Week progress:

Add predictor:

Input\_question.js:

Text

Description automatically generated

Add predictors to the json file in the order as what dataset is. Edit the patterns according to discrete\_to\_digital map.

Text

Description automatically generated

Add choice after end of prediction

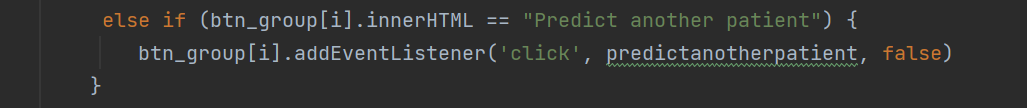
Text

Description automatically generated

Add a bot message through appendMessage()

Graphical user interface, text, application

Description automatically generated



In appendMessage() add a if and eventlistener for click corresponding button

function predictanotherpatient(){  
 add\_userMsg("Predict another patient")  
 var instruction = ""  
 var msgText = ""  
 var btnGroup = []  
 var nextques = ""  
 var pattern = "Predict"  
 input\_choice = input\_question["Predict"]  
 PERSON\_NAME="Your choice is"  
 if (pattern=="Predict")  
 {  
 appendMessage(BOT\_NAME, NURSE\_IMG, "left", "I can predict the recurrence probability of breast cancer, please tell me which year you want to predict","treatment\_year instruction",{"5 year":"5 year","10 year":"10 year","15 year":"15 year"})  
 }  
 else {  
  
 for (var i = 0; i < input\_choice.length; i++) {  
 console.log(pattern)  
 console.log(input\_choice[i].patterns)  
 console.log(Object.keys(input\_choice[i].patterns).indexOf(pattern))  
 if (Object.keys(input\_choice[i].patterns).indexOf(pattern) != -1) {  
 if (input\_choice[i].tag=="treatment\_year")  
 {  
 if (input\_choice[i].patterns[pattern]=="10")  
 {  
 input\_choice = input\_question10["Predict"]  
 }  
 if (input\_choice[i].patterns[pattern]=="5")  
 {  
 input\_choice = input\_question5["Predict"]  
 }  
 }  
 input.push(input\_choice[i].patterns[pattern])  
 nextques = input\_choice[i].nextques  
 console.log(nextques)  
 }  
 }  
 }  
 if(nextques == "none"){  
 var input\_cpoy = input  
 input = []  
 getinput(input\_cpoy)  
 appendMessage(BOT\_NAME, NURSE\_IMG, "left", "Thank you! you answered all questions, we are calculating recurrence","no information",btnGroup);  
 return  
 }  
 tag=""  
 for (var i = 0 ; i < input\_choice.length; i++) {  
 if (input\_choice[i].tag == nextques){  
 let index = Math.floor((Math.random()\*input\_choice[i].responses.length))  
 msgText = input\_choice[i].responses[index]  
 btnGroup = Object.keys(input\_choice[i].patterns)  
 instruction = input\_choice[i].instruction  
 tag = input\_choice[i].tag  
 }  
 }  
 appendMessage(BOT\_NAME, NURSE\_IMG, "left", msgText, instruction, btnGroup,tag);  
}

realize predict\_another\_patient function

Graphical user interface, text, application, chat or text message

Description automatically generated

When selecting the button

Submit survey

Text

Description automatically generated

Text

Description automatically generated

Previously, I failed to get the star and text in the frontend. So I re-write the html sentences of survey form and the function which deals with submit.

After getting the value of star and text input, use a post method to pass the value to backend.

Text

Description automatically generated

Currently I am still thinking about which database is suitable for project. To test the database I try to connect to cloud mongdb database. In the future the database can be replace to others.

'''  
connect to database  
'''  
client = pymongo.MongoClient("mongodb+srv://yangzhen:yangzhen@cluster0.wigwlyv.mongodb.net/?retryWrites=true&w=majority")  
db = client.test  
  
#name of database  
imedbot = client["imedbot"]  
#name of collection  
survey = imedbot["survey"]

@application.route("/submitsurvey", methods=['POST','GET'])  
def get\_user\_survey():  
 if request.method == 'POST':  
 star = request.form.get('star')  
 text = request.form.get('text')  
 print(star, text)  
 now = datetime.now()  
 print("Current Time =", now)  
 survey\_dict={"time":now,"star":star,"suggestion":text}  
 survey.insert\_one(survey\_dict)  
  
 return "success"

The result in database:

Graphical user interface, text, application, email

Description automatically generated

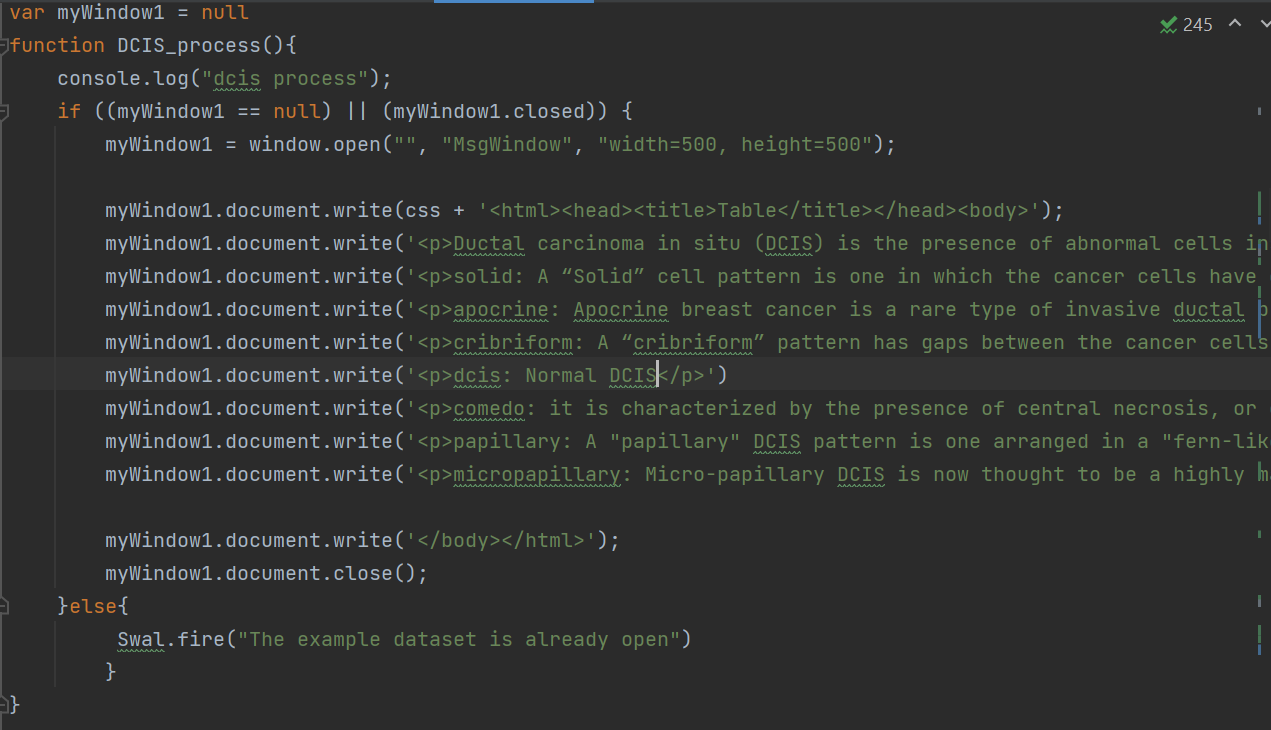
Dcis more explanation

A screenshot of a computer

Description automatically generated with medium confidence

Text

Description automatically generated



DCIS predictors are hard to understand as the others. So I choose to open a new window when clicking on the DCIS\_level link.

Delete the year selection in training service:

Delete selecting years code but new bugs happen:

1. previously when testing a new patient, I generated the below list by judging the year of training but now I need to generate the list directly from the dataset uploaded.

Graphical user interface

Description automatically generated

To fix it:

Frontend I pass the dataset name instead of year

Text

Description automatically generated

In backend I generate predictors separately from txt or csv file

Text

Description automatically generated

Fix the problem that system cannot test new patient after training new dataset by removing previous patientform information:

Text

Description automatically generated

Fix the bug that when no file chosen the bot still says view dataset:

By just move the appendMessage after successfully reading the file in function submit

