ST1505:

Are you smarter than an AI?

Soh Hong Yu
P2100775
DAAA/FT/2B/01

Table of Contents

DevOps Process

MLOps

Model Development

Web Application Development

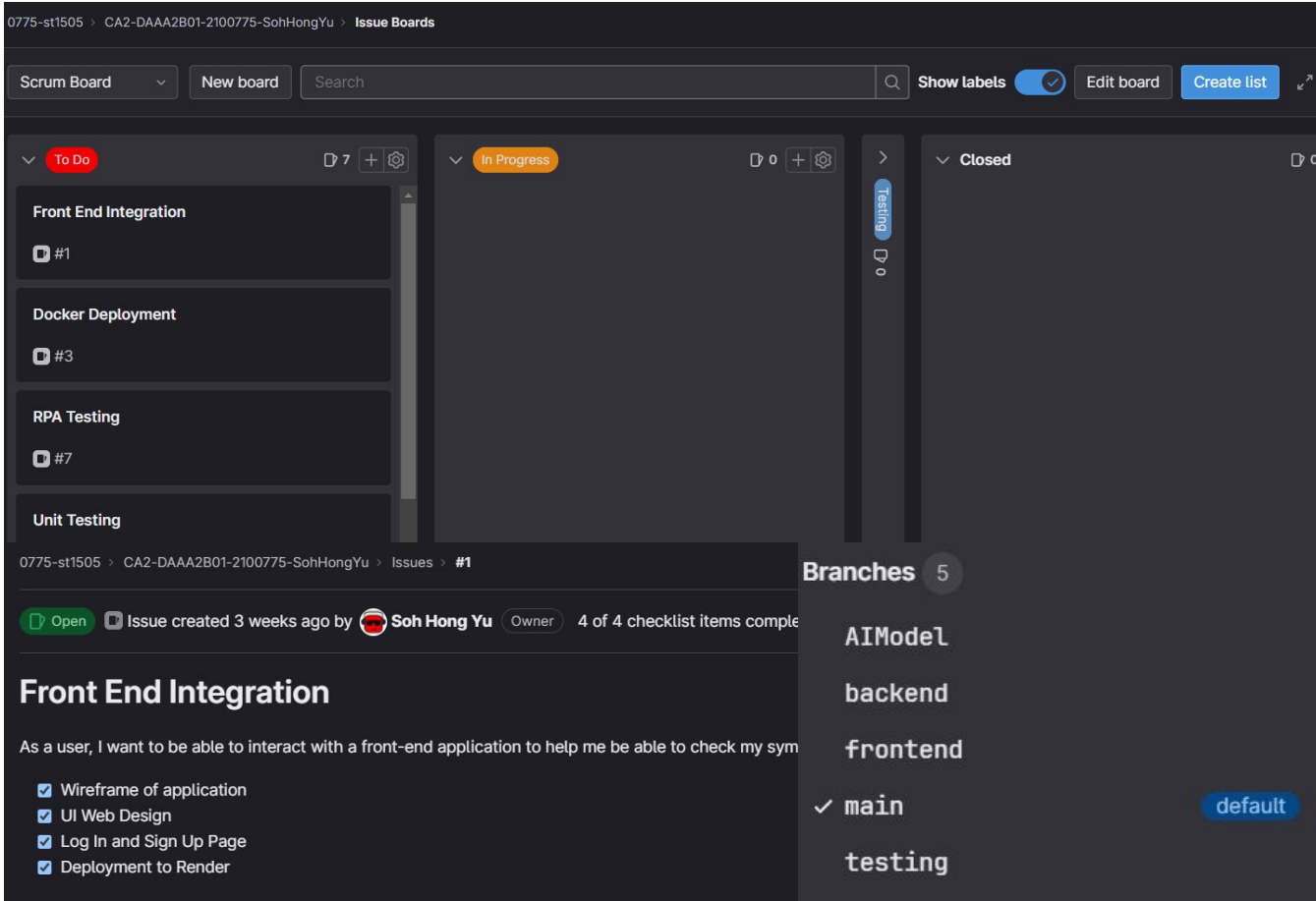
Automatic Testing

Robotic Process Automation

DevOps Process

Before developing the application, we need to do some scrum boarding. This allows us to keep track of our application development. [To Do, In Progress and Testing Labels]

After doing the scrum board, it is time to make branches. This allow us to make changes to the folder in a controlled environment.
[No of Branches: 5]



Model Development

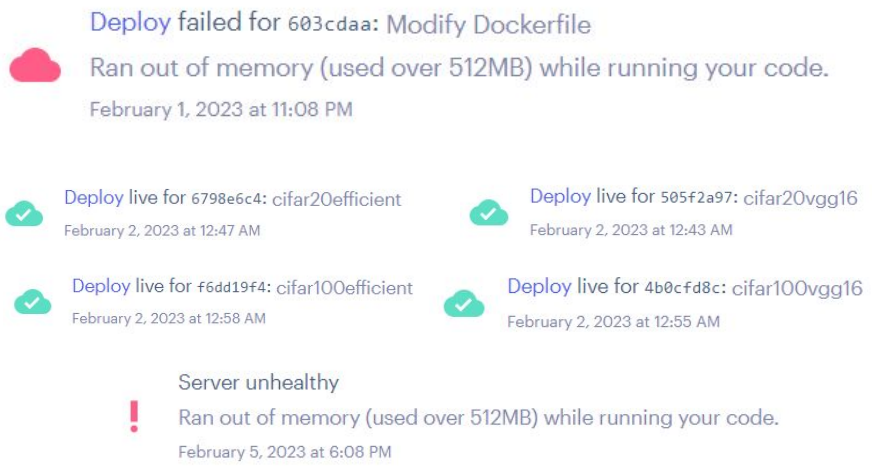
Due to Render's lack of memory and the size of the Deep Learning(DL) models, I have separated the DL models into 4 separate Render URL.

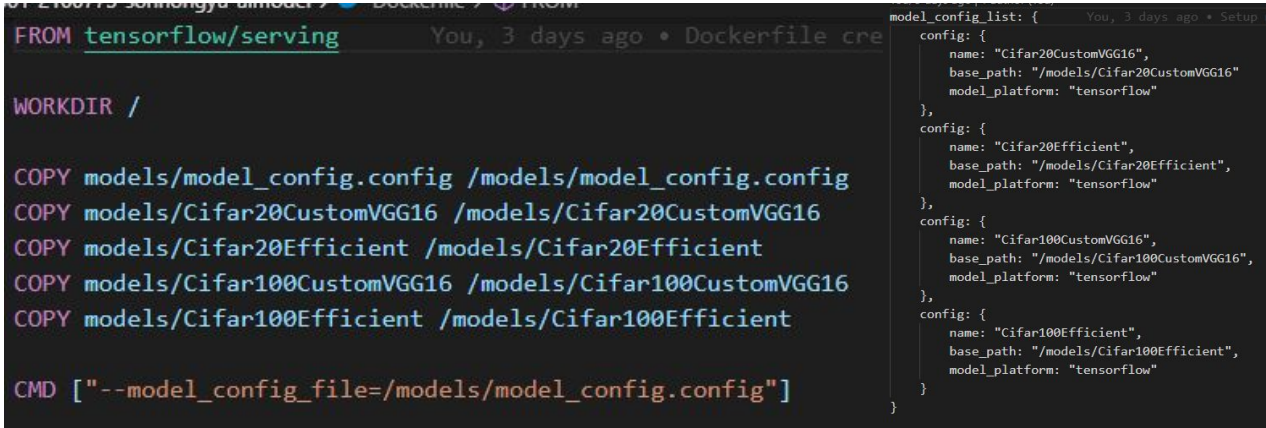
But I have also deploy the 4 models together on LOCAL DockerFile into the same URL.

We need to note that Render's processing and computational power is not very strong. Models like EfficientNetV2 and ResNetV2 will have issues with memory!

DL Models used:

1. **Fine**
   a. VGG16
   b. EfficientNetV2
2. **Coarse**
   a. VGG16
   b. EfficientNetV2





Automatic Testing

Unexpected Failure Testing

We create a test for unexpected failure testing using some of the data and send to the SQLLite Table.

We can see that all the results has passed as we did not expect any failure.

```
tests/test_application.py::test_EntryClass[predictionList0] PASSED
tests/test_application.py::test_EntryClass[predictionList1] PASSED
tests/test_application.py::test_EntryClass[predictionList2] PASSED
tests/test_application.py::test_EntryClass[predictionList3] PASSED
```

```python
@pytest.mark.parametrize("predictionList", [
    [1, "./application/static/images/saved/001586-11802-butterfly-insects.png",
    "vgg16", "fine", 15],
    [1, "./application/static/images/saved/001586-11802-butterfly-insects.png",
    "vgg16", "coarse", 15],
    [1, "./application/static/images/saved/001586-11802-butterfly-insects.png",
    "efficient", "fine", 15],
    [1, "./application/static/images/saved/001586-11802-butterfly-insects.png",
    "efficient", "coarse", 15]
])
# 3: Write the test function pass in the arguments1231
def test_EntryClass(predictionList, capsys):
    with capsys.disabled():
        now = datetime.datetime.utcnow()
        new_entry = Entry(
            user=predictionList[0],
            filePath=predictionList[1],
            modelType=predictionList[2],
            dataType=predictionList[3],
            prediction=predictionList[4],
            predicted_on=now)
        assert new_entry.user == predictionList[0]
        assert new_entry.filePath == predictionList[1]
        assert new_entry.filePath[-4:] == ".png" or new_entry.filePath[-4:
                                                                       ] == ".jpg" or new_entry.filePath[-5:] == ".jpeg"
        assert new_entry.modelType == predictionList[2]
        assert new_entry.modelType == "vgg16" or new_entry.modelType == "efficient"
        assert new_entry.dataType == predictionList[3]
        assert new_entry.dataType == "fine" or new_entry.dataType == "coarse"
        assert new_entry.prediction == predictionList[4]
        if new_entry.dataType == "fine":
            assert new_entry.prediction >= 0 and new_entry.prediction < 100
        else:
            assert new_entry.prediction >= 0 and new_entry.prediction < 20
        assert new_entry.predicted_on == now
```

Expected Failure Testing

We will be using the same function as the unexpected failure testing. This allows us to moderate and check for any of the data has any outliers that was not validated.

Issues that might occurs:

1. What happens if imgPath is invalid?
2. What happens if modelType is not equal to “vgg16" / "efficient"?
3. What happens if dataType is not equal to "fine" / "coarse"?
4. What happens if prediction is outside the range fine[0 - 19] and coarse [0 - 99]?

```
tests/test_application.py::test_EntryValidation[predictionList0] XFAIL (arguments fail due to testing)
tests/test_application.py::test_EntryValidation[predictionList1] XFAIL (arguments fail due to testing)
tests/test_application.py::test_EntryValidation[predictionList2] XFAIL (arguments fail due to testing)
tests/test_application.py::test_EntryValidation[predictionList3] XFAIL (arguments fail due to testing)
tests/test_application.py::test_EntryValidation[predictionList4] XFAIL (arguments fail due to testing)
tests/test_application.py::test_EntryValidation[predictionList5] XFAIL (arguments fail due to testing)
tests/test_application.py::test_EntryValidation[predictionList6] XFAIL (arguments fail due to testing)
tests/test_application.py::test_EntryValidation[predictionList7] XFAIL (arguments fail due to testing)
tests/test_application.py::test_EntryValidation[predictionList8] XFAIL (arguments fail due to testing)
tests/test_application.py::test_EntryValidation[predictionList9] XFAIL (arguments fail due to testing)
tests/test_application.py::test_EntryValidation[predictionList10] XFAIL (arguments fail due to testing)
tests/test_application.py::test_EntryValidation[predictionList11] XFAIL (arguments fail due to testing)
tests/test_application.py::test_EntryValidation[predictionList12] XFAIL (arguments fail due to testing)
tests/test_application.py::test_EntryValidation[predictionList13] XFAIL (arguments fail due to testing)
tests/test_application.py::test_EntryValidation[predictionList14] XFAIL (arguments fail due to testing)
tests/test_application.py::test_EntryValidation[predictionList15] XFAIL (arguments fail due to testing)
tests/test_application.py::test_EntryValidation[predictionList16] XFAIL (arguments fail due to testing)
tests/test_application.py::test_EntryValidation[predictionList17] XFAIL (arguments fail due to testing)
tests/test_application.py::test_EntryValidation[predictionList18] XFAIL (arguments fail due to testing)
tests/test_application.py::test_EntryValidation[predictionList19] XFAIL (arguments fail due to testing)
```

```python
@pytest.mark.xfail(reason="arguments fail due to testing")
@pytest.mark.parametrize("predictionList", [
    # Fail due to invalid file path
    [1, "./application/static/images/saved/001586-11802-butterfly-insects.p",
        "vgg16", "fine", 15],
    [1, "./application/static/images/saved/001586-11802-butterfly-insects.p",
        "efficient", "fine", 15],
    [1, "./application/static/images/saved/001586-11802-butterfly-insects.p",
        "vgg16", "coarse", 15],
    [1, "./application/static/images/saved/001586-11802-butterfly-insects.p",
        "efficient", "coarse", 15],
    # Fail due to modelType not equal to "vgg16"/"efficient"
    [1, "./application/static/images/saved/001586-11802-butterfly-insects.png",
        "1231vgg16", "fine", 15],
    [1, "./application/static/images/saved/001586-11802-butterfly-insects.png",
        "1231efficient", "fine", 15],
    [1, "./application/static/images/saved/001586-11802-butterfly-insects.png",
        "1231vgg16", "coarse", 15],
    [1, "./application/static/images/saved/001586-11802-butterfly-insects.png",
        "1231efficient", "coarse", 15],
    # Fail due to dataType is not equal to "fine"/"coarse"
    [1, "./application/static/images/saved/001586-11802-butterfly-insects.png",
        "vgg16", "1231fine", 15],
    [1, "./application/static/images/saved/001586-11802-butterfly-insects.png",
        "efficient", "1231fine", 15],
    [1, "./application/static/images/saved/001586-11802-butterfly-insects.png",
        "vgg16", "1231coarse", 15],
    [1, "./application/static/images/saved/001586-11802-butterfly-insects.png",
        "efficient", "1231coarse", 15],
    # Fail due to prediction is outside the range fine[0 - 19] and coarse [0 - 99]
    [1, "./application/static/images/saved/001586-11802-butterfly-insects.png",
        "vgg16", "fine", -1],
    [1, "./application/static/images/saved/001586-11802-butterfly-insects.png",
        "efficient", "fine", -1],
    [1, "./application/static/images/saved/001586-11802-butterfly-insects.png",
        "vgg16", "coarse", -1],
    [1, "./application/static/images/saved/001586-11802-butterfly-insects.png",
        "efficient", "coarse", -1],
    [1, "./application/static/images/saved/001586-11802-butterfly-insects.png",
        "vgg16", "fine", 100],
    [1, "./application/static/images/saved/001586-11802-butterfly-insects.png",
        "efficient", "fine", 100],      You, now • Uncommitted changes
    [1, "./application/static/images/saved/001586-11802-butterfly-insects.png",
        "vgg16", "coarse", 20],
    [1, "./application/static/images/saved/001586-11802-butterfly-insects.png",
        "efficient", "coarse", 20],
])
def test_EntryValidation(predictionList, capsys):
    test_EntryClass(predictionList, capsys)
```

Consistency Testing

We group arrays into 3 array of random index. The array in an array will be looped through, then the values are feed into the AI Model to test if the output prediction is consistent.

```
tests/test_application.py::test_predictAPI[bigPredictionList0] /static
PASSED
tests/test_application.py::test_predictAPI[bigPredictionList1] /static
PASSED
tests/test_application.py::test_predictAPI[bigPredictionList2] /static
PASSED
tests/test_application.py::test_predictAPI[bigPredictionList3] /static
PASSED
```

```python
@pytest.mark.parametrize("bigPredictionList", [
    [[1, "./application/static/images/saved/992451-19607-wolf-large carnivores.png",
      "vgg16", "fine", 97],
     [1, "./application/static/images/saved/992451-19607-wolf-large carnivores.png",
      "vgg16", "fine", 97],
     [1, "./application/static/images/saved/992451-19607-wolf-large carnivores.png",
      "vgg16", "fine", 97]
     ],
    [[1, "./application/static/images/saved/854316-13930-trout-fish.png",
      "vgg16", "coarse", 1],
     [1, "./application/static/images/saved/854316-13930-trout-fish.png",
      "vgg16", "coarse", 1],
     [1, "./application/static/images/saved/854316-13930-trout-fish.png",
      "vgg16", "coarse", 1], ],
    [[1, "./application/static/images/saved/775114-11802-butterfly-insects.png",
      "efficient", "fine", 14],
     [1, "./application/static/images/saved/775114-11802-butterfly-insects.png",
      "efficient", "fine", 14],
     [1, "./application/static/images/saved/775114-11802-butterfly-insects.png",
      "efficient", "fine", 14]
     ],
    [[1, "./application/static/images/saved/350882-43714-chimpanzee-large omnivores and herbivores.png",
      "efficient", "coarse", 11],
     [1, "./application/static/images/saved/350882-43714-chimpanzee-large omnivores and herbivores.png",
      "efficient", "coarse", 11],
     [1, "./application/static/images/saved/350882-43714-chimpanzee-large omnivores and herbivores.png",
      "efficient", "coarse", 11], ],
])
def test_predictAPI(client, bigPredictionList, capsys):
    predictOutput = []
    for predictionList in bigPredictionList:
        with capsys.disabled():
            with open(predictionList[1], "rb") as f:
                encoded_string = base64.b64encode(f.read()).decode('utf-8')
                encoded_string = "" + encoded_string
            predictData = {
                "user": predictionList[0],
                "imageBlob": encoded_string,
                "imageName": predictionList[1].split("/")[-1],
                "model": predictionList[2],
                "dataset": predictionList[3],
                "prediction": predictionList[4],
            }
            response = client.post('/api/predict',
                                   data=json.dumps(predictData),
                                   content_type="application/json",)
            # check the outcome of the action
            assert response.status_code == 200
            assert response.headers["Content-Type"] == "application/json"
            response_body = json.loads(response.get_data(as_text=True))
            assert response_body["id"]
            predictOutput.append(response_body["prediction"])
        assert len(set(predictOutput)) <= 1
```

Validation Failure Testing

We check if there is any issue or validation error in the parameter. We also validate and ensure the data is inside of the database beforehand.

We see as the first is a valid email and password, it is a XPASS while the rest have issues and are XFAIL

```
tests/test_auth.py::test_loginAPI[logInInfo0] /static
XPASS (Not Valid Username or Password)
tests/test_auth.py::test_loginAPI[logInInfo1] /static
XFAIL (Not Valid Username or Password)
tests/test_auth.py::test_loginAPI[logInInfo2] /static
XFAIL (Not Valid Username or Password)
tests/test_auth.py::test_loginAPI[logInInfo3] /static
XFAIL (Not Valid Username or Password)
```

```python
# Test Login API
# Validation Test
@pytest.mark.xfail(reason="Not Valid Username or Password")
@pytest.mark.parametrize("logInInfo", [
  ["sohhongyu@gmail.com","123", 0], # correct email and password
  ["sohhongyu123@gmail.com", "123", 1],  # Invalid credentials
  ["sohhongyu@gmail.com","123123", 1], # correct email but wrong password
  ["devops@gmail.com", "123", 1] # Invalid credentials
]
)
def test_loginAPI(client, logInInfo, capsys):
    with capsys.disabled():
        # prepare the data into a dictionary
        logInData = {
            "email": logInInfo[0],
            "password": logInInfo[1]
        }
    response = client.post('/api/login',
                           data=json.dumps(logInData),
                           content_type="application/json",)
    # check the outcome of the action
    assert response.status_code == 200
    assert response.headers["Content-Type"] == "application/json"
    response_body = json.loads(response.get_data(as_text=True))
    assert not response_body["isLogin"] == logInInfo[2]
```

Endpoint API Testing - Sign Up

We will be testing the sign up endpoint of the application. We will be testing a new email and a user that already exist in the database.

```
tests/test_auth.py::test_signUpAPI[details0] /static
XPASS (User already exist)
tests/test_auth.py::test_signUpAPI[details1] /static
XFAIL (User already exist)
```

```python
# Test Sign Up API
# Validation Test
@pytest.mark.xfail(reason="User already exist")
@pytest.mark.parametrize("details", [
    ["DevOps", "devops@gmail.com", "123"], # New email
    ["Hong Yu", "sohhongyu@gmail.com", "123"],  # User already exist
]
)
def test_signUpAPI(client, details, capsys):
    with capsys.disabled():
        # prepare the data into a dictionary
        signUpData = {
            "username": details[0],
            "email": details[1],
            "password": details[2]
        }
        logInData = {
            "email": details[1],
            "password": details[2]
        }
    response = client.post('/api/signup',
                           data=json.dumps(signUpData),
                           content_type="application/json",)
    response_body = json.loads(response.get_data(as_text=True))
    # check the outcome of the action
    # assert response.status_code == 200
    # assert response.headers["Content-Type"] == "application/json"

    response = client.post('/api/login',
                           data=json.dumps(logInData),
                           content_type="application/json",)
    # check the outcome of the action
    assert response.status_code == 200
    assert response.headers["Content-Type"] == "application/json"

    response = client.post(f'/user/delete/{response_body["id"]}',
                           content_type="application/json",)
    assert response.status_code == 200
    assert response.headers["Content-Type"] == "application/json"
```

Endpoint API Testing - Add

To add the data, we will be using the “/api/add” endpoint

```
tests/test_application.py::test_addAPI[predictionList0] /static
PASSED
tests/test_application.py::test_addAPI[predictionList1] /static
PASSED
tests/test_application.py::test_addAPI[predictionList2] /static
PASSED
tests/test_application.py::test_addAPI[predictionList3] /static
PASSED
```

```python
# Test Add API


@pytest.mark.parametrize("predictionList", [
    [1, "./application/static/images/saved/001586-11802-butterfly-insects.png",
        "vgg16", "fine", 15],
    [1, "./application/static/images/saved/001586-11802-butterfly-insects.png",
        "vgg16", "coarse", 15],
    [1, "./application/static/images/saved/001586-11802-butterfly-insects.png",
        "efficient", "fine", 15],
    [1, "./application/static/images/saved/001586-11802-butterfly-insects.png",
        "efficient", "coarse", 15]
])
def test_addAPI(client, predictionList, capsys):
    with capsys.disabled():
        # prepare the data into a dictionary
        predictData = {
            "user": predictionList[0],
            "imgPath": predictionList[1],
            "modelType": predictionList[2],
            "dataType": predictionList[3],
            "prediction": predictionList[4],
        }
    response = client.post('/api/add',
                           data=json.dumps(predictData),
                           content_type="application/json",)
    # check the outcome of the action
    assert response.status_code == 200
    assert response.headers["Content-Type"] == "application/json"
    response_body = json.loads(response.get_data(as_text=True))
    assert response_body["id"]
```

Endpoint API Testing - Add

To add the data, we will be using the “/api/add” endpoint

```
tests/test_application.py::test_addAPI[predictionList0] /static
PASSED
tests/test_application.py::test_addAPI[predictionList1] /static
PASSED
tests/test_application.py::test_addAPI[predictionList2] /static
PASSED
tests/test_application.py::test_addAPI[predictionList3] /static
PASSED
```

```python
# Test Add API


@pytest.mark.parametrize("predictionList", [
    [1, "./application/static/images/saved/001586-11802-butterfly-insects.png",
        "vgg16", "fine", 15],
    [1, "./application/static/images/saved/001586-11802-butterfly-insects.png",
        "vgg16", "coarse", 15],
    [1, "./application/static/images/saved/001586-11802-butterfly-insects.png",
        "efficient", "fine", 15],
    [1, "./application/static/images/saved/001586-11802-butterfly-insects.png",
        "efficient", "coarse", 15]
])
def test_addAPI(client, predictionList, capsys):
    with capsys.disabled():
        # prepare the data into a dictionary
        predictData = {
            "user": predictionList[0],
            "imgPath": predictionList[1],
            "modelType": predictionList[2],
            "dataType": predictionList[3],
            "prediction": predictionList[4],
        }
    response = client.post('/api/add',
                           data=json.dumps(predictData),
                           content_type="application/json",)
    # check the outcome of the action
    assert response.status_code == 200
    assert response.headers["Content-Type"] == "application/json"
    response_body = json.loads(response.get_data(as_text=True))
    assert response_body["id"]
```

Endpoint API Testing - Get & Delete

We will look at the different rows to look and see if the output is the same as the one in the parameters

```
tests/test_application.py::test_getAPI[predictionList0] /static
PASSED
tests/test_application.py::test_getAPI[predictionList1] /static
PASSED
tests/test_application.py::test_getAPI[predictionList2] /static
PASSED
tests/test_application.py::test_getAPI[predictionList3] /static
PASSED
```

```python
@pytest.mark.parametrize("predictionList", [
    [8, 1, "./application/static/images/saved/854316-13930-trout-fish.png",
        "vgg16", "coarse", 1],
    [17, 1, "./application/static/images/saved/951416-23530-chair-household furniture.png",
        "vgg16", "fine", 20],
    [10, 1, "./application/static/images/saved/350882-43714-chimpanzee-large omnivores and herbivores.png",
        "efficient", "coarse", 11],
    [11, 1, "./application/static/images/saved/775114-11802-butterfly-insects.png",
        "efficient", "fine", 14],
])
def test_getAPI(client, predictionList, capsys):
    with capsys.disabled():
        response = client.get(f'/api/get/{predictionList[0]}')
        ret = json.loads(response.get_data(as_text=True))
        # check the outcome of the action
        assert response.status_code == 200
        assert response.headers["Content-Type"] == "application/json"
        response_body = json.loads(response.get_data(as_text=True))
        assert response_body["id"] == predictionList[0]
        assert response_body["user"] == predictionList[1]
        assert response_body["filePath"] == predictionList[2]
        assert response_body["modelType"] == predictionList[3]
        assert response_body["dataType"] == predictionList[4]
        assert response_body["prediction"] == predictionList[5]
```

```
tests/test_application.py::test_deleteAPI[predictionList0] /static
PASSED
tests/test_application.py::test_deleteAPI[predictionList1] /static
PASSED
tests/test_application.py::test_deleteAPI[predictionList2] /static
PASSED
tests/test_application.py::test_deleteAPI[predictionList3] /static
PASSED
```

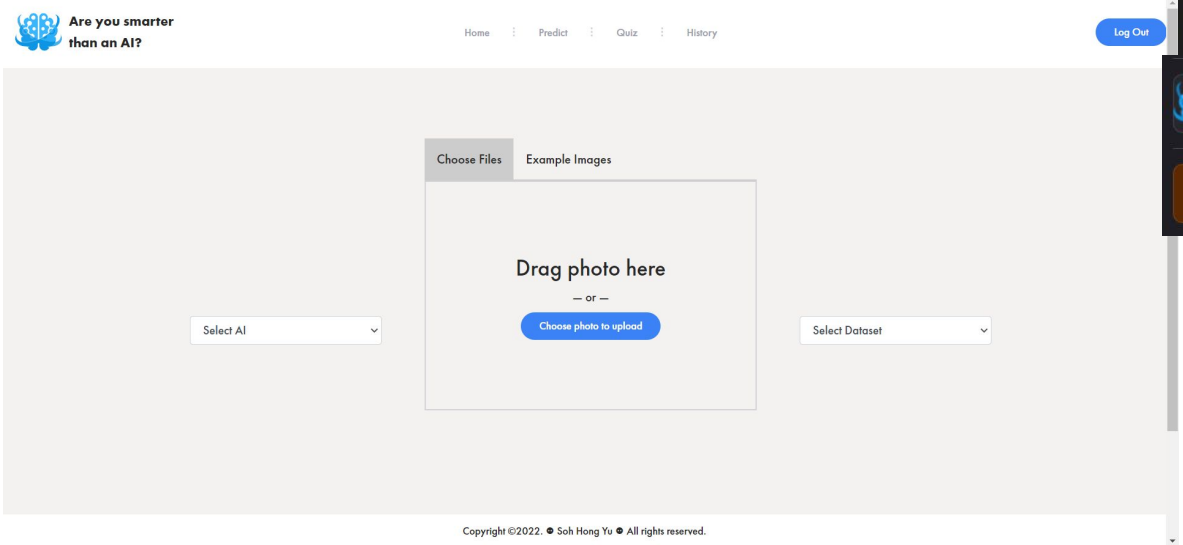
```python
@pytest.mark.parametrize("predictionList", [
    [1, "./application/static/images/saved/001586-11802-butterfly-insects.png",
        "vgg16", "fine", 15],
    [1, "./application/static/images/saved/001586-11802-butterfly-insects.png",
        "vgg16", "coarse", 15],
    [1, "./application/static/images/saved/001586-11802-butterfly-insects.png",
        "efficient", "fine", 15],
    [1, "./application/static/images/saved/001586-11802-butterfly-insects.png",
        "efficient", "coarse", 15]
])
def test_deleteAPI(client, predictionList, capsys):
    with capsys.disabled():
        # prepare the data into a dictionary
        predictData = {
            "user": predictionList[0],
            "imgPath": predictionList[1],
            "modelType": predictionList[2],
            "dataType": predictionList[3],
            "prediction": predictionList[4],
        }
    response = client.post('/api/add',
                           data=json.dumps(predictData),
                           content_type="application/json",)
    response_body = json.loads(response.get_data(as_text=True))
    assert response_body["id"]
    id = response_body["id"]
    response2 = client.get(f'/api/delete/{id}')
    ret = json.loads(response2.get_data(as_text=True))
    # check the outcome of the action
    assert response2.status_code == 200
    assert response2.headers["Content-Type"] == "application/json"
    response2_body = json.loads(response2.get_data(as_text=True))
    assert response2_body["result"] == "ok"
```

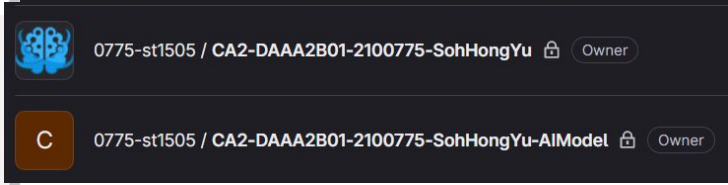
MLOps (Deployment)

To deploy the model / website to the internet, we will be using the platforms Render. The Render will run the Docker files that uses gunicorn as the WSL Server to host the website online.

https://areyousmarterthanai.onrender.com/

```
FROM python:3.8-slim
#update the packages installed in the image
RUN apt-get update -y
# Make a app directory to contain our application
RUN mkdir /app
# Copy every files and folder into the app folder
COPY . /app
# Change our working directory to app fold
WORKDIR /app
# Install all the packages needed to run our web app
RUN pip install -r requirements.txt
# Add every files and folder into the app folder
ADD . /app
# Expose port 5000 for http communication
EXPOSE 5000
# Run gunicorn web server and binds it to the port
CMD gunicorn --bind 0.0.0.0:5000 app:app
```





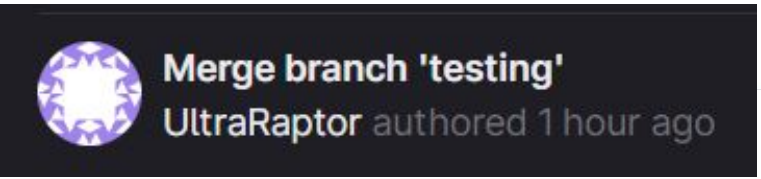
MLOps (CI/CD)

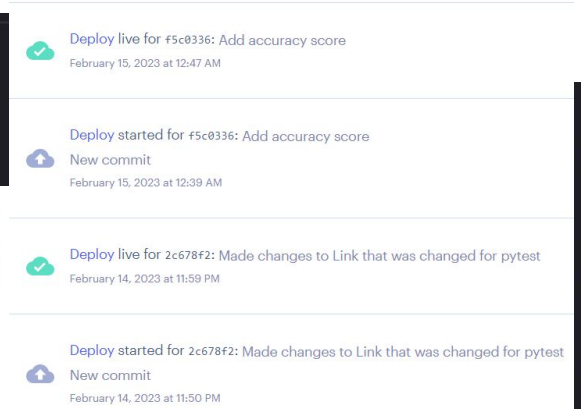
After deploying the models/website on Render, this gives us the opportunity to practice Continuous Integration and Continuous Deployment.

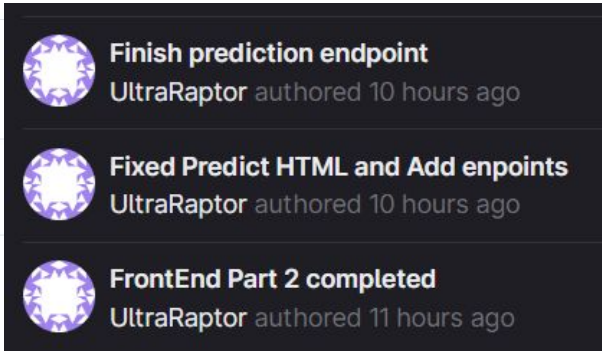
As Render has an Auto-Deployment feature, this allows the application to be deployed when a new push(changes) are made. (CD)

To assist us with Continuous Integration, we will be using a GitLab repository and the different branches to develop features and integrating it into the main branch.

Source: https://docs.aws.amazon.com/codepipeline/latest/userguide/concepts-continuous-delivery-integration.html





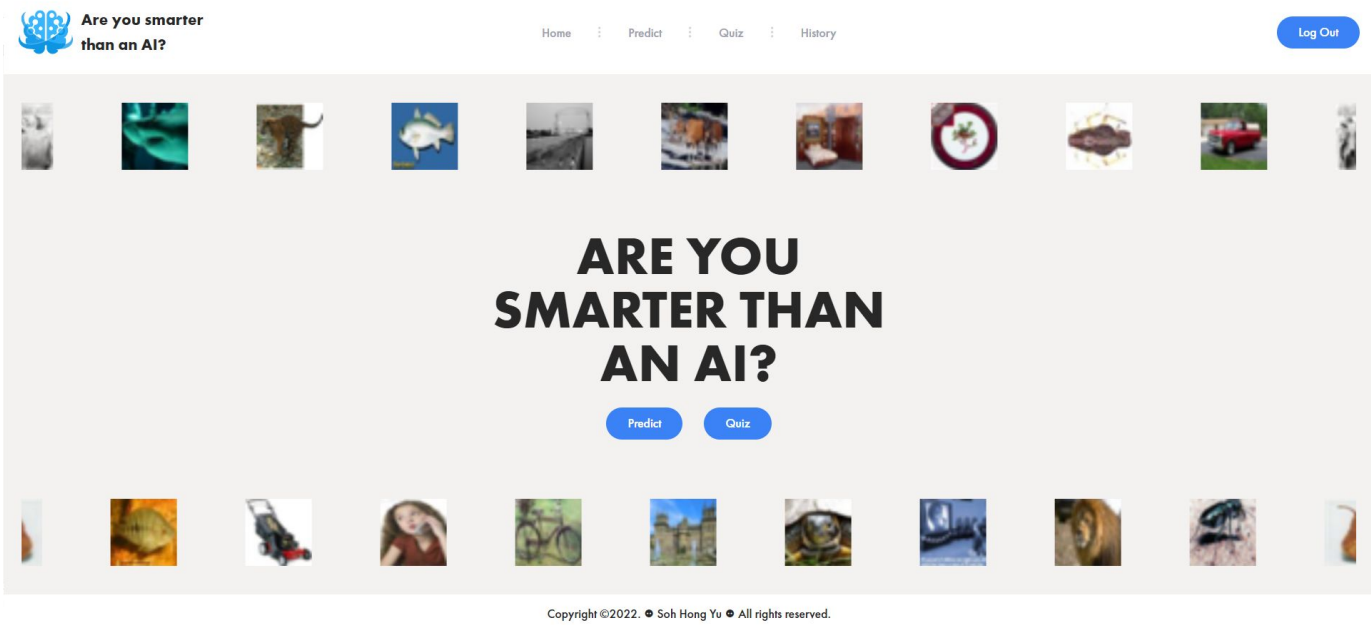


Introducing ... Are You Smarter than an AI?

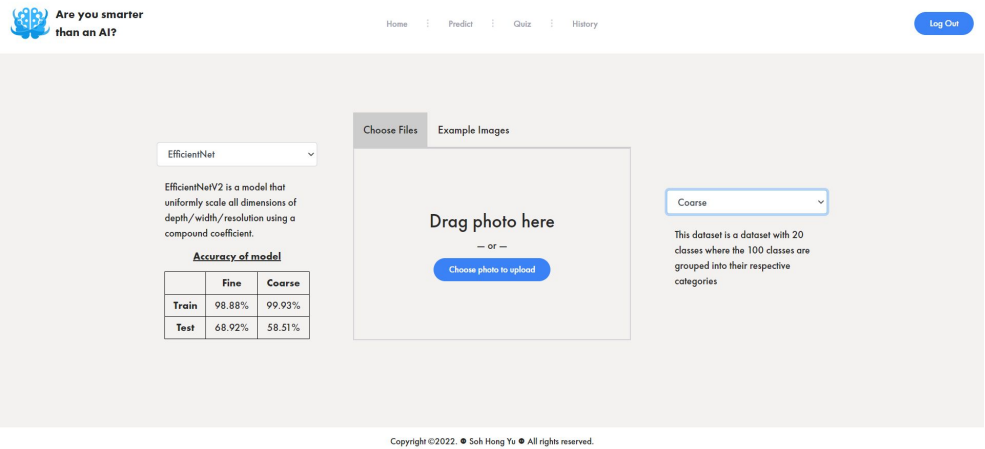
An application that users can use to predict images as well as compete in a quiz with the AI to see who is smarter [You or AI]?

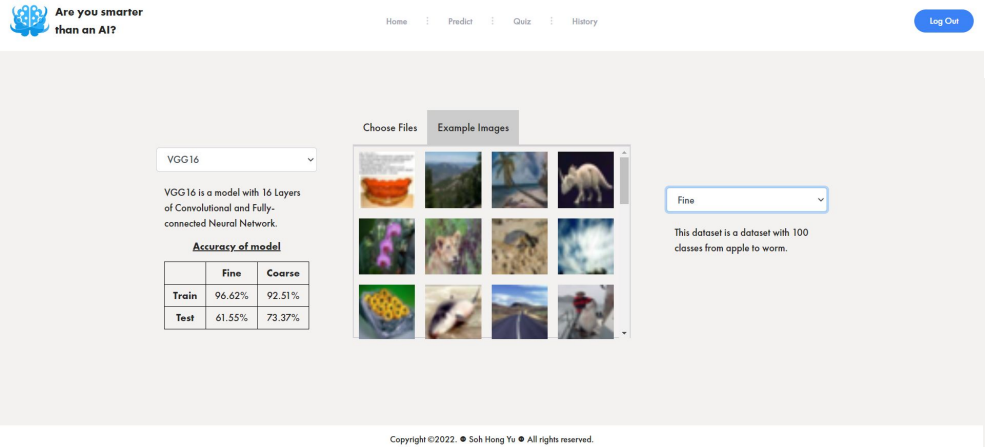
Users can also view history to review their mistakes to help improve themselves for future quizzes.

https://areyousmarterthanai.onrender.com/



Are You Smarter than an AI? - Prediction

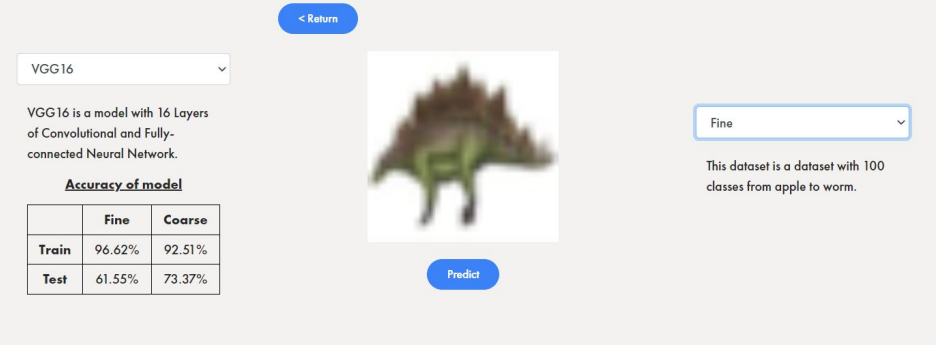




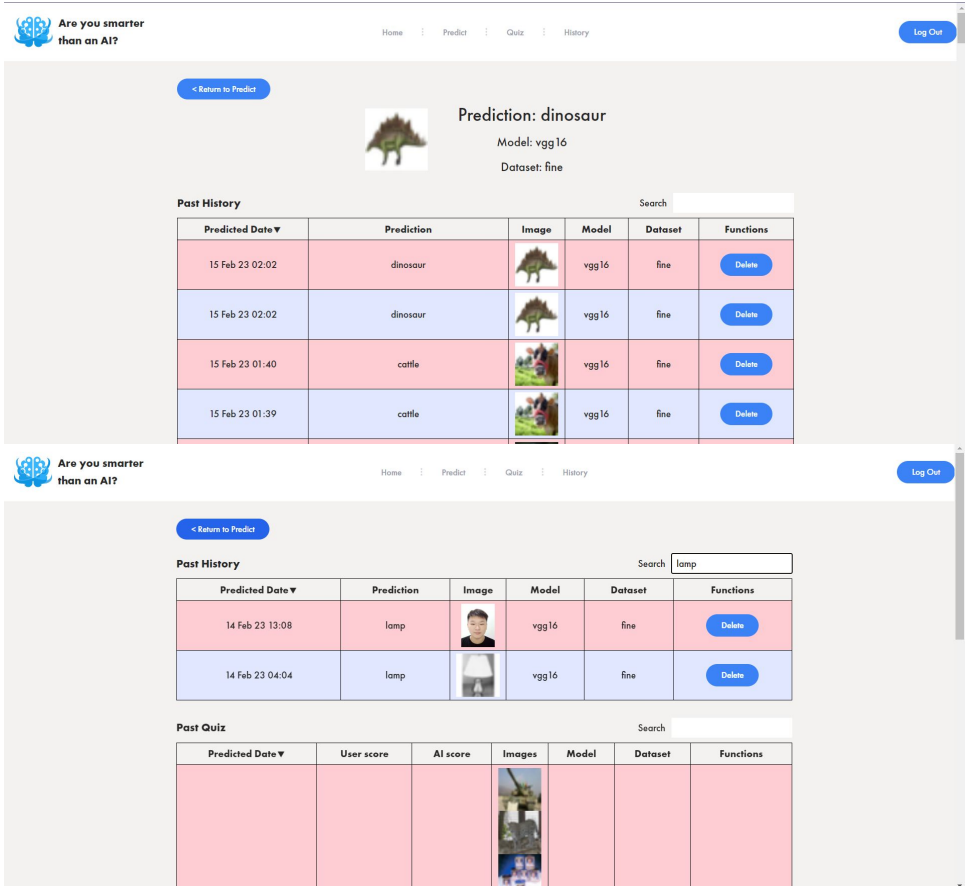
Users can choose to upload an image or choose from the existing images.

Users can also select which model VGG16 or EfficientNet as well as which Dataset is it from like Fine or Coarse.

After selection, the predict button will appear for the user to click to place the prediction.



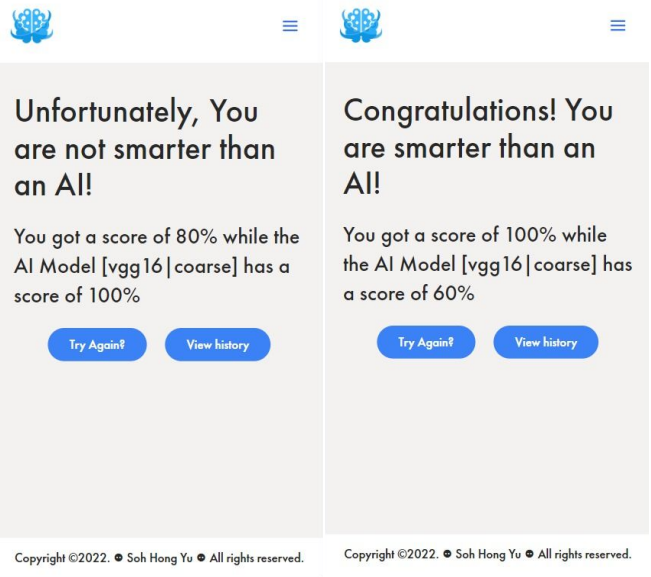
Are You Smarter than an AI? - History

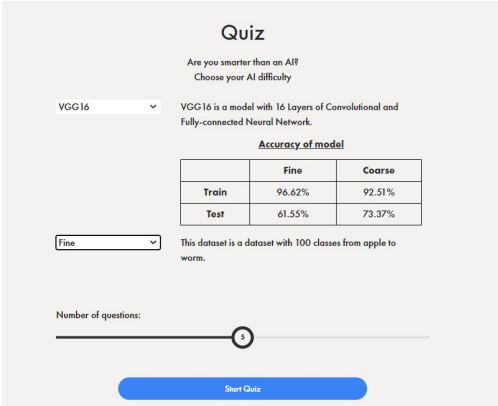


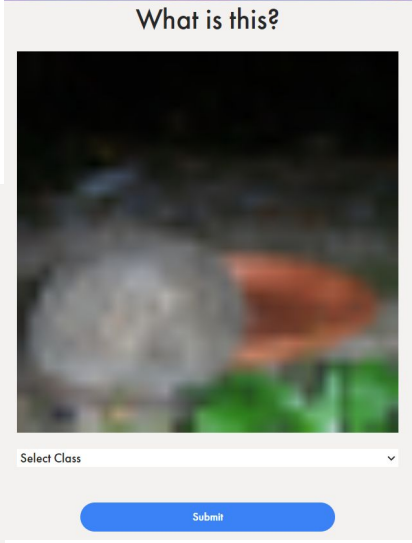
After predictions, users will be moved to the history page where the list of past prediction can be found. One can use the search function as well as delete function.

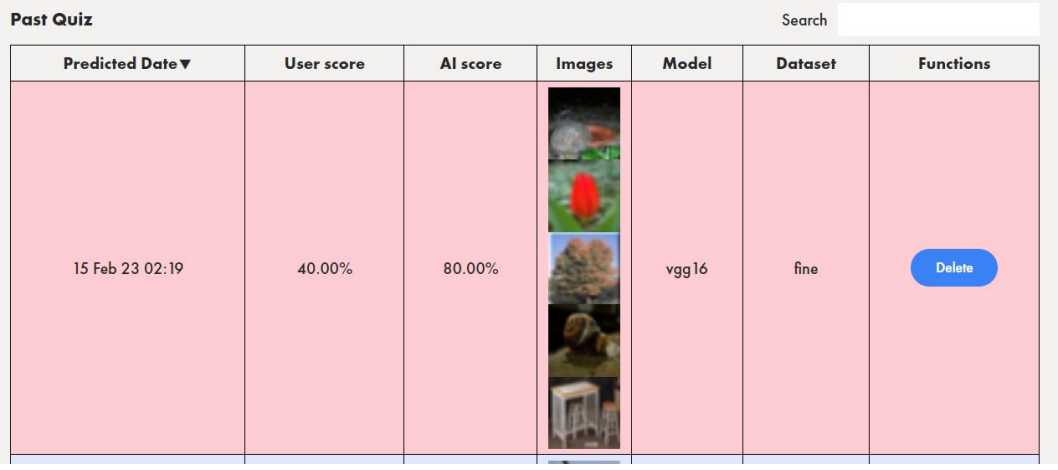
Are You Smarter than an AI? - Quiz

Users can choose the difficulty of the quiz by choosing which model they would want to compete against.
They can choose the number of questions they want to compete against!
Users will be shown an image and a dropdown to allow them to choose what they think the image is about.







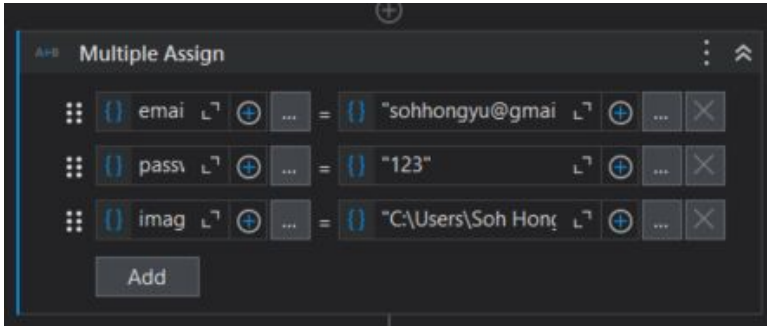


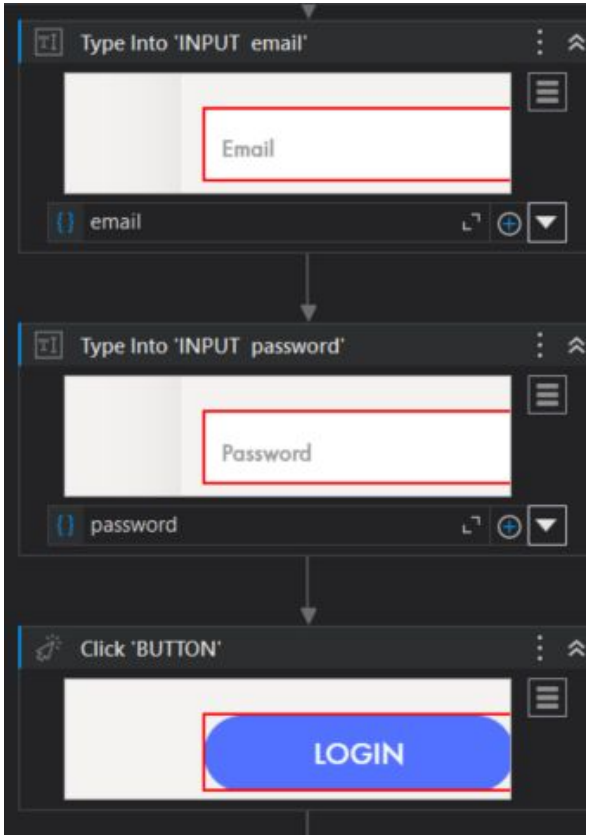
Robotics Process Automation

Login RPA

Using UIPath as our RPA robot, we will be first assigning and giving information to the bot so that it knows what to type in the input boxes.

We will use the “Type Into” activities to type into the input boxes. Afterwards, we use the “Click” activity to click on the login button.

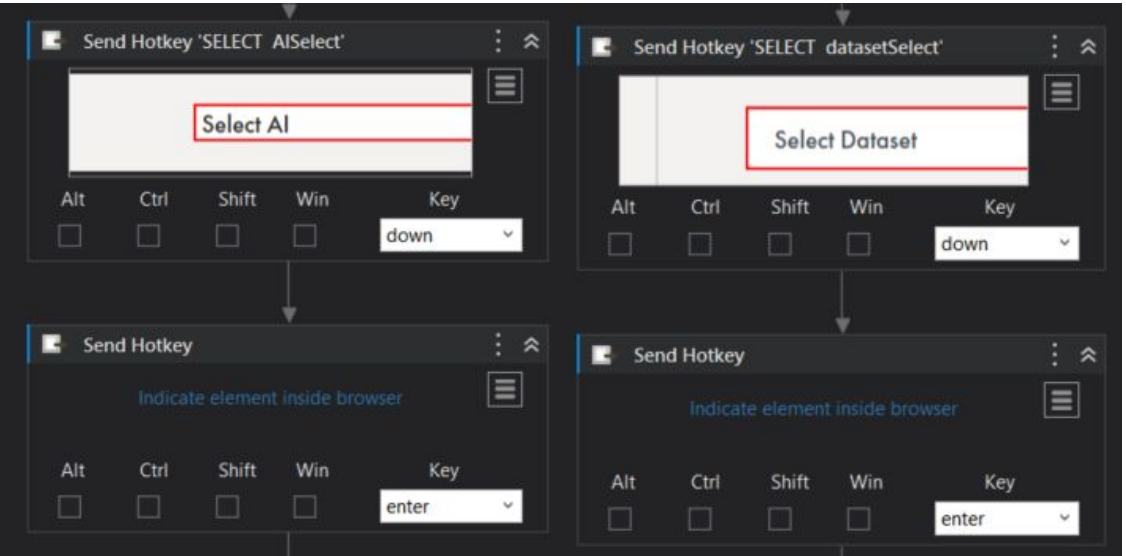


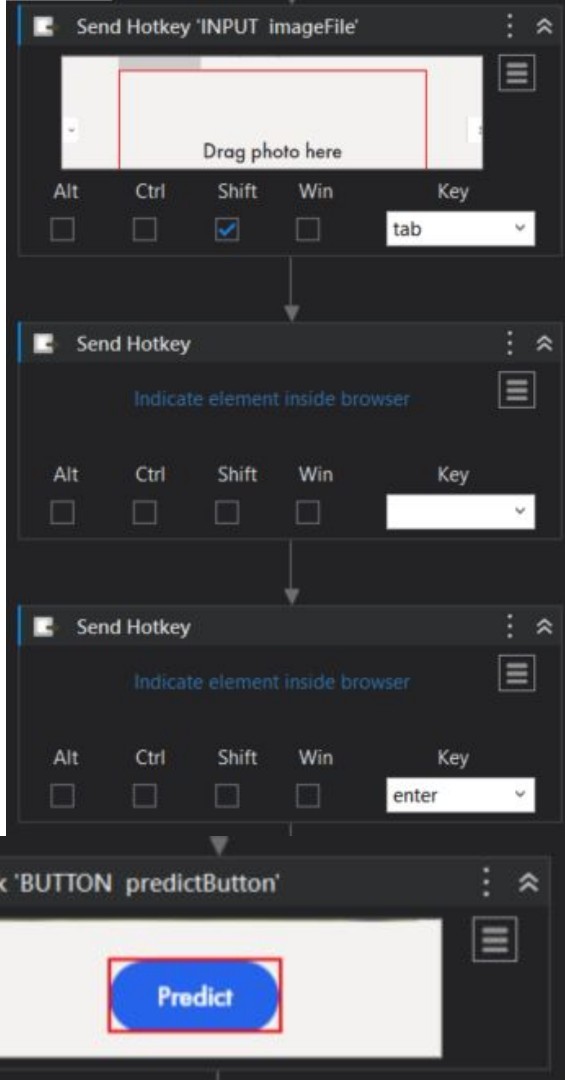


Predict RPA

As there are multiple models to predict from, we will be using the more stable model [VGG16 - Due to latency and small architecture]. (For DEMO purposes)

We will also be using the FINE dataset to allow for the model to be more accurate at pinpointing the class that the image is supposed to be.





Search + Delete RPA

After predicting, we want to be able to interact with the “Delete” buttons to delete entries! We also want to be able to see if the search for the application works!

We will be using “Type Into” activity to search for entries and “Click” activity to delete the entry