UiT The Arctic University of Norway

CLERMONT AUVERGNE INP ISIMA

Creation of a network and web-based video surveillance system for robot remote control.

Presented by : Loup RUSAK

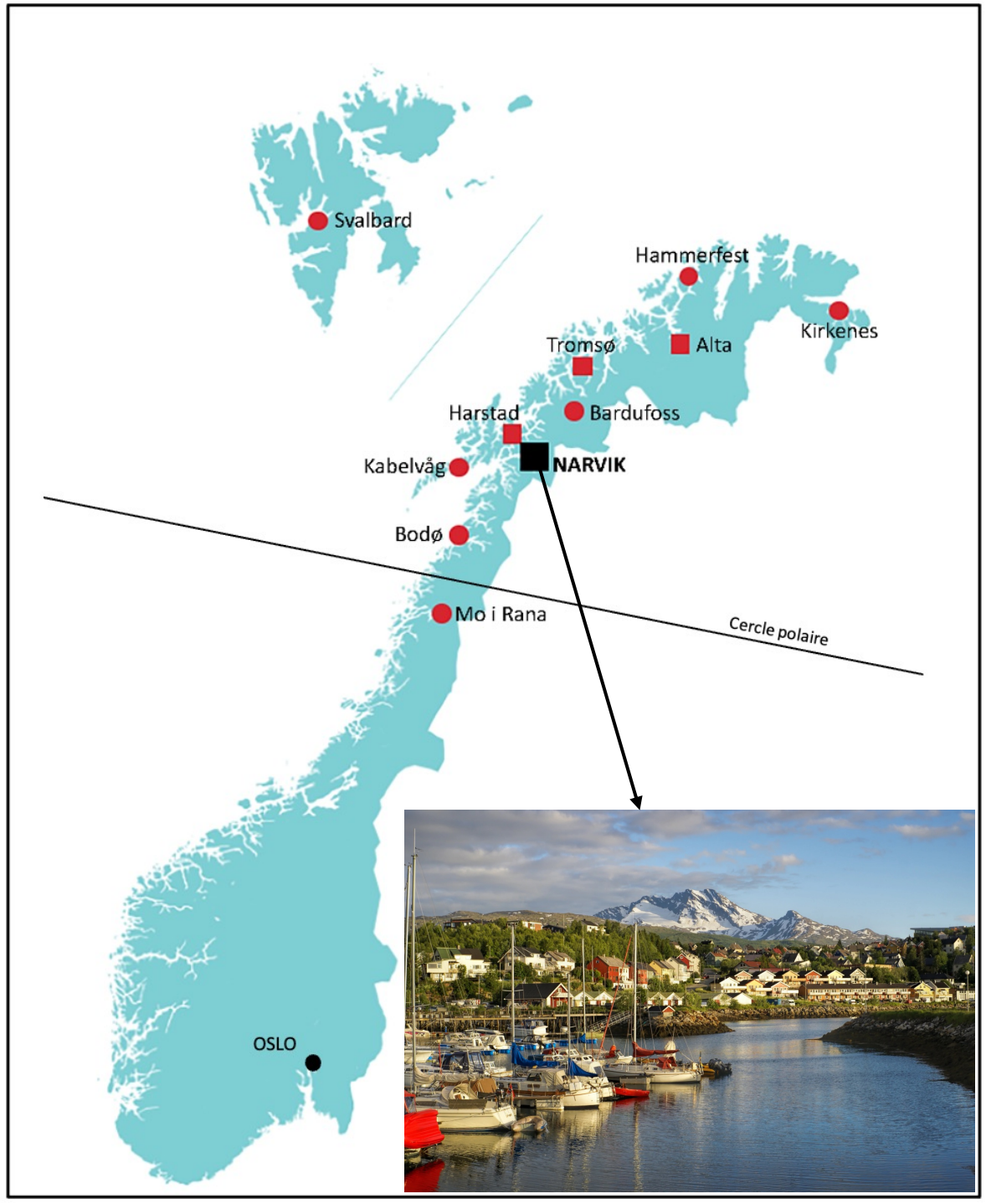
ISIMA Tutor : Romuald AUFRERE
UiT Tutors : Hao YU & Beibei SHU

28/08/2023 11h30

Introduction

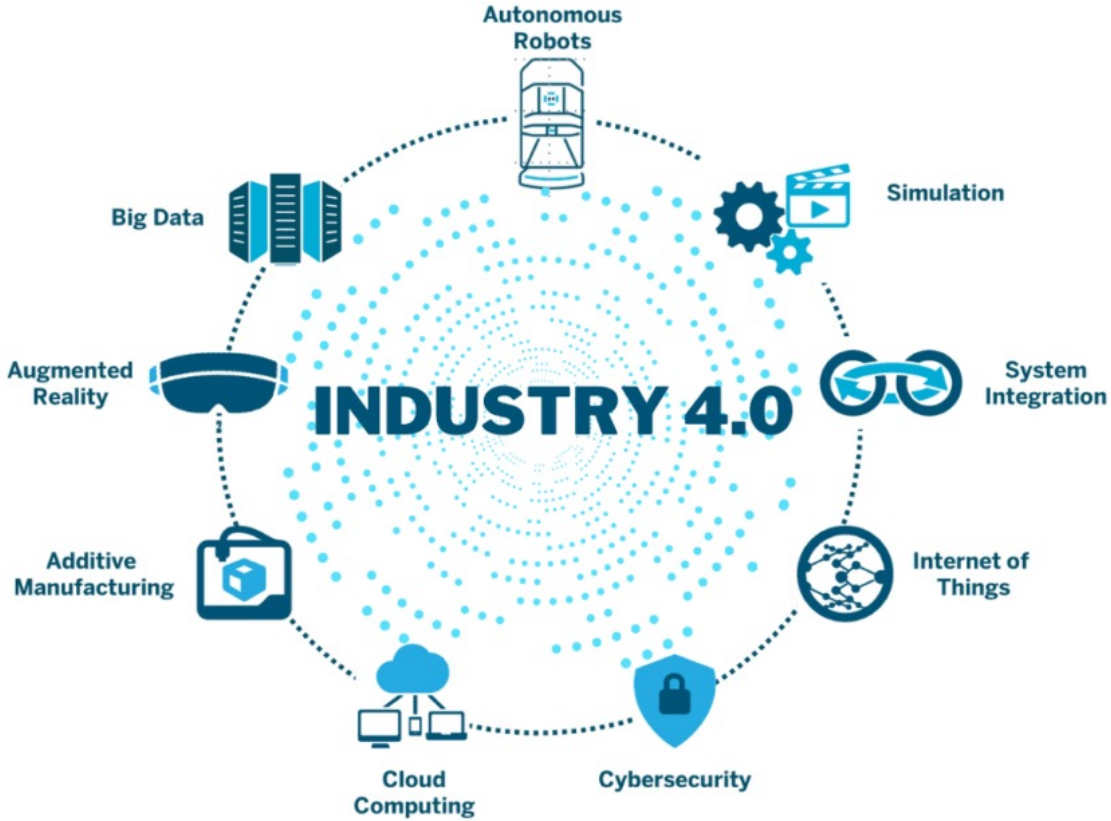
UiT The Arctic University of Norway

- **Narvik** Campus
- Founded in : **1968**
- Formely known as « **Tromsø University** »
- **Third largest** university of Norway
- **Northernmost** university in the world



Introduction

- **Industrial Engineering** Department
- Own **laboratory**
- Robot **UR10E**
- **Pickit Camera**



Develop a smart logistic system

Introduction

Web based camera view for robot remote control

- Multiples cameras
- Point the robot cell
- Different angles

Ensure safety of Human and Robot

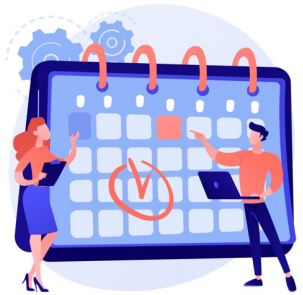
How to create from scratch a video surveillance system using Python technologies ?

Summary

1) Organization

2) Conceptualization

3) Realization



1) Organization

1) Organization

Production steps

Research and design

Research work about programmation languages, technologies and design with Figma.

Web app core

Develop app structure, pages, links, basic and advanced features.

Network streaming

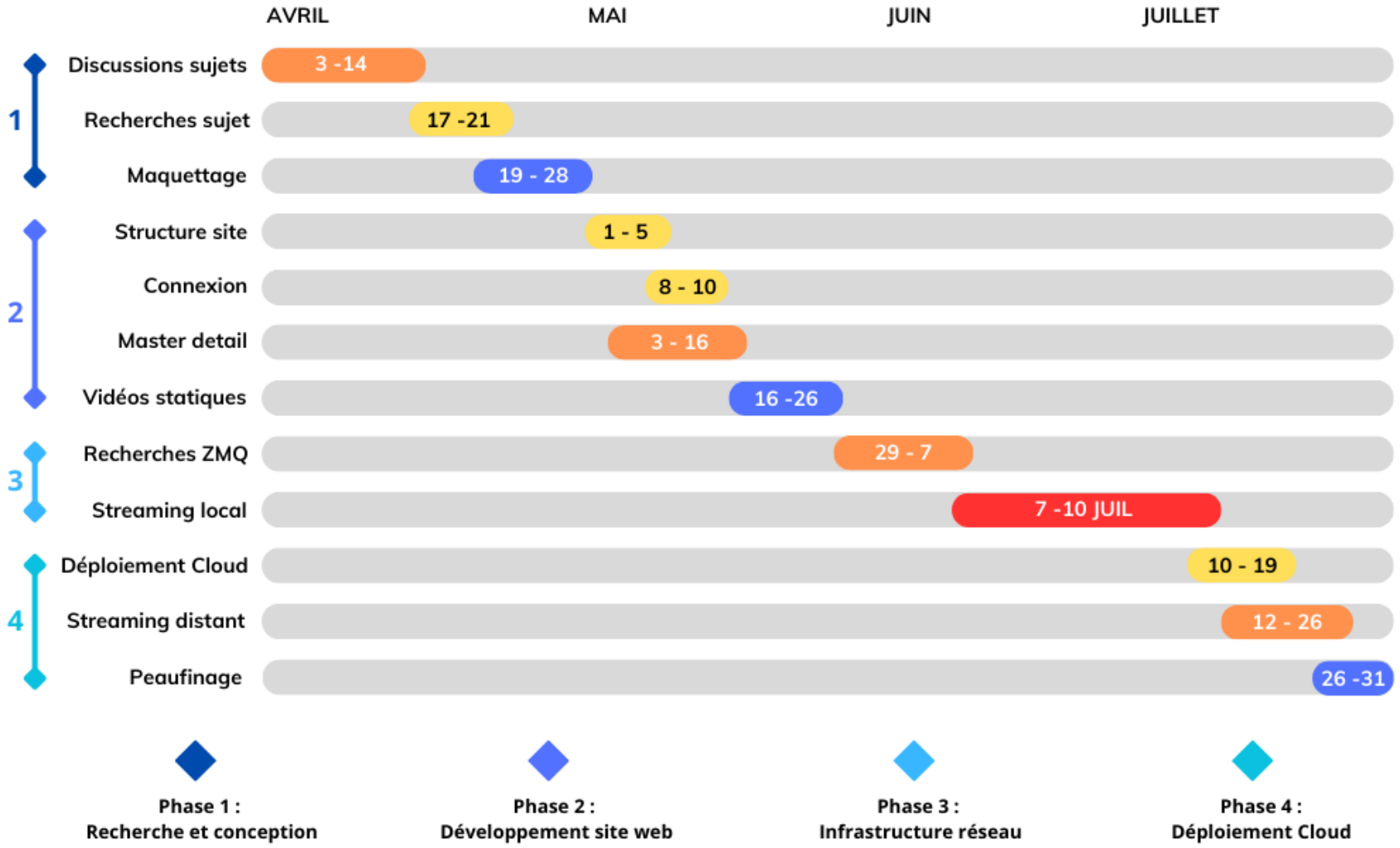
Develop server client ZMQ demo with multithreading and implement it.

Cloud deploy

Deploy the website on the cloud and camera server in local.
Fix bugs and improve app.

1) Organization

Gantt Diagram



Hardware tools - cameras

- 3 **IP wired** Cameras
- 3 view angles (right, left, top)
- **IR CCTV** Cameras
- Indoor/outdoor
- Day/night
- Embedded Linux system
- **3 streams** per camera
- **ONVIF** standard

Hardware tools - network

Switch for PC local network connection

Router for University local subnetwork connection

2) Conceptualization

2) Conceptualization

Software tools

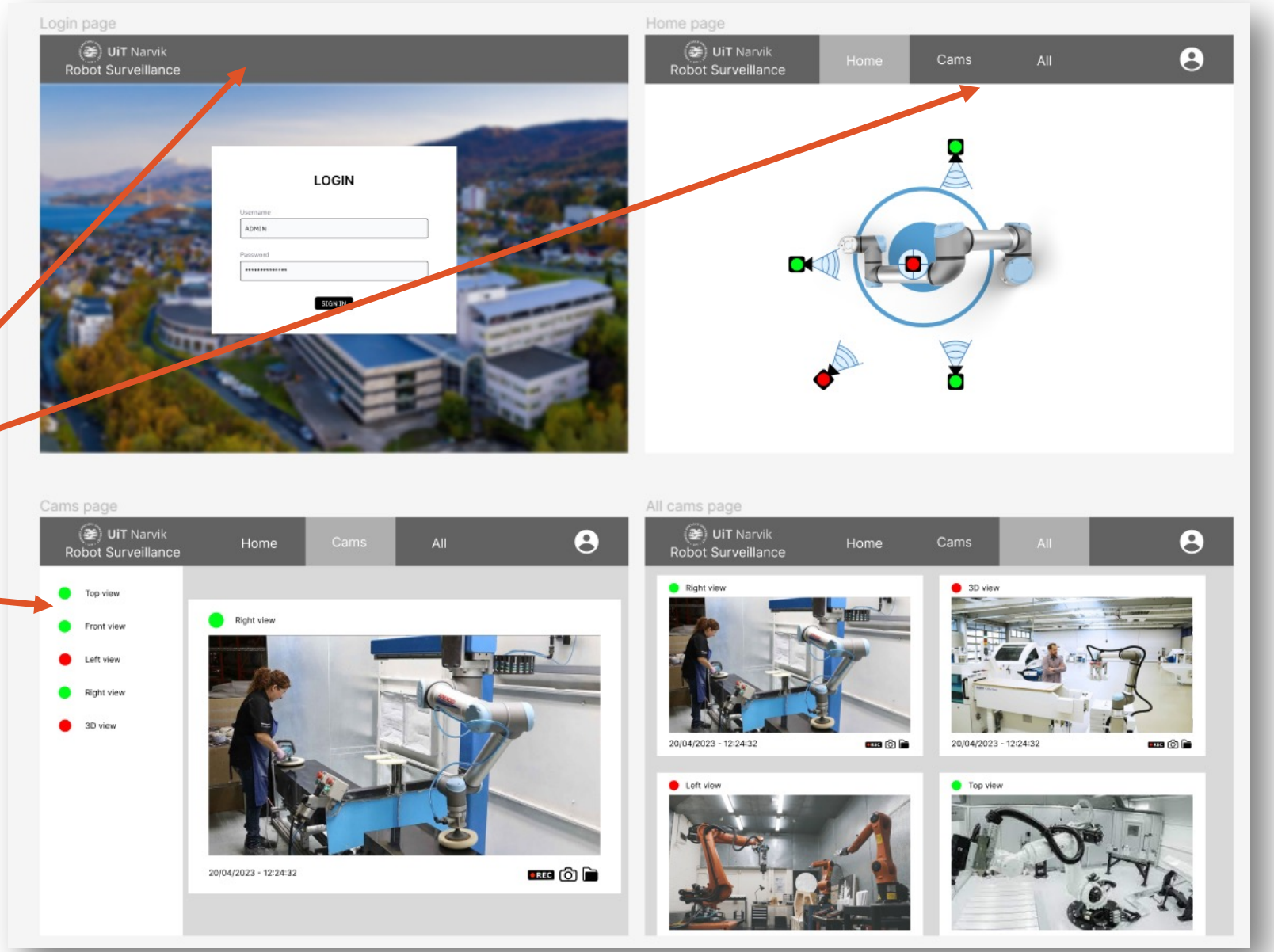
- **Python** : main programmation language.
- **Flask** : Python framework for developping web apps.
- **Bootstrap** : Website creation tools library.
- **OpenCV** : Image processing library
- **ZeroMQ (ZMQ)** : Python network library for asynchronous communication.
- **Waitress** : Python server for Flask

2) Conceptualization

Design

- Design with **Figma**
- Inspired by **real examples**

- Common **navigation bar**
- 3 **tabs**
- 2 types of **views**
- **Master Detail**



Figma

2) Conceptualization

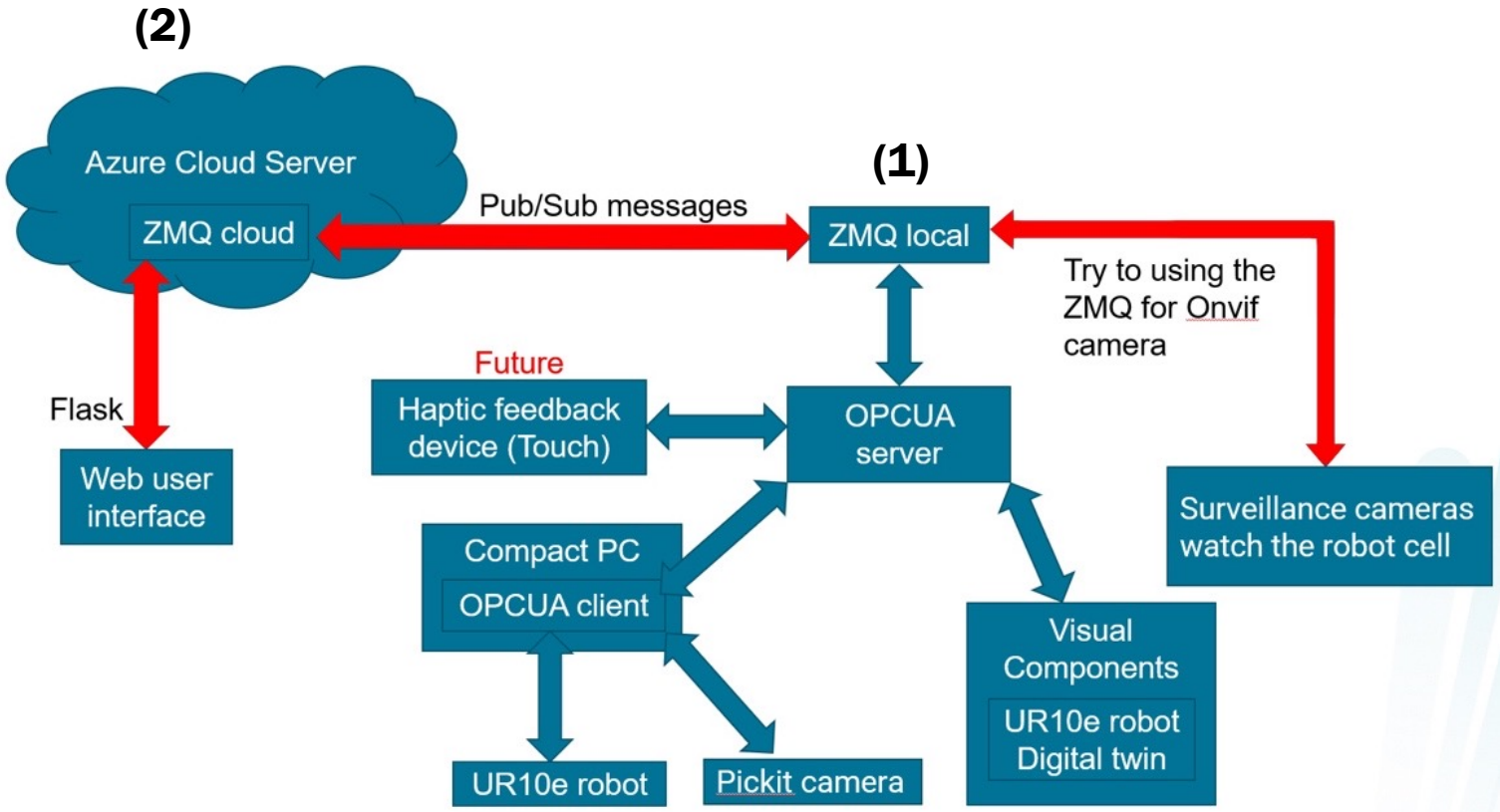
Network architecture

Local Server (1) :

- Get video from cameras
- Image processing
- Send data to Cloud

Cloud Server (2) :

- Receive data from network
- Image processing
- Display video on Flask web app



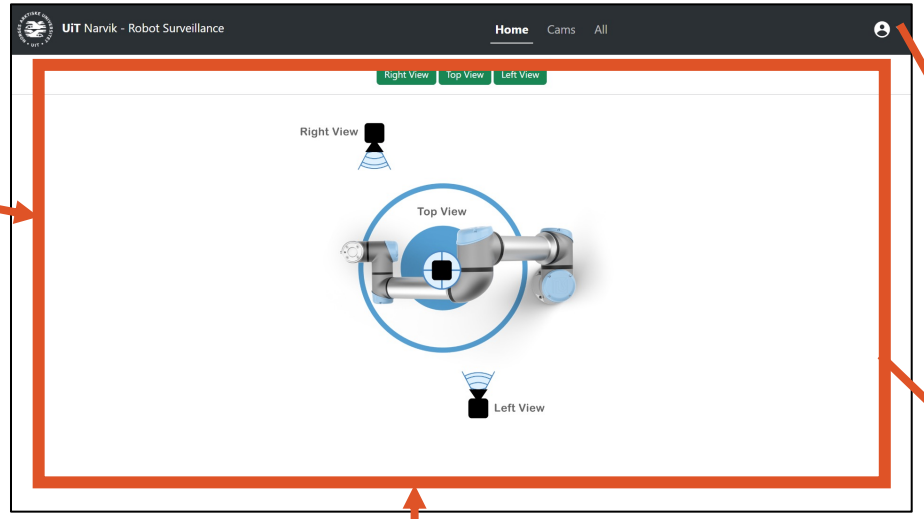
3) Realization

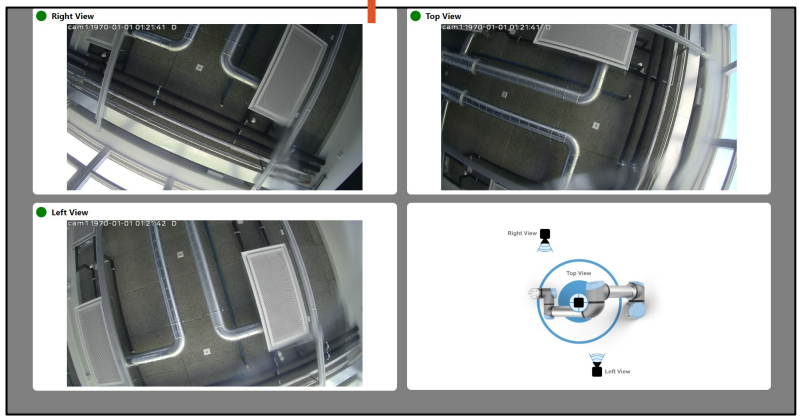
3) Realization

Website – Layout

Home view

Detail view





Global view

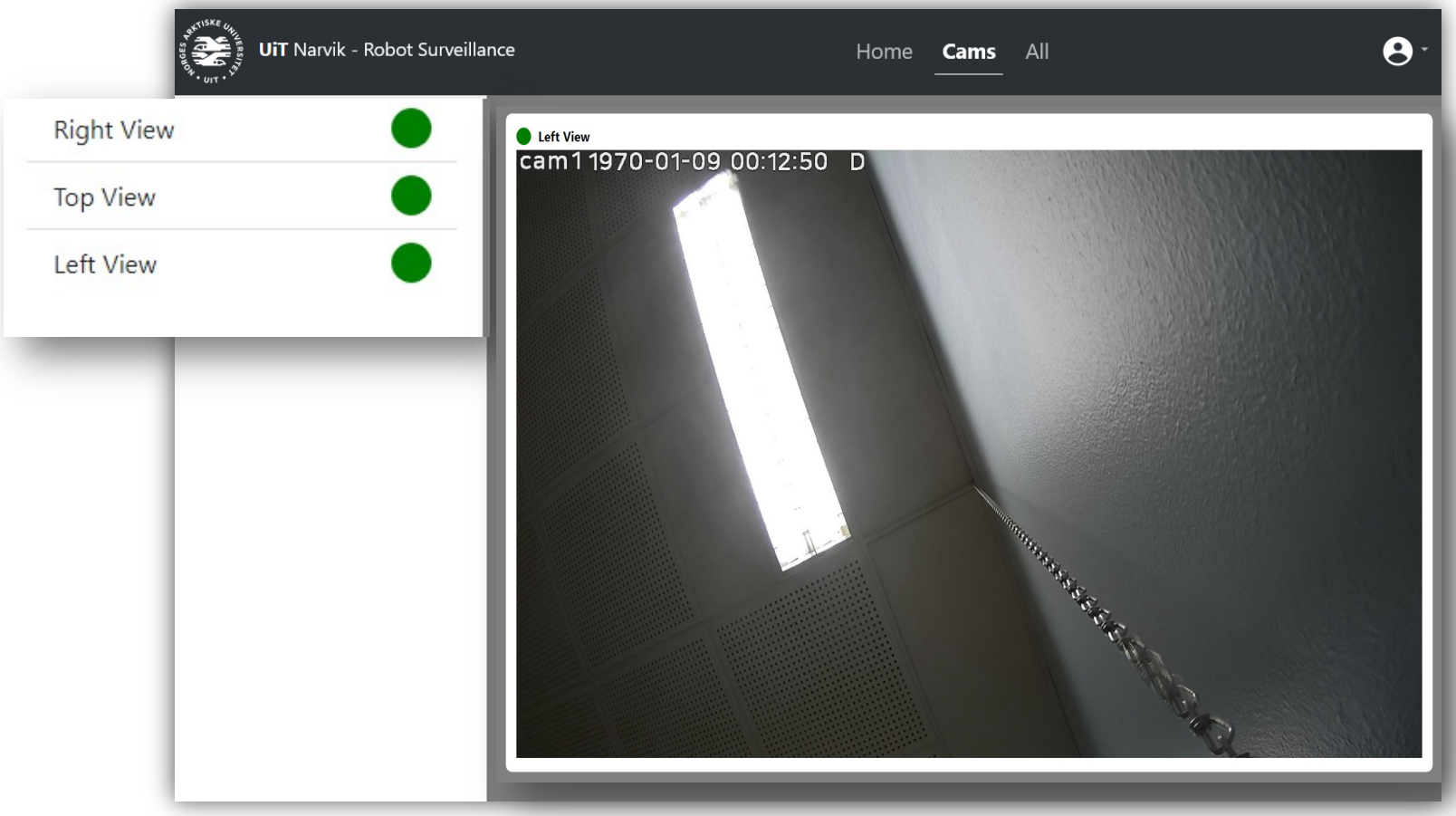
```
<!DOCTYPE html>
<html>
<head>
    <!-- Header -->
</head>

<body>
    <NavBar>
        <!-- Navigation Bar -->
    </NavBar>

    {% block content %}
        <!-- Page content -->
    {% endblock %}

</body>
</html>
```

3) Realization

Website – Master detail

```
# Cameras data
cameras = [
    {'name': 'Right View', 'status': 'active', 'src': 'video_feed_1'},
    {'name': 'Top View', 'status': 'active', 'src': 'video_feed_2'},
    {'name': 'Left View', 'status': 'active', 'src': 'video_feed_3'}
]
```

```
<div id"sidebarMenu>
    {% for cam in cameras %}
        <div>
            <a
            class="list-group-item list-group-item-action py-2
            aria-current="true"
            data-cam='{{ cam | tojson | replace("\"", "\'") }}'
            >
            {{ cam.name }}
            {% if cam.status=='active' %}
                <span class="dot cam-active"></span>
            {% else %}
                <span class="dot cam-inactive"></span>
            {% endif %}
            </a>
        </div>
    {% endfor %}
</div>
```



3) Realization

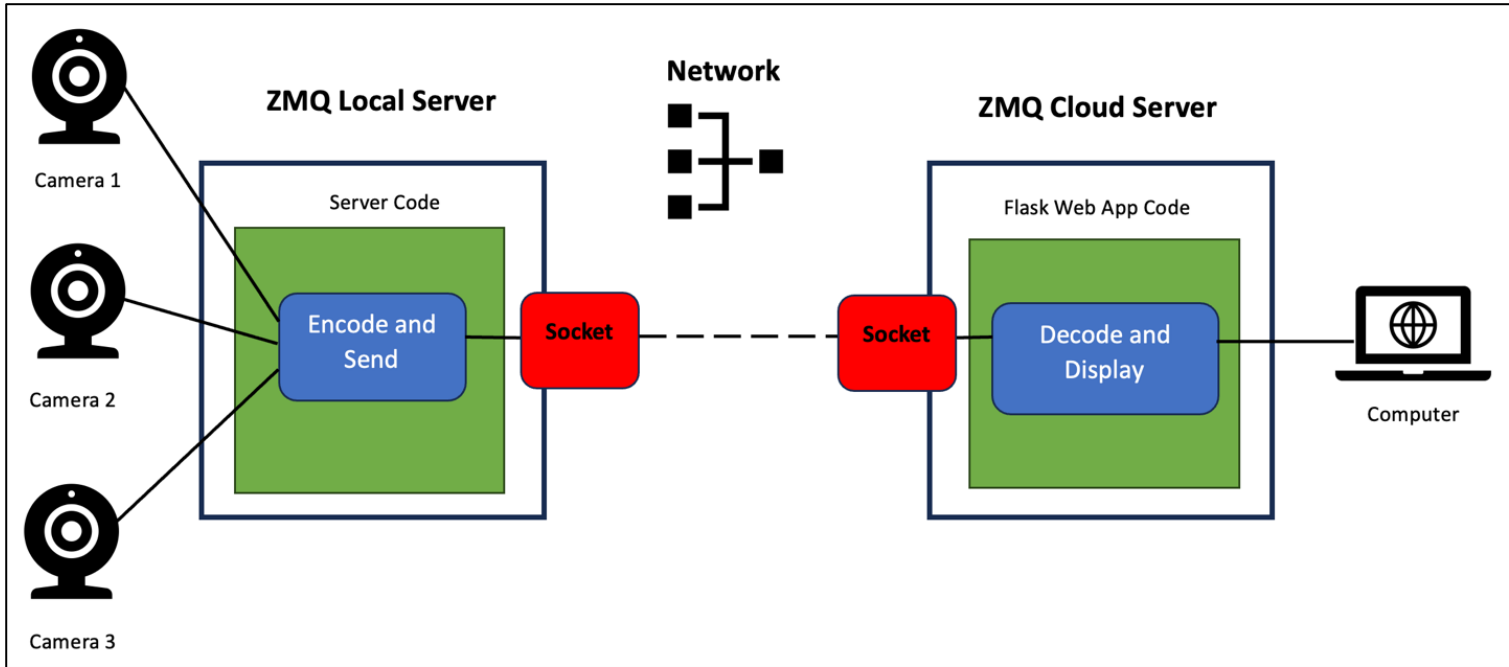
Streaming – Demo

ZMQ Local Server :

- PUB Socket
- Diffuse :
  `bind('tcp://*:5555')`
- Video capture :
  `VideoCapture / Read`
- Processing :
  `imencode('.jpg', image)`
- Send :
  `send_multipart(data)`

ZMQ Cloud Server :

- SUB Socket
- Listen :
  `connect('tcp://ipaddr:5555')`
- Receive :
  `recv-multipart()`
- Processing :
  `imdecode(data)`
- Display



3) Realization

Streaming – Issues

1) Exponentialy high video latency

- 1 or 2 seconds at the beginning
- Up to 45 seconds 1 minute later

Bottleneck somewhere ?

2) Time delay when switching views

- Leaving and come back
- Timestamp remains the same
- Non direct visualization

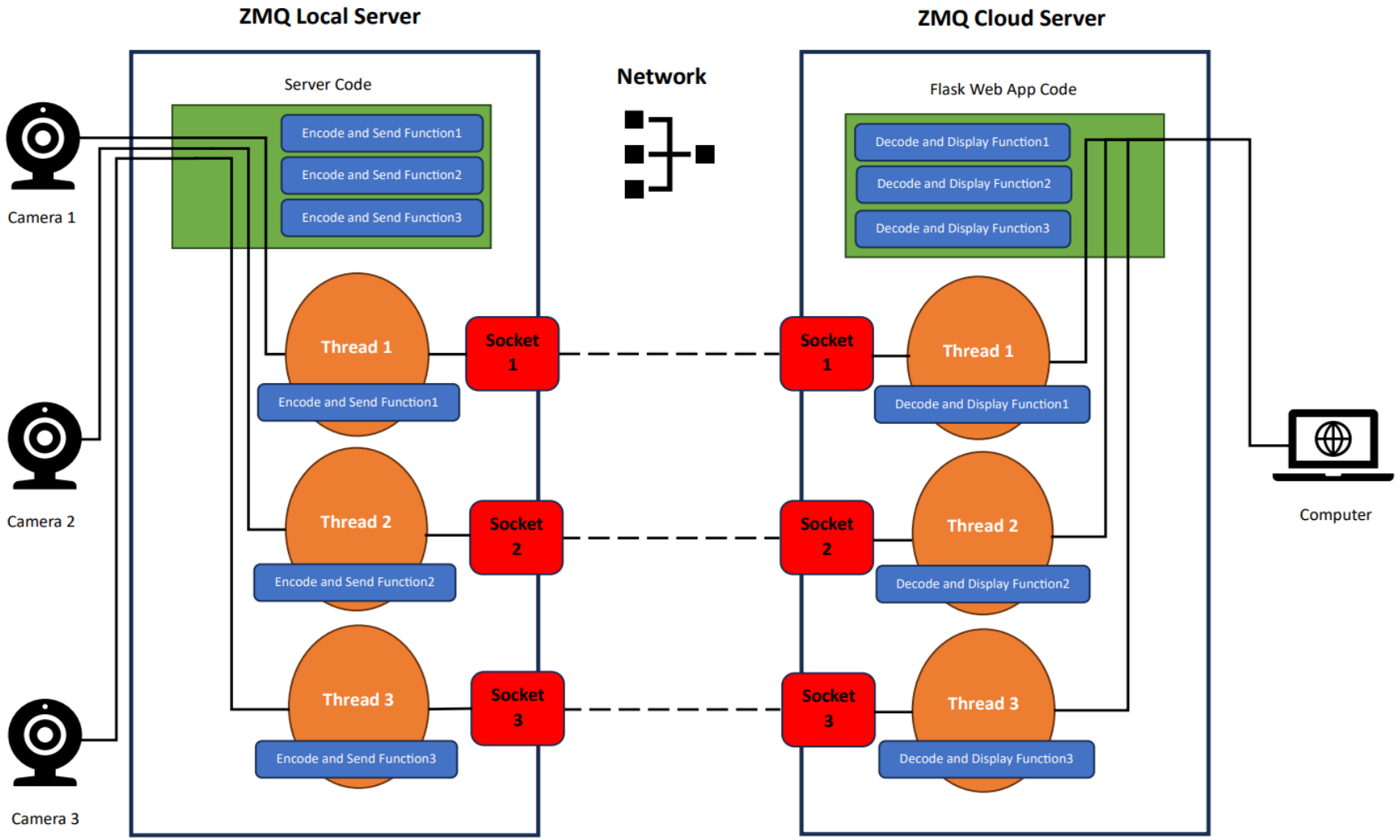
Frame **queue ?**

3) Realization

Solution 1 - Multithreading

On each side :

- 1 **socket** per camera
- 1 **thread** per camera
- 1 **threaded function** (encode/decode)



Solution 2 – Skipping frames

« **Flush** » socket waiting queue ?

Web app side :

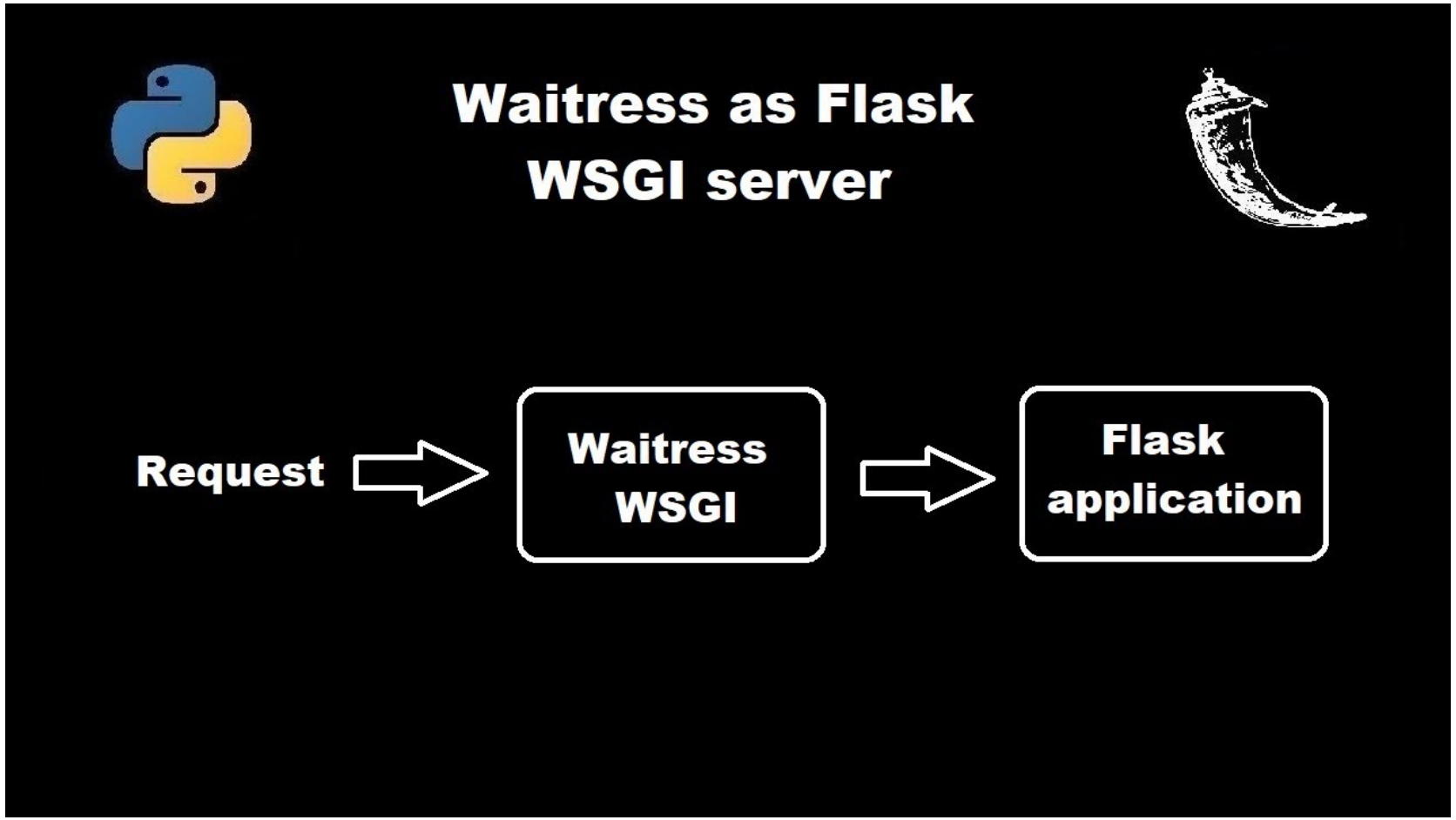
- Catch-up frames
- Skip frames

- **Compare** times
- **Calculate** number of frames
- **Receive loop** for skipped frames
- No processing

```
# Skipping lost frames
current_time = time.time()
if last_visualization_time is not None:
    elapsed_time = current_time - last_visualization_time
    skipped_frames = int(elapsed_time / frame_time)
    for _ in range(skipped_frames):
        try:
            socket.recv_multipart(zmq.NOBLOCK)
        except zmq.error.Again:
            break
last_visualization_time = current_time
```

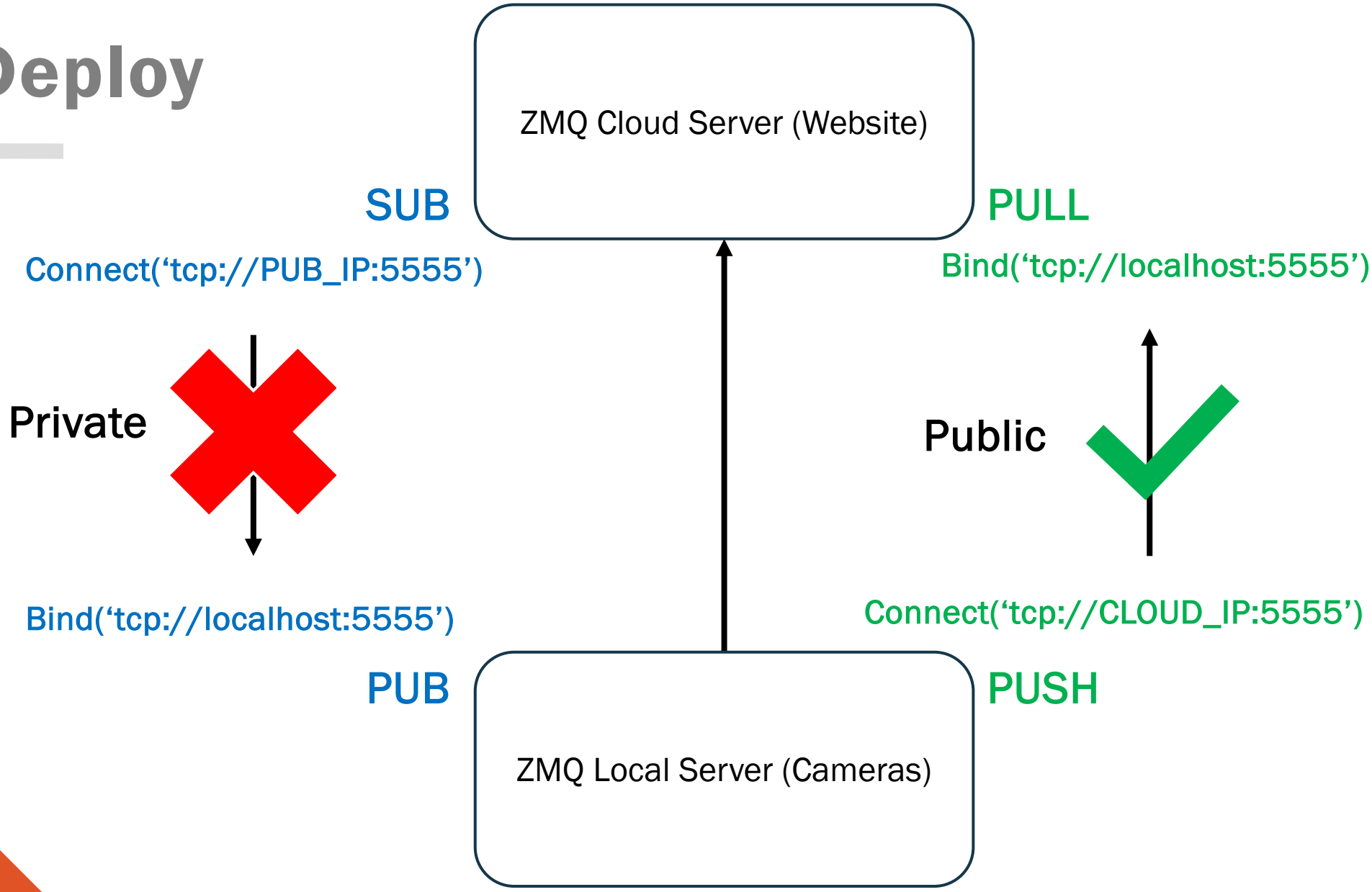
3) Realization

Cloud Deploy



3) Realization

Cloud Deploy



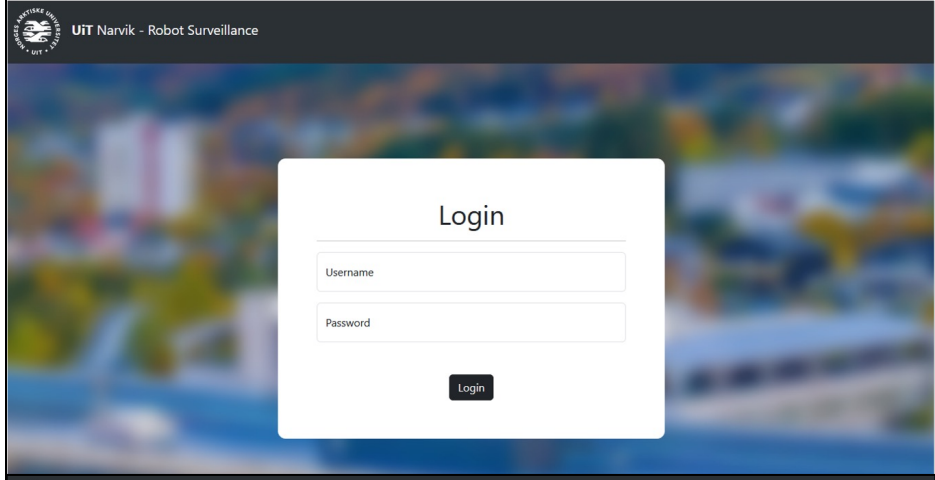
Conclusion

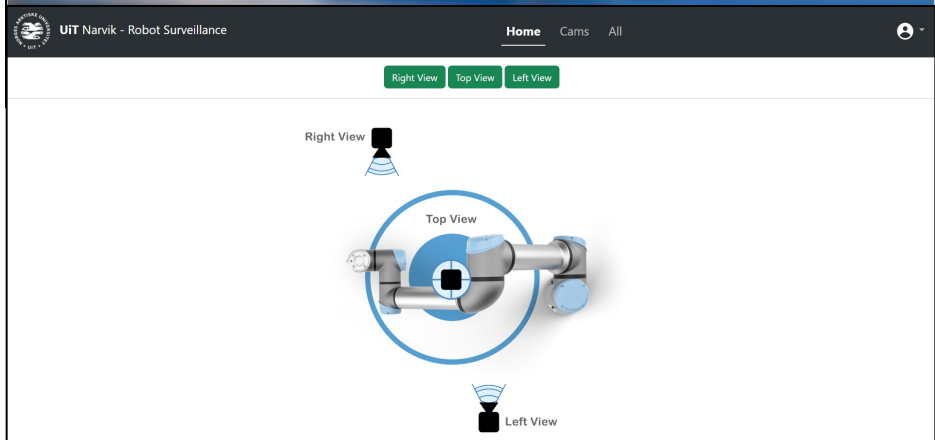
Content :

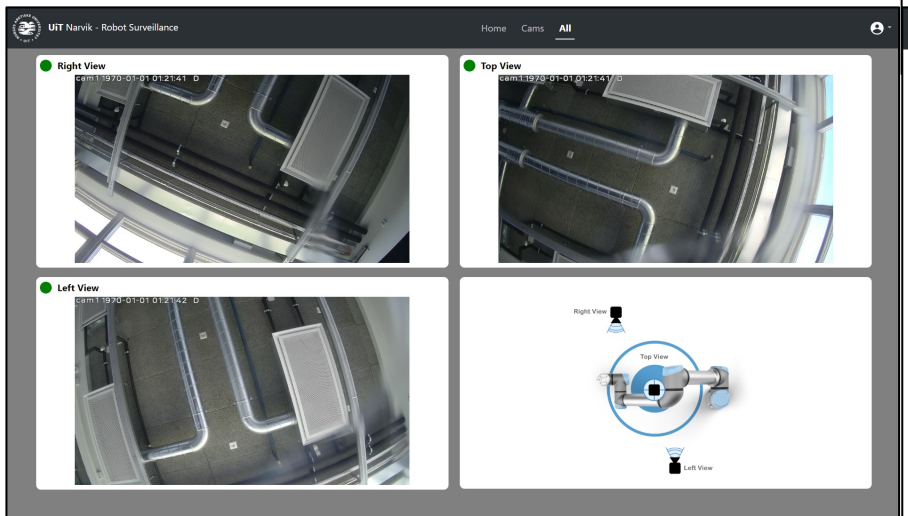
- Full web app for video surveillance
- Network architecture and servers

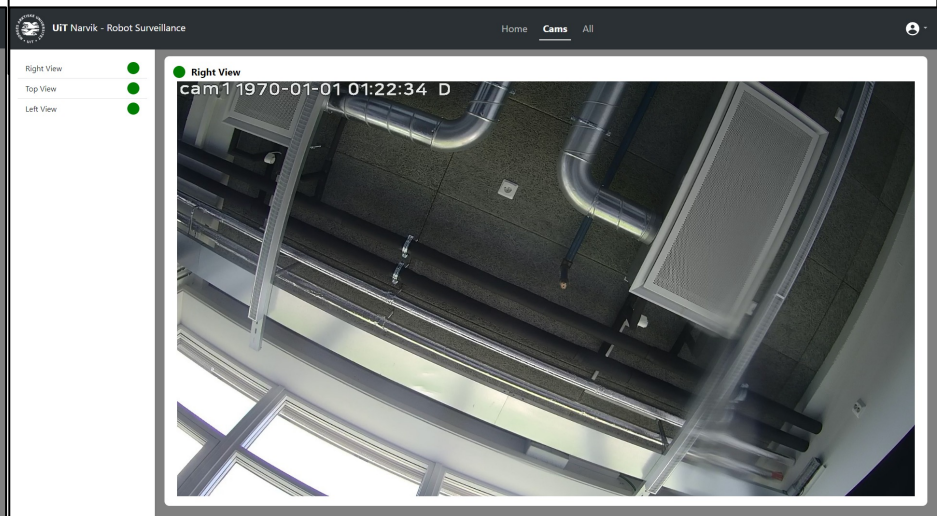
Skills and knowledge :

- Research project
- Autonomy
- Web development technologies










Thank you for your attention !