TEAM

THE OUTSIDERS

“Capstone Project_Midterm Presentation”

Agenda

00. Objective
01. Schedule milestone
02. Roles of each member
03. Implementation
04. Challenges
05. Limitation
06. Demo

00. Objective

“CNN based place recognition web app”

1. Collect data set & Build CNN models which have the best accuracy
2. Work on UI design & graphic Design
3. Apply CNN models on web app

01. Schedule milestone

Previous Plan

~10/3 Brainstorming & Confirm
Service's Concept, Collect dataset

~10/10 Make Initial UI Design, Build simple CNN model

~10/17 Graphic Design & Specify Function's details , Build various CNN model

~10/24 Make Web Structure
(Make index page & main CSS file), Improving accuracy

~11/2 Make Specific web page & Connect CNN Model

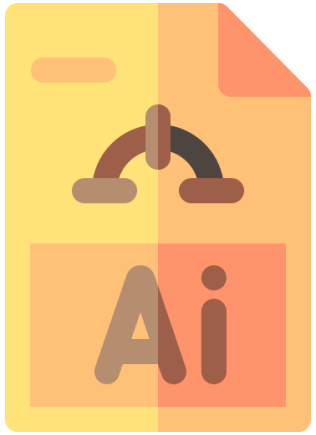
Inspection

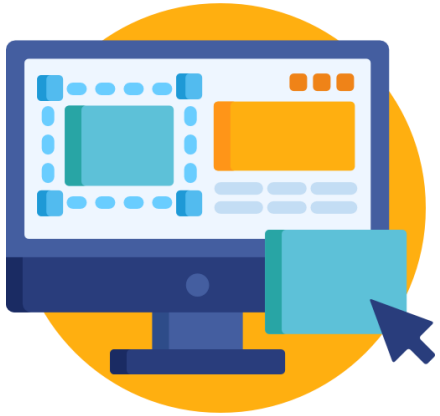
Implemented more than we had planned!

-> left all plan : completed
-> implement more function
: Start Page / SNS page /Upload Page !

02. Roles of each member

“Roles of each member in three parts”





CNN Build

- **Collect image dataset**
  CHE SEUNG YUN, HONG SEONGJUN
- **Build simple cnn model & test it**
  CHE SEUNG YUN

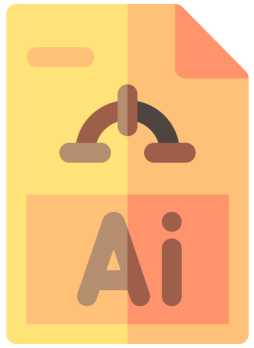
Front end

- **UI design with Figma**
  JEONG CHAEWON, LEE JI SEOP

Back end

- **Test simple CNN model in web application**
  UHM JI YONG

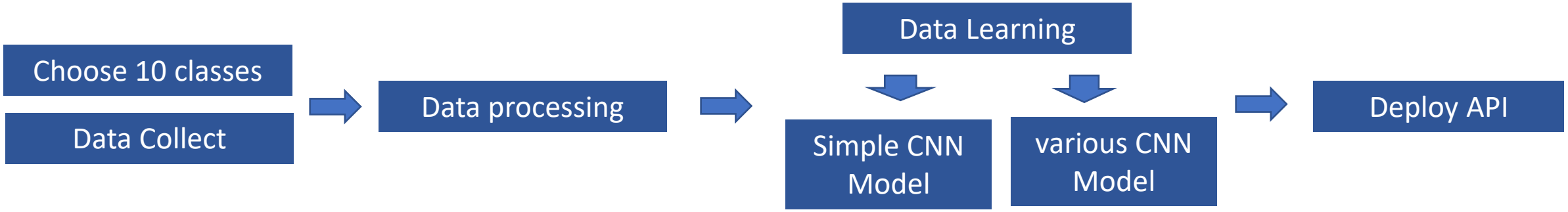
03. Implementation – CNN build



CNN Build

Our CNN model development process

1. Collect and process data necessary for learning.
2. Train an appropriate artificial intelligence model using the processed learning data.
3. Deploy the trained model to utilize it in application.



03. Implementation – CNN build

Choose 10 classes

- Dongdaemun_Design_Plaza
- Gyeongui_Line_Forest_Park
- Haebangchon
- Han_River_Sebitseom
- Ikseon_Dong_Hanok_Village
- Jamsil_Lotte_Tower
- Myeongdong_Cathedral
- Naksan_Park
- Namsan_Seoul_Tower
- The_Hyundai_Seoul_Mall

10 classses

Data Collect

Library

Selenium

Platform

jupyter

main ▾ PINPLACE / crawling.py / <> Jump to ▾

orioncsy Update crawling.py

1 contributor

62 lines (53 sloc) | 2 KB

```python
#selenium을 이용하여 구글에서 이미지를 크롤링하는 코드
from selenium import webdriver
from selenium.webdriver.common.keys import Keys
import time
import os
import urllib.request

def createFolder(directory):
    try:
        if not os.path.exists(directory):
            os.makedirs(directory)
    except OSError:
        print ('Error: Creating directory. ' +  directory)
keyword='연트럴 파크'
#검색할 키워드
createFolder('C:/Users/user/Desktop/images/'+keyword+'_img_download')
chromedriver = 'C:/Users/user/Downloads/chromedriver.exe'
driver = webdriver.Chrome(chromedriver)
driver.implicitly_wait(3)
```

- Choose 10 hot places in Seoul where MZ generation likes
- Collect image data by crawling. Using Python, selenium library.

  Collect 1000 images for each class
- Remove irrelevant images and duplicate data.

  After image processing, about 600 images remain for each class.

03. Implementation – CNN build

Data processing

```
data_aug_gen = ImageDataGenerator(rescale=1./255,
                                  rotation_range=15,
                                  width_shift_range=0.1,
                                  height_shift_range=0.1,
                                  shear_range=0.5,
                                  zoom_range=[0.8, 2.0],
                                  horizontal_flip=True,
                                  vertical_flip=False,
                                  fill_mode='nearest')
```

```
(17815, 128, 128, 3)
17815
```

- We used data augmentation and save the images.
- We have total 17,815 images

03. Implementation – CNN build

Data Learning

- We tried various models such as Alexnet, VGG16 model, Resnet50 models
- But, Alexnet, VGG16 model have so many parameters to learn. It can not be learned in google colab because of limits of ram capacity
- So, we choose Resnet50 model

```
model = ResNet50V2(include_top=True, weights=None, input_shape=(128,128,3), classes=10)
model.compile(loss='categorical_crossentropy', optimizer='Nadam', metrics=['accuracy'])
```

conv5_block3_2_conv (Conv2D)	(None, 4, 4, 512)	2359296	conv5_block3_2_pad[0][0]
conv5_block3_2_bn (BatchNormali	(None, 4, 4, 512)	2048	conv5_block3_2_conv[0][0]
conv5_block3_2_relu (Activation	(None, 4, 4, 512)	0	conv5_block3_2_bn[0][0]
[illegible]524			conv5_block3_2_relu[0][0]
[illegible]	[illegible]	0	conv5_block2_out[0][0] conv5_block3_3_conv[0][0]
post_bn (BatchNormalization)	(None, 4, 4, 2048)	8192	conv5_block3_out[0][0]
post_relu (Activation)	(None, 4, 4, 2048)	0	post_bn[0][0]
avg_pool (GlobalAveragePooling2	(None, 2048)	0	post_relu[0][0]
predictions (Dense)	(None, 10)	20490	avg_pool[0][0]

Total params: 23,585,290
Trainable params: 23,539,850
Non-trainable params: 45,440

03. Implementation – CNN build

Data Learning

We set train set, validation set, test set as 5:2:3

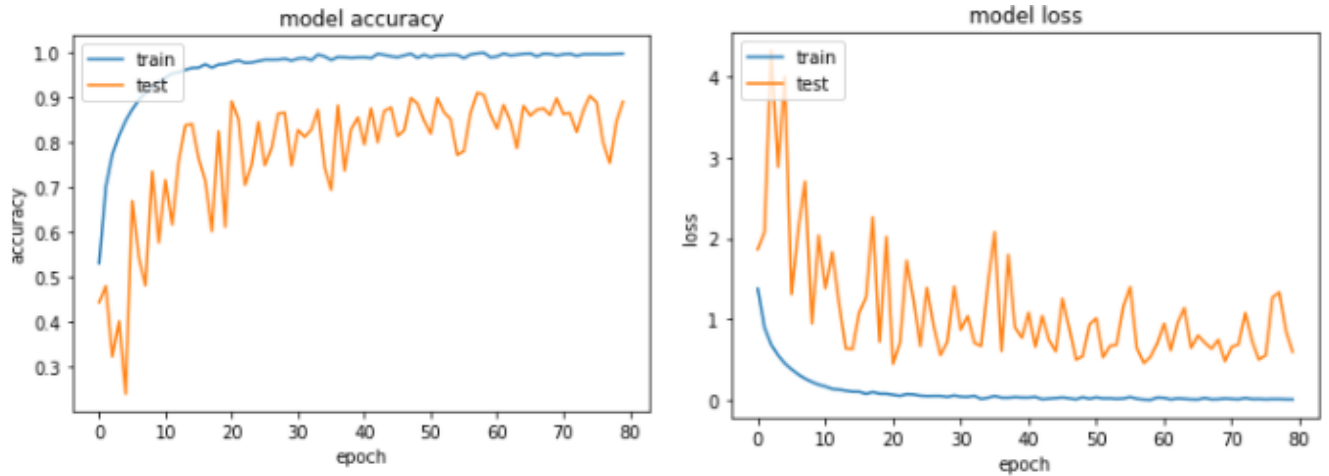
```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3)
```

```
history = model.fit(X_train, y_train, batch_size=32, epochs=80, validation_split=0.2)
```

Accuracy of model is 88.16%

```
#모델 정확도 출력
print("정확도 : %.4f" % (model.evaluate(X_test, y_test)[1]))
```

```
239/239 [==============================] - 7s 28ms/step - loss: 0.6787 - accuracy: 0.8816
정확도 : 0.8816
```



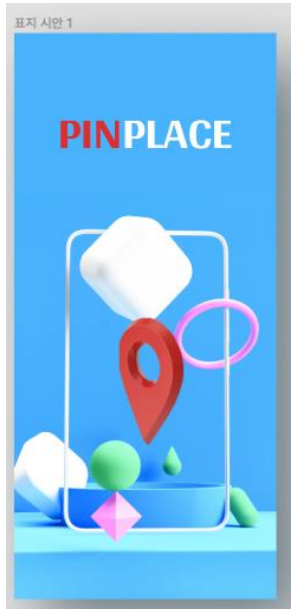
03. Implementation – front end

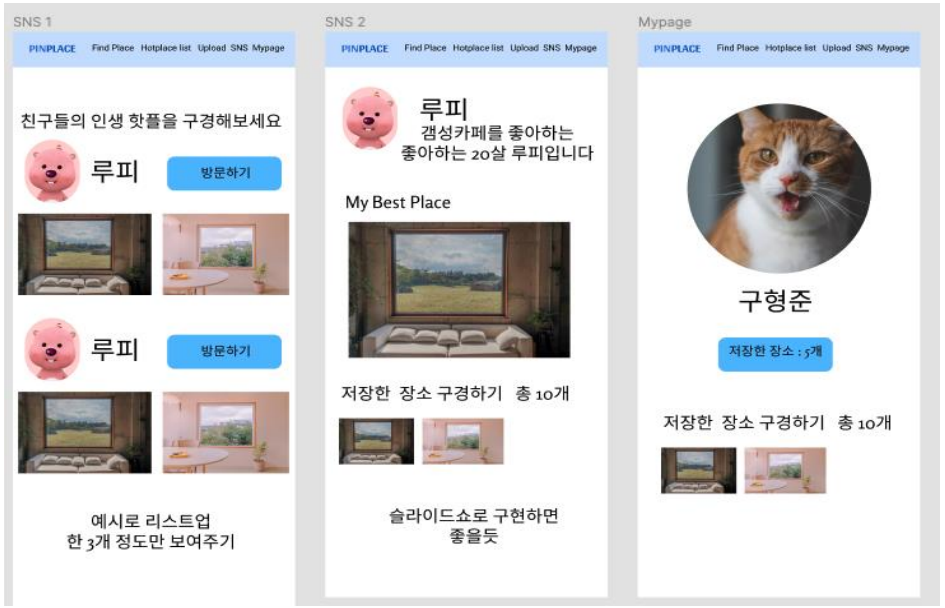
Front end

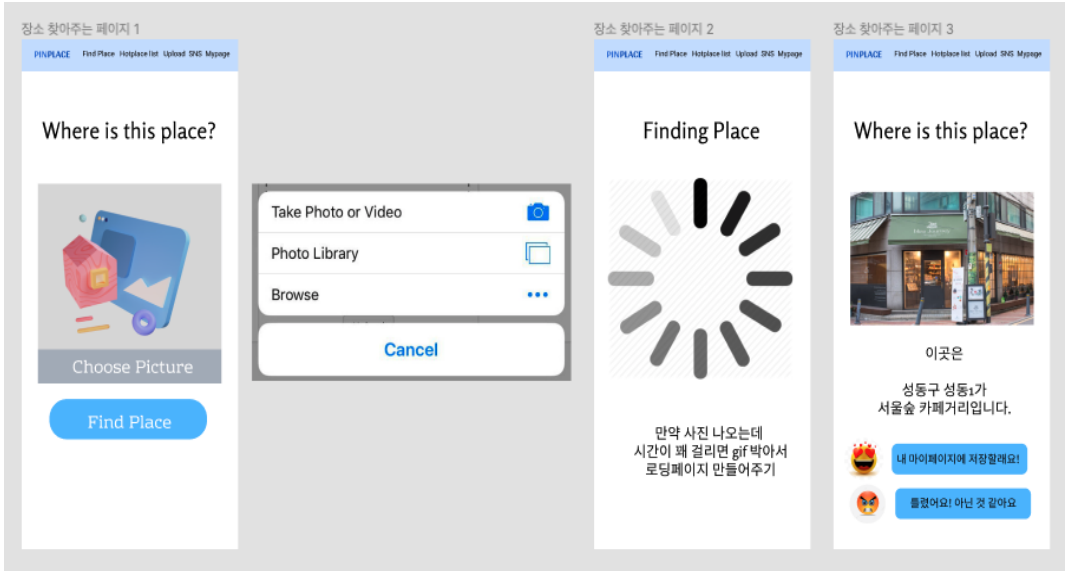
UI Design

- **The initial design Was made using Figma**

10/10 Completed







03. Implementation – front end

10 Pages

- Cover Page — 10/17 Completed
- Start page — 10/24 Completed
- Guide Page — 11/2 Completed
- Login Page — 11/2 Completed
- Find Location Page — 10/31 Completed
- List Up Page — 11/2 Completed
- Upload picture Page — 10/31 Completed
- SNS Page — 10/31 Completed Partly
- My Page — Not yet

Coming Soon

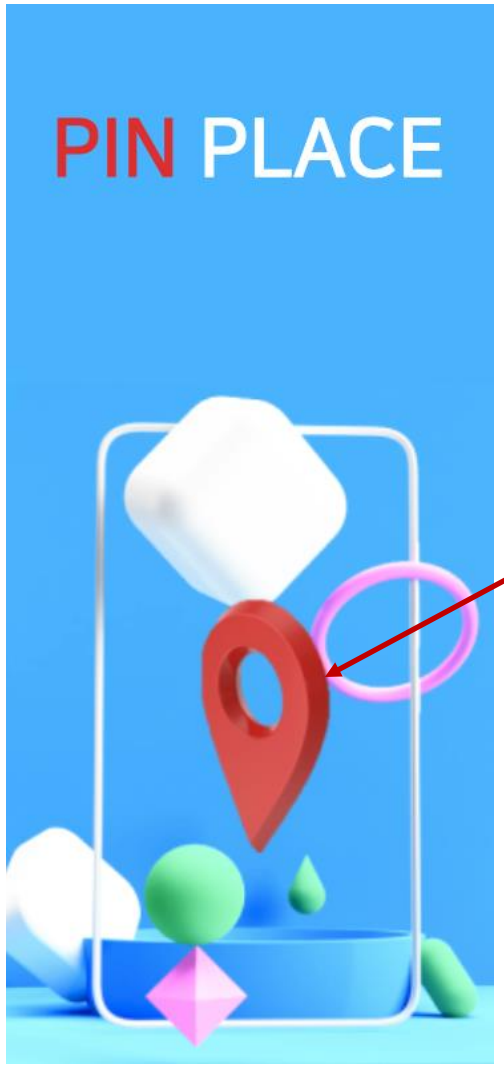
-> SNS & MY Page Implementation Perfectly

-> Refactoring & Connect Backened Part Perfectly

How to develop?

03. Implementation – front end

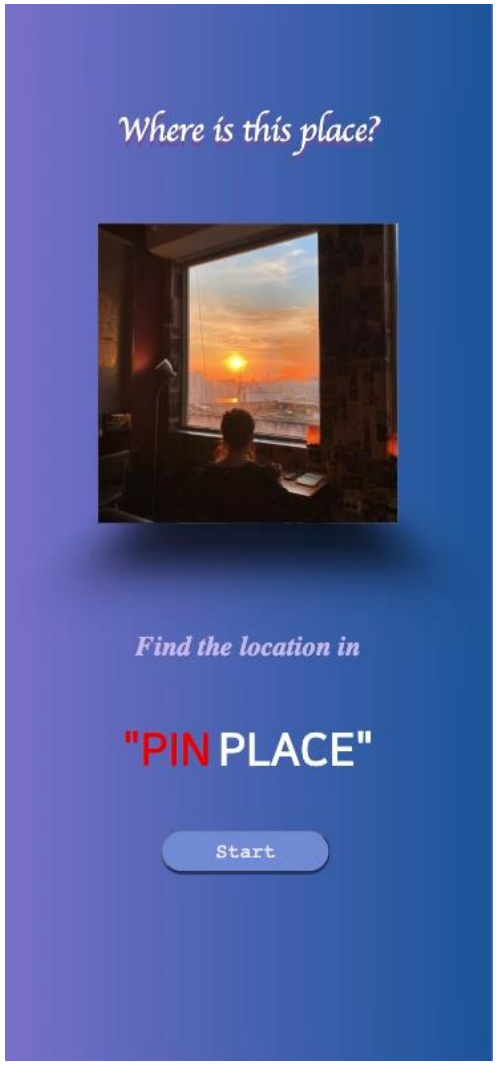
▪ Cover Page



➢ Instruction

- Found 3d graphic asset and placed it myself.
- Top -> ”PIN PLACE”
  : Our service’s Logo
- Location Pin 3d Graphic
  : included in ”Onclick Function” that could move next page
  (onclick="location.href= ' ./start.html’”)

▪ Start Page



➢ Instruction

- Found 3d graphic asset and placed it myself.
- Top -> ”PIN PLACE”
  : Our service’s Logo
- Location Pin 3d Graphic
  : included in ”Onclick Function” that could move next page
  (onclick="location.href= ' ./start.html’”)

03. Implementation – front end

▪ Guide Page

➢ Instruction

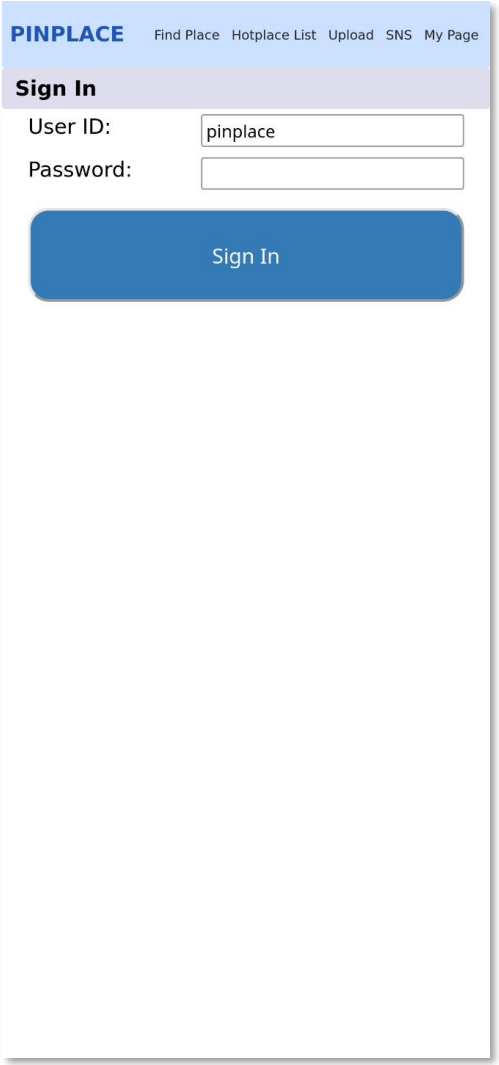
- For optimal UX, We made User Guide Page with card UI
- Every time user turn the page, the content and design are designed to be different
- Implement ‘PREV, NEXT, FINISH’

-PREV : onclick function that move previous card

-NEXT : onclick function that move next card

-Start : onclick function that connect login page

▪ Log-in Page

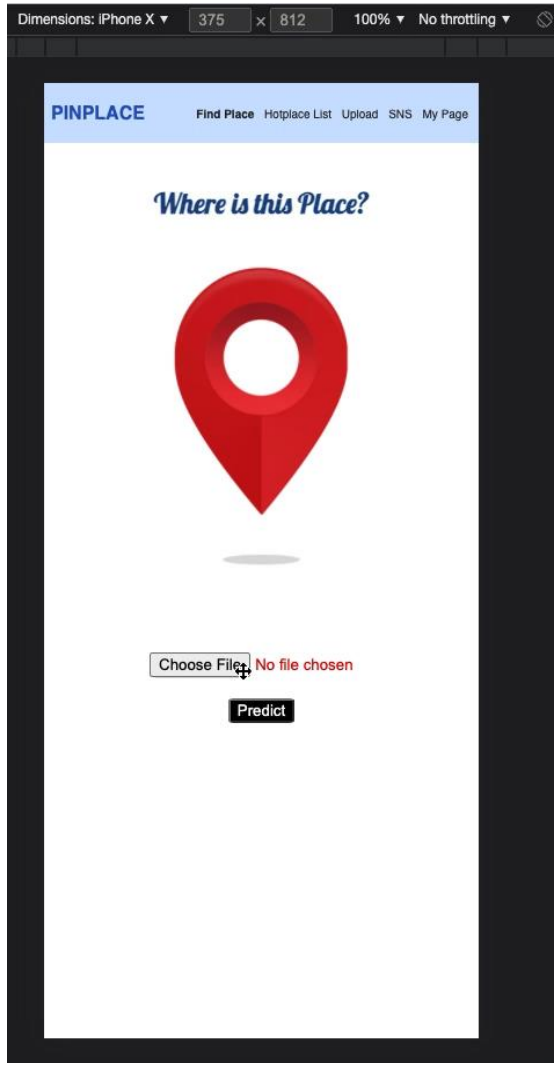


➢ Instruction

- A simple log-in form
- Checks if ID and password are valid
- Not really authenticating for now

03. Implementation – front end

▪ Find Location Page

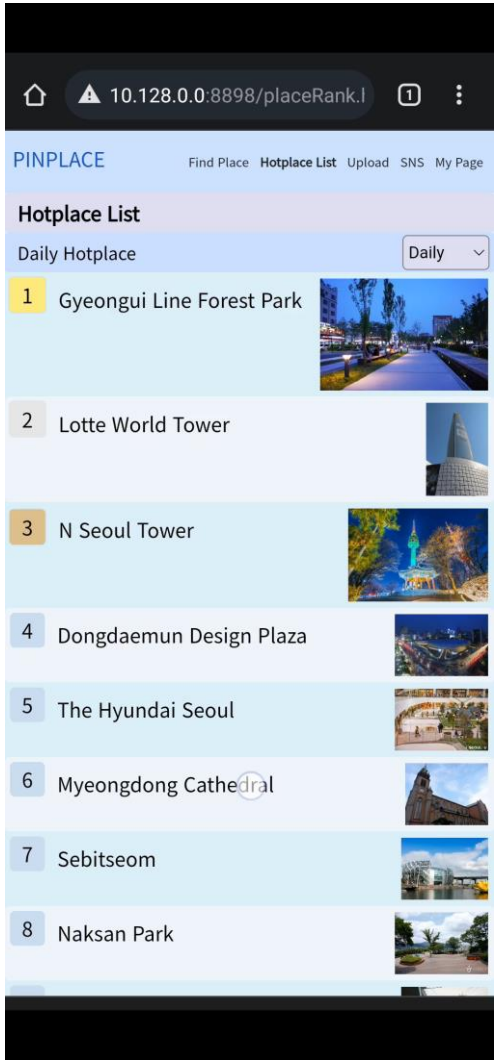


➢ Instruction

- Core Function Page
- Connect with CNN Model that we made ourselves
- User Flow

Click Choose File Button
-> Put Input File
(regardless of file's extension)
-> Click Predict Button
-> Appear Output(location)

▪ List Up Page



➢ Instruction

- Sort by popularity
  - Daily, Weekly, Monthly…
  - Actual measurement not implemented
- Page for each place:
  - 'ejs' Node.js module for automated generation
  - Images sorted by uploaded time

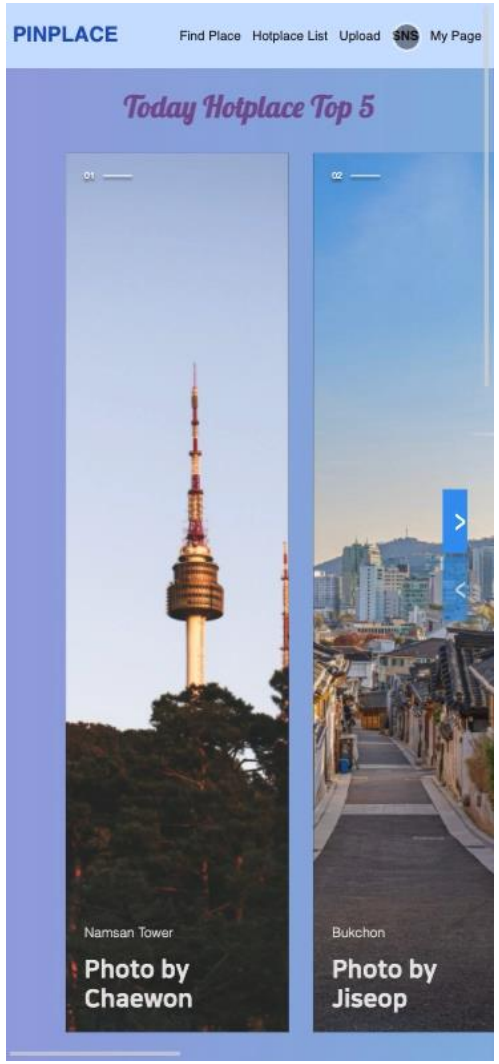
03. Implementation – front end

▪ Upload Page

➢ Instruction

- With predefined locations
- Image preview right after file selection
- Submitted files are stored in a server with location identifiers

▪ SNS Page



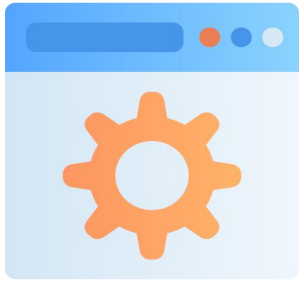
➢ Instruction

- Partly Implementation (now)
- Will implement function more after this week
- Develop Image Slider that moves automatically (with js code)
- Implement ‘PREV, NEXT BTN’

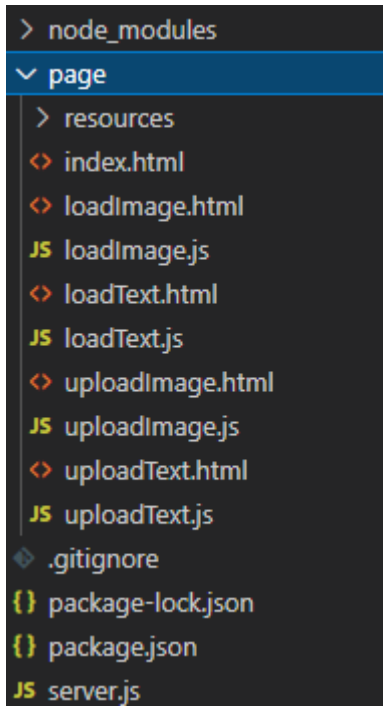
-PREV(<) : onclick function that move previous card

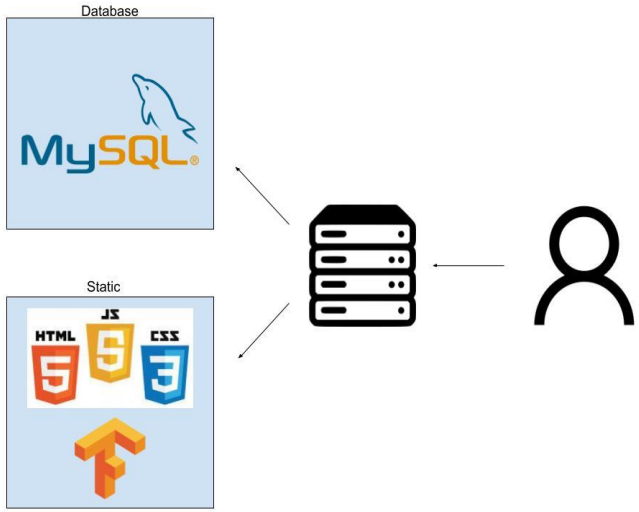
-NEXT(>) : onclick function that move next card

03. Implementation – back end



Back end






```
app.post("/textupload", (req, res) => {
  console.log(req.body.id);
  console.log(req.body.content);
  console.log(req.body.test);
  console.log(`Received request to post Text from ${req.body.id}`);
  const uploadText = (id, cont) => {
    con.query(
      `INSERT INTO cn_userdatabase.text (iduser, content) VALUES (${id},"${cont}")`,
      (serr, sres, sfield) => {
        if (serr) {
          res.end("failed");
          throw serr;
        }
      }
    );
  };
  uploadText(req.body.id, req.body.content);
  res.end("success");
});
```

Uploading data into MySQL

- Server sends static files to user
- Requests to upload or load use the database
- Currently response is just set to “success”
- Better method would be to have a formatted json file to use as response

03. Implementation – back end

server.js

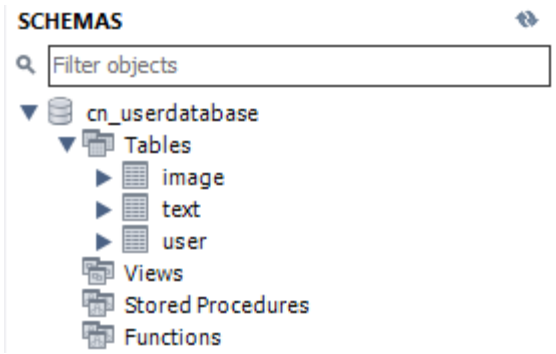
BackEnd > SimpleSNS > server.js > ...

```
const express = require("express");
const mysql = require("mysql");
const fs = require("fs");
const multer = require("multer");
let storage = multer.diskStorage({
  destination: (req, file, cb) => {
    cb(null, "./page/resources");
  },
  filename: (req, file, cb) => {
    cb(null, `${Date.now()}-${file.originalname}`);
  },
});
let upload = multer({ storage: storage });
const PORT = 8899;

const con = mysql.createConnection({
  host: "localhost",
  user: "",
  password: "",
  database: "cn_userdatabase",
  multipleStatements: true,
});

con.connect((e) => {
  if (e) throw e;
  console.log("Connected to mySQL");
});

let app = express();
app.use((req, res, next) => {
  console.log(`${new Date()}: ${req.method}=>${req.url}`);
  next();
});
app.use(express.urlencoded({ extended: false, limit: "50mb" }));
```

- Open Http Server with set port
- Connect to MySQL Database
- Add response to requests

- Still at prototype state
- Does not offer use of requests with params
- Code too long should make more modular

03. Implementation – back end



	idimage	iduser	image
▶	1	1	./resources/1635584018693-1-lot.jpg
*	NULL	NULL	NULL

	idtext	iduser	content
▶	1	1	abc
	2	2	abc
	3	1	hello world
*	NULL	NULL	NULL

	iduser	username	password	isAdmin
▶	1	admin	admin	1
	2	testid	testpw	0
*	NULL	NULL	NULL	NULL

```sql
DROP SCHEMA IF EXISTS `cn_userdatabase`;
CREATE SCHEMA `cn_userdatabase`;

USE `cn_userdatabase`;
CREATE TABLE `user` (
    `iduser` INT NOT NULL AUTO_INCREMENT,
    `username` VARCHAR(45) NOT NULL,
    `password` VARCHAR(45) NOT NULL,
    `isAdmin` TINYINT NOT NULL DEFAULT 0,
    PRIMARY KEY(`iduser`)
);

INSERT INTO `user` (`username`, `password`, `isAdmin`) VALUES ("admin", "admin", "1");
INSERT INTO `user` (`username`, `password`, `isAdmin`) VALUES ("testid", "testpw","0");

CREATE TABLE `text` (
    `idtext` INT NOT NULL AUTO_INCREMENT,
    `iduser` INT NOT NULL,
    `content` VARCHAR(500) NOT NULL,
    PRIMARY KEY(`idtext`)
);

CREATE TABLE `image` (
    `idimage` INT NOT NULL AUTO_INCREMENT,
    `iduser` INT NOT NULL,
    `image` VARCHAR(500) NOT NULL,
    PRIMARY KEY(`idimage`)
);
```

- Currently stored data is simple
- SQL does not store the image but instead its path
- SQL script for setup and resetting the database was made

“Challenges in CNN building”

Collect image dataset

- We have small amount of dataset, about total 6,000 images in 10 classes
- So, we tried k-fold cross validation. However, this can not help to improve the accuracy
- So, we tried data augmentation. We have total 17,815 images for training data set

Low accuracy

- At first, we made simple CNN models and this have low accuracy, about 60%
- So, we tried to various models. But Alexnet and VGG16 model has too many parameters to learn. We choose Resnet50 model which has relatively low parameters about 23,000,000
- Then we can have quite great accuracy

“Challenges in Front end”

Difficulties in Designing

- Designing is difficult for us who are not UI/UX designers
- Testing with multiple devices are necessary to work on various environments
- Had to deal with backend stuff for functions interacting with server

“Challenges in Back end”

Initial Plan

```
const PORT = 8899;
const userID = 1; //set default admin
const btn = document.querySelector(".post");
const content = document.querySelector(".content");
btn.addEventListener("click", () => {
  console.log("button clicked");
  console.log(content.value);
  const xhr = new XMLHttpRequest();
  xhr.open("POST", `http://localhost:${PORT}/textupload`);
  xhr.setRequestHeader("Content-Type", "application/x-www-form-urlencoded");
  xhr.send(`id=${userID}&content=${content.value}&test=abc`);
  content.value = "";
  console.log("success");
});

```

Upload Code format for uploading text to database

- For simplicity did not use forms
- Sends encoded requests with content and id

04. Challenges - CNN

“Challenges in Back end”

Problems

- Could not translate encoded data to image correctly
- Saving large data into SQL did not seem efficient

Solution

- Use forms to encode the data
- Encoded data is read and saved through multer
- SQL saves the path to file

Actual Code

```
const imgSelector = document.querySelector(".img");
const btn = document.querySelector(".post");
const form = document.querySelector(".sendform");
const PORT = 8899;
const userId = 1;
let url;
let blob;

form.addEventListener("submit", (event) => {
  event.preventDefault();
  let data = new FormData(event.target);
  console.log(data);
  console.log(data.get("img-file"));
  console.log(data.get("img-file").name);
  console.log(data.get("img-file").stream());
  data.set(
    "img-file",
    data.get("img-file"),
    `${userId}-${data.get("img-file").name}`
  );

  //add additional content here
  data.append("test", "test");
  data.append("test2", "test2");

  console.log(data.get("test"));
  const xhr = new XMLHttpRequest();
  xhr.open("POST", "/imguploadform");
  xhr.onload = () => {
    console.log(xhr.responseText);
  };
  xhr.send(data);
});
```

“Limitation in current project”

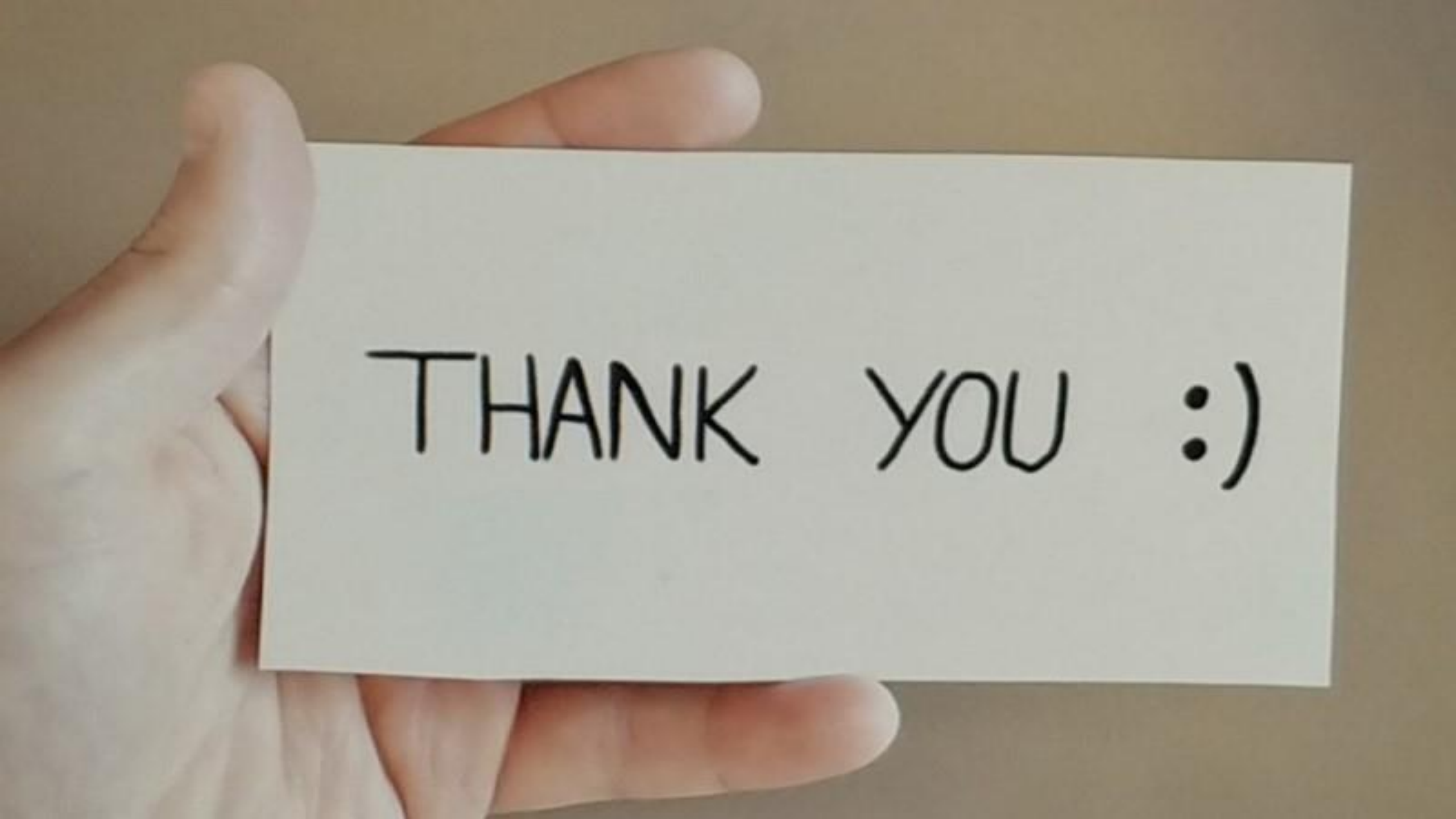
Users give only valid inputs

- Assumption that users are good and give only valid inputs
  - Every forms are vulnerable so that the server may stop for invalid inputs
  - Will cover exceptional cases and “evil user” later

CNN models accuracy

- The CNN model still does not have good accuracy
  - The model cannot predict some of pictures correctly.
  - So, next week, we will use confusion matrix to find the problem

Demo


THANK YOU :)