the camera to a point in 3D coordinate space, (s)he configures x,y and z coordinates and clicks on the move button
Data	3x1 integer vector from client to server
Preconditions	OpenFlexure Connect must be already connected to the host server. Displaying live stream must be triggered before.
Stimulus	
Basic Flow	Step 1 – User clicks the navigation interface Step 2 – User fills in the blanks for x, y, and z coordinates Step 3 – User clicks on the move button
Alternative Flow#1	Step 1 – User clicks the navigation interface Step 2 – User clicks on the increment or the decrement button Step 3 – User clicks on the move button
Alternative Flow#2	–
Exception Flow	–
Post Conditions	Camera is relocated on the desired coordinates

Table 14: Move Camera To a Coordinate Function

Use-Case Name	Focus camera automatically
Actors	Microscopists, Web API
Description	If a user wants the camera to maximize its resolution/sharpness automatically, (s)he clicks on the auto button and the camera zooms in or out in the z axis
Data	Image frame from Raspberry-Pi camera
Preconditions	OpenFlexure Connect must be already connected to the host server. Displaying live stream must be triggered before.
Stimulus	
Basic Flow	Step 1 – User clicks the navigation interface Step 2 – User clicks on the auto button
Alternative Flow#1	–
Exception Flow	–
Post Conditions	Camera is focused

Table 15: Focus Camera Automatically Function

Use-Case Name	Click to move
Actors	Microscopist, Web API
Description	If a user wants to migrate the camera to a point in 2D coordinate space, (s)he clicks on the point from camera screen in the user interface
Data	Live stream of camera frames from server to client
Preconditions	OpenFlexure Connect must be already connected to the host server. Displaying live stream must be triggered before. Also, the stage must be calibrated.
Stimulus	
Basic Flow	Step 1 – User clicks the navigation interface Step 2 – User clicks on the camera screen in the user interface Step 3 – Camera moves in the x-y coordinate plane
Alternative Flow#1	–
Exception Flow	–
Post Conditions	Camera is relocated on the selected pointed

Table 16: Click to Move Function

3.3 Usability Requirements

1. A user shall use the system once an network connection is stably available.
2. A user shall connect to multiple microscopes concurrently.
3. A user shall install and download project without heavy tech-field knowledge.
4. A user shall script experiment without having to reimplement the hardware control code.
5. A user shall abort scanning procedure without losing or corrupting images that are already acquired.

3.4 Performance Requirements

- 2 GB of the server's memory shall be kept empty so that the system works properly.
- Microscope shall operate normally after an event of disconnection between client and server.
- Microscope shall be connected by multiple users.
- Microscope shall be operated by multiple users.

- Microscope shall be available on the network not only for internet but also local network.
- Captured images shall appear in gallery interface in less than 2 seconds.
- Microscope shall be open for API requests while on an operation. Operations shall not block API requests.
- Microscope camera shall not be out of focus less than 5 seconds during scanning.
- Sending server a GET request shall take 17 ms, a PUT request shall take 22 ms on average.
- Basic auto focus operation shall take 10 to 20 seconds.
- Fast auto focus operation shall take 5 seconds.
- Client user interface shall continue interacting with users while another action is running.

3.5 Logical Database Requirements

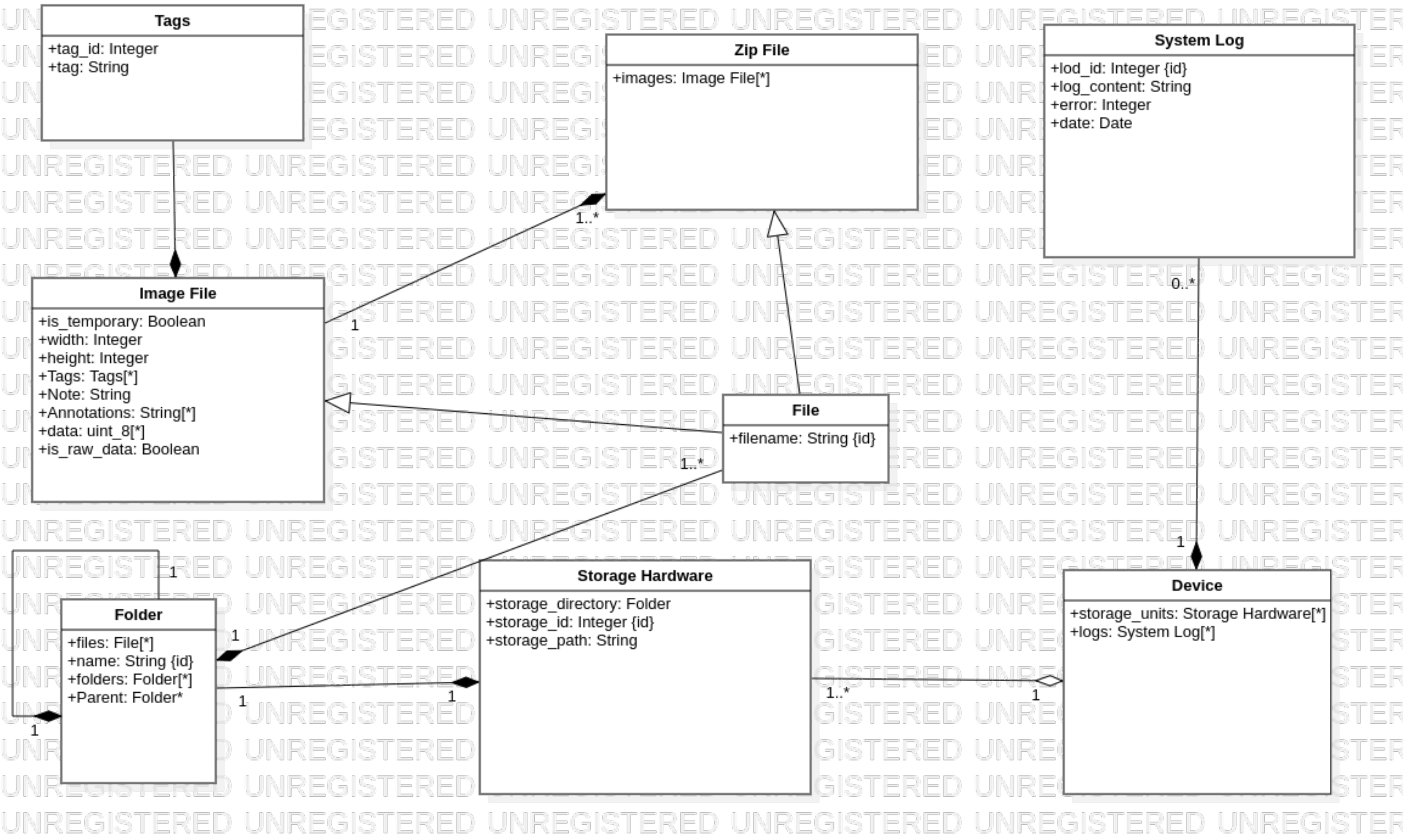


Figure 8: Logical Database Class Diagram

- Database shall be initialised after OpenFlexure Microscope is set up.
- A device shall be created after user saves a device.

- An image is created under certain folder every time a capture is taken.
- During the scan operation every captured frame is stored in that second in the database so that stopping scan does not effect already taken frames.
- OpenFlexure Microscope takes log history about established connections to the server every time a new connection is initialised.
- Database is accessible via server command line interface.

3.6 Design Constraints

System concerns to serve users in a cheap and free way therefore open source hardware designs and open source software development methods are chosen and licensed under GNU General Public License, version 3.

3.7 Software System Attributes

a. Reliability:

- Data loss in camera data shall be less than 0.05.
- Failure of the system's hardware components shall be less than 5 minutes in a month.
- In case of disconnection for more than 5 seconds server side continues operating normally until a connection is established.

b. Availability:

- In case of a restart, a Raspberry Pi takes 2 minutes to be fully available.
- Latest stable version of the system will be always available for users however it doesn't update automatically.
- System stores images in local storage therefore in case of a system corruption, images and zip files are not lost and recovered after restart.

c. Security:

- System shall not store any user information.
- OpenFlexure Microscope can be controlled by anyone who knows the IP address of the Raspberry Pi.
- OpenFlexure Microscope system uses HTTP for communication and shall not be vulnerable against attacks coming over network.
- Server keeps log of the OpenFlexure Microscope system.

d. Maintainability:

- System designed such that it is capable of integration of new cameras and writing extensions.
- OpenFlexure Project is supported and maintained by community in Gitlab platform.
- Documentation of the system shall be found on project website.

e. Portability:

- Web application shall be runnable on every device that runs web browsers.

- OpenFlexure Connect shall support Windows and Linux operating systems.
- Programming language which is chosen for the development of the OpenFlexure Microscope system shall not depends on OS.
- Libraries that used in development shall be available for various programming languages.

3.8 Supporting Information

- The OpenFlexure Microscope project aims to provide an accessible solution for identifying and quantifying Malaria in regions with restricted access to conventional microscopes.
- Mac users can build software side of the system from source code as addressed in Gitlab.

Chapter 4

Verification

Appendix A

Appendices

A.1 Assumptions and dependencies

A.2 Acronyms and abbreviations