Software Design Description

OpenFlexure Microscope

Onat ÖZDEMİR - 2310399

Contents

List of Figures

List of Tables

Revision History

Date	Reason For Changes	Version
28.05.2021	Initial Draft	0.1
12.06.2021	Final Version	1.0

Table 1: Revision History of Software Design Description Document

1 Introduction

This document is the Software Design Description for the OpenFlexure Microscope project introduced by the group of researchers from the University of Bath. The document is written with respect to the specifications of the [1].

1.1 Purpose of the System

The purpose of this project is to design a customizable, low-cost and locally-manufacturable [2] microscope that can be easily built by people from all over the world. Unlike complex and expensive microscopes, OpenFlexure Microscope can be built using 3d machines in low-cost and even an inexperienced person can easily assemble and use it.

1.2 Scope

- System will have a server that runs an internally stored OpenFlexure Web API. Connected users will be able to send requests using this Web API.
- Users will be able to connect to the system both locally and remotely.
- System will have interface for users. Users will be able to navigate microscope stage, view camera livestream, capture images and record videos using this interface. They will also be able to manage and modify the saved media.
- System will have interface for system admins. System admins will be able to manage system settings. Additionally, they will be able to view connected users and manage these connections using this interface.
- System will have interface for maintainers. Maintainers will be able to view detailed system logs which contain description, time and type information. Additionally, they will be able to check software updates and install these updates to the system using this interface.
- System will have interface for maintainers. Maintainers will be able to view detailed system logs which contain description, time and type

information. Additionally, they will be able to check software updates and install these updates to the system using this interface.

- The system has many information such as connected user IP addresses, admin and maintainer information, saved media and system logs to save to and retrieve from database. To provide secure and stable connection, the system will also have Database Management Interface.

- Since microscopes are error-prone systems, to hande these errors system will have Back-end Failure Management Interface. When an error occurs, the system will save the detailed log of this error in the protected area of the database and maintainers will be able view these system failure logs.

However,

- Designing the hardware components of the OpenFlexure Microscope

- Designing and implementing the OpenFlexure Connect desktop application

- Designing and implementing the OpenFlexure Web API

- Designing and implementing the OpenFlexure Update System and Update API

are out of the scope of this project.

1.3 Stakeholders and Their Concerns

There are several different stakeholders within the system.

1. Users
Users (can be seen as end-users) are the people who use the microscope for their daily tasks. The main concerns of the users are the easy usability of the microscope, the privacy of their IP addresses, and using the microscope without any technical issues. They want a simple user interface, secure database and to be able to connect to the system both remotely and locally.

2. Admins
Admins are authorized people who are responsible for the management of the microscope. Their main concerns are managing the system settings to customize microscope preferences to provide a better and controlled user experience. They also want to see details of the connected users and be able to block/unblock users to maintain the security of the microscope. Finally, since they use admin keys to login to the system, they want these passwords to be kept securely.

3. Maintainers
Maintainers are authorized people who are responsible for keeping system software up to date and solving the technical and software-related issues in the system. They want to see the latest software releases, be able to update system software and view detailed system logs related to the problems that occured in the system. Finally, since they use maintainer keys to login to the system, they want these passwords to be kept securely.

2 References

[1] "IEEE Standard for Information Technology–Systems Design–Software Design Descriptions," in IEEE STD 1016-2009 , vol., no., pp.1-35, 20 July 2009, doi: 10.1109/IEEESTD.2009.5167255.

[2] https://openflexure.gitlab.io/microscope-handbook/learn-about-the-project/local-manufacturing.html

3 Glossary

Admin	An authorized person who manages the microscope
API	An application programming interface (API) is an interface that defines interactions between multiple software applications
Command-line Interface	A command-line interface (CLI) processes commands to a computer program in the form of lines of text
GUI	The graphical user interface is a form of user interface that allows users to interact with electronic devices through graphical icons and audio indicator
Maintainer	An authorized person who is responsible for keeping system software up to date and solving the technical and software-related issues in the system.
Metadata	Higher-level data that describes or annotates another data
OpenFlexure Update System	A system responsible for serving the OpenFlexure software updates to users
SSH	The Secure Shell Protocol is a cryptographic network protocol for operating network services securely over an unsecured network
User	An unauthorized person who connects to the microscope for daily tasks

Table 2: Glossary

4 Architectural Views

4.1 Context View

4.1.1 Context Diagram

OpenFlexure is not a part of a larger system. The communication between the OpenFlexure and Users is maintained by the OpenFlexure Web API. Users will be able to send their requests and get responses from the server using the Web API.

There are three user types in the system: Users (can be seen as end-user), Admins, and Maintainers. The system will provide different interfaces to each of them so that they can do the functions defined to them. Since Admins and Maintainers have private permissions, they will need to log in to the system using the keys defined to them to use their functions. While Admins and Maintainers will be responsible for keeping the system stable, Users will be able to use the microscope.

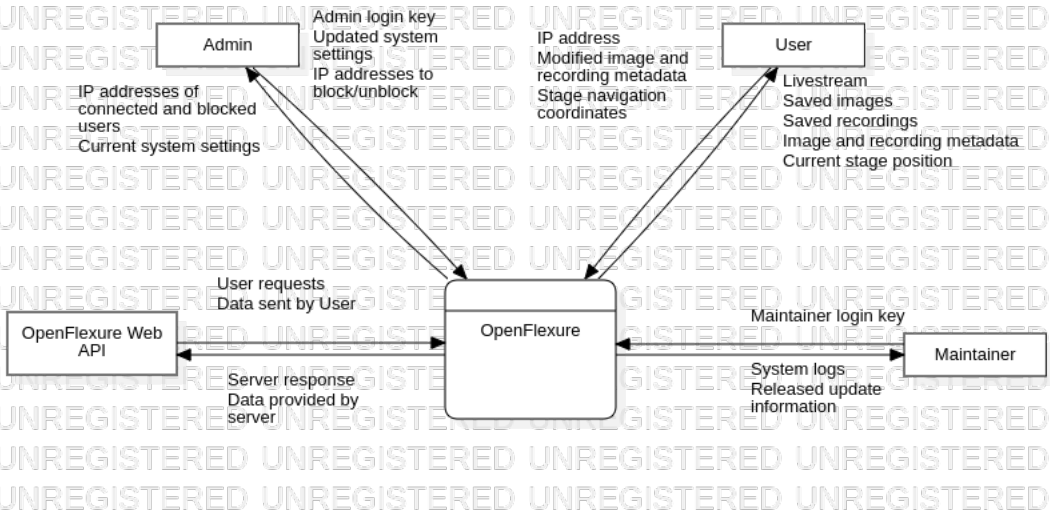

Figure 1: System Context Diagram for OpenFlexure

4.1.2 Use-Case Diagram

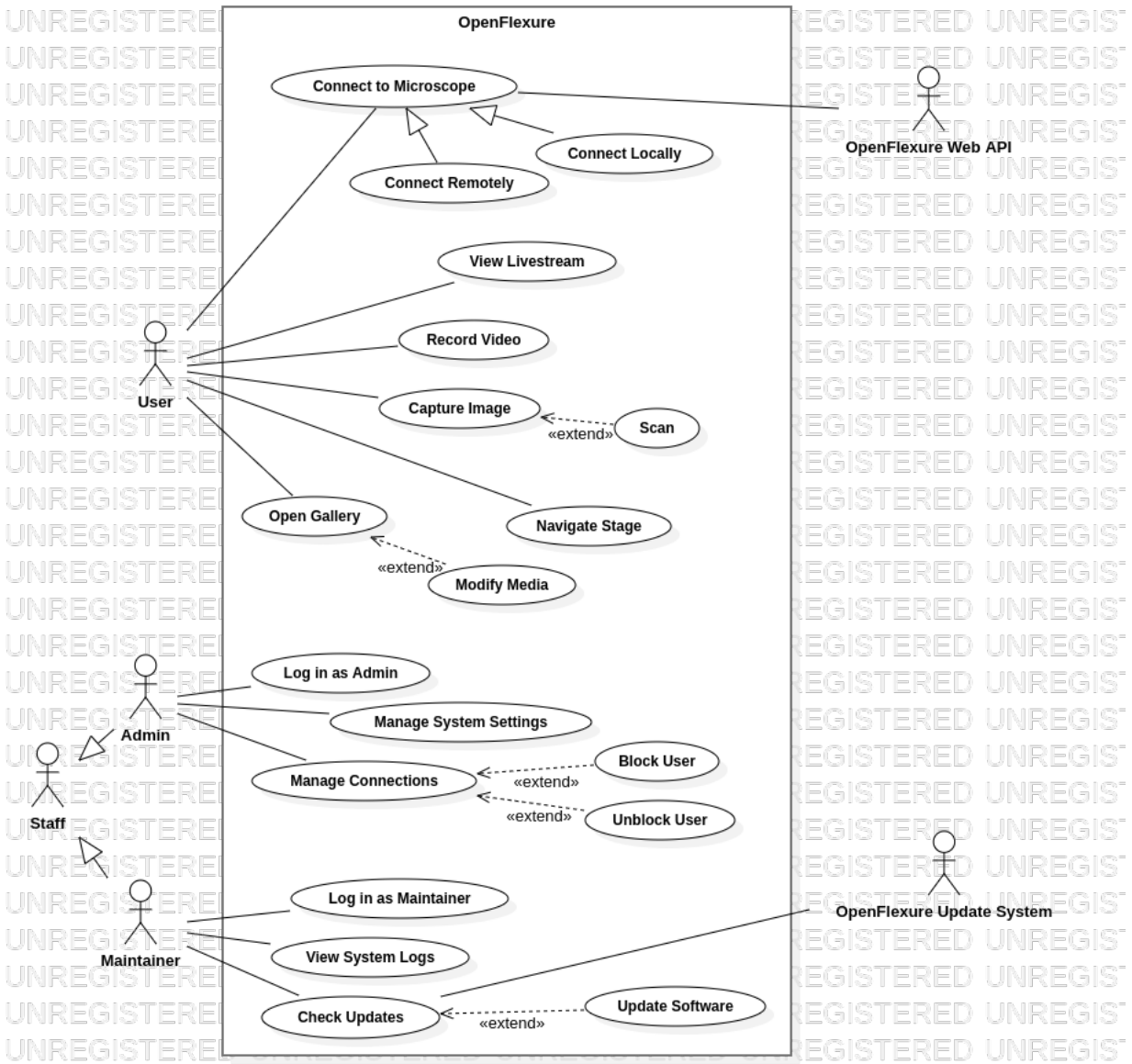

Figure 2: Use Case Diagram for OpenFlexure

4.1.3 Use-Case Descriptions

Use-Case Name	Connect to Microscope
Actors	User
Description	The user connects to the microscope's server.
Data	IP address of the user's computer
Preconditions	Desktop application must be running.
Stimulus	The user clicks on "Connect to server" button placed in the home screen of the desktop application.
Basic Flow	Step 1 – The popup window contains the connection screen is opened. Step 2 – The user chooses "Connect remotely" option from the option boxes. Step 3 – Desktop application searches for the nearby microscope servers and lists the found ones. Step 4 – The user selects one of the microscopes listed in. Step 5 – The user clicks on "Connect" button.
Alternative Flow#1	Step 2 – The user chooses "Connect locally" option. Step 3 – The desktop application searches for the web api internally stored in the Raspberry Pi, and when it finds, "Connect" button becomes available. Step 4 – The user clicks on "Connect" button.
Alternative Flow#2	-
Exception Flow	If desktop application cannot found any nearby servers or the web api, the system displays an error message saying that "No device were found". If the IP address of the user is blocked, the system displays an error message saying that "Your IP address is blocked".
Post Conditions	The user connects to the microscope.

Table 3: Connect to Microscope

Use-Case Name	View Livestream
Actors	User
Description	The user views the live camera feed.
Data	Camera feed
Preconditions	User must be connected to the system. Camera connection must be provided.
Stimulus	The user clicks on "View" button.
Basic Flow	Step 1 – The user clicks on "View" button from the side pine. Step 2 – Server reaches Raspberry Pi camera. Step 3 – Camera livestream is displayed in graphical user interface.
Alternative Flow#1	-
Alternative Flow#2	-
Exception Flow	If the camera connection is not established, the system displays an error message saying that "Check the camera connection".
Post Conditions	The livestream is displayed.

Table 4: View Livestream

Use-Case Name	Record Video
Actors	User
Description	The user starts recording the livestream.
Data	Recorded video
Preconditions	The user must be connected to the system. Camera connection must be provided.
Stimulus	The user clicks on "Record" button.
Basic Flow	Step 1 – The user clicks on "Record" button from the side pane. Step 2 – The tab that contains live camera feed and menu for recording settings opens. Step 3 – When user wants to start recording the livestream, he/she clicks on "Start Recording" button placed in the menu. Step 4 – The system sends "record" command to the "Camera System". Step 5 – The system displays a blinking led to indicate the recording. Step 6 – The user clicks on "Stop Recording" button to stop the recording. Step 7 – The system receives the recording from the "Camera System" and displays it in a popup window. Step 8 – The user gives a name to the recording by filling the textbox placed in the popup window and specifies the time interval if he/she wants to cut the recording before saving it. Step 9 – The user clicks on "Save" button to save the recording. Step 10 – The system saves the video to the location specified in system settings and create database record for the video using the Database Management Interface.
Alternative Flow#1	Step 8 – The user clicks on the "Exit" button placed in the popup window to exit without saving the recording.
Alternative Flow#2	-
Exception Flow	If the user tries to save the recording without naming it, the system displays an error message. If there is not enough available storage to store the recording, the system displays an error message.
Post Conditions	The recording is saved to the location specified in the settings and the popup window is closed.

Table 5: Record Video

Use-Case Name	Capture Image
Actors	User
Description	The user captures the image on the screen.
Data	Captured image
Preconditions	The user must be connected to the system. Camera connection must be provided.
Stimulus	The user clicks on "Capture" button.
Basic Flow	Step 1 – The user clicks on "Capture" button from the side pane. Step 2 – The tab that contains live camera feed and menu for image settings opens. Step 3 – When user wants to capture the image on the screen, he/she clicks on "Capture image" button placed in the menu. Step 4 – The system sends "capture" command to the "Camera System". Step 5 – The system receives the captured image from the "Camera System" and displays it in a popup window. Step 6 – The user gives a name to the captured image by filling the textbox placed in the popup window. Step 7 – The user clicks on "Save" button to save the captured image. Step 8 – The system saves the image to the location specified in system settings and create database record for the image using the Database Management Interface.
Alternative Flow#1	Step 6 – The user clicks on the "Exit" button to exit without saving the captured image.
Alternative Flow#2	-
Exception Flow	If the user tries to save the image without naming it, the system displays an error message. If there is not available storage to store the image, the system displays an error message.
Post Conditions	The image is saved to the location specified in the system settings and the popup window is closed.

Table 6: Capture Image

Use-Case Name	Scan
Actors	User
Description	The system generates a scan of the area bounded by the coordinates provided by the user.
Data	Generated scan
Preconditions	The user must be connected to the system. Camera connection must be provided. Capture tab must be opened.
Stimulus	The user clicks on "Scan" button.
Basic Flow	Step 1 – The user opens "Capture" tab from the side pane. Step 2 – The user activates "Scan" option by clicking the option box placed in image settings menu. Step 3 – The textboxes showing the coordinates that define the boundary of the area to be scanned becomes available. Step 4 – The user types the coordinates to the textboxes. Step 5 – Until the entire area has been scanned, at each step, the system sends a command to the "Motor Control System" to move by the step size specified in system settings and to the "Camera System" to capture the current image. Step 6 – The system merges all captured images to generate a bigger image of the area and displays it in a popup window. Step 7 – The user gives a name to the generated scan. Step 8 – The user clicks on "Save" button to save the generated scan. Step 9 – The system saves the scan image to the location specified in system settings and create database record for the image using the Database Management Interface.
Alternative Flow#1	Step 7 – The user clicks on the "Exit" button to exit without saving the scan.
Alternative Flow#2	-
Exception Flow	If the user tries to save the scan without naming it, the system displays an error message. If there is no available storage, the system displays an error message.
Post Conditions	The scan is saved to the location specified in the system settings.

Table 7: Scan

Use-Case Name	Navigate Stage
Actors	User
Description	The user adjusts the position of the stage.
Data	Coordinates of the stage in x, y and z axes
Preconditions	User must be connected to the system.
Stimulus	The user clicks on "Navigate" button.
Basic Flow	Step 1 – The user clicks on "Navigate" button from the side pane. Step 2 – Tab that contains live camera feed and navigation menu opens. Step 3 – The user chooses the new x, y and z coordinates by filling the three textboxes placed in navigation menu. Step 4 – The user clicks on "Apply" button. Step 5 – The system sends the given coordinates to the "Motor Controller"
Alternative Flow#1	-
Alternative Flow#2	-
Exception Flow	If the motor connection is not established, the system displays an error message saying that "Check the motor connection". If the given x, y, z coordinates outside of the range, the system displays an error message saying that "Coordinates are out of the range".
Post Conditions	The position of the stage is physically changed.

Table 8: Navigate Stage

Use-Case Name	Open Gallery
Actors	User
Description	The user displays the saved images and recorded videos.
Data	Images, recordings and metadata
Preconditions	User must be connected to the system.
Stimulus	The user clicks on "Gallery" button
Basic Flow	Step 1 – The user clicks on "Gallery" button from the side pane. Step 2 – The server reaches to the database and retrieves the media records with the user's IP address in the authorIPAddress field. Step 3 – The system displays a list of images and recordings in a new tab as a grid style gallery where each entry has thumbnail and metadata of the corresponding media.
Alternative Flow#1	-
Alternative Flow#2	-
Exception Flow	If there is no saved media associated with the user, the system displays a message saying that "No saved media were found."
Post Conditions	The user views the media saved by him/her.

Table 9: Open Gallery

Use-Case Name	Modify Media
Actors	User
Description	The user modifies the saved media.
Data	Metadata of the saved images and recordings
Preconditions	User must be connected to the system. Gallery tab must be active
Stimulus	The user clicks on one of the media entries in the gallery.
Basic Flow	Step 1 – The user clicks the media entry he/she wants to modify in the gallery. Step 2 – The popup window contains metadata information of the clicked media opens. Step 3 – The user changes the metadata of the media. Step 4 – The user clicks on "Save" button to save the updated metadata. Step 5 – The system updates the corresponding database records using the Database Management Interface.
Alternative Flow#1	Step 3 – The user clicks on "Delete" button. Step 4 – The media is deleted from the storage location and corresponding database records is deleted from the database.
Alternative Flow#2	-
Exception Flow	-
Post Conditions	The media is either deleted or its metadata is updated.

Table 10: Modify Media

Use-Case Name	Log in as Admin
Actors	Admin
Description	The admin connects to server with his/her login key.
Data	Admin login key
Preconditions	-
Stimulus	The admin sends login request to server using the terminal.
Basic Flow	Step 1 – The server receives the login request. Step 2 – The server asks for the admin login key. Step 3 – Maintainer types his/her login key to the terminal. Step 4 – The system reaches to the local database and verifies the given login key. Step 5 – The system signs in the admin.
Alternative Flow#1	-
Alternative Flow#2	-
Exception Flow	If the given key is invalid, the system displays an error message saying that "Given key is invalid". The system saves the report of the failed login attempt and the IP address of the user to the system logs. Then, the flow starts again from Step 2.
Post Conditions	The admin command line interface opens.

Table 11: Log in as Admin

Use-Case Name	Manage System Settings
Actors	Admin
Description	The admin manages and updates the system settings.
Data	Current settings, updated setting
Preconditions	The admin must be logged in.
Stimulus	The admin gives "manage settings" command using the admin command line interface.
Basic Flow	Step 1 – The system reaches to the local database and retrieve the system settings record. Step 2 – The system displays the current settings as a table where each row consists of the setting name and current value of the setting. Step 3 – The admin gives "set {setting name} {new value}" command to change the value of the specified setting. Step 4 – The system reaches to database and changes the value of the given settings field.
Alternative Flow#1	-
Alternative Flow#2	-
Exception Flow	-
Post Conditions	The system settings are updated.

Table 12: Manage System Settings

Use-Case Name	Manage Connections
Actors	Admin
Description	The admin views the list of connected and blocked users.
Data	IP addresses, types (user, maintainer or admin) and the statuses (connected, blocked) of the users.
Preconditions	The admin must be logged in.
Stimulus	The admin gives "manage connections" command using the admin command-line interface.
Basic Flow	Step 1 – The system reaches to the database and gathers records of the connected and blocked users. Step 2 – The system displays the list of IP addresses, connection times and the statuses (connected, blocked) of the users.
Alternative Flow#1	-
Alternative Flow#2	-
Exception Flow	If there is no connected or blocked user, the system displays an error message saying that "No data were found".
Post Conditions	The admin views the connected and blocked users.

Table 13: Manage Connections

Use-Case Name	Block User
Actors	Admin
Description	The admin blocks a user with given IP address.
Data	IP address of the user
Preconditions	The admin must be logged in. The admin must have given the "manage connections" command beforehand.
Stimulus	The admin types "block user" command in the admin command-line interface.
Basic Flow	Step 1 – The system asks for the IP address of the user to block. Step 2 – The admin gives the IP address of the user obtained from the list displayed after "manage connections" command. Step 3 – Server reaches to the database and sets the blockStatus field of the user record with the given IP Address as "blocked"
Alternative Flow#1	-
Alternative Flow#2	-
Exception Flow	If there is no connected user with the given IP address, the system displays an error message saying that "Invalid Ip address". If the user with the given IP address has already been blocked, the system displays an error message saying that "The given IP address has already been blocked".
Post Conditions	The user with the given IP address will not be able to connect to the server until it is unblocked.

Table 14: Block User

Use-Case Name	Unblock User
Actors	Admin
Description	The admin unblocks a user with given IP address.
Data	IP address of the user
Preconditions	The admin must be logged in. The admin must have given the "manage connections" command beforehand.
Stimulus	The admin types "unblock user" command in the admin command-line interface.
Basic Flow	Step 1 – The system asks for the IP address of the user to unblock. Step 2 – The admin gives the IP address of the user obtained from the list displayed after "manage connections" command. Step 3 – Server reaches to the database and sets the blockStatus field of the user record with the given IP Address as "unblocked"
Alternative Flow#1	-
Alternative Flow#2	-
Exception Flow	If there is no blocked user with the given IP address, the system displays an error message saying that "No blocked users were found".
Post Conditions	The user with the given IP address will be able to connect to the server again.

Table 15: Unblock User

Use-Case Name	Log in as Maintainer
Actors	Maintainer
Description	The maintainer connects to server with his/her login key.
Data	Maintainer login key
Preconditions	-
Stimulus	The maintainer sends login request to server from the terminal.
Basic Flow	Step 1 – The server receives the login request. Step 2 – The server asks for the maintainer login key. Step 3 – Maintainer types his/her login key to the terminal. Step 4 – The system reaches to the local database and verifies the given login key. Step 5 – The system signs in the maintainer.
Alternative Flow#1	-
Alternative Flow#2	-
Exception Flow	If the given key is invalid, the system displays an error message saying that "Given key is invalid". The system saves the log of the failed login attempt and the IP address of the user to the database. Then, the flow starts again from Step 2.
Post Conditions	The maintainer command-line interface opens.

Table 16: Log in as Maintainer

Use-Case Name	View system logs
Actors	Maintainer
Description	The maintainer can list and view the system logs sorted by their dates.
Data	System logs
Preconditions	Maintainer must be logged in.
Stimulus	The maintainer gives "view logs" command using the command-line interface.
Basic Flow	Step 1 – The server takes the request. Step 2 – The server gathers the system logs from the database. Step 3 – The system displays the system logs in command-line interface.
Alternative Flow#1	-
Alternative Flow#2	-
Exception Flow	If there is no system logs in database, the system displays a message saying that "No system logs were found".
Post Conditions	The system logs are displayed in the maintainer's interface.

Table 17: View System Logs

Use-Case Name	Check Updates
Actors	Maintainer, OpenFlexure Update System
Description	The maintainer checks for the software updates.
Data	System's current software version, released update information
Preconditions	Maintainer must be logged in.
Stimulus	The maintainer gives "check updates" command using the command line interface.
Basic Flow	Step 1 – The server retrieves the version number of the software from the database. Step 2 – The server sends the current version number of the software to the OpenFlexure Update System. Step 3 – The server receives the information about the latest update returned by the OpenFlexure Update System. Step 4 – The server displays the version number, size, release date and changelog of the update in command-line interface.
Alternative Flow#1	-
Alternative Flow#2	-
Exception Flow	If the system is already using the latest version, an error message saying that "No updates were found" is displayed.
Post Conditions	Information about the latest update is displayed in the command-line interface.

Table 18: Check Updates

Use-Case Name	Update Software
Actors	Maintainer, OpenFlexure Update System
Description	The maintainer updates the system software.
Data	Installable update package
Preconditions	Maintainer must be logged in. Maintainer must have given the "check updates" command beforehand.
Stimulus	The maintainer gives "update" command using the command line interface.
Basic Flow	Step 1 – The server requests the update package from the OpenFlexure Update System. Step 2 – The package is downloaded to the Raspberry Pi. Step 3 – The server installs the downloaded package. Step 4 – The server saves the version number and the release date of the update to the corresponding database table.
Alternative Flow#1	-
Alternative Flow#2	-
Exception Flow	If an error is encountered during the installation, for example, if the downloaded package is corrupted, the system saves the detailed error log into the database.
Post Conditions	The system is updated to the latest version.

Table 19: Update Software

4.2 Composition View

The Composition View section provides information about the components of the system. Detailed information can be found corresponding parts of the section.

4.2.1 Component Diagram

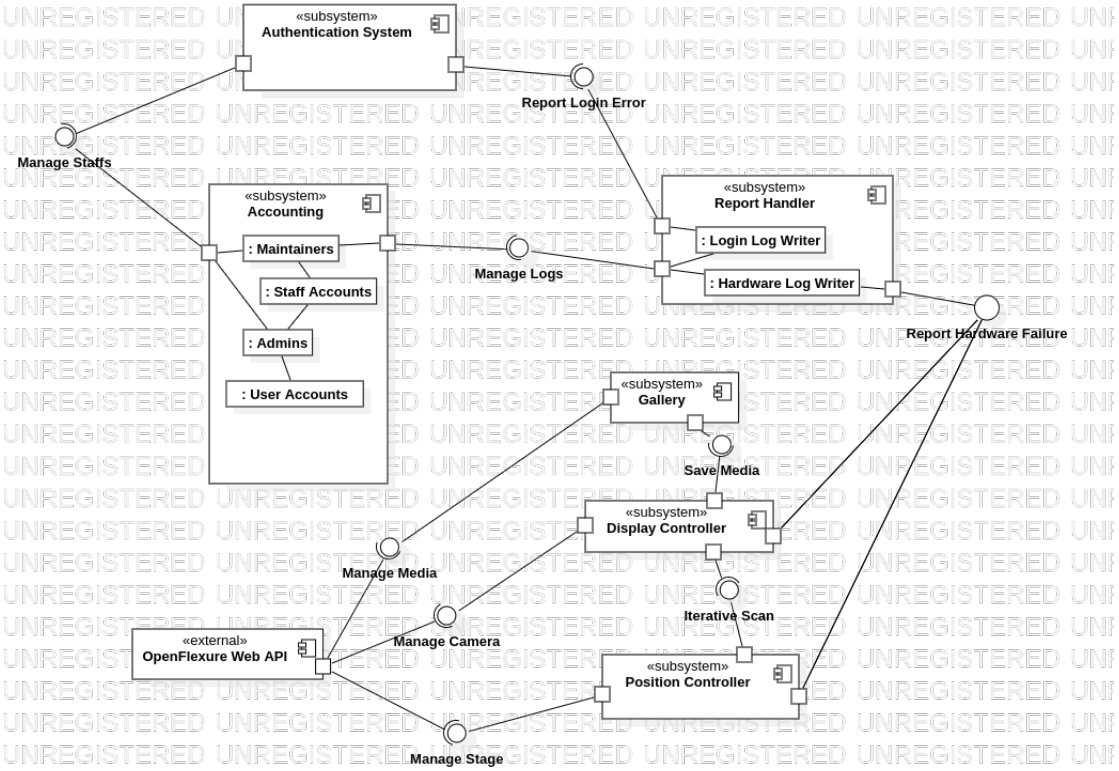

Figure 3: Component Diagram for OpenFlexure

4.2.2 Deployment Diagram

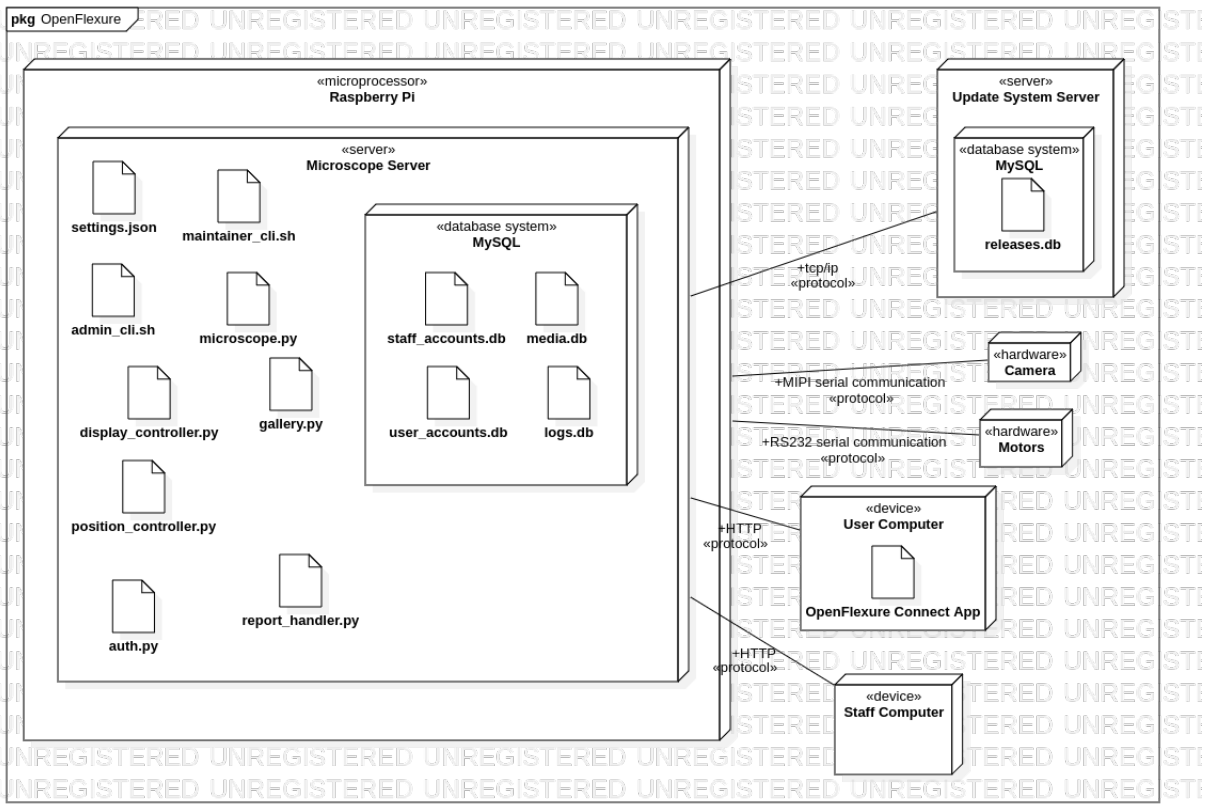

Figure 4: Deployment Diagram for OpenFlexure

4.2.3 Design Rationale

4.2.3.1 Component Diagram

- To allow the admins to use Block User and Unblock User functions defined in use-case diagram, Admins part is associated with the User Accounts part.

- To allow the admins to use Block User and Unblock User functions defined in use-case diagram, Admins part is associated with the User Accounts part.

- To allow the Authentication Systems to check the provided password information, Accounting provides an interface.
- Report Handler provides two different interfaces which are used by the Authentication System, Position Controller and Display Controller to write and store Login related and Hardware related issues.
- To be able to store the media (images and videos) generated by the Display Controller, Gallery provides an interface.

4.2.3.2 Deployment Diagram

- MySQL is used as database management system to store System Logs, Staff Accounts, User Connection Records and saved Media Metadata.
- System Settings are stored in json file, since it is easier and faster to process json file.
- To provide Admin Commmand-line Interface and Maintainer Command-line Interface two bash programs are stored in the server, admin_cli.sh and maintainer_cli.sh, respectively.
- Authentications of the staff personal are done by auth.py
- For Raspberry PI Camera hardware and Motor hardware MIPI and RS232 serial communication protocols are used.

4.3 Information View

The Information View section contains information about the organization and the relations of the data stored in the system.

4.3.1 Class Diagram

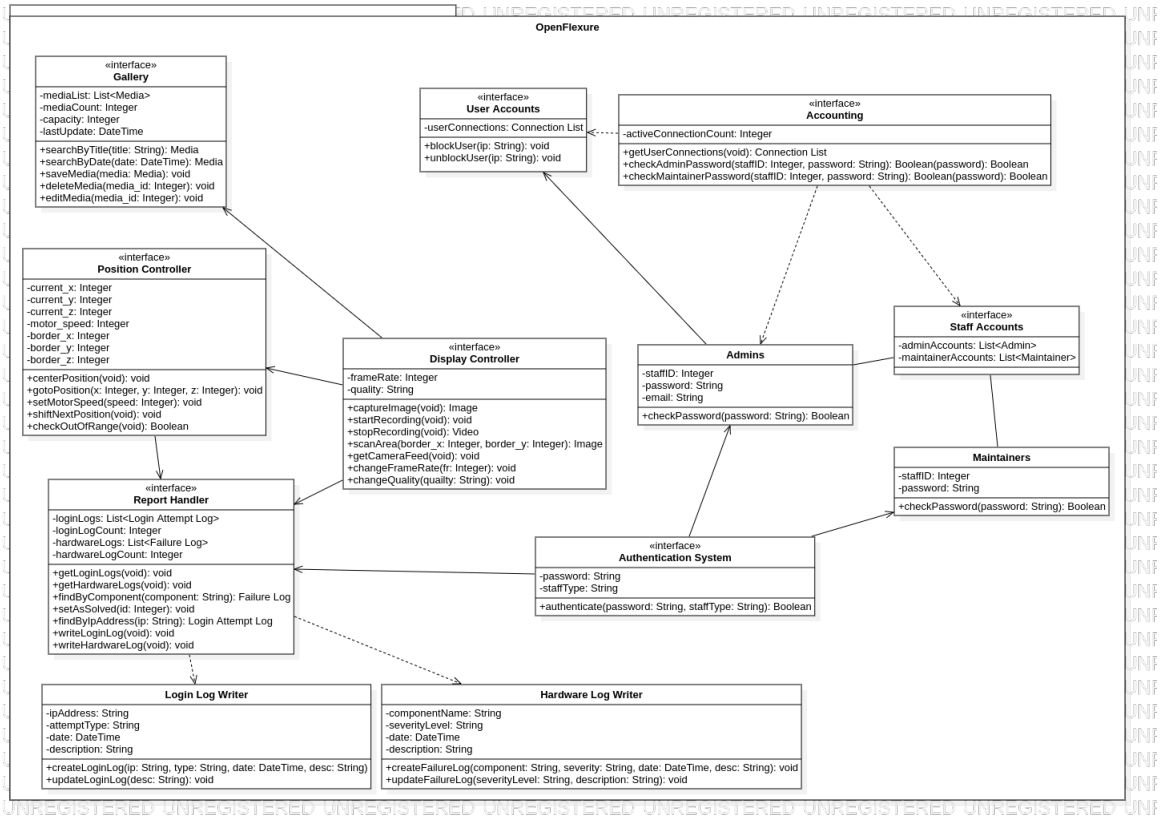

Figure 5: Interface Class Diagram for OpenFlexure

Operation	Description
searchByTitle	returns the media with given title
searchByDate	returns the media with given date
saveMedia	saves the given media to the gallery media list
deleteMedia	delete the media with given id from the gallery media list
editMedia	edits the media with given id
centerPosition	positions the stage to the center
gotoPosition	positions the stage to the point with the given coordinates
setMotorSpeed	sets the motor speed
shiftNextPosition	shifts the stage position to the next one by incrementing the current coordinates
checkOutOfRange	checks if the current coordinates are out of range
captureImage	captures the displayed livestream and returns the captured image
startRecording	starts recording the livestream
stopRecording	stops the ongoing recording and returns the recorded video
scanArea	scans the area with the given border coordinates and returns it as image
getCameraFeed	streams the live camera feed
changeFrameRate	changes the frame rate of the camera
changeQuality	changes the quality of the camera
getLoginLogs	returns the list of Login Attempt Logs
getHardwareLogs	returns the list of Failure Logs
findByComponent	returns the Failure Log with the given component name
setAsSolved	sets the status of the log with given id as solved
findByIpAddress	returns the Login Attempt Log with the given component name
writeLoginLog	writes the login log created by Login Log Writer to the loginLogs list
writeHardwareLog	writes the hardware log created by Hardware Log Writer to the hardwareLogs list
createLoginLog	creates a Login Attempt Log and fills it with the given parameters

updateLoginLog	updates the Login Attempt Log with the given id
createFailureLog	creates a Failure Log and fills it with the given parameters
updateFailureLog	updates the Failure Log with the given id
authenticate	checks the password of the user with given staff type and authenticates him/her
getUserConnections	returns the list of Connections
checkAdminPassword	compares the given password with the password of the admin with given staffID and returns the result of the comparison
checkMaintainerPassword	compares the given password with the password of the maintainer with given staffID and returns the result of the comparison
blockUser	sets the blockStatus of the user with the given ip as True
unblockUser	sets the blockStatus of the user with the given ip as False
checkPassword	compares the given password with the password of the admin/maintainer

Table 20: Operation Descriptions

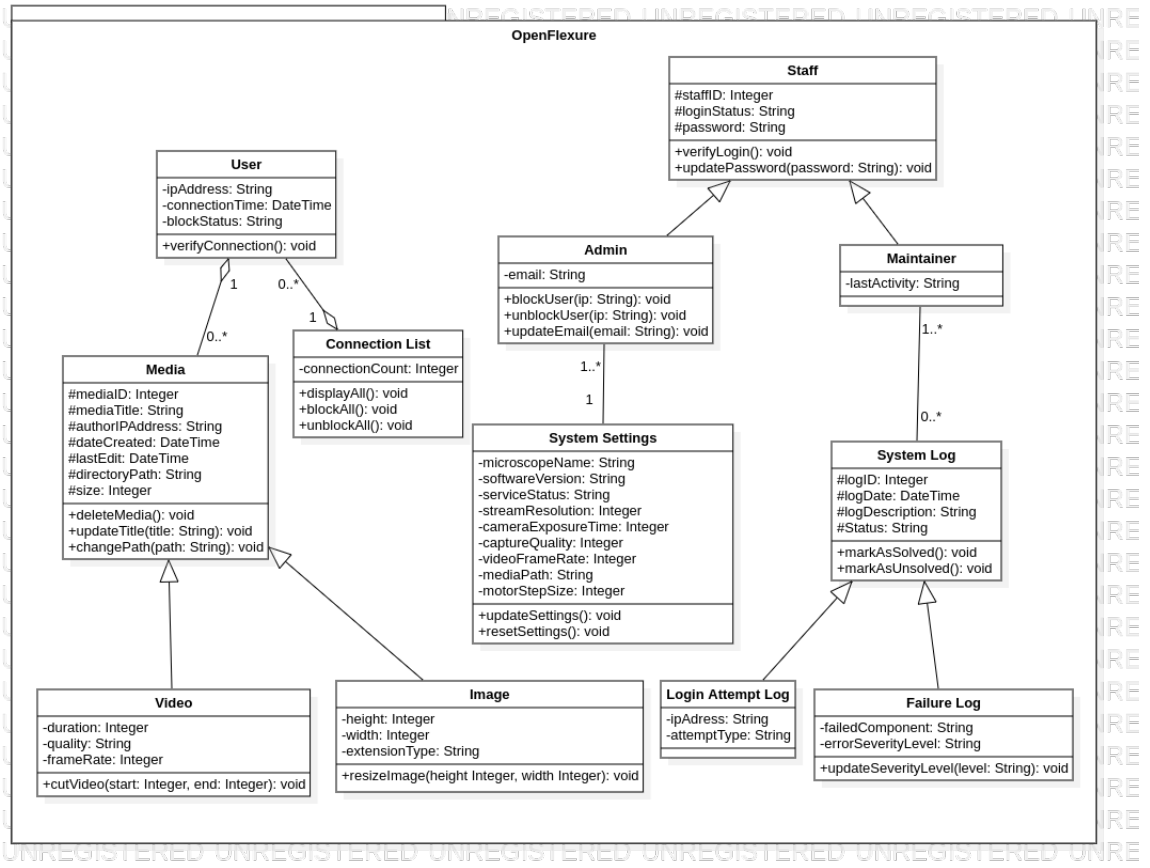

Figure 6: Database Class Diagram for OpenFlexure

4.3.2 Database Operations

Operation	**CRUD Operations**
verifyLogin	CREATE: READ: `Staff` UPDATE: `Staff` DELETE:
updatePassword	CREATE: READ: UPDATE: `Staff` DELETE:
updateEmail	CREATE: READ: UPDATE: `Admin` DELETE:
blockUser	CREATE: READ: UPDATE: `User` DELETE:
unblockUser	CREATE: READ: UPDATE: `User` DELETE:
updateSettings	CREATE: READ: UPDATE: `Settings` DELETE:
resetSettings	CREATE: READ: `Settings` UPDATE: `Settings` DELETE:
markAsSolved	CREATE: READ: UPDATE: `System Log` DELETE:

markAsUnsolved	CREATE: READ: UPDATE: `System Log` DELETE:
updateSeverityLevel	CREATE: READ: UPDATE: `Failure Log` DELETE:
verifyConnection	CREATE: READ: `User` UPDATE: `User` DELETE:
deleteMedia	CREATE: READ: UPDATE: DELETE: `Media`
updateTitle	CREATE: READ: UPDATE: `Media` DELETE:
changePath	CREATE: READ: UPDATE: `Media` DELETE:
cutVideo	CREATE: READ: UPDATE: `Video` DELETE:
resizeImage	CREATE: READ: UPDATE: `Image` DELETE:
displayAll	CREATE: READ: `Connection List`, `User` UPDATE: DELETE:

blockAll	CREATE: READ: `Connection List` UPDATE: `User` DELETE:
unblockAll	CREATE: READ: `Connection List` UPDATE: `User` DELETE:

Table 21: CRUD Operations

4.3.3 Design Rationale

4.3.3.1 Interface Class Diagram

- Report Handler is responsible for creating and storing logs related to hardware failures and login errors.
- Both Display Controller and Position Controller associated with the Report Handler to report hardware related problems.
- Authentication System is associated with the Report Handler to report suspicious login attempts.
- To allow the maintainers to use View System Logs function defined in use-case diagram, Maintainers component uses the interface provided by the Report Handler.
- To allow the admins to use Block User and Unblock User functions defined in use-case diagram, Admins part is associated with the User Accounts part.
- To be able to store the captured images, scans and recorded videos, Display Controller is associated with the Gallery.

4.3.3.2 Database Class Diagram

- For easy and fast usability of the microscope, users do not have to authenticate to the system. Therefore, for user records, password information is omitted.

- To avoid repetition of the attributes, Media, System Log and Staff parent classes are defined.
- To be able to handle different types of errors, Login Attempt Log and Failure Log classes are created.
- To allow the maintainer to use View System Logs function defined in use-case diagram, Maintainer and System Logs classes are linked to each other.

4.4 Interface View

The Interface View section provides information about the internal and external interfaces of the system.

4.4.1 Internal Interfaces

4.4.1.1 Interface between Accounting and Authentication System

The interface provided by Accounting component is used by the Authentication System to verify the provided staffID and password.

Design Rationale:

- Authentication System sends the given staffID and password parameters to the Accounting component.
- Based on the type of the staff Accounting checks the password using checkPassword operation belongs to either Admins or Maintainers parts.
- Returns the result of the checkPassword operation.

4.4.1.2 Interface between Authentication System and Report Handler

The interface provided by the Report Handler is used by the Authentication System to report suspicious and wrong login attempts.

Design Rationale:

- If the boolean result returned by the Accounting, Authentication System sends the provided information to the Report Handler.
- Report Handler saves the Login Attempt Log created by the Login Log Writer part in the loginLogs list.

4.4.1.3 Interface between Accounting and Report Handler The interface provided by the Report Handler is used by the Accounting and especially its subcomponent Maintainers to display and check the written System Logs.

Design Rationale:

- Report Handler provides the loginLogs list and hardwareLogs list to the Accounting and specifically Maintainers part to allow maintainers to use View System Logs function.

4.4.1.4 Interface between Report Handler and Display Controller and Position Controller The interface provided by the Report Handler is used by both Display and Position Controllers to report any hardware related technical issues.

Design Rationale:

- Since Display and Position Controllers are related to hardware devices (camera and motors), it is possible to have hardware related issues. This interface is for handling the hardware related problems.
- Report Handler saves the Failure Log created by the Hardware Log Writer part in the hardwareLogs list.

4.4.1.5 Interface between Display Controller and Position Controller The interface provided by the Position Controller is used by the Display Controller for the scanArea operation. In this operation by using this interface Display Controller is able to iteratively move the camera position to scan the area.

Design Rationale:

- To be able to use Scan function defined in the use-case diagram Display Controller serves scanArea function.
- To use the scanArea function position of the stage should be shifted iteratively. Position Controller serves this interface to allow Display Controller to shift stage position in each iteration.

4.4.1.6 Interface between Display Controller and Gallery The interface provided by the Gallery is used by the Display Controller to save the captured Images, scans, recorded videos.

Design Rationale:

- Display Controller is responsible to implement Capture Image, Record Video and Scan functions defined in use-case diagram.
- To be able to store the media generated by the Display Controller, Gallery serves this interface.

4.4.2 External Interfaces

4.4.2.1 User Interfaces

- **User Web Interface**
  Users use a web interface written in javascript using Vue.js framework which is running inside the local microscope server. User can reach to the User Web Interface by just typing the ip address and designated port of the microscope's server in their browser. Users are able to control the microscope and view the camera feed using the user interface. User Web Interface provides live stream camera feed, navigation menu to change the position of the stage and gallery to view and update the saved media.
  Design Rationale:
  1. User web interface has a very simple design to allow inexperienced users to use the provided features easily.
  2. It has a help window that contains solutions for the frequently faced problems to guide users faced with such problems.

3. The icons have been placed inside the tab buttons in addition to button labels to ensure that the desired feature can be found easily.

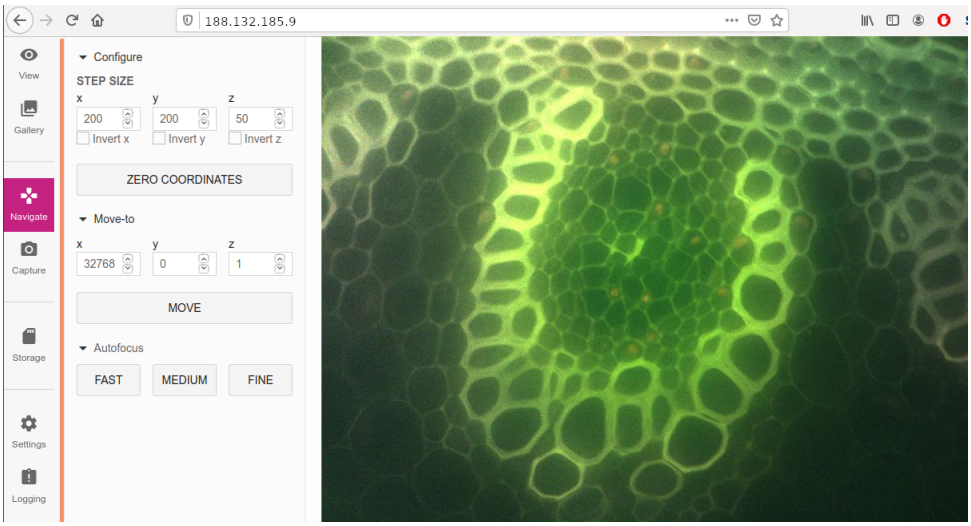

Figure 7: OpenFlexure User Web Interface

- **Admin Command-line Interface**
  System administrators use a command-line interface to manage the system. Administrators are able to list the detailed information about connected/blocked users such as User IPs, block/unblock status, and connection time. Using the commands provided by this interface, the administrators can view/manage/update system settings and block/unblock users. Admin command-line interface is provided by a bash script placed inside the local microscope server and starts to run when an admin connects to the microscope using SSH.
  Design Rationale:

  1. The reason to use a command-line interface rather than GUI is related to speed. Administrators should be able to apply their actions immediately. Since the command-line interface is a more quick way to apply actions, it is preferred over GUI.

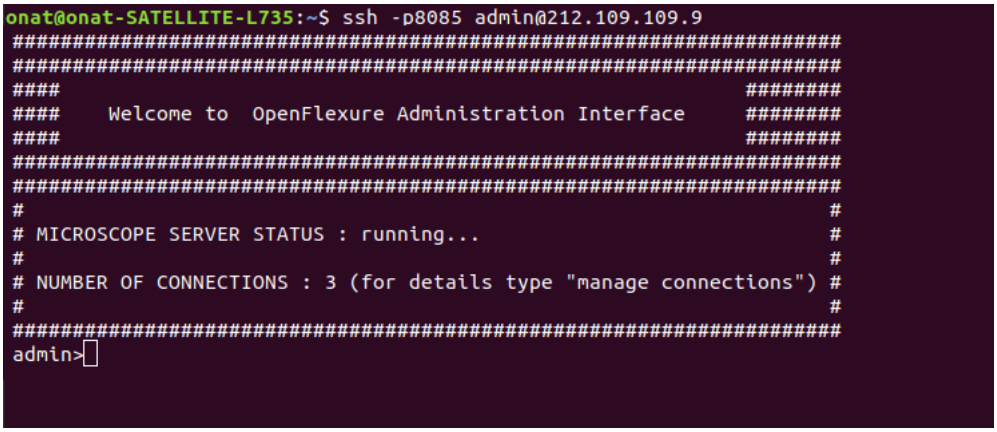

Figure 8: OpenFlexure Administration Command-line Interface

- **Maintainer Command-line Interface**
  Similar to system administrators, maintainers use a separate maintainer command-line interface to do their tasks, due to the similar reasons. Since the permissions of the maintainers are limited to viewing released updates, updating the system and retrieving the system logs, commands available to maintainers are also limited. Maintainers can list saved error logs, system details such as the current software version and the information about the latest software updates. Furthermore, maintainers are also able to update the system software using the command provided by this interface. Maintainer command-line interface is provided by a bash script placed inside the local microscope server and starts to run when a maintainer connects to the microscope using SSH.

Design Rationale:

1. The reason to use a command-line interface rather than GUI is related to speed. Maintainers should be able to apply their actions immediately. Since the command-line interface is a more quick way to apply actions, it is preferred over GUI.

4.4.2.2 System/Service Interfaces

4.4.2.3 Interface between Web API and Gallery The interface provided by the Gallery is used by the Web API when a user sends a request to the Web API to display/edit/manage the saved media materials.

Design Rationale:

- User sends a request to the Web API to view/edit the stored media.
- To allow the Web API to use the media facilities that user wants to use, Gallery provides this interface and based on the request sent to the Web API, corresponding operation of the Gallery is activated.
- This interface is related to the Open Gallery and Modify Media functions defined in the use-case diagram.

4.4.2.4 Interface between Web API and Position Controller The interface provided by the Position Controller is used by the Web API when a user sends a request to the Web API to move the position of the stage.

Design Rationale:

- User sends a request to the Web API to navigate the positions of the microscope stage.
- To allow the Web API to change the position of the stage to the desired position, Position Controller provides this interface and changes the positions of the stage using the provided coordinates.
- This interface is related to the Navigate Stage function defined in the use-case diagram.

4.4.2.5 Interface provided Web API and Display Controller The interface provided by the Display Controller is used by the Web API when a user sends a request to display the livestream, capture the current view, record to the livestream or scan an area.

Design Rationale:

- User sends a request to the Web API to use facilities provided by the Display Controller such as viewing livestream, capturing images, recording videos etc.

- To allow the Web API to use the mentioned facilities that user wants to use, Display Controller provides this interface and based on the request sent to the Web API, corresponding operation of the Display Controller is activated.
- This interface is related to the View Livestream, Capture Image, Record Video and Scan functions defined in the use-case diagram.

4.4.3 Sequence Diagrams

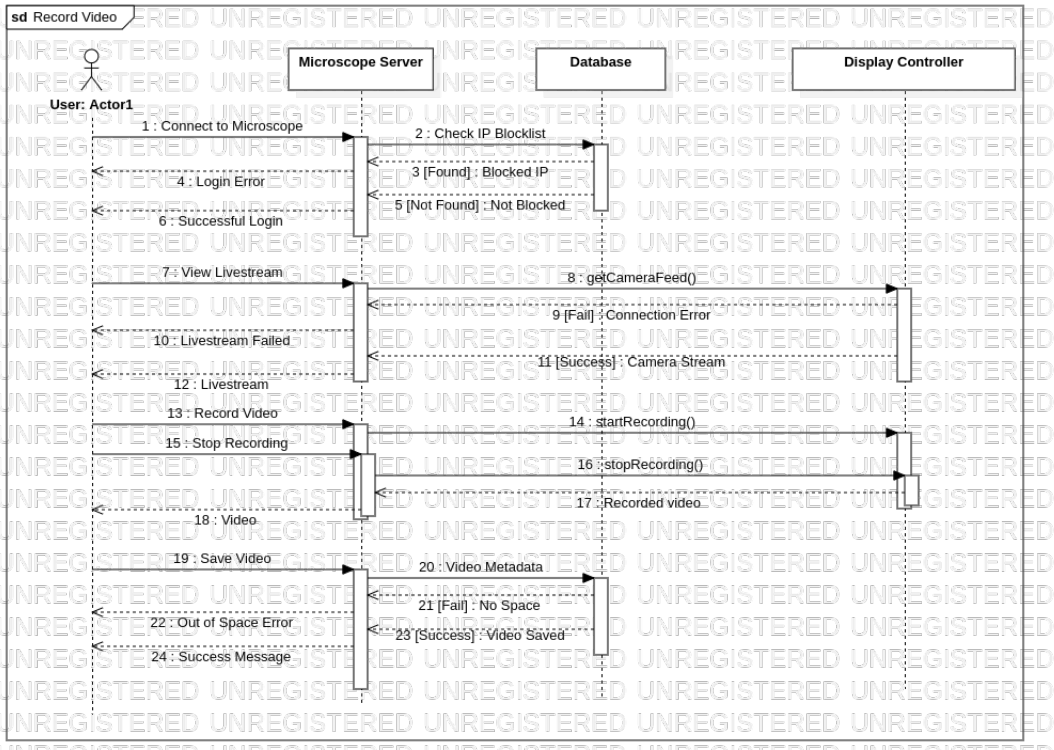

Figure 9: Sequence Diagram of Record Video Function

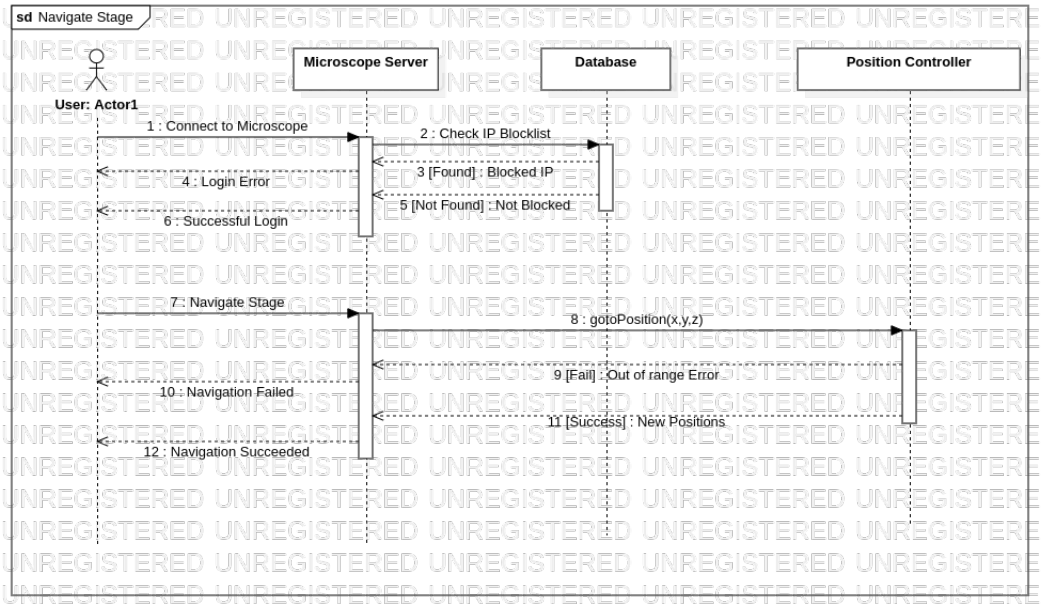

Figure 10: Sequence Diagram of Navigate Stage Function

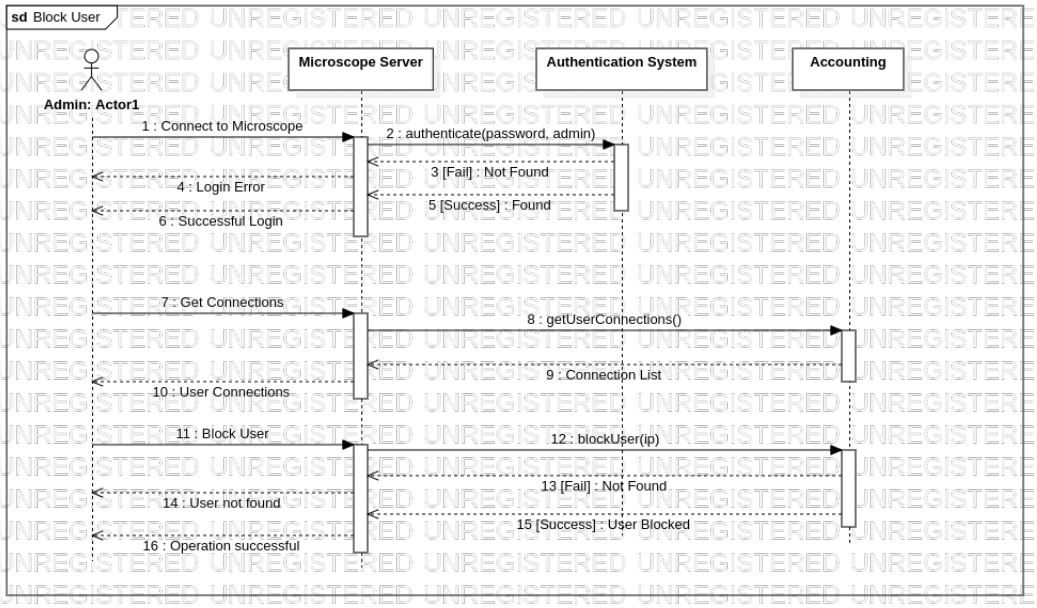

Figure 11: Sequence Diagram of Block User Function

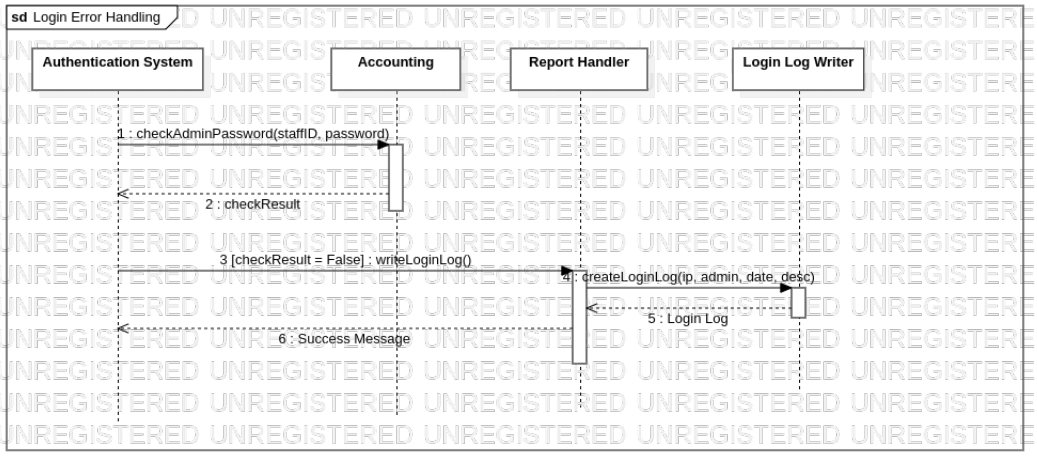

Figure 12: Sequence Diagram of Login Error Handling

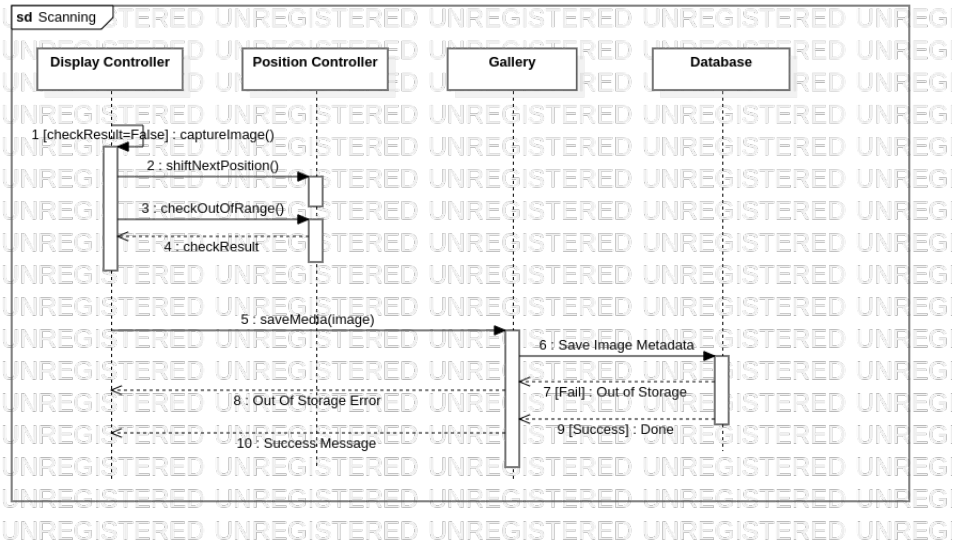

Figure 13: Sequence Diagram of Scanning

4.4.4 Design Rationale

Design rationales are specified where each interface is defined above.