Labmate

Midterm Presentation

A smart lab companion for graduate students

Nishadi Prasangini
Yonwoo Choi
Hyoyoung Lho
Emma Pruvost

Summary

1.
Initial idea review

Previously...

Reduce student's stress - Make lab life easier

FEATURES

GPU monitoring

☹ Already existing solutions

Meal recommendation

☹ Hard to build good recommender

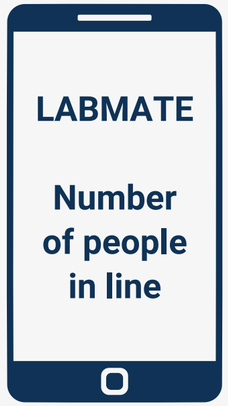
Seat and queue checker at cafeteria

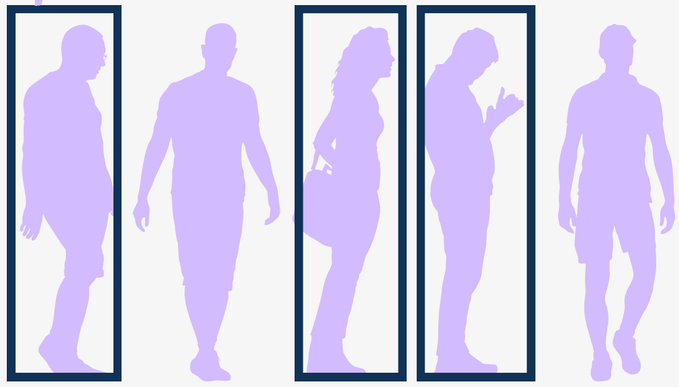
2. Refining: Queue checker

Queue checker

~~Seat checker~~

Queue checker





Gusia

User case

Should we go to Gusia now?

Should we make delivery?

Students from 300 buildings

With Labmate

Goals

Features	Success evaluation
User managment	Login / register / delete account working
Real time number of people in queue	Average latency between image capture and phone display Accuracy, limited to Gusia images

3. System overview

System diagram

RDS

MySQL

DATABASE

Add/ Delete / query user

EC2

android

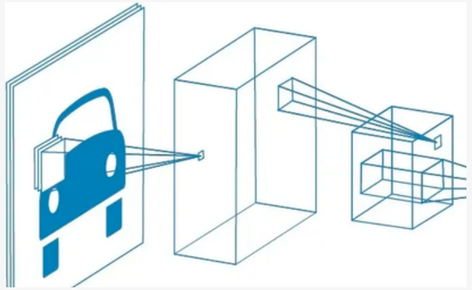
Port 1

SERVER

Success/Fail
Nb people in line

APP

Login/Join



MODEL

Number of people in line

Images

Port 2

Flask
web development,
one drop at a time

Send frames 30fps

CAMERA

3.1. Vision model

Model selection

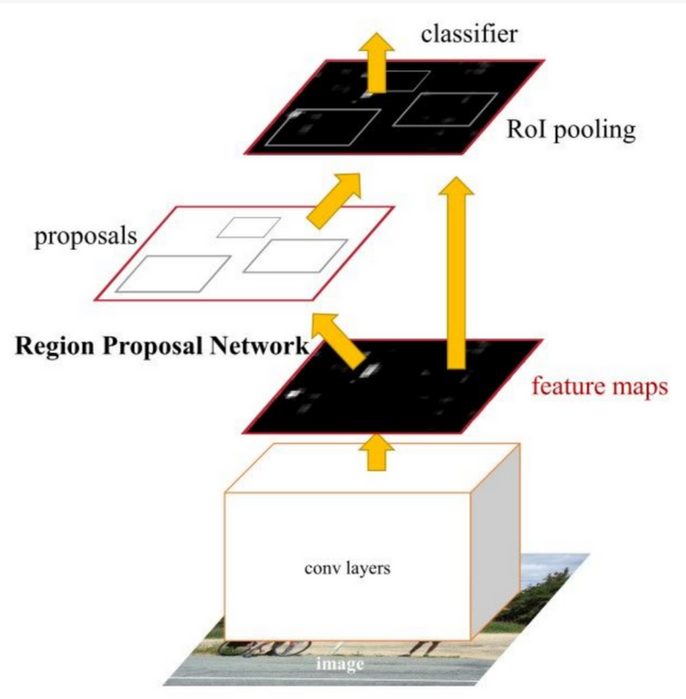
Silhouette detection

Yolov3

- State of the art model, great accuracy
- Fast
- 2018
- Training code released for public
- Darknet

Faster RCNN

- Region proposal model + CNN
- Good for retraining
- 2015
- Script used from Detectron2
- ResNet



➡ Need to retrain for head detection for improvement

Post of oct 28 2021 on businessware technologies "Facial recognition for detecting people in queues"
Detectron2: (2020) https://github.com/facebookresearch/detectron2

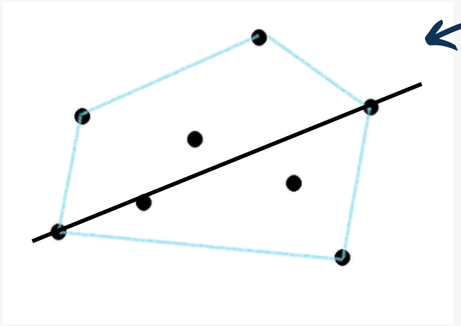
Methods

Detect only humans

Filter outputs

- Image framing

-> detect only people inside 2 lines



- Convex hull for queue orientation

-> infer the queue direction line and keep people with perpendicular distance to line < threshold

- Single line perpendicular distance

- Fine tuning

Count number of people

- Count for each image

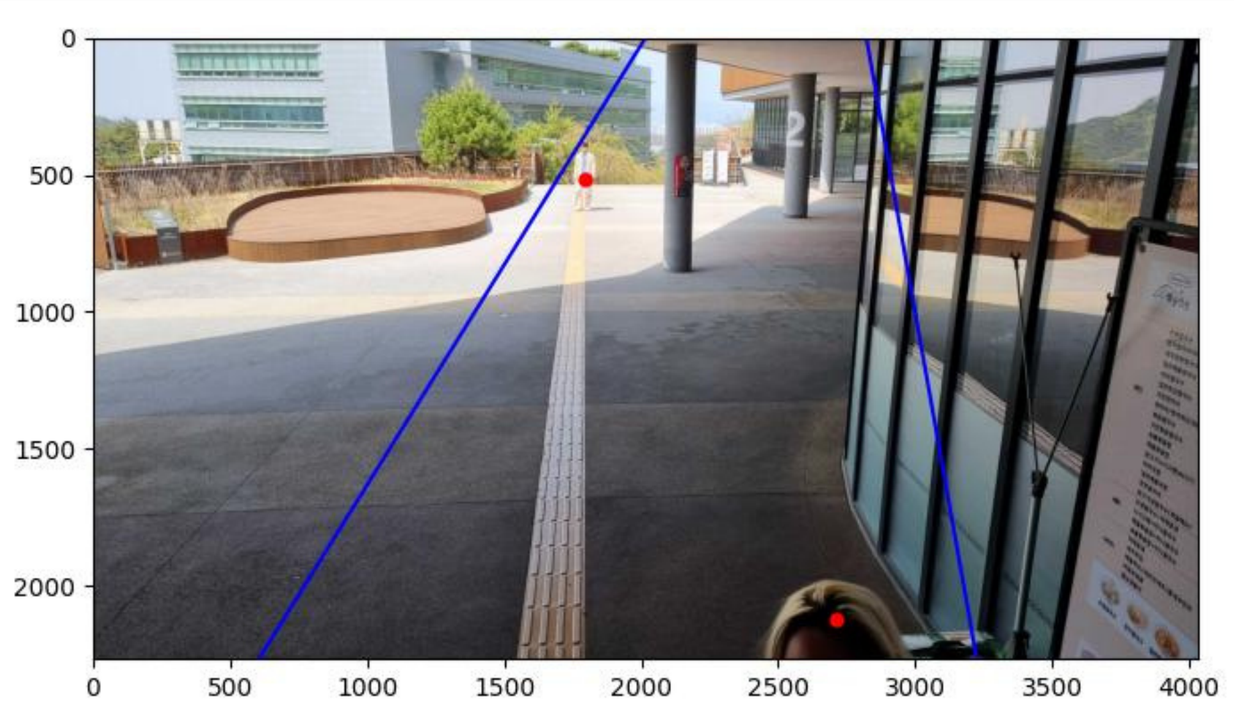
- Base counting frequency on previous images and hour of the day

- Count for each image but keep highest occurence value over interval

Results

Using faster RCNN

Framing method



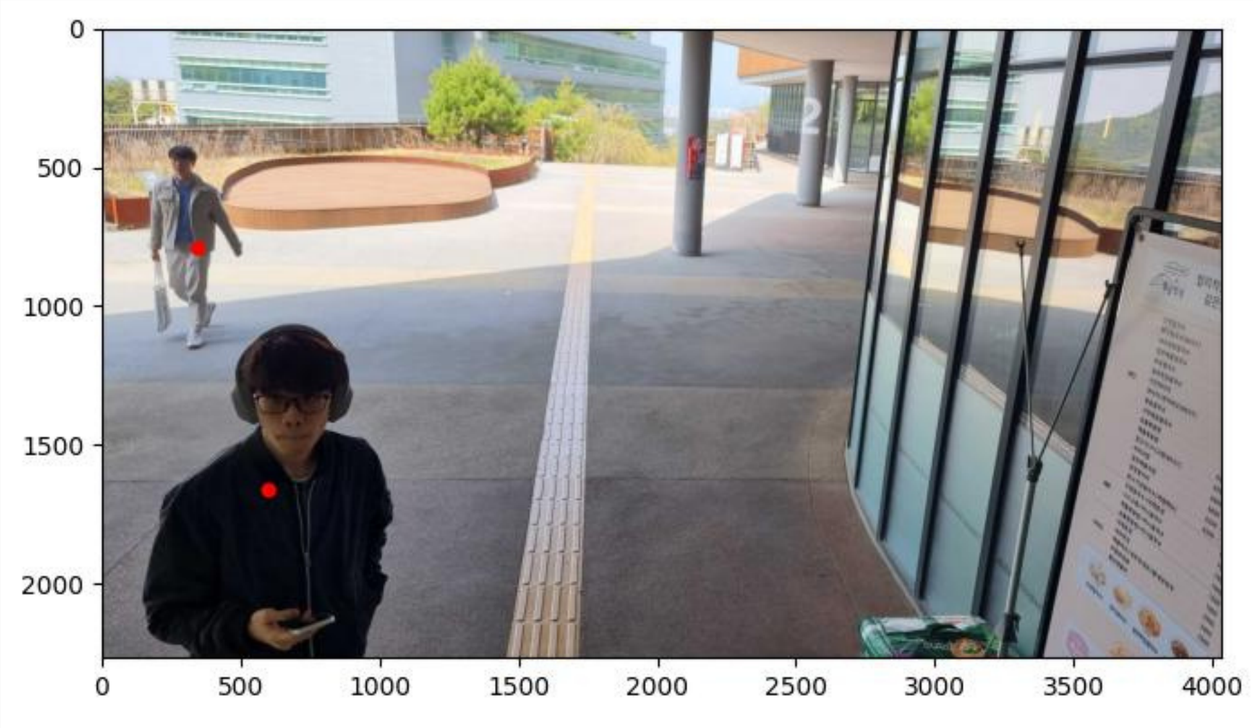
Limitations:

- Detects everything between the lines
- Strong assumptions: camera position is fixed + queue is a straight line

Results

Using faster RCNN

Queue orientation



Limitations:

- finds queue even if there is not
- assums queue is a straight line

Improvements:

- threshold based on standard deviation
- do not detect line if single person or 2 person very distant
- force beginning and end of line in specific area / switch to single line perpendicular distance method

Need data for quantitative evaluation

3.2. Server and database

Cloud computing

Why cloud computing?

- Not run locally because not sure of the capacity of devices in Gusia
- Independant from our lab's servers
- To extend the functionality of mobile app

2 ports:

- User managment
- Sensor data and model inference

AWS (amazon web services)

EC2

Run virtual servers in the cloud, providing scalable and cost-effective compute capacity

RDS

(Relational Database Service)

Set up, operate and scale a relational database in the cloud with ease

User managment

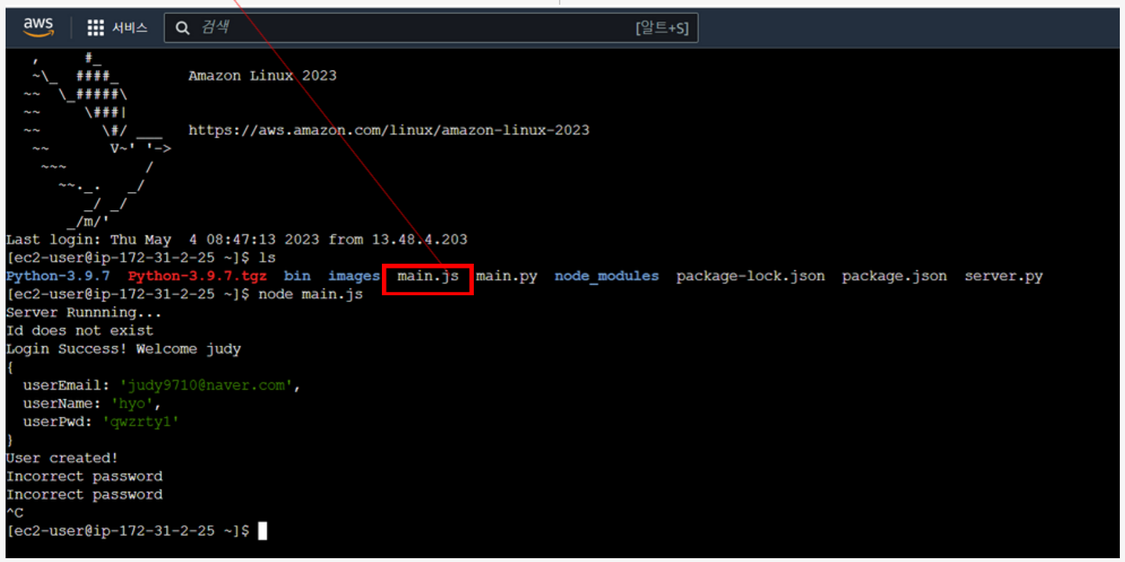
AWS RDS & EC2

Done

login, signup (main.js)
(same as the tutorial)

To do

delete account



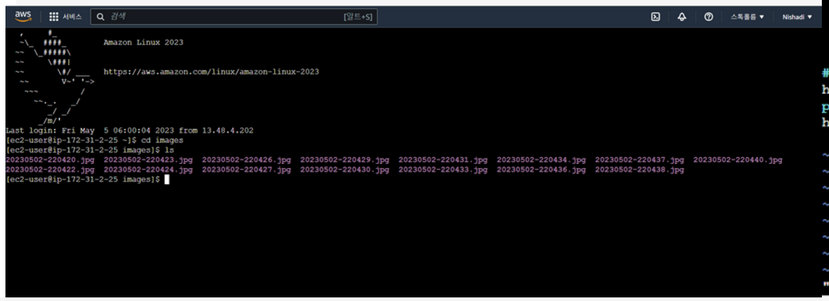
Data collection and storage

Send & receive captured frames

```python
import cv2
import requests
import numpy as np
import time

# initialize the video capture object
cap = cv2.VideoCapture(0)

# set the IP address and port of the EC2 instance
ec2_ip = 'ec2-13-50-249-234.eu-north-1.compute.amazonaws.com'
ec2_port = '3000'

# loop to capture and send frames
while True:
    # read a frame from the video capture object
    ret, frame = cap.read()

    # encode the frame into JPEG format
    _, jpeg = cv2.imencode('.jpg', frame)

    # convert the JPEG-encoded frame to bytes
    img_bytes = jpeg.tobytes()

    # create a unique filename based on the current time
    filename = time.strftime('%Y%m%d-%H%M%S') + '.jpg'

    # send the frame to the EC2 instance
    url = 'http://{}:{}/{}'.format(ec2_ip, ec2_port, filename)
    response = requests.post(url, data=img_bytes)

    # wait for 0.1 seconds before capturing the next frame
    time.sleep(0.1)

    # break the loop if the 'q' key is pressed
    if cv2.waitKey(1) == ord('q'):
        break
```

AWS

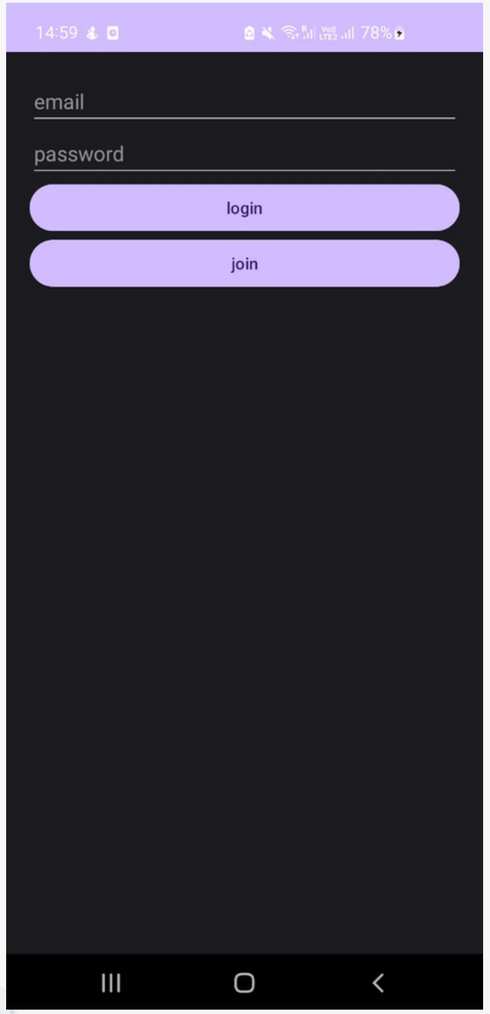
Considering to use S3 bucket, in order to store the data and use it for retraining

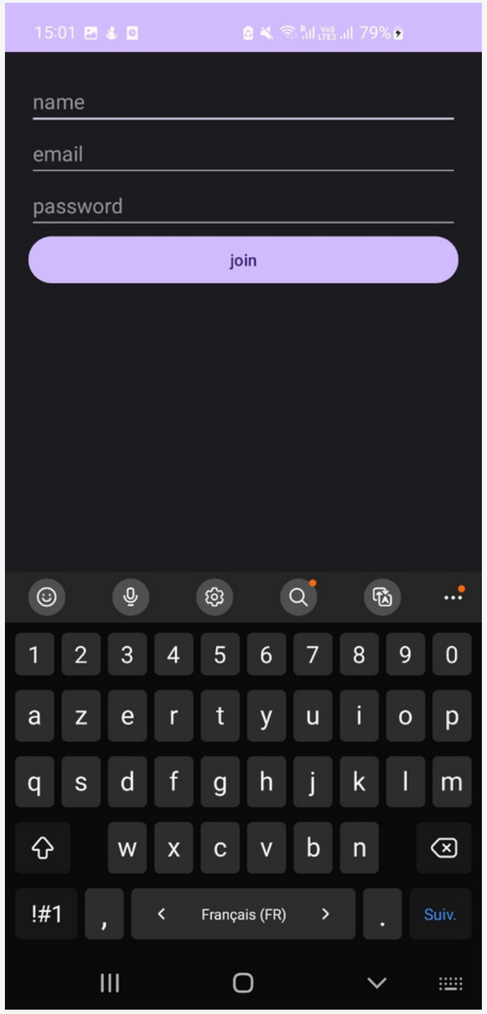


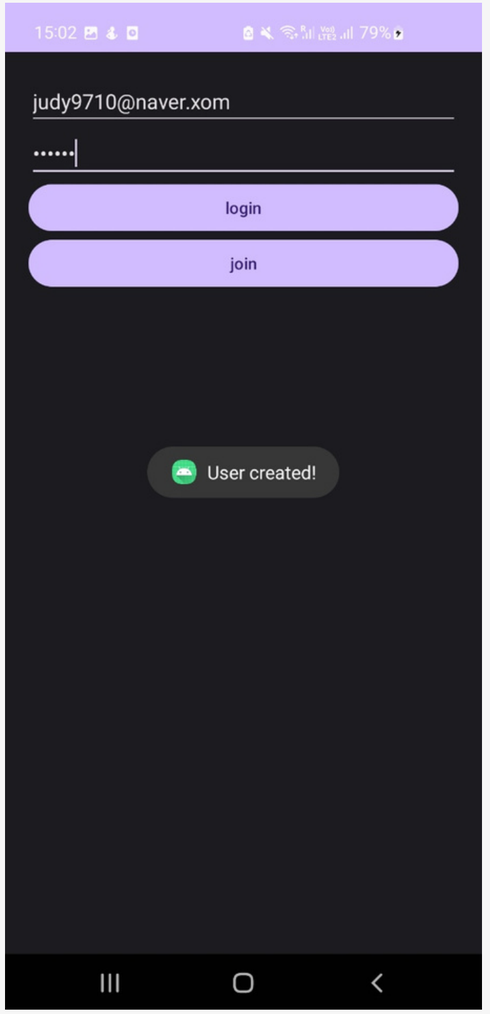

```python
from http.server import BaseHTTPRequestHandler, HTTPServer
import os

# define the port to listen on
PORT = 3000

# define the directory to save the frames to
SAVE_DIR = './images'

# define a custom request handler that saves the received frames to disk
class RequestHandler(BaseHTTPRequestHandler):
    def do_POST(self):
        # get the filename from the URL path
        filename = self.path[1:]

        # read the frame bytes from the request body
        content_length = int(self.headers.get('Content-Length'))
        frame_bytes = self.rfile.read(content_length)

        # save the frame bytes to a file
        with open(os.path.join(SAVE_DIR, filename), 'wb') as f:
            f.write(frame_bytes)

        # send a response back to the client
        self.send_response(200)
        self.end_headers()

# create an HTTP server object and start listening for incoming requests
httpd = HTTPServer(('0.0.0.0', PORT), RequestHandler)
print('Listening on port', PORT)
httpd.serve_forever()

~
~
~
~
~
~
~
~
"server.py" 32L, 1003B
```

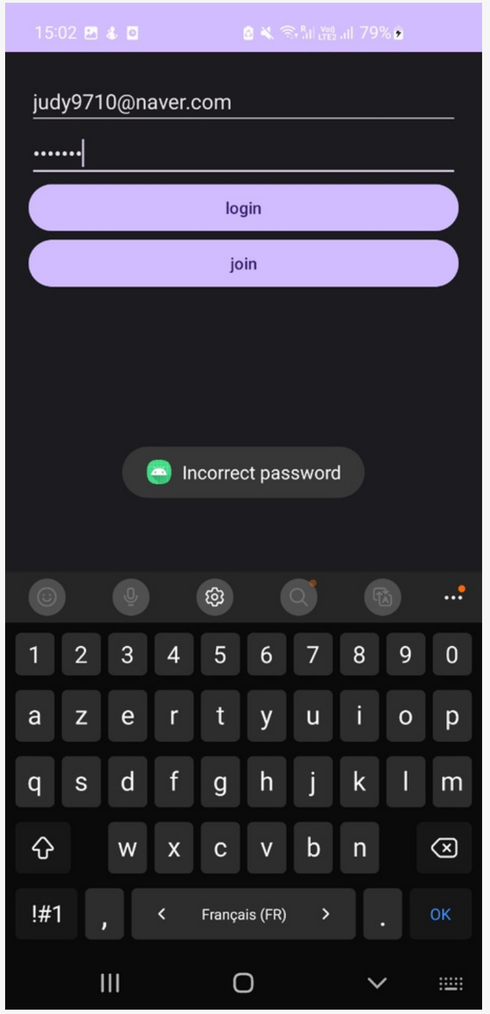
3.3. UI

Login & registration page

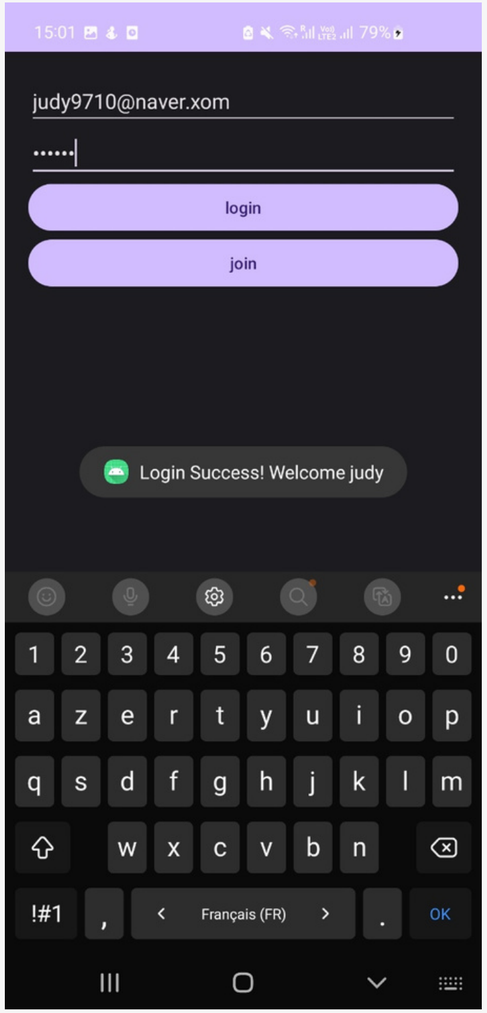


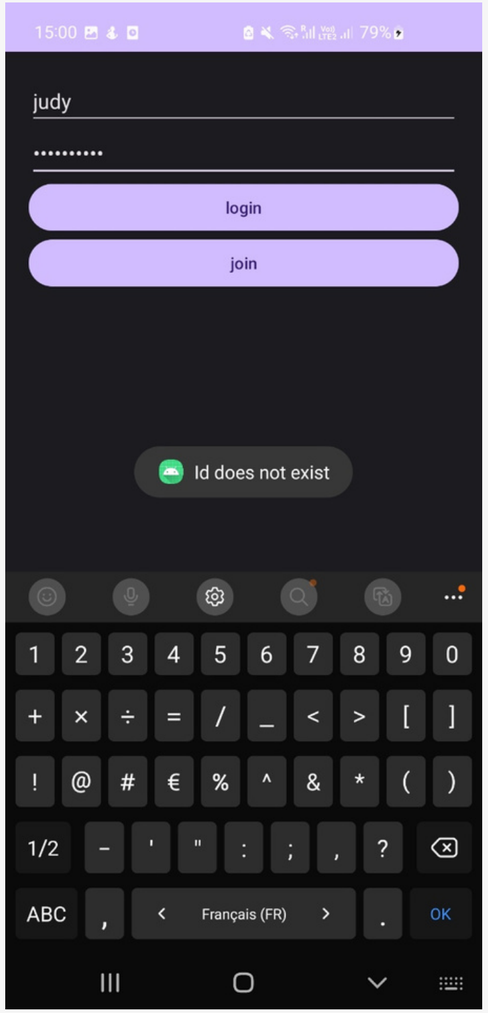






Homepage


judy9710@naver.xom
login
join
Login Success! Welcome judy


judy
login
join
Id does not exist


Hi user judy!
logout
G
U
S
I
A
people in queue :
15
waiting time
expectation
15 minutes

4.
Workload

Tasks

Done

- tutorials (Y,H,N,E)
- faster-rcnn and yolov3 inference (Y,E)
- framing output filter script finished (Y,E)
- queue orientation output filter in progress (Y,E)
- script to capture image in server (N,H)
- user login and registration (script in server + connected to app) (N,H)
- basic UI (N,H)

To do

- merge model inference scripts and image capturing script in server for real time image analysis (flask) and connect to app
- adapt yolo outputs to the output filter functions similarly as for faster-rcnn
- find/make dataset + model accuracy evaluation + model selection
- improve queue orientation output filter
- model retraining and fine tuning
- evaluate final app (tests + latency measure)
- refactor, reformat, comment scripts etc
- add user delete account function
- improve UI

Y: Yonwoo, H: Hyoyoung, N: Nishadi, E: Emma

Project schedule

	10	11	12	13	14	15
Merge scripts	H+N	H+N				
Adapt yolo outputs	E+Y	E+Y				
Data collection and model selection	ALL	ALL				
Improve queue orientation output filter			E+Y			
Model fine tuning				E+Y		
Evaluate final app					ALL	ALL
Refactor, reformat, comment scripts				H+N		
Add user delete account function			H+N	H+N		
Improve UI					H+N	