People's Democratic Republic of Algeria
Ministry of Higher Education and Scientific Research

University of Béjaia

Faculty of Exact Sciences
Informatics Department

This thesis is submitted in partial fulfillment of the requirements for the diploma of Master of Software Engineering

Thesis theme:

Intelligent Cloud Computing Solution for Real Time Object Recognition

Supervisors:
Dr. BELAID Ahror
Dr. AKILAL Abdellah

Authors:
AITOUAKLI Hichem
MEKHAZNI Fouad

Members of the Jury:
Dr. ALOUI Soraya
Dr. MOKTEFI Mohand

Academic year 2021/2022

ACKNOWLEDGEMENTS

Above all else, we would like to thank Allah for giving us the strength and his blessings throughout our lives, allowing us to complete to thesis.

We wish to express our sincerest gratitude to Dr. BELAID Ahror and Dr.AKILAL Abdellah, our supervisors, for providing us the opportunity to learn and gain valuable expertise that will hopefully serve us in the future and for sharing with us their knowledge and guiding us throughout this project.

Our gratitude goes also the the members of the jury who accepted to examine and evaluate our work.

Finally, we express our deepest gratitude to our family for their love and encouragement.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

ABBREVIATIONS

- **ML** : Machine Learning
- **MLOps** : Machine Learning Operations
- **IaaS** : Infrastructure as a Service
- **PaaS** : Platform as a Service
- **SaaS** : Software as a Service
- **FaaS** : Function as a Service
- **CPU** : Central Processing Unit
- **GPU** : Graphics Processing Unit
- **TPU** : Tensor Processing Unit
- **WebRTC** : Web Real Time Communication
- **SSH** : Secure Shell Protocol
- **UML** : Unified Modelling Language
- **UI** : User Interface
- **HTTP** : Hypertext Transfer Protocol
- **API** : Application Programming Interface
- **RFC** : Request For Comment

- **ICE** : Interactive Connectivity Establishment
- **STUN** : Session Traversal Utilities for NAT
- **NAT**: Network Address Translator
- **TURN** : Traversal Using Relays around NAT
- **SDP** : Session Description Protocol
- **AKS** : Azure Kubernetes Service
- **CI / CD** : Continuous Integration / Continuous Deployment
- **K8S** : Kubernetes
- **SDK** : Software Devlopment Kit
- **ANN** : Artificial Neural Network
- **DNN** : Deep Neural Network
- **YOLO** : You Only Look Once
- **LBPH** : Local Binary Pattern Histogram

INTRODUCTION

A common problem data scientist and machine learning experts are facing emerge from deploying a model in production, which is extremely difficult. As it is reported by[1], 87 percent of data science projects never make it into production.

These scientists put considerable time and effort into developing models and algorithms. However, their efforts are vain, their projects are abandoned due to the lack of production environments and tooling.

To help data scientists excel in their roles, companies don't only need to direct resources in the right direction, and also understand what machine learning models are all about. One possible solution is that leaders get some introductory training to data science themselves, so they can put this knowledge into practice at their companies. That's where MLOps comes in.

MLOps allows companies to easily deploy, monitor, and update ML models in production.

The goal of this project is to develop a cloud based web application that uses machine learning models to perform real time video processing.

CHAPTER I)

MACHINE LEARNING OPERATIONS

1 - Introduction

In this chapter, we will describe the evolution software development and its infrastructures, the challenges of traditional software development, the impact of machine learning in software development, MLOps and its workflow.

2 - The evolution of software development related infrastructure

There has been a rise in software applications since the coming of the modern internet age,ranging from operating systems such as Windows 95 to the Linux operating system and websites such as Google and Amazon, which have been serving the world (online) for over two decades. This has resulted in a culture of continuously improving services by collecting, storing, and processing a massive quantity of data from user interactions. These developments have been for long time shaping the evolution of IT infrastructure and software .

Resource sharing via networks that provide access to on-demand infrastructure, services, platforms and applications is quickly and increasingly becoming important, at the expense of sharing it through hardwired connections.[2]

3 - Cloud Computing

Cloud Computing is a model for enabling convenient network access to a shared set of configurable computing resources hosted on the internet such as servers, databases, software, virtual storage, and networking, among others. These resources can be rapidly provisioned and released with minimal management effort or service provider interaction.

a) Types of cloud services

All software, infrastructure, platforms and technologies that are accessible to users over the Internet, without the need to download additional software, can be considered cloud services, including the following "aaS" (as-a-Service) solutions:

- **IaaS** (Infrastructure-as-a-Service): provides users with networking, computing and storage resources.
- **PaaS** (Platform-as-a-Service): provides users with a platform on which to run applications, as well as the computing infrastructure needed to run them.

- **SaaS** (Software-as-a-Service): primarily provides users with a cloud application, as well as the platform on which it runs, in addition to the infrastructure underlying the platform.
- **FaaS** (Function-as-a-Service): is an event-driven execution model, allows developers to create, execute and manage application packages as functions, without having to worry about maintaining the infrastructure.[3]

The following figure[4] shows the different types:

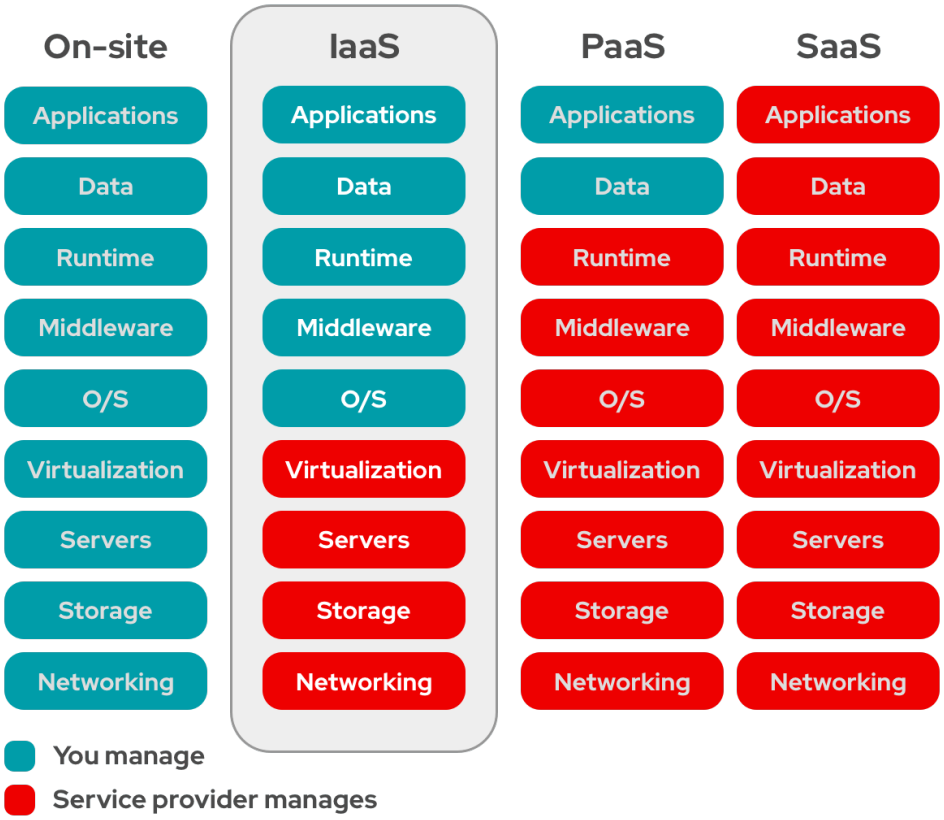


Figure 1.1: "as-a-Service" infrastructures per type

b) Machine learning on the cloud

Machine learning is one of the most demanded technology that companies chase after. Cloud computing can solve a lot of problems for machine learning analyists and engineers:

- Requires little to no knowledge of machine learning and data science
- Saves infrastructure cost computing.
- Availability of power resources such as CPUs and GPUs/TPUs.
- Resources are scalable without changing code or environment[5].

4 - The challenges of traditional software development

Software processes have progressed immensely since software engineers began to follow a disciplined flow of activities to improve quality and productivity during development. Various software development process models, methodologies, methods and/or practices have been proposed, adopted or implemented.

For many years, there has been conflict over whether to follow a completely traditional ("classical") model or become more agile. Each approach has its strengths and weaknesses, and each has its proponents and detractors However, the current diversity of software projects and the advancement of technology have led to debates about what types of software process approaches are most effective in the context.

5 - Machine learning engineering

Machine learning (ML) is a branch of artificial intelligence (AI) and computer science which focuses on the use of data and algorithms to imitate the way that humans learn, gradually improving its accuracy.

ML is a very profitable but using it to solve real world problems is very complex due to the sheer amount of algorithms, tools and activities involved in building models, which makes it intimidating for companies. To solve this complexity problem, Machine Learning Engineering applies a system that uses a set of tools, processes and methodologies that aims to optimize the chances of abandoning a ML model project.

Machine learning is an important component of the growing field of data science. Through the use of statistical methods, algorithms are trained to make classifications or predictions, uncovering key insights within data mining projects. These insights subsequently drive decision making within applications and businesses, ideally impacting key growth metrics. As big data continues to expand and grow, the market demand for data scientists will increase, requiring them to assist in the identification of the most relevant business questions and subsequently the data to answer them.[6]

6 - MLOps and its workflow:

MLOps, an acronym for Machine Learning Operations, is one of the most popular trends in the industry today. Sometimes referred to as ModelOps, it is an engineering discipline whose workflow we will see behind it and the steps that are in the workflow pipeline:

a) What is MLOps:

MLOps is an emerging method that allows to fuse machine learning with software development.This is done by integrating multiple domains as MLOps combines with ML engineering, DevOps, and data engineering.
It aims to build, deploy, and maintain ML systems in production reliably and efficiently. The figure[6] below describes MLOps and its components:

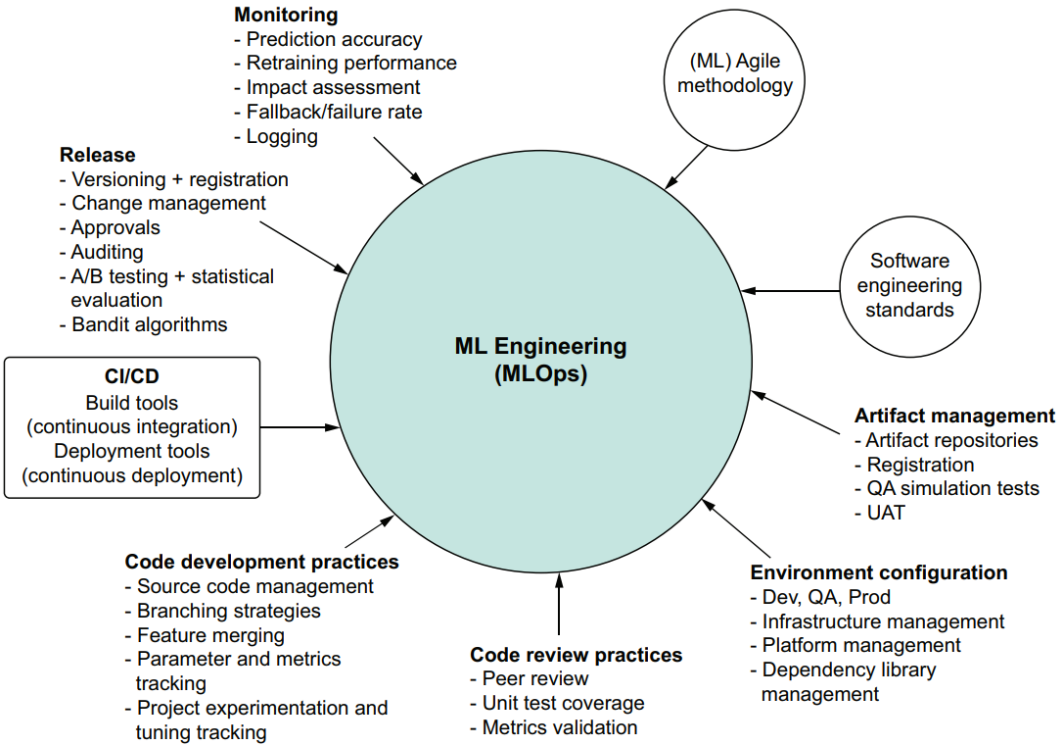


Figure 1.2: MLOps Components

b) Workflow:

Figure[7] shows a generic MLOps workflow. It focuses on optimizing ML solutions or build proofs of concepts while being modular and flexible:

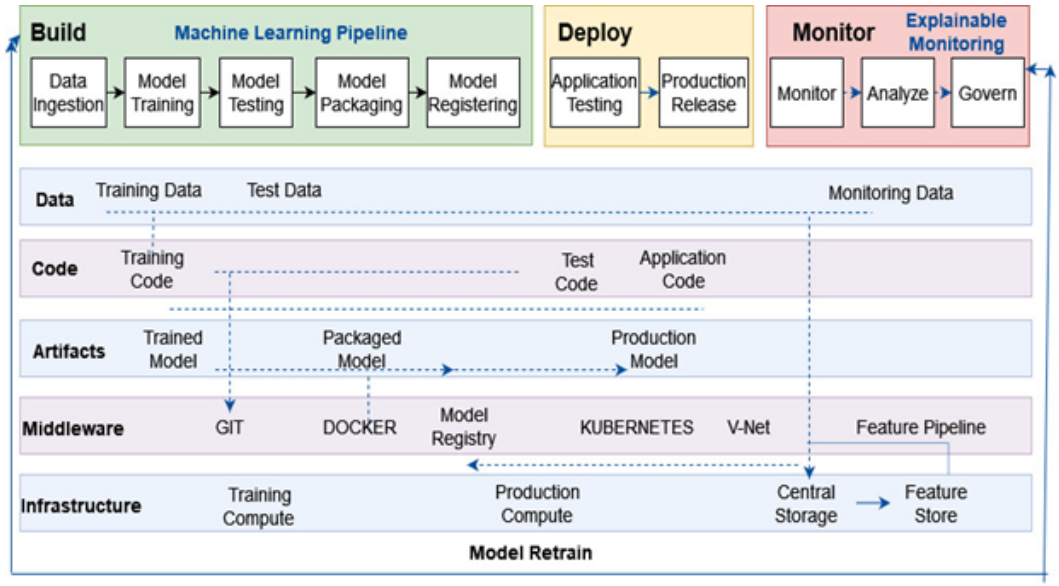


Figure 1.3: MLOps workflow

The workflow is split into two modules or layers: the upper layer is the MLOps pipeline and the lower layer are the drivers.

The pipeline is enabled by drivers such as data, code, artifacts, middleware and infrastructure.

By using this pipeline, a company can quickly prototype, test, validate and deploy machine learning models to production at scale very efficiently.

7 - Conclusion

In this chapter, we mainly talked about cloud computing and types of cloud services, MLOps and its workflow, and machine learning.
In the next chapter, we will discuss the project requirements and the various configurations needed to implement the anticipated solutions.

CHAPTER II)

CONCEPTION AND DEVELOPMENT

1 - Introduction

The objective of this chapter is to collect and analyze all assorted ideas related to defining a systems, its requirements with respect to the project contract. In short, this chapter is going to provide a detailed overview of the system's behaviour and dynamics.

We will establish and analyze the functional and non-functional requirements, as well as emphasize the behavior, dynamics, and main concept of the system. We will also try to explain various design decisions and different phases of development, namely how the WebRTC connection was done and how we deployed the system into the Cloud.

2 - Requirements analysis:

We were able to establish and analyze the functional and non-functional system requirements, which can be cited as:

a) Functional requirements:

These are the basic functionalities that the system should offer to the end users. They must be incorporated in the final product:

1) Real time / Webcam Face Capture detection:

The system should allow face detection in real time using a webcam video stream, or a single capture.

2) Real time / Webcam Gender Capture detection:

Using the previous functionality, the system must deduce a users gender from the detected face.

3) Real time / Webcam Object Capture detection:

Similarly, the system must have the necessary built-in machine learning models that allows the detection of objects.

4) Upload custom machine learning model:

The most important feature of the system and the purpose of this project, is to offer an interface to edit/input customised models for the end user (Machine learning expert). Once uploaded, a

new container must be created solely for the model.

5) Remote SSH web console:

The system should include a terminal interface in order to allow machine learning experts or administrators to access a remote SSH terminal.

b) Non-functional requirements:

The final product must adhere and satisfy the following conditions as part of the project contract. By order of priority, they are:

1) Performance:

Since the system offers a real time processing functionality, it must take into account areas such as latency, load and resource utilisation such as CPU and GPU capacity (some machine learning models rely heavily on GPUs).

2) Scalability:

The system must be able to accommodate with a large and ever-increasing user base.

3) Portability:

The system must be able to run efficiently on multiple devices.

4) Flexibility:

The system must adapt to different user expectations, and be able to process a myriad of machine learning models.

5) Security:

The SSH web console must be protected by a authentication system.

6) Robustness:

The system should function correctly even in case of invalid inputs and under heavy server stress.

3 - Design and conception:

The diagrams shown in this sections are all made in accordance to the UML (Unified Modelling Language). Their purpose is to present a graphical overview of the functionality provided by the system in terms of actors, goals, and any dependencies between those use cases.

a) Use case diagram

The figure below represent the use case of the system.

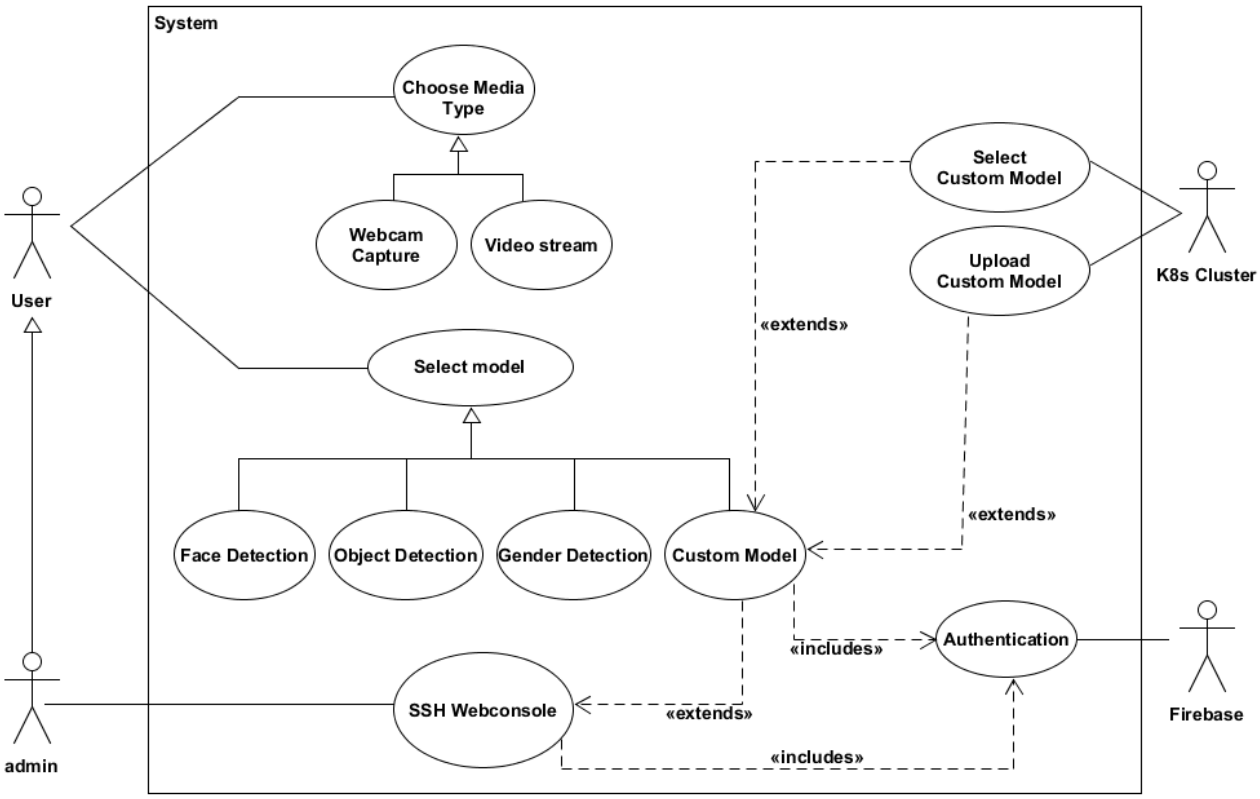


Figure 2.1: Use case diagram of the system

b) Use cases textual description:

Authentication:

Table II.1 shows the textual description of the authentication use case.

Use case title :		Authentication
Actors:		User, admin.
Pre-conditions:		- Access to web application
Normal Flow:	**Description:**	- The user clicks the authentication button. - Authentication UI is rendered - The user enters his email/password. - The user click on the login button
	Postconditions:	Registered / Logged in.
Alternative flows and exceptions:		Invalid email/password, connection loss, backend internal server error

Table II).1: Textual Description of the Authentication use case

Real time detection:

The table below show textual Description of the Realtime Detection use case.

Use case title :		Realtime Object Detection
Actors:		User
Pre-conditions:		- Access to web application and selected the video stream media type and desired ML model - Webcam connected
Normal Flow:	**Description:**	The user establishes a webrtc connection with the backend and sends a video stream with its metadata. The backend processes each frame of the video stream and sends it back processed.
	Postconditions:	Video stream is processed in real time.
Alternative flows and exceptions:		Connection loss, backend internal server error, webcam unavailable.

Table II).2: Textual Description of the Realtime Detection use case.

Webcam Capture Face Detection:

The table hereafter presents textual Description of the Webcam Capture Detection use case.

Use case title :		Webcam Capture Detection
Actors:		User
Pre-conditions:		- Access to web application and selected the webcam media type and desired ML model - Webcam connected
Normal Flow:	**Description:**	- The user opens the webcam and captures the desired image to process. - The image is converted into a base64 blob and then sent to the backend via HTTP. - The backend converts the image into a numpy frame then processes it with a ML model. - The frame is converted back into a base64 image blob and then sent back to the front-end via HTTP.
	Postconditions:	A processed image is rendered on the UI.
Alternative flows and exceptions:		Connection loss, backend internal server error, webcam unavailable.

Table II).3: Textual Description of the Webcam Capture Detection use case.

Upload custom model:

The table below show textual Description of the Upload custom model use case.

Use case title :		Upload custom model
Actors:		User, admin.
Pre-conditions:		- Access to web application. - User is authenticated.
Normal Flow:	**Description:**	- User selects the custom model transformation type, - Custom model editor UI is rendered. - User selects the model type. - User either edits the configuration file or inputs his own. - User inputs the weights file. - User click on the deploy button. - The two files are sent to the back-end. - A new docker image is built and containerized with the custom model - The IP address of the container is sent to the database along with ID of the user.
	Postconditions:	Model deployed.
Alternative flows and exceptions:		Invalid file extension/type, connection loss, backend internal server error

Table II).4: Textual Description of the Upload custom model use case

Remote SSH web console

The table below presents textual Description of the remote SSH web console use case.

Use case title :		Remote SSH web console
Actors:		User, admin.
Pre-conditions:		- Access to web application. - User is authenticated.
Normal Flow:	**Description:**	- User clicks the ssh terminal button. - User is then redirected to the web console page - The front-end establishes a websockets connection with the back-end server. - A web console user interface is rendered. - User enters SSH key.
	Postconditions:	Remote access to the cluster's terminal
Alternative flows and exceptions:		SSH key invalid, connection loss, backend internal server error

Table II).5: Textual Description of the remote SSH web console use case

c) Class Diagram

The figure below represents the class diagram of the system. It shows the different components needed for the deployment of a machine learning model.

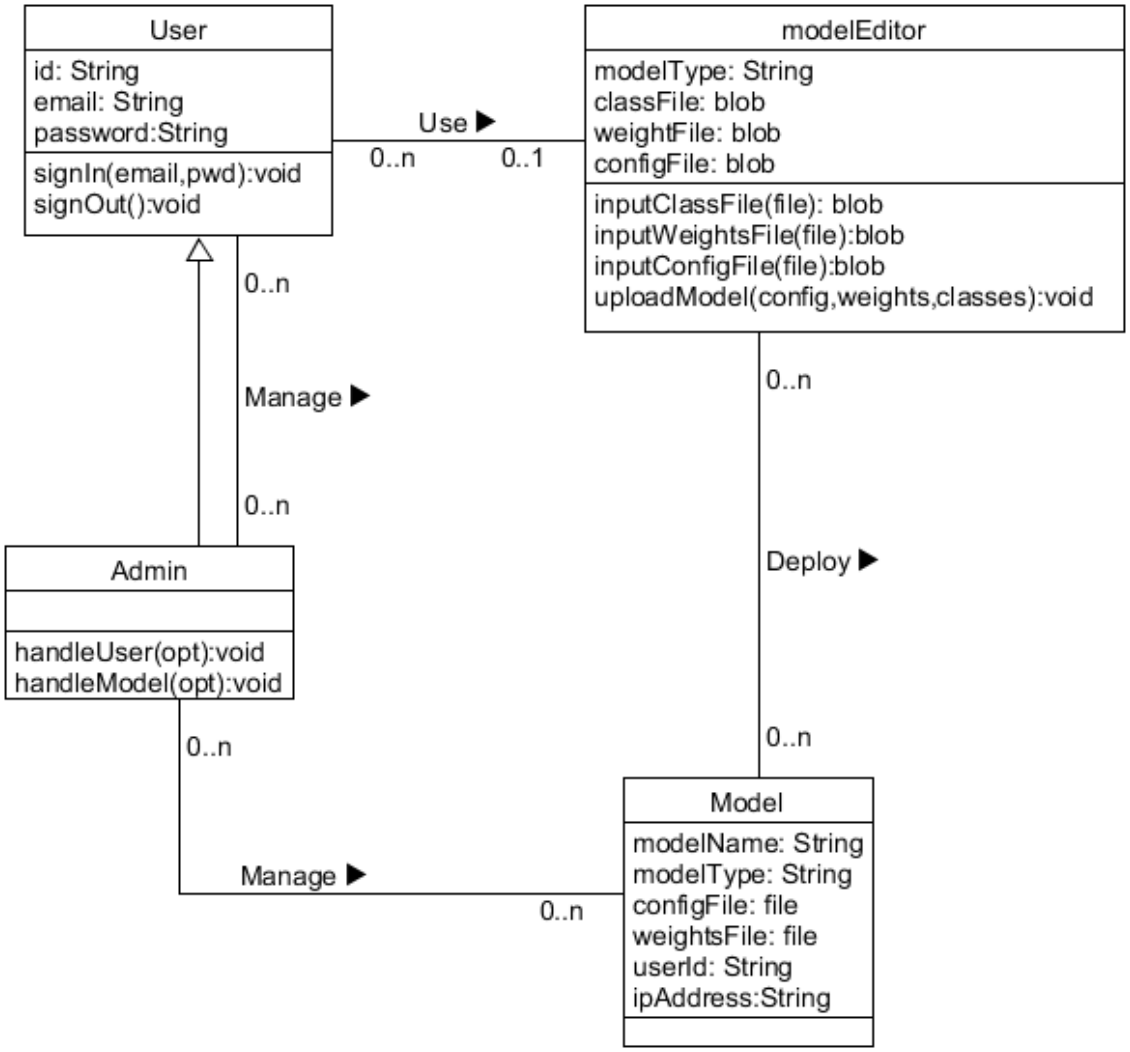


Figure 2.2: Class diagram of the system

d) Sequence Diagrams

Due to the numerous sequence diagrams included in the system's conception prefer to describe only these following 4 diagrams:

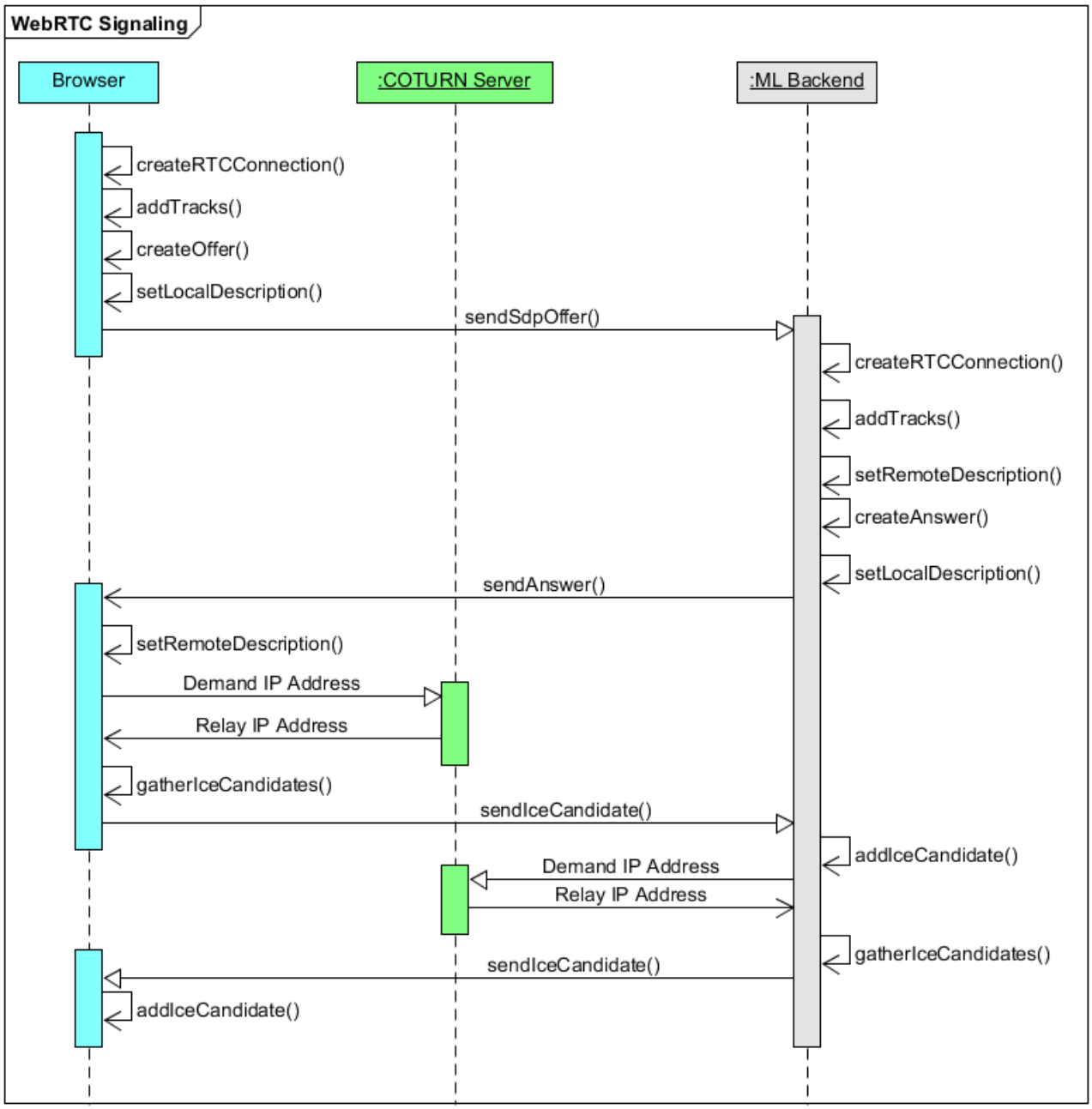


Figure 2.3: WebRTC Signaling Sequence Diagram

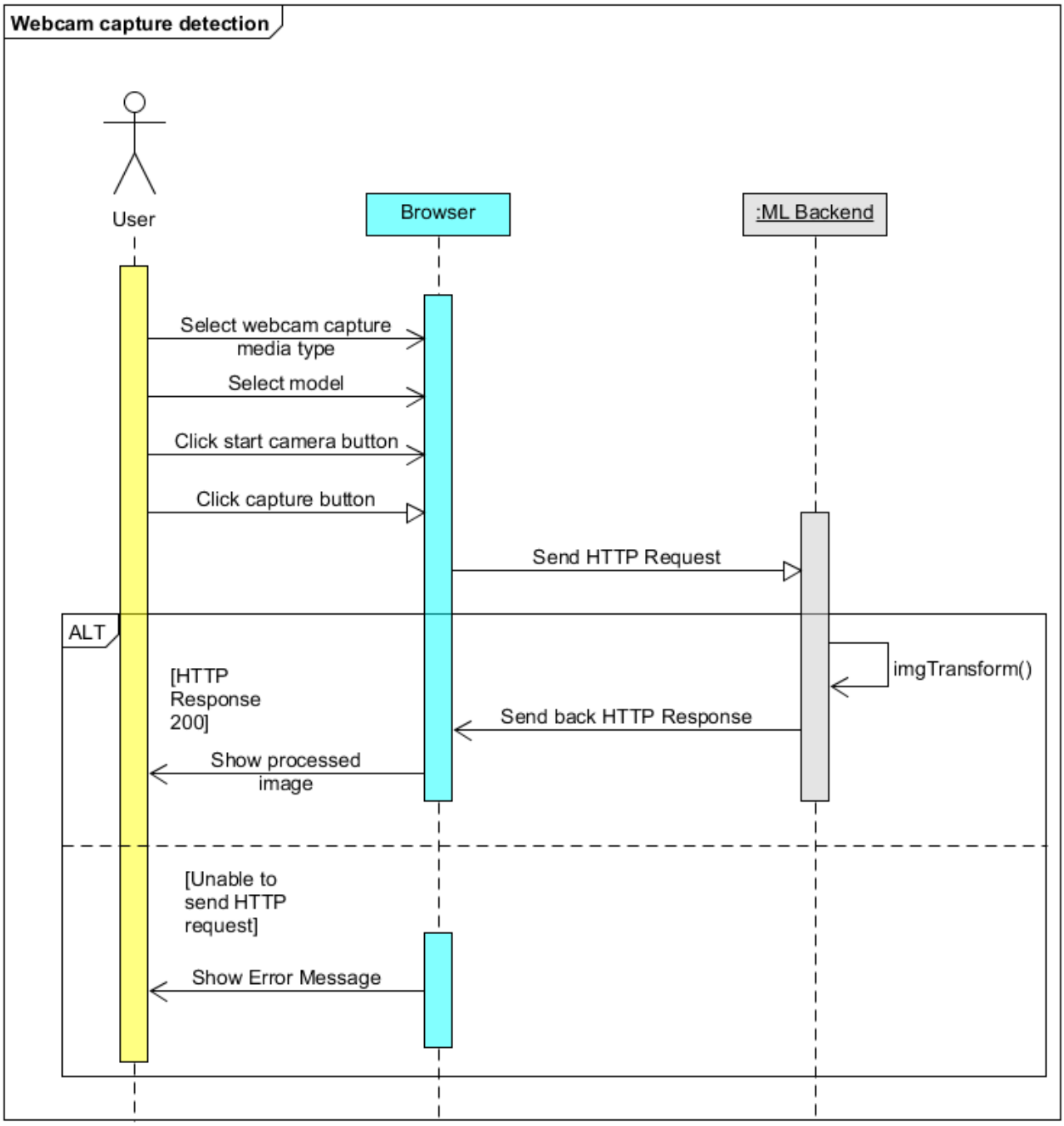


Figure 2.4: Webcam capture detection sequence diagram

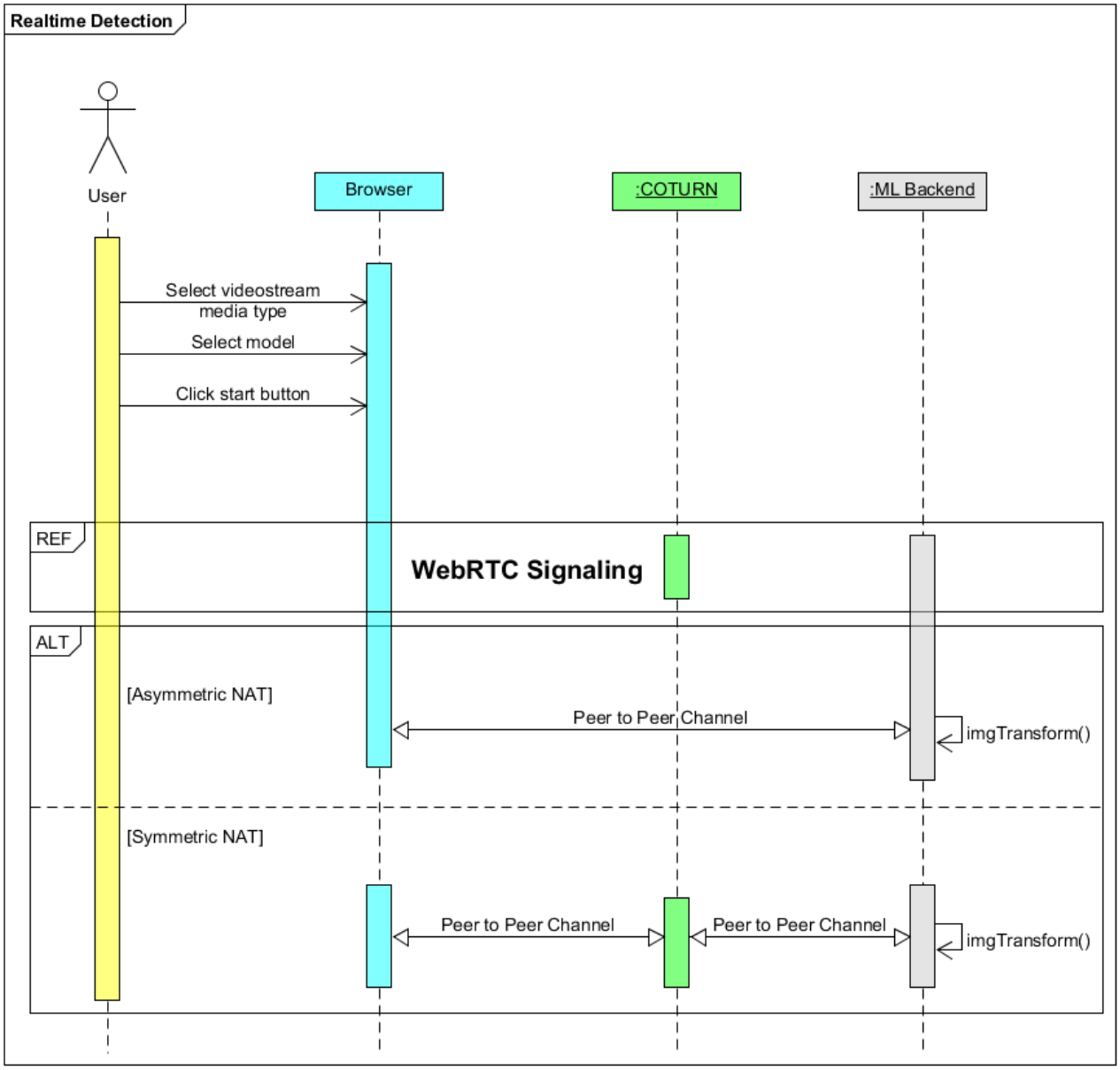


Figure 2.5: Real time detection sequence diagram

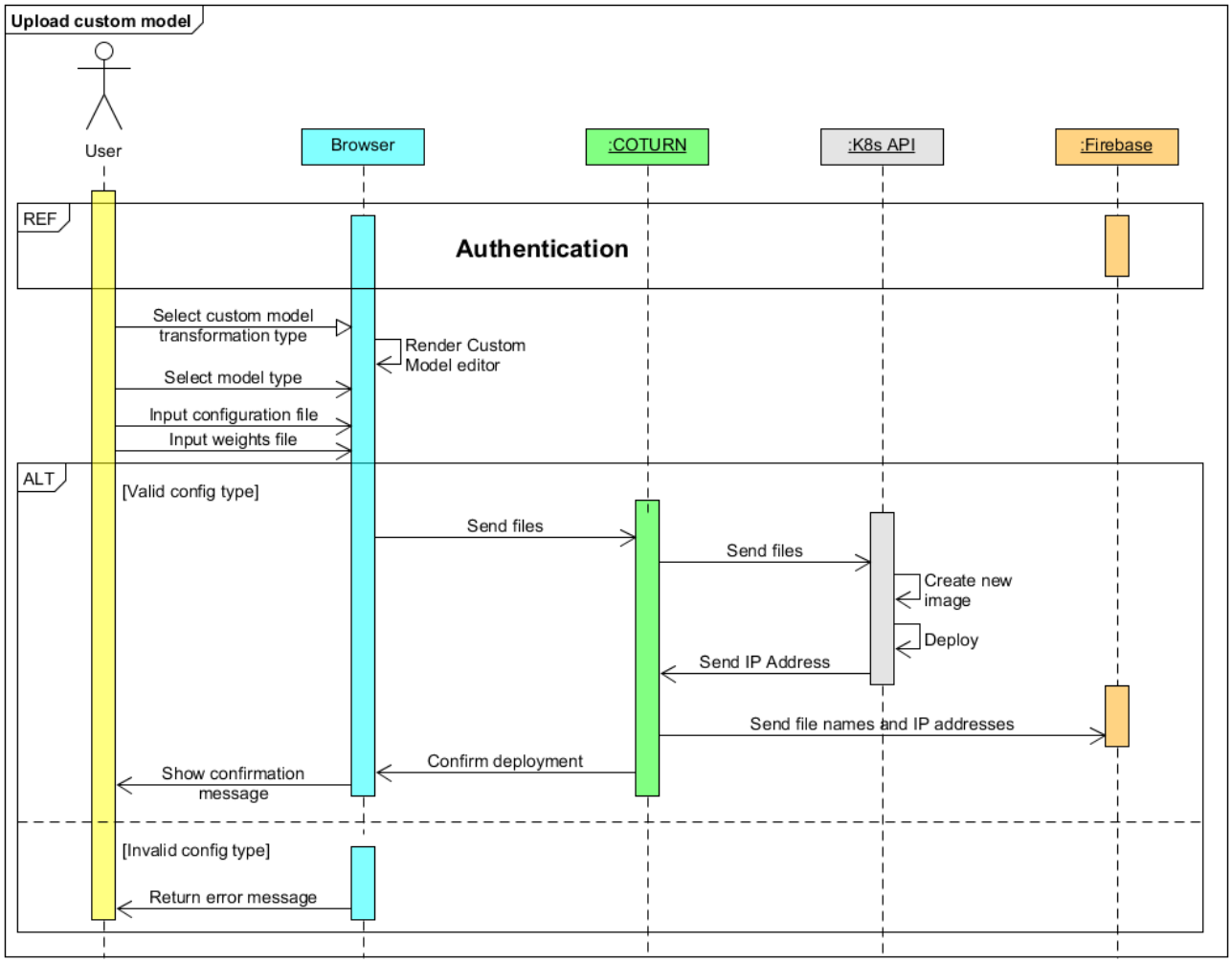


Figure 2.6: Upload custom model sequence diagram

e) The system's activity diagram

The user will first access the web page by entering a URL in a browser and then presented with a choice on how to proceed. This figure below shows all the possible paths a user can take upon accessing the web application.

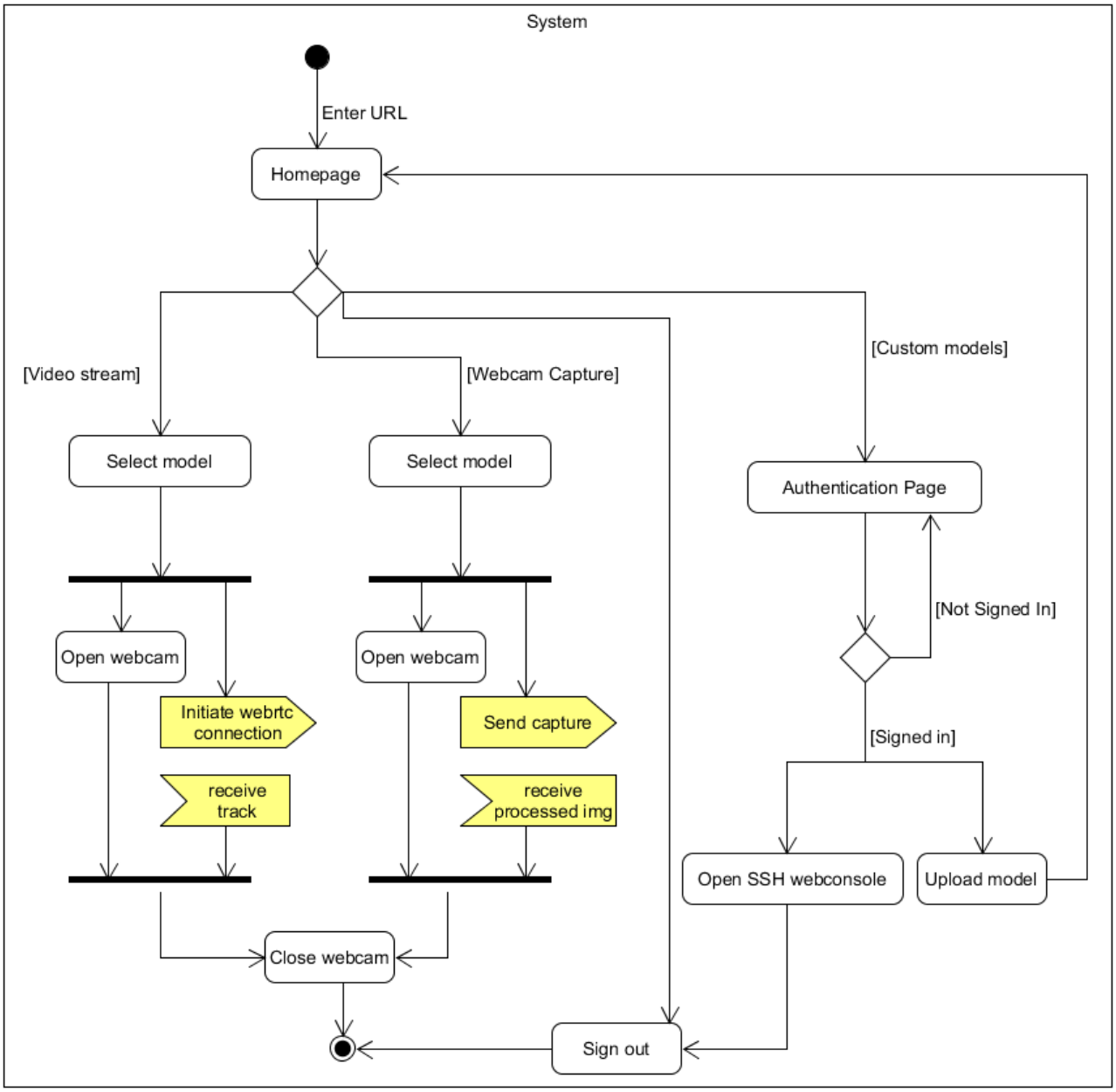


Figure 2.7: Activity diagram of the system

4 - Implementation

The three main paradigms used during the development of the system are:

- WebRTC data streaming
- Docker containerization
- Deployment on a cloud-based Kubernetes cluster

The web application will use these technologies to send video frames to a machine learning processing back end container which will process each frame with a selected model and send them back in real time. A proof of concept will be developed in order to test this feature before deploying a managed Kubernetes cluster in Microsoft Azure.

a) WebRTC Implementation

WebRTC is used to add real-time communication capabilities to the application. It works on top of an open standard. It supports video, voice, and generic data to be sent between peers. It allows developers to build powerful voice- and video-communication solutions.This technology is available on all modern browsers as well as on native clients for all major platforms. The technologies behind WebRTC are implemented as an open web standard and available as regular JavaScript APIs in all major browsers.[8]

The WebRTC API is built upon these following protocols:

ICE protocol:

The ICE Protocol (Interactive Connectivity Establishment) (RFC5245)[9] is a framework allows web browsers to generate media traversal candidates to connect with peers. It has two main roles: gathering candidates and finding the most efficient path for two peers to communicate. ICE candidates contains the information about the methods available for the peer to use to make a connection, like the IP address of the peer and available ports. ICE uses STUN and/or TURN servers to accomplish this, as described below.

STUN protocol:

The Session Traversal Utilities for NAT (STUN) protocol (RFC5389) [10] allows a host application to discover the presence of a network address translator on the network, and in such a

case to obtain the allocated public IP and port tuple for the current connection. To do so, the protocol requires assistance from a configured, third-party STUN server that must reside on the public network.

NAT:

The IP Network Address Translator (NAT) (RFC1631) [11] is a router/firewall function that is used to assigns a public IP address to a device/group of devices. NAT implementations may vary in their specific behavior in various addressing cases and their effect on network traffic, notably:

- Normal (Full Cone) NATs (or Asymmetric NATs): all requests sent from the same internal IP address and port are assigned to the same external IP address and port.
- Symmetric NATs : all requests coming from the same internal IP address and port and going to a specific destination IP address and port are mapped to a unique external IP address and port

Symmetric NATs makes establishing a WebRTC difficult: if a device behind a symmetric NAT were to send a request to two different STUN servers, it will receive the same IP address but two different ports, which makes this specific type of NAT's port mapping behaviour inconsistent.

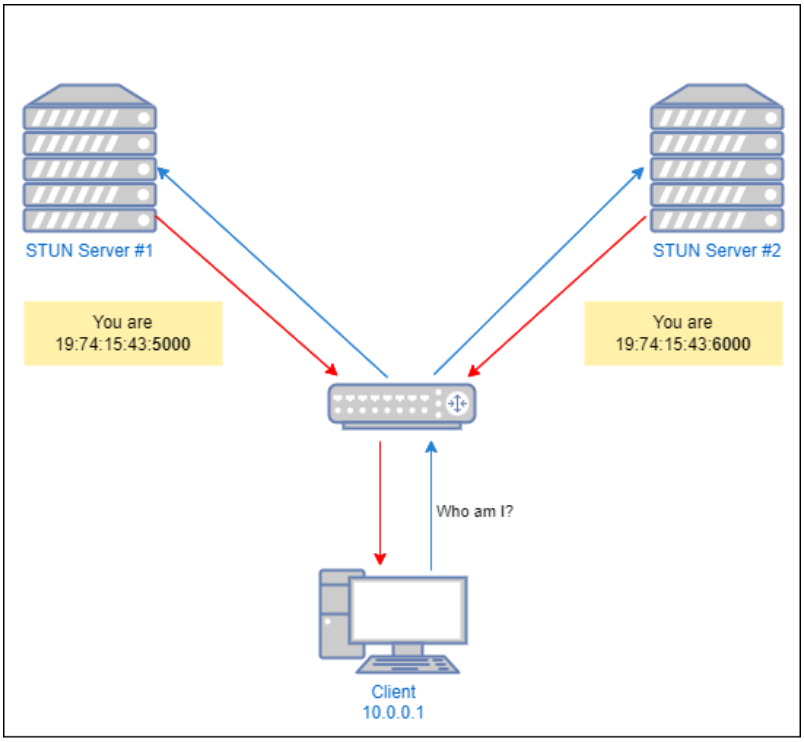


Figure 2.8: The problem with symmetric NATs

TURN protocol:

The Traversal Using Relays around NAT (TURN) protocol (RFC5766)[12] allows a host behind a NAT to obtain a public IP address and port from a relay server residing on the public Internet. Thanks to the relayed transport address, the host can then receive media from any peer that can send packets to the public Internet.

SDP standard:

The Session Description Protocol (SDP) (RFC4566)[13] provides a standard representation of media details, transport addresses, and other session description metadata required in order to initiate multimedia teleconferences, voice-over-IP calls, streaming video or other media.

It is only a format for session description and not a transport protocol, as it is inteded to be general purpose so that it can be used in a wide range of network environments and applications.

```
v=0
o=- 3865081140 3865081140 IN IP4 0.0.0.0
s=-
t=0 0
a=group:BUNDLE 0
a=msid-semantic:WMS *
m=video 57023 UDP/TLS/RTP/SAVPF 127 121 108 109
c=IN IP4 172.17.4.218
a=sendrecv
a=extmap:2 http://www.webrtc.org/experiments/rtp-hdrext/abs-send-time
a=extmap:9 urn:ietf:params:rtp-hdrext:sdes:mid
a=mid:0
a=msid:9be77b15-88eb-43be-853b-eff747966c6f 8f7901de-2ca7-45bd-9dae-ace5444c495e
a=rtcp:9 IN IP4 0.0.0.0
a=rtcp-mux
a=ssrc-group:FID 1197212952 3165833727
a=ssrc:1197212952 cname:5c5e3c6c-8d49-4202-b411-6558357a4c82
a=ssrc:3165833727 cname:5c5e3c6c-8d49-4202-b411-6558357a4c82
a=rtpmap:127 H264/90000
a=rtcp-fb:127 nack
a=rtcp-fb:127 nack pli
a=rtcp-fb:127 goog-remb
a=fmtp:127 packetization-mode=1;level-asymmetry-allowed=1;profile-level-id=42001f
a=rtpmap:121 rtx/90000
a=fmtp:121 apt=127
a=rtpmap:108 H264/90000
a=rtcp-fb:108 nack
a=rtcp-fb:108 nack pli
a=rtcp-fb:108 goog-remb
a=fmtp:108 packetization-mode=1;level-asymmetry-allowed=1;profile-level-id=42e01f
a=rtpmap:109 rtx/90000
a=fmtp:109 apt=108
a=candidate:77fd886e97df2d3cec8c2d09d59d3cde 1 udp 2130706431 172.17.4.218 57023 typ host
a=candidate:c8745481d4561f3e6f94acd2634b7c5b 1 udp 1694498815 3.251.80.188 57023 typ srflx raddr 172.17.4.218 rport 57023
a=end-of-candidates
a=ice-ufrag:Ffhu
a=ice-pwd:LhT9vcsP4ZZCWqeOQz1oFB
a=fingerprint:sha-256 3D:6E:49:39:D6:BD:3F:0A:F9:B7:5D:07:E9:46:13:4E:D2:F5:42:0A:B9:B7:D7:3B:42:22:46:99:57:70:FC:7D
a=setup:active
```

Figure 2.9: Example of a SDP answer

WebRTC transmission behind an asymmetric NAT:

The client first makes a request to a STUN server in order to obtain its public IP address, and the generates an offer in SDP and send it to the other peer, which in turn this other peer will send back an answer containing its own local description in SDP format.

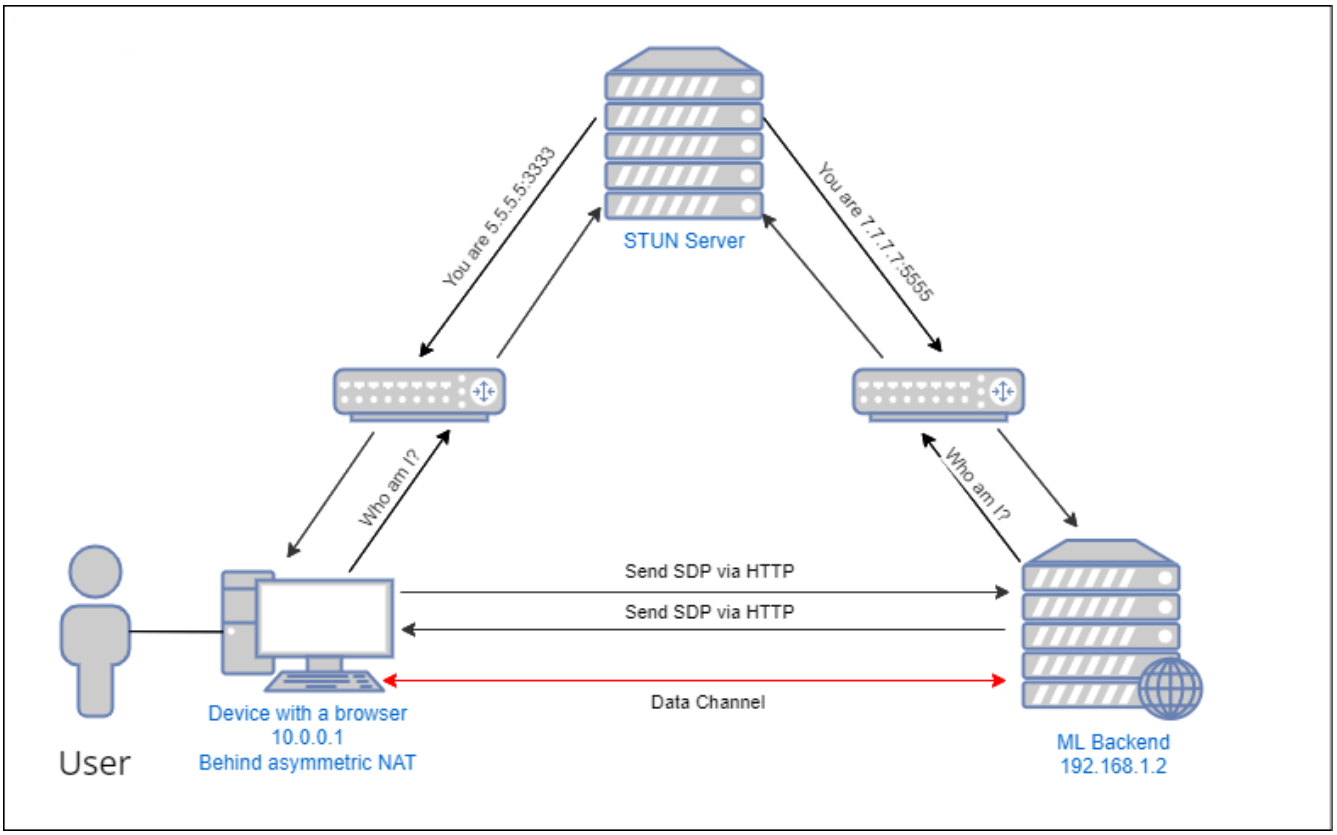


Figure 2.10: Diagram of the WebRTC transmission behind an asymmetric NAT

WebRTC transmission behind a symmetric NAT:

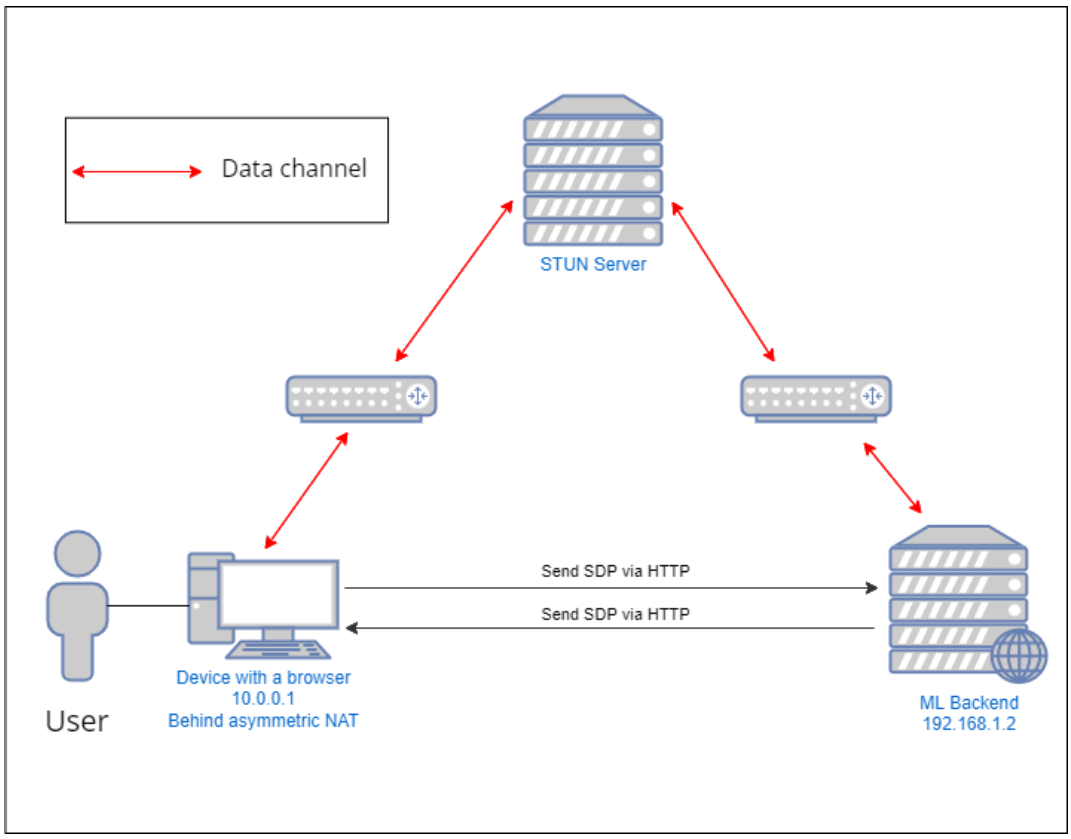


Figure 2.11: Diagram of the WebRTC transmission behind an symmetric NAT

b) Stage 1: Proof of Concept

A quick PoC is developed in a typical case using pre-built machine learning models to showcase and validate the use case and prove that the concept is feasible using WebRTC.

In our hypothetical use case, a prototype with Python FastAPI app, is deployed using Heroku, developed that does the following:

- Fetch video stream from the prototype Nextjs front-end, deployed on Vercel, using WebRTC.
- Execute OpenCV function that uses pre-built machine learning models to process each frame of the video stream
- Send back the processed frames to the front-end.

The proof of concept deployment diagram:

This diagram shows the execution architecture of the system's proof of concept

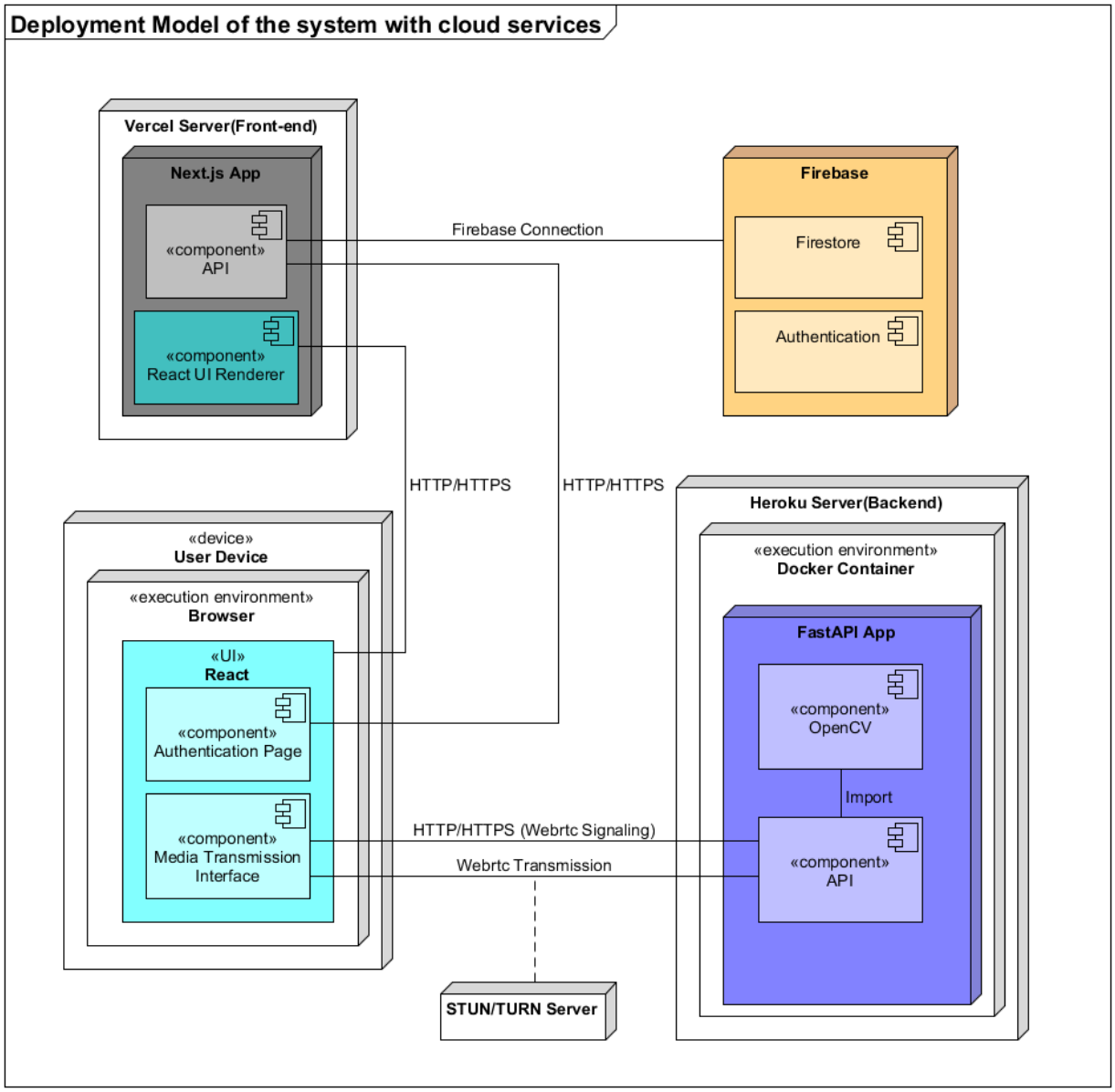


Figure 2.12: Deployment diagram of the deployed FastAPI Heroku application

c) Stage 2: Kubernetes cluster deployment

For the purpose of testing the application in a K8s (Kubernetes) environment, we used the Azure Kubernetes Cloud service which allowed us to quickly deploy a multi container application composed of a Docker container running a Next.js front-end, and another container running a FastAPI application and a container with a STUN/TURN service and a Node.js application that acts as a middleman between the front-end and back-end.

To create and deploy a AKS (Azure Kubernetes Service) cluster, we used the Azure CLI that can be installed in Windows, macOS or in a Docker container.

We will also use the AKS Deployment Center to generate Github action workflows to create deployment pipeplines for the Github source code into the AKS cluster. The following figure[14] describes the order of this CI/CD pipeline:

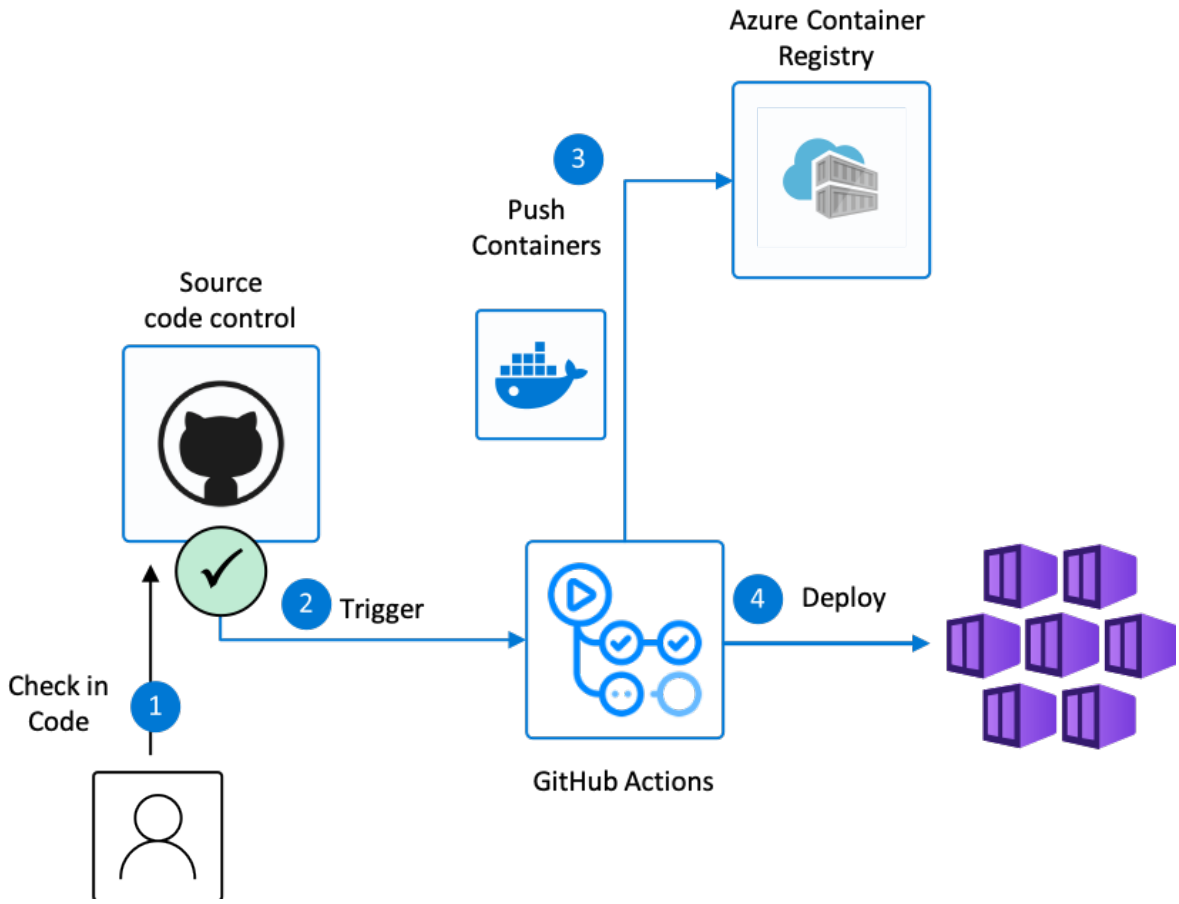


Figure 2.13: AKS Deployment Center Workflow

Deployment diagram for the Azure Kubernetes Service cluster

This diagram shows the architecture of the system deployed in a Kubernetes cluster

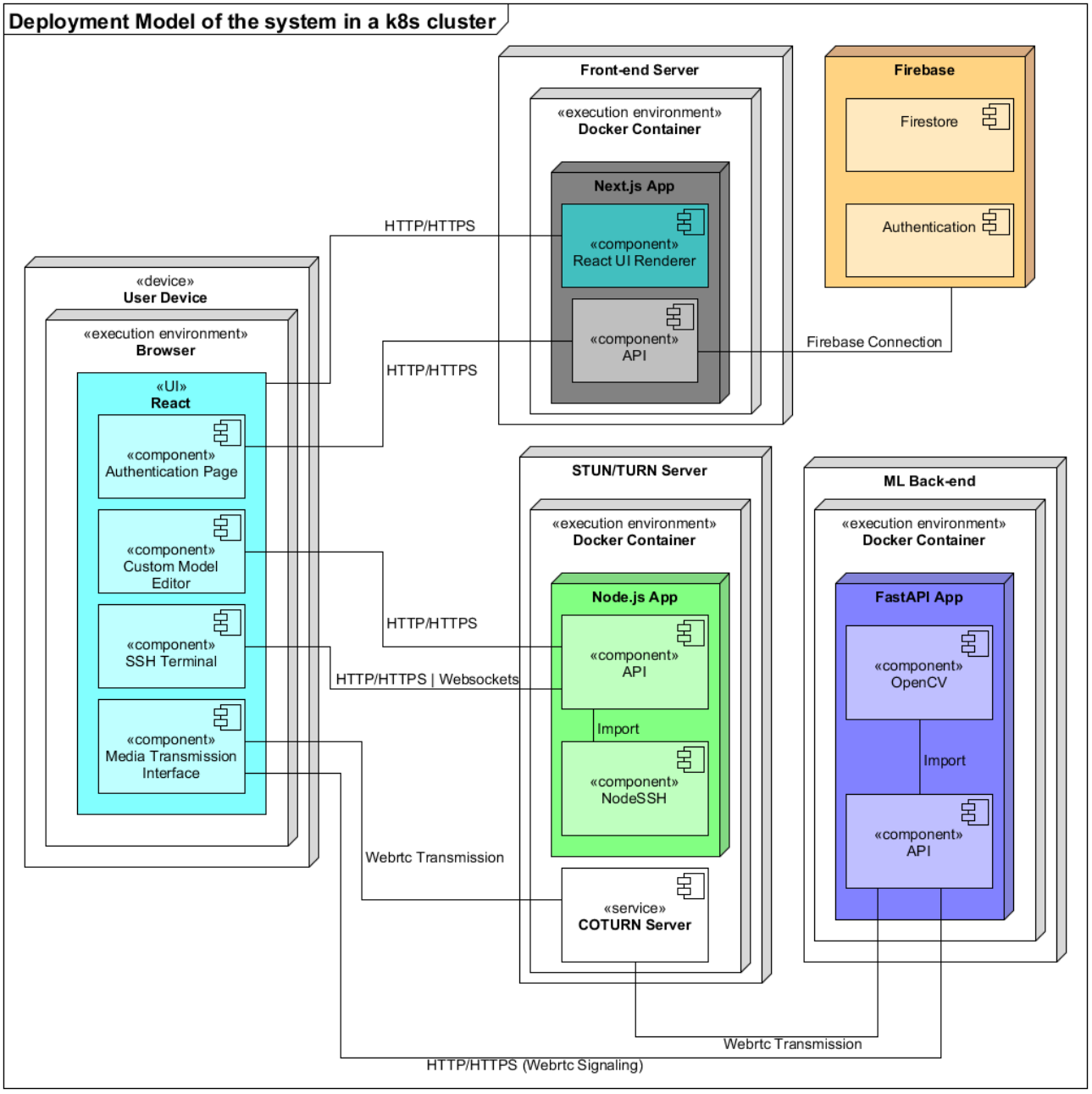


Figure 2.14: Deployment diagram of the application deployed in a K8s cluster

5 - Deployment configurations:

In this section, we will show all the files needed to deploy 3 services(Back-end, STUN/TURN server and front-end) in a Kubernetes cluster. These files will include Dockerfile files, which are text documents containing all the commands a user could call on the command line to assemble an image[15], and the Kubernetes YAML files needed to deploy the services in the cluster.

a) Back-end configuration:

Since there are a lot of python packages that need to be downloaded, we decided to use a minimal python image. The requirements.txt file may be found in Appendix A.

Dockerfile:

```
# Python base image
FROM python:3.10-slim-buster

# copy source code
COPY . /app
WORKDIR /app

# Copy python libraries requirements
COPY requirements.txt requirements.txt

# update linux packages
RUN apt-get update

# Install additional linux packages in order to compile some python librarires
RUN apt-get install -y --no-install-recommends gcc libsasl2-dev python-dev \
    libldap2-dev libssl-dev libsnmp-dev \
    && rm -rf /var/lib/apt/lists/*

# Install python libraries
RUN pip install -r requirements.txt

# Delete the additional linux packages since they are no longer needed for compiling
RUN apt-get purge -y --auto-remove gcc libsasl2-dev python-dev libldap2-dev \
```

```
    libssl-dev libsnmp-dev

# Expose port
EXPOSE 80

# Launch the fastAPI server
CMD ["uvicorn","server:app", "--host", "0.0.0.0","--port", "80" ]
```

Kubernetes deployment yaml file:

```
apiVersion : apps/v1
kind: Deployment
metadata:
  name: "recogaks-backend"
spec:
  replicas: 2
  selector:
    matchLabels:
      app: "recogaks-backend"
  template:
    metadata:
      labels:
        app: "recogaks-backend"
    spec:
      containers:
        - name: "recogaks-backend"
          image: "recogacr.azurecr.io/recogaks"
          ports:
          - containerPort: 80
```

Kubernetes service yaml file:

```
apiVersion: v1
kind: Service
metadata:
    name: "recogaks-backend"
```

```
    labels:
        app: "recogaks-backend"
spec:
    type: LoadBalancer
    ports:
    - port: 80
      targetPort: 80
      protocol: TCP
      name: http
    selector:
        app: "recogaks-backend"
```

b) STUN/TURN server configuration:

Dockerfile:

The default port for sending (or listening to) STUN/TURN requests is 3478.

```
# Node.js image
FROM node:18

# Update and upgrade linux packages
RUN apt-get update
RUN apt-get upgrade -y

# Install necessary packages for compiling and building
RUN apt-get install build-essential checkinstall zlib1g-dev gcc -y
RUN apt-get upgrade libstdc++6

# Install the openSSL package
RUN apt-get -y install openssl

# Install the coturn package
RUN apt-get install -y coturn && apt-get clean && rm -rf /var/lib/apt/lists/* /tmp/* /var/tmp/*

# Define environment variables
ENV TURN_PORT 3478
ENV TURN_PORT_START 10000
```

```
ENV TURN_PORT_END 20000
ENV TURN_SECRET mysecret
ENV TURN_SERVER_NAME coturn
ENV TURN_REALM recog.server

# Copy source code
COPY . .

# Install node packages
RUN npm install

# Change script bash permissions
RUN chmod +x start_coturn.sh

# Expose ports
EXPOSE 3478

# Execute the script
CMD [ "./", "start_coturn.sh" ]
```

Kubernetes deployment yaml file:

```yaml
apiVersion : apps/v1
kind: Deployment
metadata:
  name: "recogaks-coturn"
spec:
  replicas: 2
  selector:
    matchLabels:
      app: "recogaks-coturn"
  template:
    metadata:
      labels:
        app: "recogaks-coturn"
    spec:
      containers:
        - name: "recogaks-coturn"
          image: "acrecog1.azurecr.io/recogaks"
          ports:
          - containerPort: 3478
```

Figure 2.15: K8s deployment file for the STUN/TURN Server

Kubernetes service yaml file:

```yaml
apiVersion: v1
kind: Service
metadata:
    name: "recogaks-coturn"
    labels:
        app: "recogaks-coturn"
spec:
    type: LoadBalancer
    ports:
    - port: 3478
      targetPort: 3478
      protocol: TCP
      name: http
    selector:
        app: "recogaks-coturn"
```

Figure 2.16: K8s service file for the STUN/TURN Server

c) Front-end configuration:

Dockerfile:

```
FROM node:18-alpine

ENV PORT 3000

# Create app directory
RUN mkdir -p /usr/src/app
WORKDIR /usr/src/app

# Installing dependencies
COPY package*.json /usr/src/app/
RUN npm install

# Copying source files
COPY . /usr/src/app

# Building app
RUN npm run build
EXPOSE 3000

# Running the app
CMD "npm" "run" "dev"
```

Kubernetes deployment yaml file:

```
apiVersion : apps/v1
kind: Deployment
metadata:
  name: "recogaks-front"
spec:
  replicas: 2
  selector:
    matchLabels:
      app: "recogaks-front"
```

```
  template:
    metadata:
      labels:
        app: "recogaks-front"
    spec:
      containers:
        - name: "recogaks-front"
          image: "recogacr.azurecr.io/recogaks"
          ports:
          - containerPort: 3000
```

K8s service yaml file:

```
apiVersion: v1
kind: Service
metadata:
    name: "recogaks-front"
    labels:
        app: "recogaks-front"
spec:
    type: LoadBalancer
    ports:
    - port: 3000
      targetPort: 3000
      protocol: TCP
      name: http
    selector:
        app: "recogaks-front"
```

6 - Testing

Since the application will scale according to the number of users, we decided to load test and stress test the front-end using JMeter performance testing.

We first created a thread group with the following thread properties:

- **Number of threads:** 100
- **Loop count:** 10
- **Ramp-up period:** 100

The number of threads determines the number of concurrent users simulated connecting to a website. The loop count on the other hand simulates how many times each user connects to the website. And finally the ramp-up period dictates how long JMeter must wait before starting to simlate the next user.

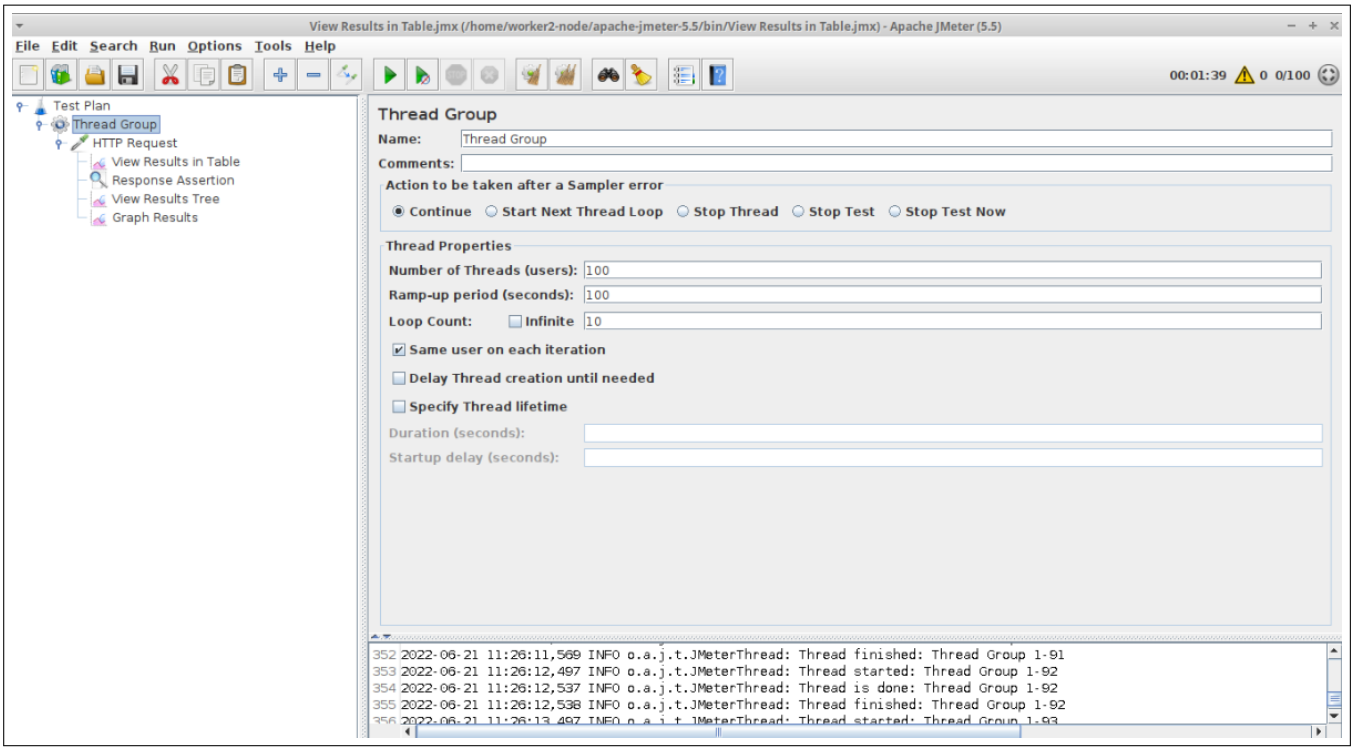


Figure 2.17: Thread group panel

After that, we added and configured the template HTTP request that each user will send. The figure below is the result of the test:

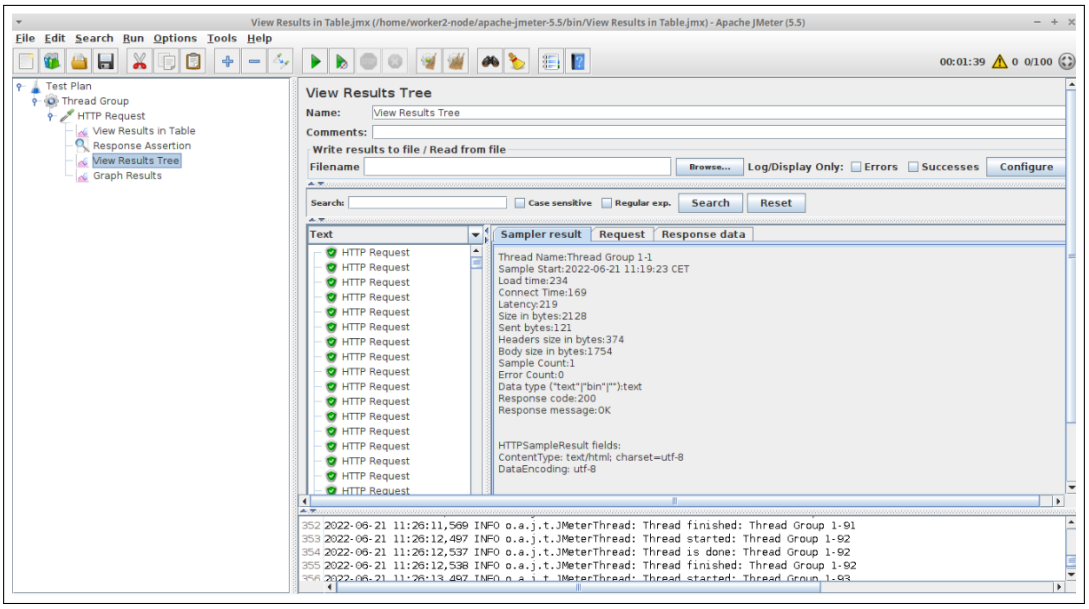


Figure 2.18: View results tree

JMeter creates requests and sends it to a server. It will then collect all the responses coming from it and visualize them in a graph, as the figure below shows:

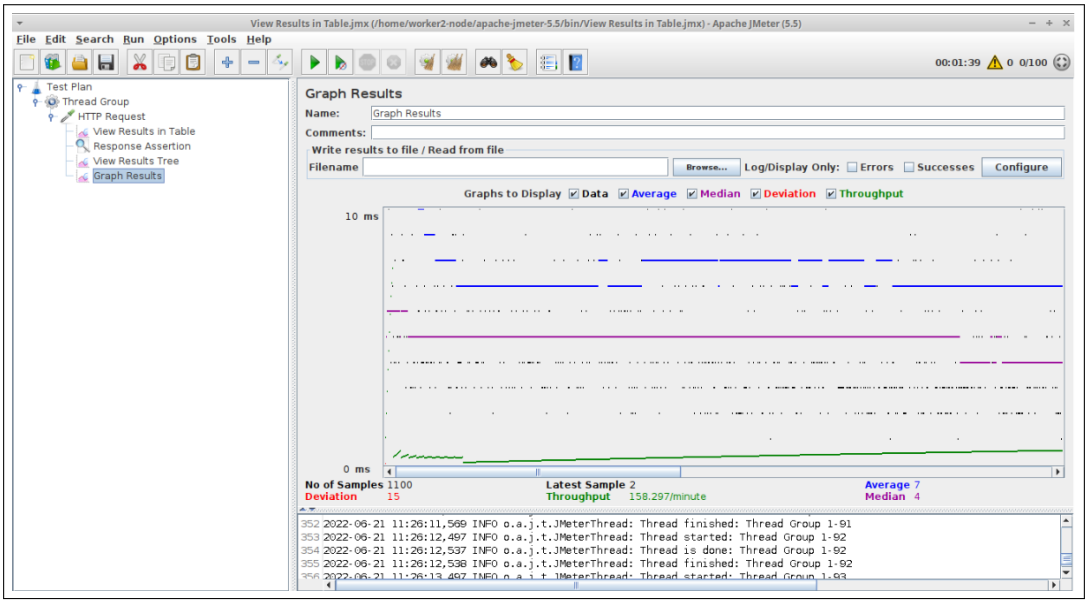


Figure 2.19: Graph results

7 - Conclusion:

During this chapter, we have analysed the system's requirements, conceptualized its functionalities, explained the different paradigms adopted for the implementation, described the different configurations needed to deploy each service and finally, performed performance tests for the front-end.

CHAPTER III)

RESOURCES AND TOOLS USED

1 - Introduction

In this chapter, we will introduce all the tools and resources that we used in order to develop the web application and services.

2 - Front-end

a) Next.js

Next.js is a React.js "Meta-framework" [16] that provides additional structures, features and optimizations to create web applications by handling the tools and configurations needed for a React UI.

Next.js solves common application requirements such as routing, data fetching, integrations - all while improving the developer and end-user experience.[17]

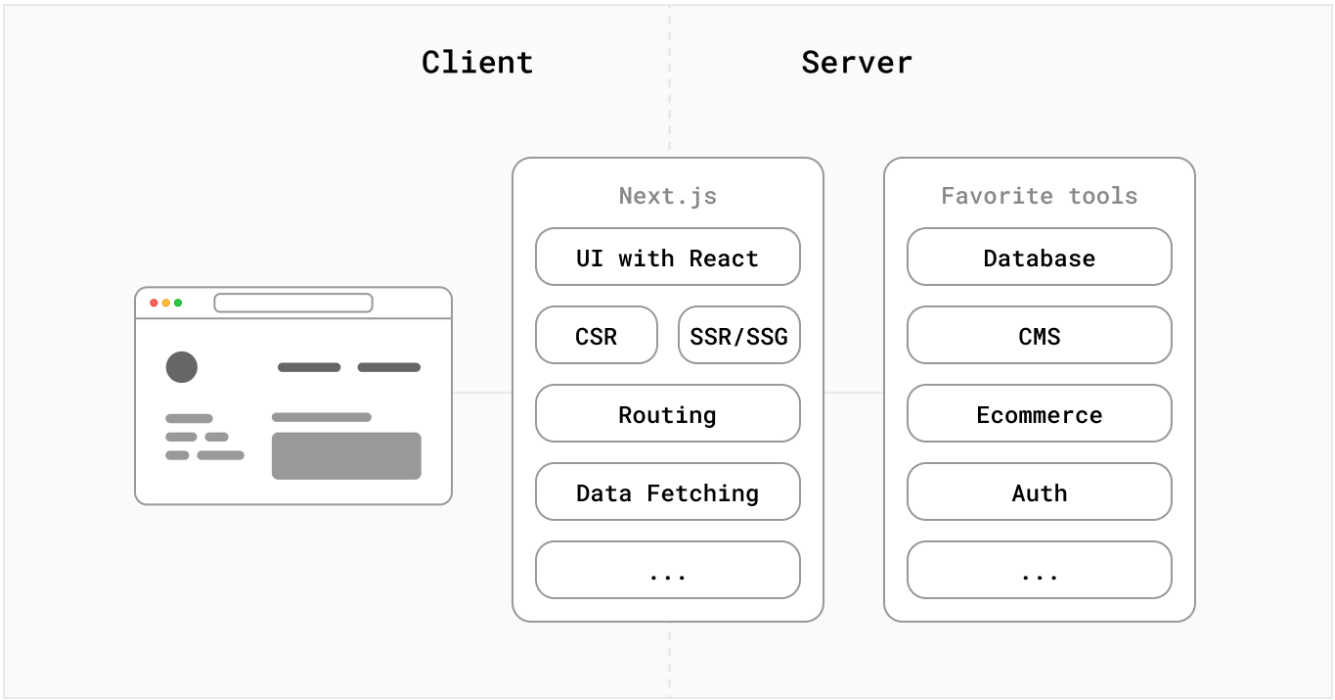


Figure 3.1: Next.js web application

Advantages:

- Short page load time thanks to the HTML static generation (the page is generated at build time and is used on each request [18]
- Next.js can statically generate pages without data so the static site doesn't have a direct connection to user sensitive information like database authentication data, which makes it safe.
- Since Next.js is a React.js framework, we can migrate an old React app without building everything from scratch.

b) Material UI

Material UI is a open-source tool developed by Google in 2014. It provides a simple, customizable, and accessible library of React UI components. It can be used with all JavaScript frameworks like AngularJS, VueJS, and libraries like ReactJS, to have an efficient and responsive application.

Advantages

- Provides powerful tools to build UI Components.
- It has a very detailed documentation to navigate easily in the framework [19].
- It is the most popular React UI Components framework in the world.

3 - Back-end

For the backend, we wanted to use Django as our application's Python web framework, but we quickly realised that it is far from being the most efficient REST API framework that are available, especially if we consider the performance requirement our application needs.

a) FastAPI

FastAPI is an open-source,high performance, production ready Python Web Framework used for building REST API endpoints. It uses ASGI (asynchronous server gateway interface) instead of the old WSGI. [20] In particular, FastAPI is used for use cases where speed is a priority (Since our project relies heavily on WebRTC streaming, a very fast API communication is **paramount**).

Advantages:

- Blazing fast performance (one of the fastest API frameworks[21]).
- Allows the validation of data types even within JSON requests.
- Very easy to learn since it has a simple and intuitive API.

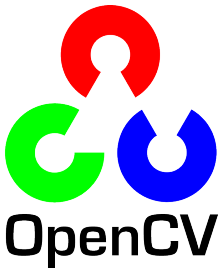
```python
from fastapi import FastAPI

app = FastAPI()


@app.get("/")
async def hello_world():
    return {"hello": "world"}
```

Figure 3.2: Minimal FastAPI Endpoint

b) OpenCV



OpenCV(Open Source Computer Vision) is a software library for computer vision and machine literacy software library. Originally created to give a common infrastructure for computer vision related operations and to increase the use of artificial perception in marketable products.It allows companies to use and modify the code software. All these algorithms being efficiently optimized. It supports real-time vision operations. Its algorithms are fluently enforced in Java, Python ...etc. It allows the collect of data, perform data processing, conditioning and eventually train and educate a model to understand how to distinguish faces according to their size, their eyes etc.[22]

Advantages

- Free of cost
- Low RAM usage
- A vast collection of algorithms

c) aiortc

An open-source Python library that implements WebRTC and ORTC(Object Real-Time Communication) for Python using asyncio. It allows the exchange of audio, video and data channels and interoperability is regularly tested against both Chrome and Firefox.[23]

Advantages

- Enables low-latency sending and receiving of video, audio and arbitrary data streams over the network by Web servers and clients.
- Easy to create innovative products by leveraging the multitude of Python packages.
- Extensive test suite to ensure best-in-class code quality

d) Node.js

Node.js is an asynchronous even-driven JavaScript runtime and platform for interpreting JavaScript code and running scalable network applications. Since Node.js is built with the Google Chrome JavaScript engine (a tool used to interpret JavaScript into useful computer commands), it is considered to be powerful and capable of supporting JavaScript as a server-side language.[24]

Figure 3.3: Node.js logo

Advantages

- Offers high performance for Real-time Applications.
- It is Easy to Learn and Quick to Adapt.
- Offers Extensibility to Meet Customized Requirements.
- Offers Easy Scalability for Modern Applications.

4 - Database / Authentication

In order to allow users to centralize and share their information that can be reliably organized, queried and improved, we have used the following tool in our project

a) Firebase

Firebase is a database that allows front-end developers to easily integrate a back-end into their application, without having to create API routes and other back-end code. It can also be said that Firebase is not just a database but a set of tools; often referred to as a back-end-as-a-service (BaaS) that contains a multitude of services, including:[25]

- Authentication: Login and user identity
- Real-time database: Real-time NoSQL database, hosted in the cloud.
- Cloud Firestore: Real-time NoSQL database, hosted in the cloud.
- ML Kit: An SDK for common machine learning tasks.

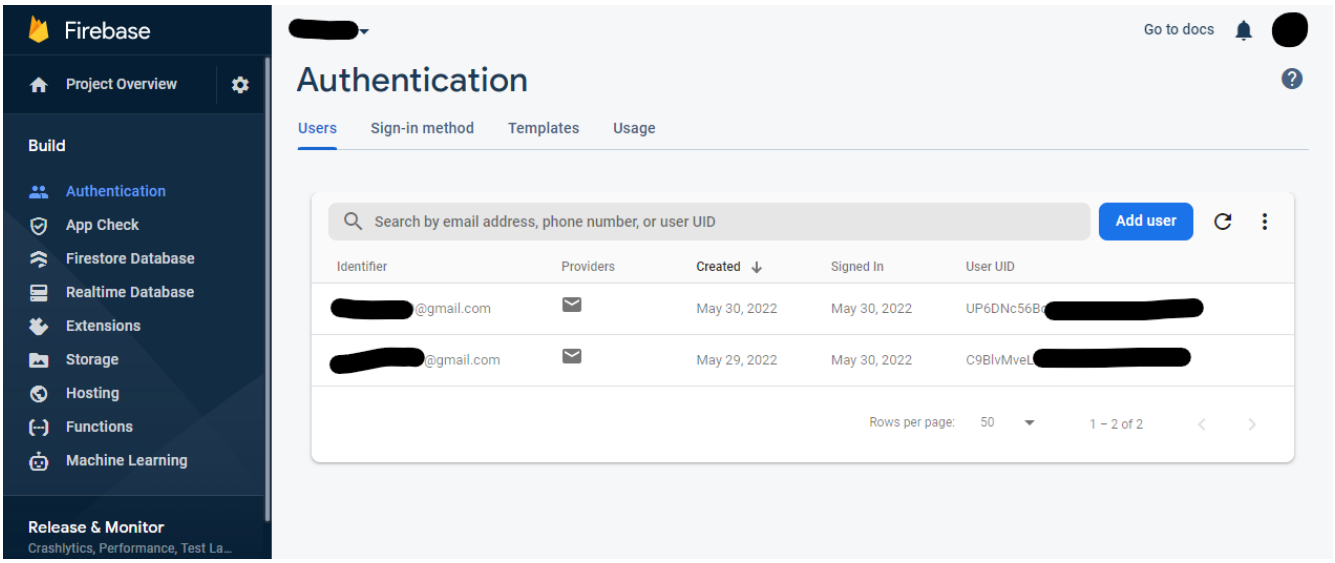


Figure 3.4: Firebase Authentication Console

Advantages

- Fast and secure hosting.
- Real-time database.
- Provides a free start.
- Free use of Firebase dynamic links

5 - DevOps tools

To apply a process to install or update our website on its environment we used the following tools:

a) GitHub

Figure 3.5: GitHub logo

GitHub is a popular website for hosting, developing and sharing software and computer code. It offers a web interface and provides features and a mix of free and paid services for working with such repositories .[26]

Advantages

- It facilitates the contribution to open source projects.
- Free service, although it also has paid services.
- Large community and easy to find help.
- In order to present our work. facilitates excellent documentation.
- It offers practical tools for cooperation and good integration with Git.

b) Github Actions

GitHub Actions is a CI/CD platform that automates software workflows. It allows developers to build, test, and deploy their code right from GitHub and make code reviews, branch management, and issue triaging. All GitHub Actions automations are handled via workflows,

which are YAML files placed under the .github/workflows directory in a repository that define automated processes.[27]

Advantages

- Setting-up a CI/CD pipeline is very simple
- Community-powered, reusable workflows
- Support for any platform, any language, and any cloud.

c) Docker

Docker is an open source application container engine, allows developers to package their applications and dependencies in a portable mirror, And publish them on any Linux Or windows on machine operating system, virtualization can also be achieved.

Advantages

- Docker containers assure a return on investment and saves cost compared to virtual machines
- Decreases deployment time
- The environment on a container is highly secure since it is isolated
- Highly scalable
- Consistent environment so less configuration/compatibility problems.

d) Kubernetes

Kubernetes, also known as K8s, is a portable, extensible, open source platform for managing containerized workloads and services, that facilitates both declarative configuration and automation. It has a large, rapidly growing ecosystem. Kubernetes services, support, and tools are widely available.[28]

Advantages

- Automatic rollouts and rollbacks to ensure the health of all instances at the same time.
- Provides an intuitive service discovery mechanism to pods and automatically load-balance across them.
- Orchestrates storage.
- Secure configuration management.
- Can automatically scale up or down based on resources usage.[29]

6 - Cloud services

a) Heroku

Heroku is a polyglot cloud application platform that provides tremendous flexibility in choosing an appropriate programming language to develop web apps. Heroku provides platform support for Ruby, Ruby on Rails, Java, Node.js, Clojure, Scala, Python, and PHP as of early 2013.

Advantages

- Heroku supports many programming languages such as python, java, node.js, etc.
- Heroku community is so big that if you have any problem, you can contact people online.

- It uses our local computer as a console.
- It offers simple and easy deployment, environment configuration and manageability.

b) Vercel

Vercel is a feature-rich platform that allows developers to easily create, pre-interface and deploy their sites as well as serverless features. It provides friction-less developer experience to take care of the hard things: deploying instantly, scaling automatically, and serving personalized content around the globe, and make it easy for frontend teams to develop, preview, and ship delightful user experiences, where performance is the default.

Advantages

- Requires no configuration and works with any type of web framework.
- Has a generous free tier
- Delivers fast site performance with simple, upgradeable deployments.

c) Microsoft Azure

Microsoft Azure is a cloud computing platform that was launched on February 2010. It allows its users to access and manage cloud services and resources provided by Microsoft. Azure provides more than 200 services, are divided into 18 categories. These categories include computing, networking, storage, IoT, migration, mobile, analytics, containers, artificial intelligence, and other machine learning, integration, management tools, developer tools, security, databases, DevOps, media identity, and web services[30]

Advantages

- Microsoft Azure has a strong focus on security.

- It outsources the maintenance of our infrastructure to experts who take care of upgrades and problems.
- Integrated Environment with Other Microsoft Tools.
- Scalability is the backbone of Azure .

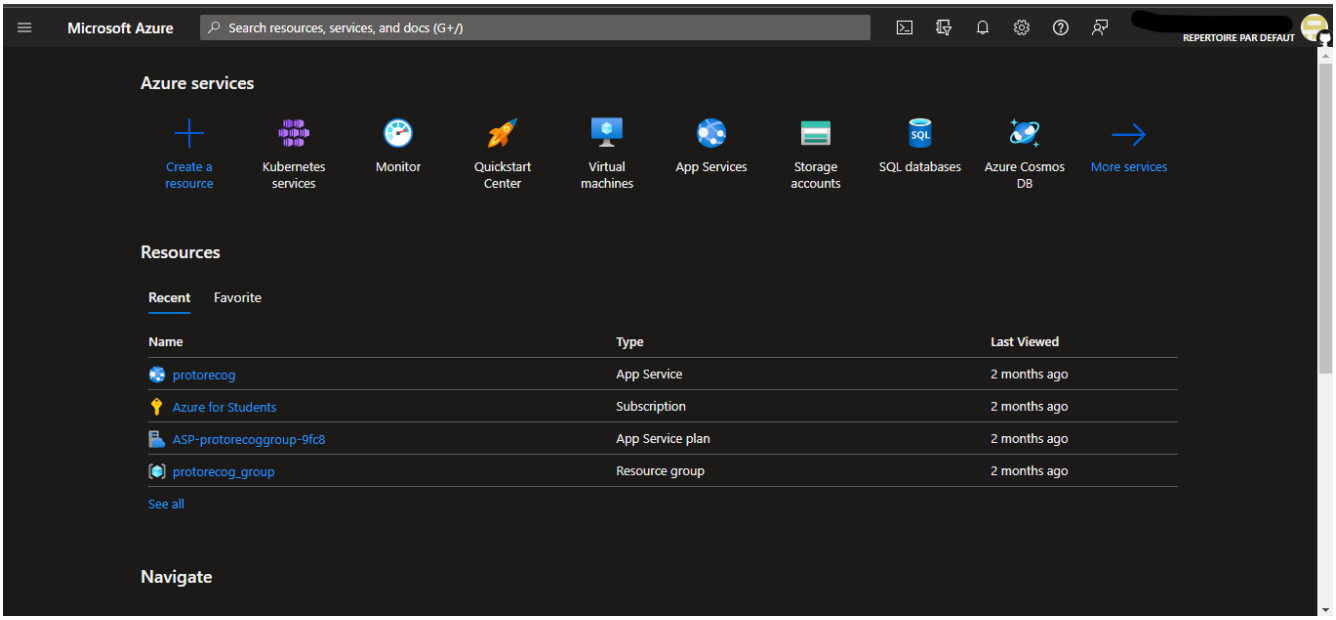


Figure 3.6: Microsoft Azure User Portal

7 - Testing tools:

a) Apache JMeter

JMeter is a pure Java open source software, designed to analyze and measure the performance of a web application or a variety of services. Performance testing consists of testing a web application against heavy load, multiple simultaneous user traffic, and for functional testing, database server testing.[31]

Advantages:

- Completely free.
- Easy to learn and use.
- Test results can be converted into different formats.
- Can also evaluate database performance.

8 - Conclusion

In this chapter we have listed all differents resources and tools that we have used for the development of our web application, and all the advantages that encouraged us to make these choices.

CHAPTER IV)

CUSTOM MODEL DEPLOYMENT

1 - Introduction

This chapter will present the front end of the application and how to use it. It will also describe the available machine learning model types that could be deployed and used.

2 - Front-end interfaces:

a) Video streaming interface

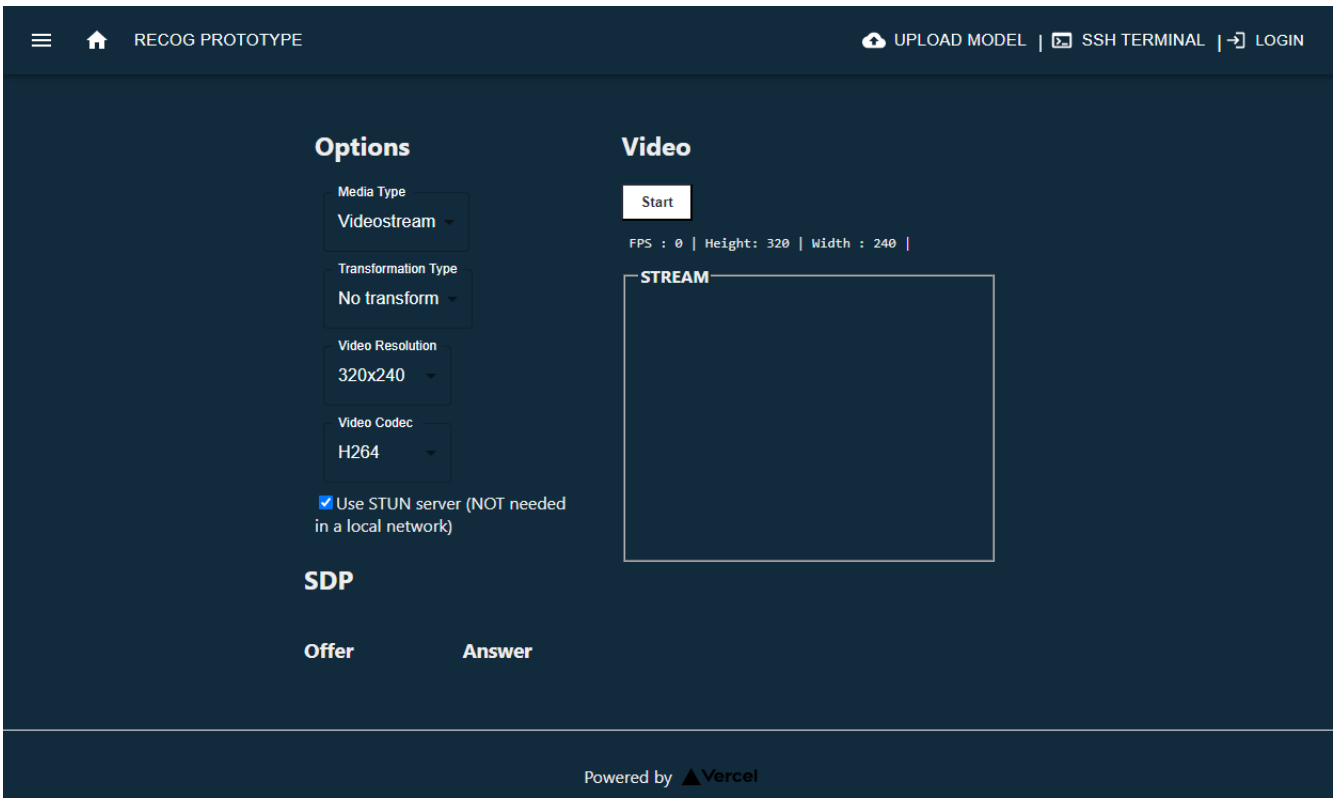


Figure 4.1: Application media streaming page

- 1) Choose videostream media type
- 2) Choose the machine learning type model to use
- 3) Choose video resolution
- 4) Choose the video codec
- 5) Only uncheck the STUN server option if you think your NAT is symmetric

- 6) Click the start button to begin the webRTC video stream and wait for the backend to process the stream using the machine learning model in real time.

b) Authentication interface

These interfaces are intuitive as it is with most authentication interfaces that websites use therefore a step guide is not needed in order to understand how to interact with it.

Login page

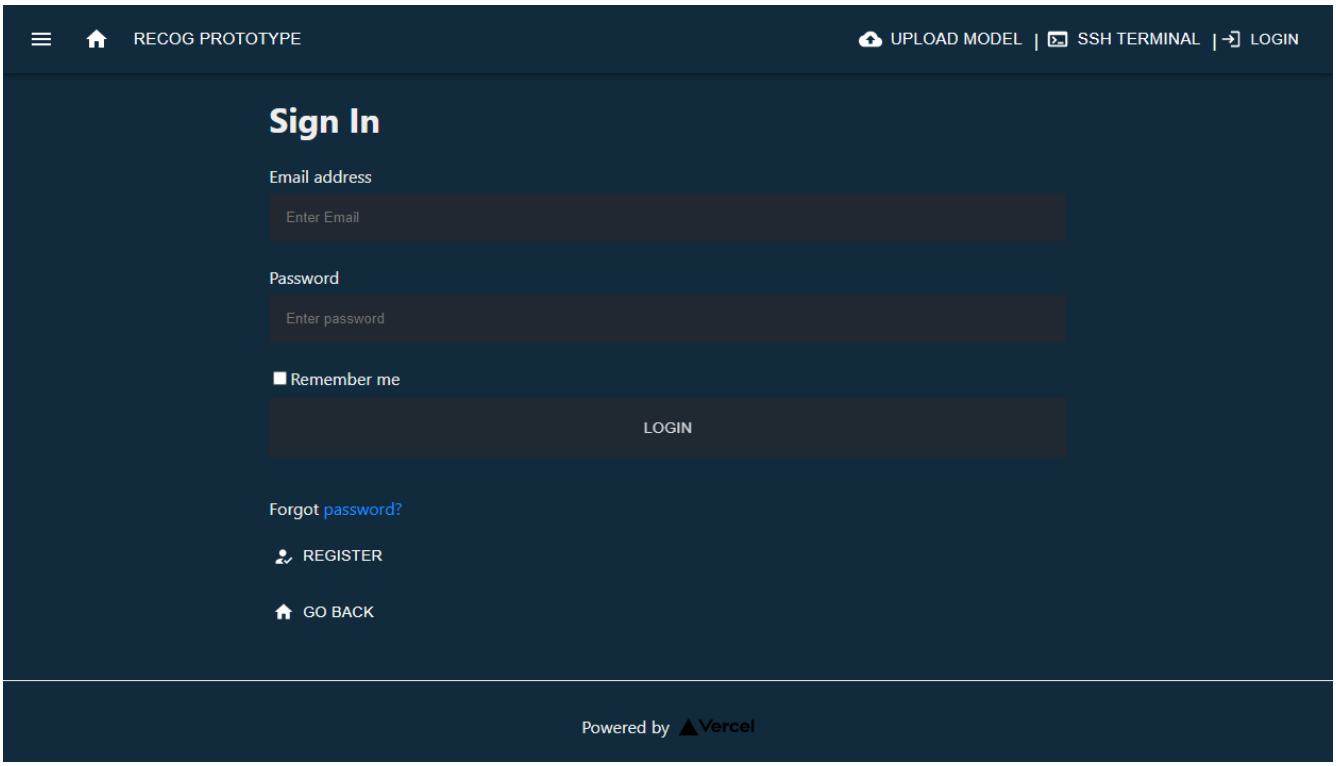


Figure 4.2: Login Page

Register page

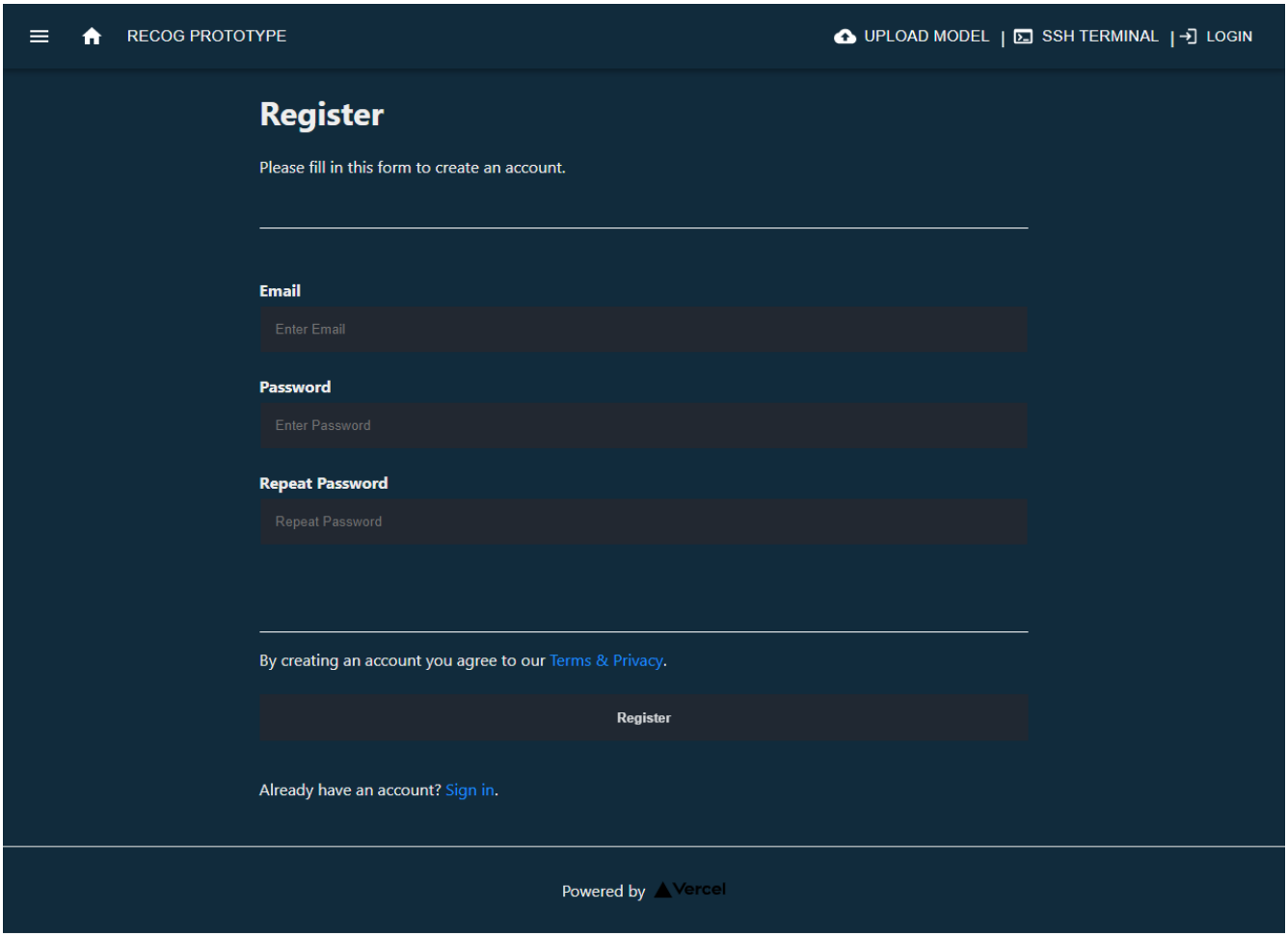


Figure 4.3: Register Page

c) Custom model configuration editor interface

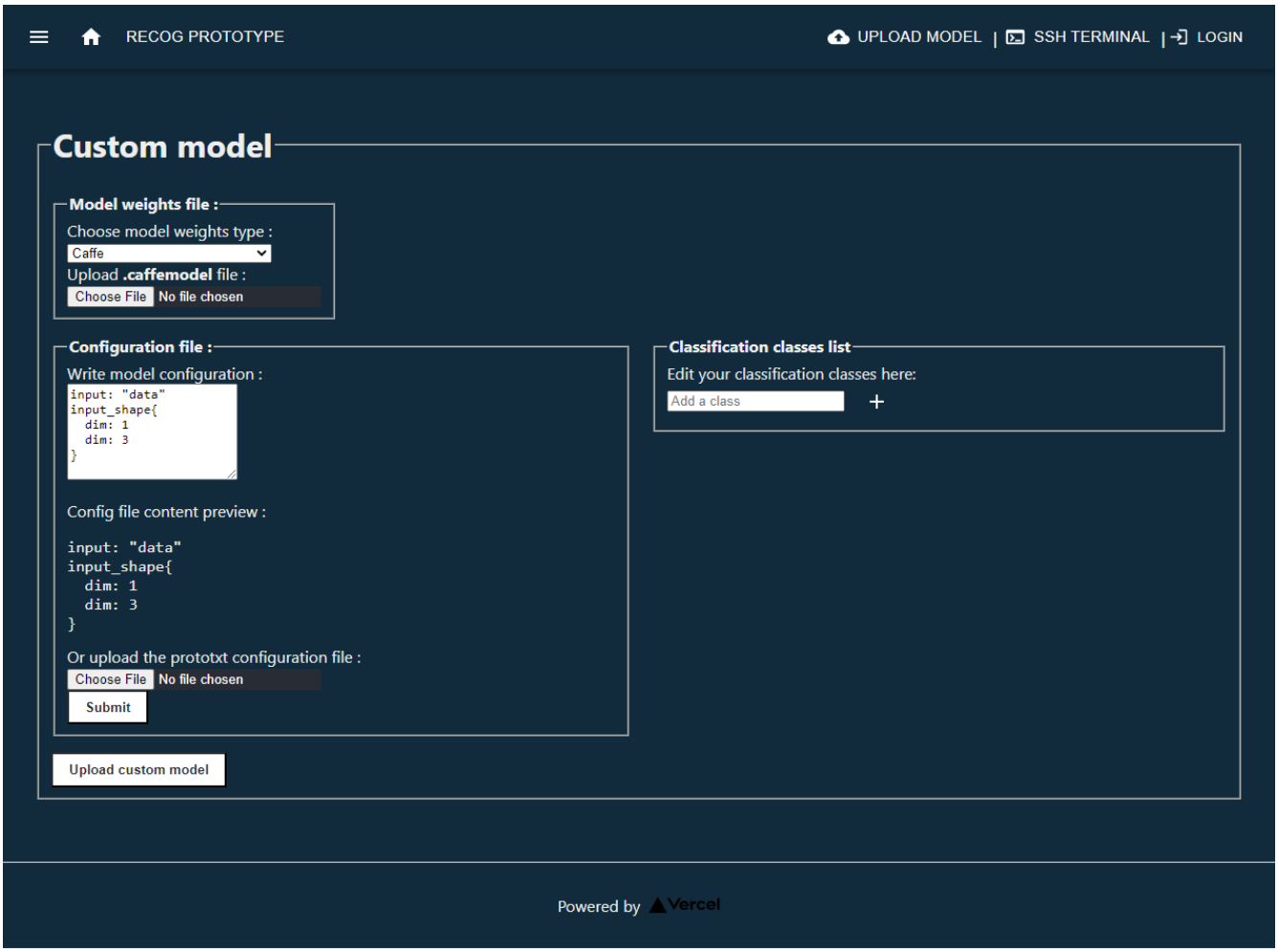


Figure 4.4: Custom model input

- 1) Select the desired OpenCV framework (or weight type) for the model.
- 2) Input the model file.
- 3) Edit the configuration file or upload it.
- 4) Add classification classes for the model
- 5) Click the ”Upload custom model” button in order to upload the model.

We are going to go in depth about the types of custom models in the next section.

3 - Deep learning and neural networks

Before getting into the types of machine learning models OpenCV supports, we must first explain what neural networks are.

Artificial neural networks (ANN) are a subset of machine learning and are at the heart of deep learning algorithms. They are computing systems which names and structures are inspired by the human brain, mimicking the way that biological neurons signal to one another. [32]

They are composed of a node layers, containing an input layer, one or more hidden layers, and an output layer. Each node, or artificial neuron, connects to another and has an associated weight and threshold. If the output of any individual node is above the specified threshold value, that node is activated, sending data to the next layer of the network. Otherwise, no data is passed along to the next layer of the network.

This figure[33] below illustrates the different layers of an ANN.

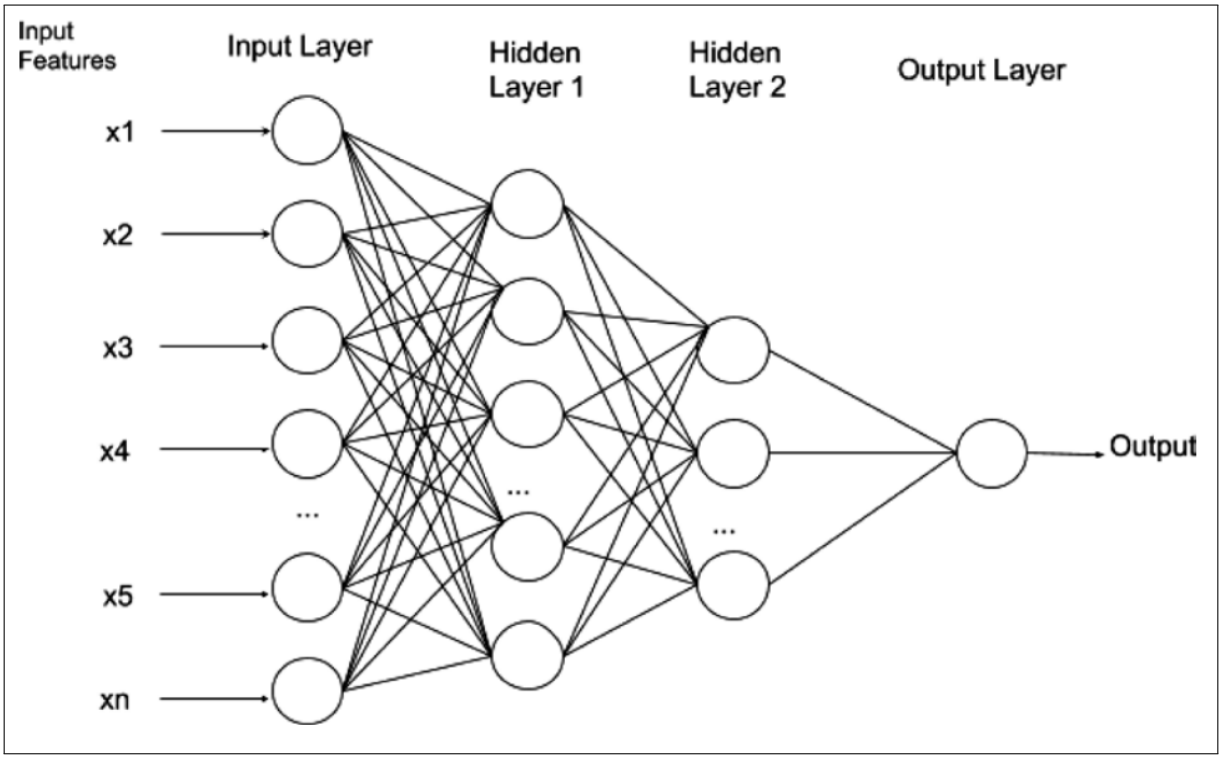


Figure 4.5: The layers of an artificial neural network

4 - OpenCV Deep Neural Network frameworks:

While it does not support the training of models, the OpenCV DNN module supports deep learning inference on images and videos.

It also supports many popular deep learning frameworks:

a) Caffemodel:

Caffe is a deep learning framework made with expression, speed, and modularity in mind. It is developed by Berkeley AI Research (BAIR) and by community contributors. It provides multimedia scientists and practitioners with a clean and modifiable framework for state-of-the-art deep learning algorithms and a collection of reference models. The framework is a BSD-licensed C++ library with Python and MATLAB bindings for training and deploying general-purpose convolutional neural networks and other deep models efficiently on commodity architectures.

Users create and save their models as plain text PROTOTXT files and then trains and refines their model using the Caffe algorithm, and then it is saved as a .caffemodel file(). ".caffemodel" files are binary protocol buffer files (protobuf), thus it is impossible to read or edit them. [34]

Advantages:

- Clear and intuitive architecture that encourages application and innovation
- Can freely switch between using CPU or GPU by setting a single flag
- Huge community thanks to its extensible code that fosters active development
- One of the fastest models: Caffe can process over 60 million images per day with a single GPU (provided its a high-end GPU).

b) Tensorflow:

TensorFlow is an open source platform created by Google that provides a high-level and easy-to-use API to create machine learning models.It is also an execution engine for Keras, a high-level neural network API written in Python. [35]

Two files are needed in order to load pre-trained TensorFlow models: a model weights file and a protobuf text

file contains the model configuration. The weight file has a .pb extension which is a protobuf file containing all the pre-trained weights.

c) Darknet:

Darknet is an open source neural network framework written in C and CUDA. It is fast, easy to install, and supports CPU and GPU computation.[36]

YOLO:

YOLO (You only look once) is a fast, real-time, and multi-object detection algorithm that consists of a single convolutional neural network that predicts simultaneously the bounding boxes and class probabilities of objects within them. YOLO trains on the full image, and the network is set up to solve regression problems to detect objects. Therefore, YOLO does not need a complex processing pipeline, which makes it extremely fast. [37]

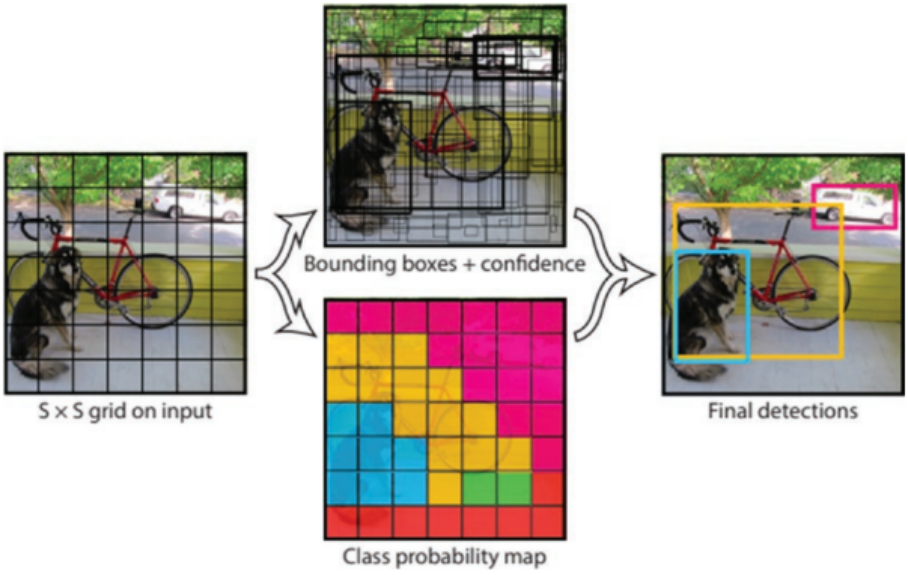


Figure 4.6: YOLO object detection example

d) LBPH Face recognition:

LBPH (Local Binary Pattern Histogram) is a powerful face recognition algorithm used to recognize the face of a person from both the front and the side. [38]

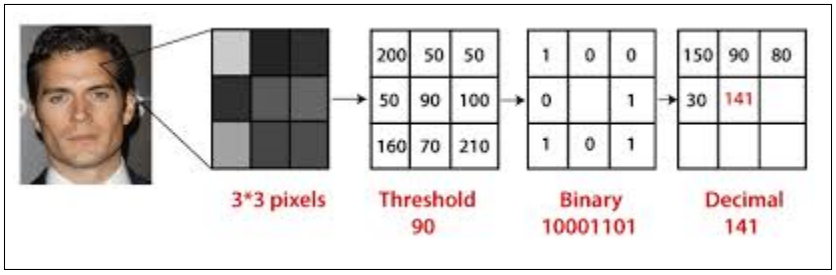


Figure 4.7: LBP Computation

5 - Conclusion

In this last chapter, we have presented the most relevant interfaces as well as the different machine learning model frameworks supported by the system.

CONCLUSION

The objective of this thesis was the implementation of a web application designed for facilitating the deployment of and use them for real time face/recognition.

The key points of this work were:

- Developing a web application that exploits the protocols and standards of WebRTC for the purpose of transmitting a webcam video and se OpenCV's image processing functionalities on each frame of the video.
- Containerization of the application with Docker.
- Deploying the application using cloud services.

In the first chapter, we talked about software development infrastructure and its evolution and have given a general idea on cloud computing as an infrastructure and the different types of services it provides. We also talked about machine learning engineering and MLOps. The second chapter provided an assessment of the system's requirements and described its expected behaviour. We mentioned all the ressources and tools we have used in the third chapter. Finally, the fourth chapter illustrated the different interfaces of the system.

In conclusion, the research carried out in this master's thesis addresses the feasibility of developing a system that facilitates the deployment of custom machine learning models for face recognition, using cloud services as an infrastructure.

BIBLIOGRAPHY

[1] VB Staff. Why do 87% of data science projects never make it into production?, jul 2019. [Accessed: 06/06/2022]. URL: https://venturebeat.com/2019/07/19/why-do-87-of-data-science-projects-never-make-it-into-production/.

[2] Red Hat. Understanding cloud computing, may 2022. [Accessed: 01/06/2022]. URL: https://www.redhat.com/en/topics/cloud.

[3] Inc. Red Hat. What is faas, jan 2020. URL: https://www.redhat.com/fr/topics/cloud-native-apps/what-is-faas.

[4] Red Hat. What is iaas?, aug 2019. [Accessed: 10/06/2022]. URL: https://www.redhat.com/en/topics/cloud-computing/what-is-iaas.

[5] Noah Gift. *Cloud Computing for Data Analysis, The missing semester of Data Science*. Leanpub, jul 2021.

[6] Ben Wilson. *Machine Learning Engineering in Action*. Manning Publications Co., 20 Baldwin Road, PO Box 761, Shelter Island, NY 11964, 2022.

[7] Emmanuel Raj. *Engineering MLOps, Rapidly build, test, and manage production-ready machine learning life cycles at scale*. Packt, Livery Place, 35 Livery Street, Birmingham B3 2PB, UK., first edition, apr 2021.

[8] MDN Contributors. Webrtc api. [Accessed 05/06/2022]. URL: https://developer.mozilla.org/en-US/docs/Web/API/WebRTC_API.

[9] Jonathan Rosenberg. Interactive Connectivity Establishment (ICE): A Protocol for Network Address Translator (NAT) Traversal for Offer/Answer Protocols. RFC 5245, April 2010. URL: https://www.rfc-editor.org/info/rfc5245, doi:10.17487/RFC5245.

[10] Philip Matthews, Jonathan Rosenberg, Dan Wing, and Rohan Mahy. Session Traversal Utilities for NAT (STUN). RFC 5389, October 2008. URL: https://www.rfc-editor.org/info/rfc5389, doi:10.17487/RFC5389.

[11] Kjeld Borch Egevang and Paul Francis. The IP Network Address Translator (NAT). RFC 1631, May 1994. URL: https://www.rfc-editor.org/info/rfc1631, doi:10.17487/RFC1631.

[12] Philip Matthews, Jonathan Rosenberg, and Rohan Mahy. Traversal Using Relays around NAT (TURN): Relay Extensions to Session Traversal Utilities for NAT (STUN). RFC 5766, April 2010. URL: https://www.rfc-editor.org/info/rfc5766, doi:10.17487/RFC5766.

[13] Colin Perkins, Mark J. Handley, and Van Jacobson. SDP: Session Description Protocol. RFC 4566, July 2006. URL: https://www.rfc-editor.org/info/rfc4566, doi:10.17487/RFC4566.

[14] Microsoft. Deploy using github actions into azure kubernetes service. [Accessed 20/06/2022]. URL: https://azure.github.io/kube-labs/1-github-actions.html#objective-of-the-lab.

[15] Docker. Dockerfile reference. [Accessed 25/06/2022]. URL: https://docs.docker.com/engine/reference/builder/.

[16] Michael Rambeau. 2021 javascript rising stars. [Accessed: 11/06/2022]. URL: https://risingstars.js.org/2021/en.

[17] Vercel. What is next.js? [Accessed: 11/06/2022]. URL: https://nextjs.org/learn/foundations/about-nextjs/what-is-nextjs.

[18] Vercel. Pages. [Accessed: 11/06/2022]. URL: https://nextjs.org/docs/basic-features/pages.

[19] Material ui documentation. [Accessed: 11/06/2022]. URL: https://mui.com/material-ui/getting-started/overview/.

[20] Tiangolo. Fastapi. [Accessed: 12/06/2022]. URL: https://fastapi.tiangolo.com/.

[21] Web framework benchmark. [Accessed: 12/06/2022]. URL: https://www.techempower.com/benchmarks/.

[22] OpenCV Maintainers. Opencv. [Accessed 05/06/2022]. URL: https://vovkos.github.io/doxyrest-showcase/opencv/sphinx_rtd_theme/index.html.

[23] Jeremy Lainé. aiortc. [Accessed: 04/06/2022]. URL: https://aiortc.readthedocs.io/en/latest/.

[24] OpenJS Foundation. About node.js. [Accessed 15/06/2022]. URL: https://nodejs.org/en/about/.

[25] Google. Firebase. [Accessed 15/06/2022]. URL: https://firebase.google.com/.

[26] Github. What is github actions? benefits and examples. [Accessed 17/06/2022]. URL: https://resources.github.com/downloads/What-is-GitHub.Actions_.Benefits-and-examples.pdf.

[27] Github. What is github actions? benefits and examples. [Accessed 17/06/2022]. URL: https://resources.github.com/downloads/What-is-GitHub.Actions_.Benefits-and-examples.pdf.

[28] Kubernetes. What is kubernetes? [Accessed: 14/06/2022]. URL: https://kubernetes.io/docs/concepts/overview/what-is-kubernetes/.

[29] Kubernetes. Kubernetes features. [Accessed: 13/06/2022]. URL: https://kubernetes.io/.

[30] Microsoft Co. Azure services platform. [Accessed: 12/06/2022]. URL: http://www.microsoft.com/azure.

[31] Apache Software Foundation. Apache jmeter™. [Accessed 26/06/2022]. URL: https://jmeter.apache.org/.

[32] IBM Cloud Education. What are neural networks?, aug 2020. [Accessed 23/06/2022]. URL: https://www.ibm.com/cloud/learn/neural-networks.

[33] Shamshad Ansari. *Building Computer Vision Applications Using Artificial Neural Networks: With Step-by-Step Examples in OpenCV and TensorFlow with Python*. Apress Media, Centreville, VA, USA, first edition, jul 2020.

[34] OpenCV. Load caffe framework models. URL: https://docs.opencv.org/4.x/d5/de7/tutorial_dnn_googlenet.html.

[35] Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S. Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Ian Goodfellow, Andrew Harp, Geoffrey Irving, Michael Isard, Yangqing Jia, Rafal Jozefowicz, Lukasz Kaiser, Manjunath Kudlur, Josh Levenberg, Dandelion Mané, Rajat Monga, Sherry Moore, Derek Murray, Chris Olah, Mike Schuster, Jonathon Shlens, Benoit Steiner, Ilya Sutskever, Kunal Talwar, Paul Tucker, Vincent Vanhoucke, Vijay Vasudevan, Fernanda Viégas, Oriol Vinyals, Pete Warden, Martin Wattenberg, Martin Wicke, Yuan Yu, and Xiaoqiang Zheng. TensorFlow: Large-scale machine learning on heterogeneous systems, 2015. Software available from tensorflow.org. URL: https://www.tensorflow.org/.

[36] Joseph Redmon. Darknet: Open source neural networks in c. http://pjreddie.com/darknet/, 2013–2016. [Accessed: 15/06/2022].

[37] Joseph Redmon and Ali Farhadi. Yolov3: An incremental improvement. *arXiv*, 2018.

[38] T. Ahonen, A. Hadid, and M. Pietikainen. Face description with local binary patterns: Application to face recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 28(12):2037–2041, 2006. doi:10.1109/TPAMI.2006.244.

APPENDIX A

Python requirements.txt file:

```
absl-py==1.0.0
aioice==0.7.6
aiortc==1.3.1
anyio==3.5.0
asgiref==3.5.0
astunparse==1.6.3
autopep8==1.6.0
av==9.1.1
cachetools==5.0.0
certifi==2021.10.8
cffi==1.15.0
charset-normalizer==2.0.12
click==8.1.2
colorama==0.4.4
cryptography==36.0.2
cvlib==0.2.7
dnspython==2.2.1
ecdsa==0.17.0
email-validator==1.1.3
et-xmlfile==1.1.0
fastapi==0.75.0
flatbuffers==2.0
```

```
gast==0.5.3
google-auth==2.6.4
google-auth-oauthlib==0.4.6
google-crc32c==1.3.0
google-pasta==0.2.0
grpcio==1.44.0
gunicorn==20.1.0
h11==0.13.0
h5py==3.6.0
httptools==0.4.0
idna==3.3
imageio==2.16.2
imutils==0.5.4
Jinja2==3.1.1
keras==2.8.0
Keras-Preprocessing==1.1.2
libclang==13.0.0
Markdown==3.3.6
MarkupSafe==2.1.1
netifaces==0.11.0
numpy==1.22.3
oauthlib==3.2.0
opencv-python-headless==4.5.5.64
openpyxl==3.0.9
opt-einsum==3.3.0
passlib==1.7.4
Pillow==9.1.0
progressbar==2.5
protobuf==3.20.0
pyasn1==0.4.8
pyasn1-modules==0.2.8
pycodestyle==2.8.0
pycparser==2.21
pydantic==1.9.0
pyee==9.0.4
pylibsrtp==0.7.1
pymongo==4.1.0
```

```
python-dotenv==0.20.0
python-jose==3.3.0
python-multipart==0.0.5
PyYAML==6.0
requests==2.27.1
requests-oauthlib==1.3.1
rsa==4.8
six==1.16.0
sniffio==1.2.0
starlette==0.17.1
tensorboard==2.8.0
tensorboard-data-server==0.6.1
tensorboard-plugin-wit==1.8.1
tensorflow==2.8.0
tensorflow-io-gcs-filesystem==0.24.0
termcolor==1.1.0
tf-estimator-nightly==2.8.0.dev2021122109
toml==0.10.2
typing_extensions==4.1.1
urllib3==1.26.9
uvicorn==0.17.6
watchgod==0.8.2
websockets==10.2
Werkzeug==2.1.1
wrapt==1.14.0
```

Github Actions to AKS CI/CD pipeline workflow:

```yaml
# This workflow uses actions that are not certified by GitHub.
# They are provided by a third-party and are governed by
# separate terms of service, privacy policy, and support
# documentation.

name: Build and deploy to Azure Kubernetes Service

env:
  # set this to the name of your container registry
  AZURE_CONTAINER_REGISTRY: MY_REGISTRY_NAME
   # set this to your project's name
  PROJECT_NAME: MY_PROJECT_NAME
  # set this to the resource group containing your AKS cluster
  RESOURCE_GROUP: MY_RESOURCE_GROUP
  # set this to the name of your AKS cluster
  CLUSTER_NAME: MY_CLUSTER_NAME
  # set this to the URL of your registry
  REGISTRY_URL: MY_REGISTRY_URL
  # If you build using helm:
   # set this to the path to your helm file
  CHART_PATH: MY_HELM_FILE
  # set this to an array of override file paths
  CHART_OVERRIDE_PATH: MY_OVERRIDE_FILES

on: [push]

jobs:
  build:
    runs-on: ubuntu-latest
    steps:
    - uses: actions/checkout@v3

    - name: Azure Login
      uses: azure/login@89d153571fe9a34ed70fcf9f1d95ab8debea7a73
```

```
      with:
        creds: ${{ secrets.AZURE_CREDENTIALS }}

    - name: Build image on ACR
      uses: azure/CLI@7378ce2ca3c38b4b063feb7a4cbe384fef978055
      with:
        azcliversion: 2.29.1
        inlineScript: |
          az configure --defaults acr=${{ env.AZURE_CONTAINER_REGISTRY }}
          az acr build -t  -t ${{ env.REGISTRY_URL }}/${{ env.PROJECT_NAME }}:${{ github.sha }}

    - name: Gets K8s context
      uses: azure/aks-set-context@4e5aec273183a197b181314721843e047123d9fa
      with:
          creds: ${{ secrets.AZURE_CREDENTIALS }}
          resource-group: ${{ env.RESOURCE_GROUP }}
          cluster-name: ${{ env.CLUSTER_NAME }}
      id: login
    - name: Configure deployment
      uses: azure/k8s-bake@773b6144a3732e3bf4c78b146a0bb9617b2e016b
      with:
        renderEngine: 'helm'
        helmChart: ${{ env.CHART_PATH }}
        overrideFiles: ${{ env.CHART_OVERRIDE_PATH }}
        overrides: |
          replicas:2
        helm-version: 'latest'
      id: bake

    - name: Deploys application
    - uses: Azure/k8s-deploy@c8fbd76ededaad2799c054a9fd5d0fa5d4e9aee4
      with:
        manifests: ${{ steps.bake.outputs.manifestsBundle }}
        images: |
          ${{ env.AZURE_CONTAINER_REGISTRY }}.azurecr.io/${{ env.PROJECT_NAME }}:${{ github.sha }}
        imagepullsecrets: |
          ${{ env.PROJECT_NAME }}
```

OpenCV image processing functions:

```
import cv2

import numpy as np

import cvlib as cv
from cvlib.object_detection import draw_bbox


def face_detect(img):
    faceCascade = cv2.CascadeClassifier(
        './utils/cascades/haarcascade_frontalface_default.xml')
    # Detect faces with Haarcascade
    gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
    faces = faceCascade.detectMultiScale(
        gray,
        scaleFactor=1.2,
        minNeighbors=5,
        minSize=(20, 20)
    )
    for (x, y, w, h) in faces:
        cv2.rectangle(img, (x, y), (x+w, y+h), (0, 255, 0), 2)

    return img


def face_detect_cvlib(img):
    # apply face detection
    faces, confidences = cv.detect_face(img)
    # loop through detected faces
    for idx, f in enumerate(faces):
        (startX, startY) = f[0], f[1]
        (endX, endY) = f[2], f[3]
        # draw rectangle over face
        cv2.rectangle(img, (startX, startY),
```

```
                      (endX, endY), (0, 255, 0), 2)
        text = "{:.2f}%".format(confidences[idx] * 100)
        Y = startY - 10 if startY - 10 > 10 else startY + 10
        # write confidence percentage on top of face rectangle
        img = cv2.putText(img, text, (startX, Y), cv2.FONT_HERSHEY_SCRIPT_COMPLEX, 0.7,
                          (0, 255, 0), 2)
    return img


def object_detect_cvlib(img):
    # Perform the object detection
    bbox, label, conf = cv.detect_common_objects(
        img, confidence=0.25, model='yolov3-tiny', enable_gpu=False)
    return draw_bbox(img, bbox, label, conf)


def gender_recog_cvlib(img):
    padding = 20
    # apply face detection
    faces, confidences = cv.detect_face(img)
    # loop through detected faces
    for idx, f in enumerate(faces):
        (startX, startY) = max(0, f[0]-padding), max(0, f[1]-padding)
        (endX, endY) = min(
            img.shape[1]-1, f[2]+padding), min(img.shape[0]-1, f[3]+padding)

        # draw rectangle over face
        cv2.rectangle(img, (startX, startY),
                      (endX, endY), (0, 255, 0), 2)

        face_crop = np.copy(img[startY:endY, startX:endX])

        # apply face detection
        (label, confidence) = cv.detect_gender(face_crop)

        idx = np.argmax(confidence)
        label = label[idx]
```

```
72
73          label = "{}: {:.2f}%".format(label, confidence[idx] * 100)
74
75          Y = startY - 10 if startY - 10 > 10 else startY + 10
76
77          # write detected gender and confidence percentage on top of face rectangle
78          img = cv2.putText(img, label, (startX, Y), cv2.FONT_HERSHEY_SIMPLEX, 0.7,
79                            (0, 255, 0), 2)
80      return img
81
82
83  def edge_detect(img):
84      # perform edge detection
85      return cv2.cvtColor(cv2.Canny(img, 100, 200), cv2.COLOR_GRAY2BGR)
86
87
88  def rotate_track(img, time):
89      # rotate image
90      rows, cols, _ = img.shape
91      M = cv2.getRotationMatrix2D(
92          (cols / 2, rows / 2), time * 45, 1)
93      return cv2.warpAffine(img, M, (cols, rows))
94
95
96  def cartoon_effect(img):
97      # prepare color
98      img_color = cv2.pyrDown(cv2.pyrDown(img))
99      for _ in range(6):
100         img_color = cv2.bilateralFilter(img_color, 9, 9, 7)
101     img_color = cv2.pyrUp(cv2.pyrUp(img_color))
102     # prepare edges
103     img_edges = cv2.cvtColor(img, cv2.COLOR_RGB2GRAY)
104     img_edges = cv2.adaptiveThreshold(
105         cv2.medianBlur(img_edges, 7),
106         255,
107         cv2.ADAPTIVE_THRESH_MEAN_C,
108         cv2.THRESH_BINARY,
```

```
        9,
        2,
    )
    img_edges = cv2.cvtColor(img_edges, cv2.COLOR_GRAY2RGB)

    # combine color and edges
    return cv2.bitwise_and(img_color, img_edges)
```