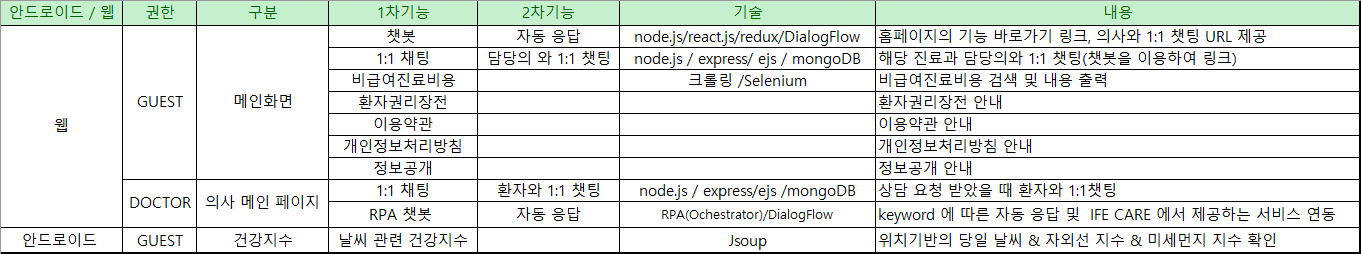
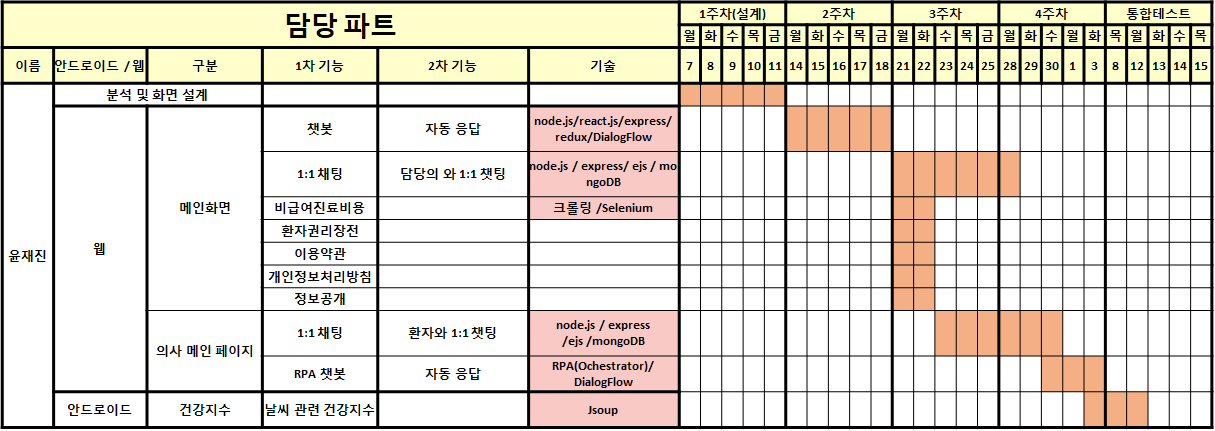
**-기능정의서**

****

기술할 1차기능 : 챗봇, 1:1채팅, 비급여진료비용, 날씨 관련 건강지수

**-개발일정\_챗봇**

****

**홈페이지>챗봇 (모든 페이지에서 접근가능)**

**–사용된 기술 : Node.js, React.js, Express, Redux, DalogFlow**

Node.js를 활요한 챗봇

1. Node.js 설정방법

-Nodejs 설치 : 공식 사이트에서 LTS(안전한버전)을 설치

-개발 툴 : VS Code를 설치

-Start : VS Code에서 new Terminal로 툴에 터미널 이식 -> 프로젝트 진행할 폴더를 열고

터미널에 npm을 사용하기 위해 Npm Init을 입력 후 대략적인 정보를 입력

-Npm으로 모듈 다운로드 npm install “모듈명”을 입력하여 install 한다

(모듈 명 : body-parser, dialogflow, dialogflow-fulfillment, express)

-npm은 Nodejs package manager라는 뜻으로 nodejs 모듈을 관리해주는 시스템이다.

-body-parser는 현재 express에서 재공하는 json매소드가 있어 사용되지 않는다.

1. React.js 설정방법

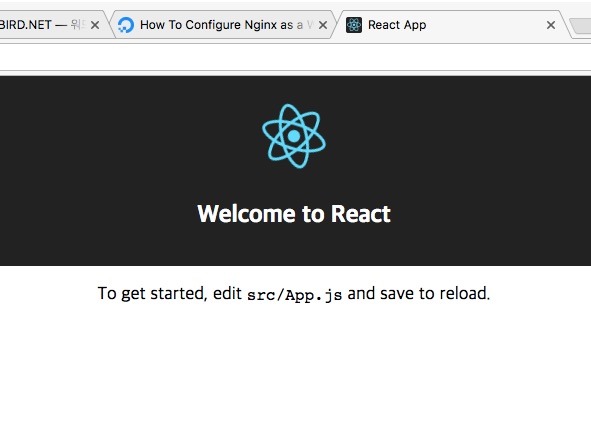
-최상위 프로젝트 폴더안에 client 폴더 생성

-터미널로 client 폴더에 접근

-npx create-react-app .으로 리엑트설치

-리엑트 라이브러리가 깔린 폴더에 접근하여 npm run start를 실행 했을 때

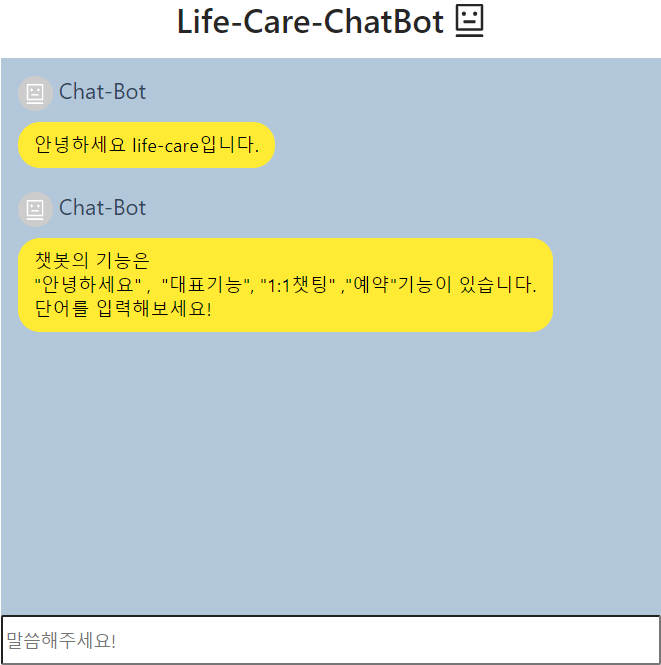
기본 페이지가 나오면 성공



이후 리엑트 모듈이 설치된 폴더에서 다음 모듈을 설치한다.

(antd, axios, prop-types, redux, redux-promis, redux-thunk 등을 설치한다)

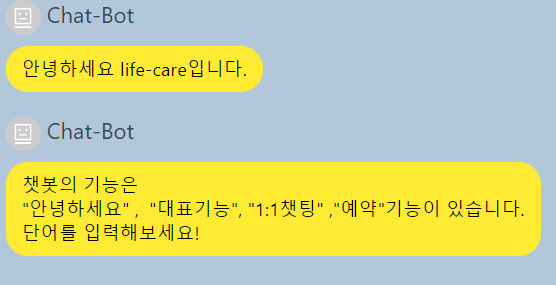
3.챗봇 매인 화면



|  |  |  |  |
| --- | --- | --- | --- |
| 작성자 | 윤재진 | 작성일 | 2020.10.19 |
| ChatBot.js | Client.Chatbot.ChatBot.js | | |
| return (      <div>        <div style={{ display: 'flex', justifyContent: 'center', marginTop: '2rem' }}>            <Title level={2} >Life-Care-ChatBot&nbsp;<Icon type="robot" /></Title>        </div>        <div style={{ display: 'flex', justifyContent: 'center' }}>          <Chatbot />        </div>      </div>    )  간단설명 : 챗봇의 메인화면 코드 (리엑트 컴포넌트)  -최초 챗봇을 키면 매인화면을 뿌려주는 콤포넌트 <Chatbot> 은 메시지와 메시지를보내는 콤포넌트가 입력되는 부분 | | | |

4.랜딩 메시지 뿌려주는 방법

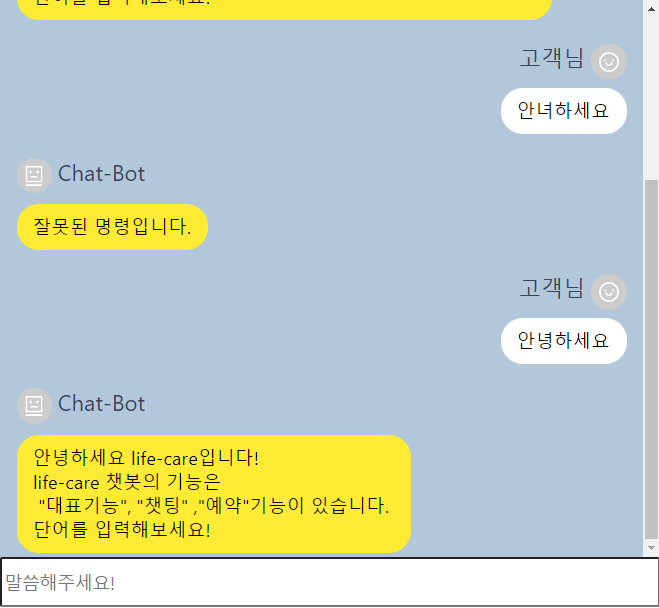
구글 다이알로그플로우를 통해 이벤트 response를 받아 (최초 진입 시 실행되는 이벤트)다음과 같은 메시지를 뿌려 줌



---->

|  |  |  |  |
| --- | --- | --- | --- |
| 작성자 | 윤재진 | 작성일 | 2020.10.19 |
| ChatBot.js | Client.Chatbot.ChatBot.js | | |
| const eventQuery = async (event) => {          // We need to take care of the message Chatbot sent          const eventQueryVariables = {              event          }          try {              //I will send request to the textQuery ROUTE              const response = await Axios.post('/api/dialogflow/eventQuery', eventQueryVariables)              for (let content of response.data.fulfillmentMessages) {                  let conversation = {                      who: 'Chat-Bot',                      content: content                  }                  dispatch(saveMessage(conversation))              }  최초 챗봇 실행시 뿌려주는 eventQuery실행 코드 | | | |

5.메시지를 보내고 받는 방법

-메시지 보내는 법

다음과같이 Intent에 맞는 질문을 보내면

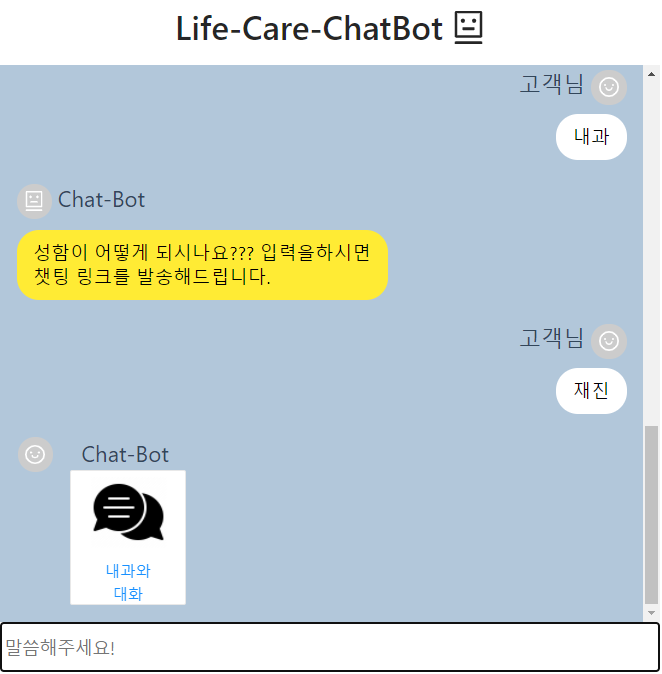
Dialogflow 에서 지정된 Response의 대답이 돌아온다.

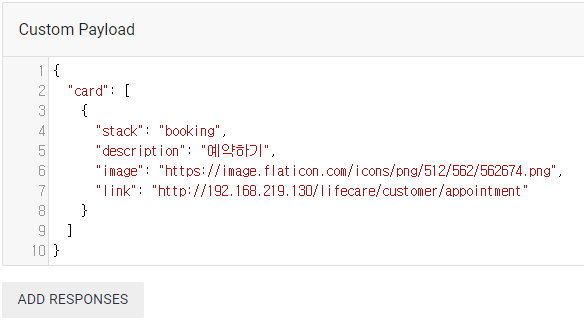
|  |  |  |  |
| --- | --- | --- | --- |
| 작성자 | 윤재진 | 작성일 | 2020.10.19 |
| ChatBot.js | Client.Chatbot.ChatBot.js | | |
| const keyPressHanlder = (e) => {          if (e.key === "Enter") {              if (!e.target.value) {                  return alert('질문을 입력해주세요!')              }              //we will send request to text query route              textQuery(e.target.value)              e.target.value = "";          }      }  위와 같은 핸들러로 메새지를 DialogFlow으로 보낼 수 있다. | | | |

-메시지 받는 법

|  |  |  |  |
| --- | --- | --- | --- |
| 작성자 | 윤재진 | 작성일 | 2020.10.19 |
| Message.js | Client.Chatbot.Section.Message.js | | |
| function Message(props) {         const AvatarSrc = props.who ==='Chat-Bot' ? <Icon type="robot" /> : <Icon type="smile" />              if(props.who==='Chat-Bot'){          return (     <ul className="sent">      <p style={{fontSize:'20px',marginBottom:'0'}}> {<Avatar icon={AvatarSrc}/>}&nbsp;{props.who}</p>     <li><pre className="p">{props.text}</pre></li>     </ul>          )      }else{          return (          <ul className="replies">              <p className="r">{props.who}&nbsp;{<Avatar icon={AvatarSrc}/>}</p>              <br/><br/>              <li><pre className="message">{props.text}</pre></li>          </ul>          )      }  }  export default Message  전달받은 who의 값이 Chat-bot이면 챗봇 전용 CSS와 아이콘을 데이터에 입히고 return한다  전달받은 who의 값이 Chat-bot이 아니면 사용자 전용 CSS와 아이콘들 데이터에 입히고 return한다. | | | |

6. 링크 카드 뿌려주기(DialogFlow의 CustomPayLoad(JSON)를 사용)

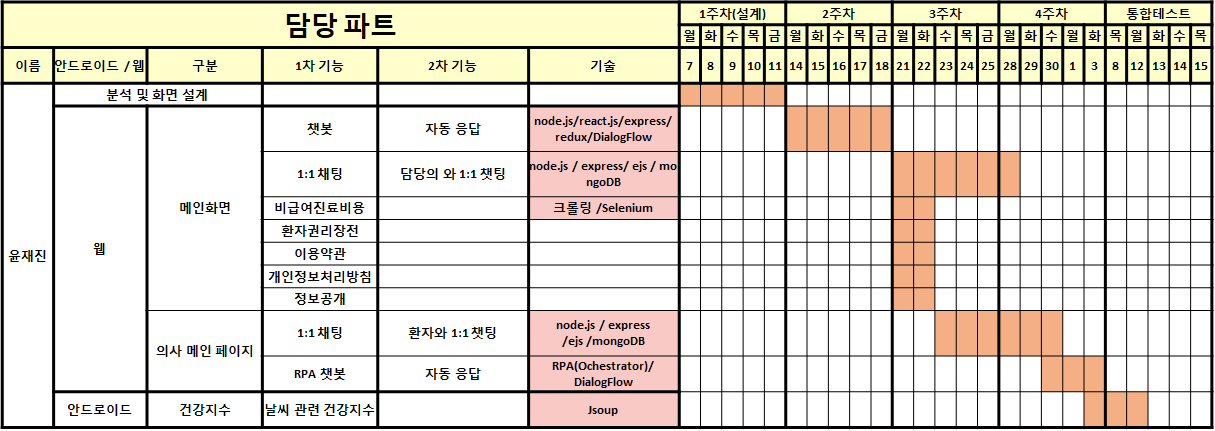




--->

|  |  |  |  |
| --- | --- | --- | --- |
| 작성자 | 윤재진 | 작성일 | 2020.10.19 |
| Card.js | Client.Chatbot.Section.Card.js | | |
| import React from 'react'  import { Card, Icon } from 'antd';  require("../css/test.css");  const { Meta } = Card;  function CardComponent(props) {      return (          <Card              style={{ width: 105,height:123 }}              cover={                  <a target="\_blank" style={{margin:'0px auto',width:'70px',height:'110px'}} rel="noopener noreferrer" href={props.cardInfo.fields.link.stringValue}>                      <img className="card"                          alt={props.cardInfo.fields.description.stringValue}                          src={props.cardInfo.fields.image.stringValue} />                         <div className="text">{props.cardInfo.fields.description.stringValue}</div>                  </a>              }         </Card>      )  }  export default CardComponent  CustomPayLoad으로 받은 값을 props으로 접근하여 값을 얻어 카드 형태로 return | | | |

**-개발일정\_1:1챗팅**

****

**홈페이지>의사메인페이지>환자와의 챗팅**

**-사용된 기술 : Node.js, EJS, Websocket, MongoDB, Express**

Node.js를 활요한 채팅

1. Node.js 설정방법

-Nodejs 설치 : 공식 사이트에서 LTS(안전한버전)을 설치

-개발 툴 : VS Code를 설치

-Start : VS Code에서 new Terminal로 툴에 터미널 이식 -> 프로젝트 진행할 폴더를 열고

터미널에 npm을 사용하기 위해 Npm Init을 입력 후 대략적인 정보를 입력

-Npm으로 모듈 다운로드 npm install “모듈명”을 입력하여 install 한다

(모듈 명 : body-parser, ejs, mongoose, mongoose-auto-increment, Express, Socket.IO)

-npm은 Nodejs package manager라는 뜻으로 nodejs 모듈을 관리해주는 시스템이다.

-구현이유

의사와 환자의 1:1 상담을 위해 구현하였습니다.

-사용방법

환자는 챗봇에서 보낸 링크로 자신이 선택한 과의 개인 챗팅방으로 접속됨과 동시에 챗팅방이 생성 됩니다.

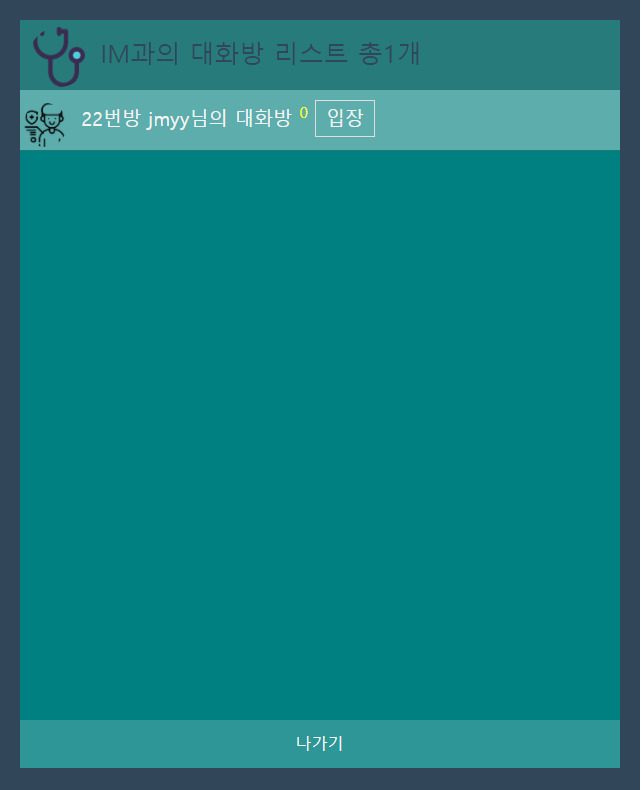
의사는 자신이 속한 과에 만들어친 챗팅방을 통해 환자와 챗팅이 시작되도록 구현하였습니다.

-구현방법

실시간 정보교환이 이뤄지도록 Websocket.io를 활용하여 챗팅기능을 구현하였고

챗팅로그를기록하기 위해 mongoDB를 사용하였습니다.

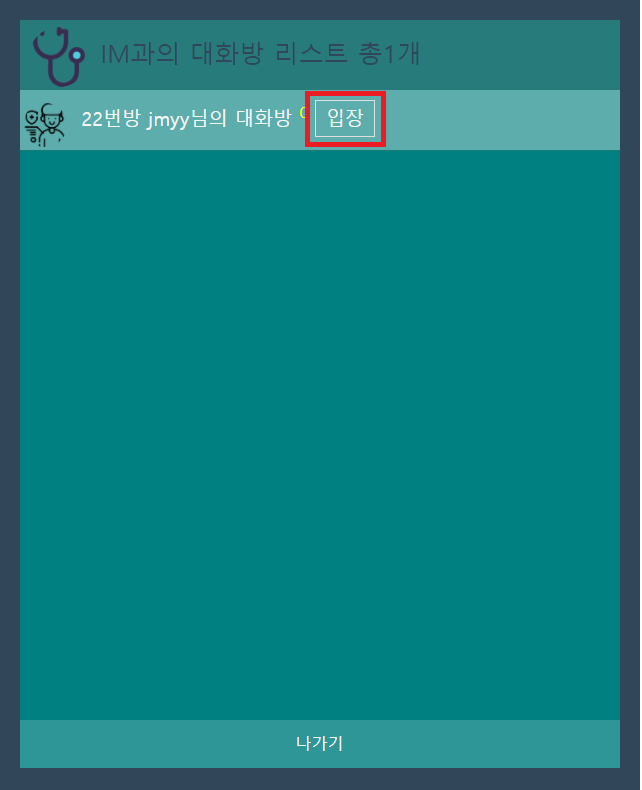
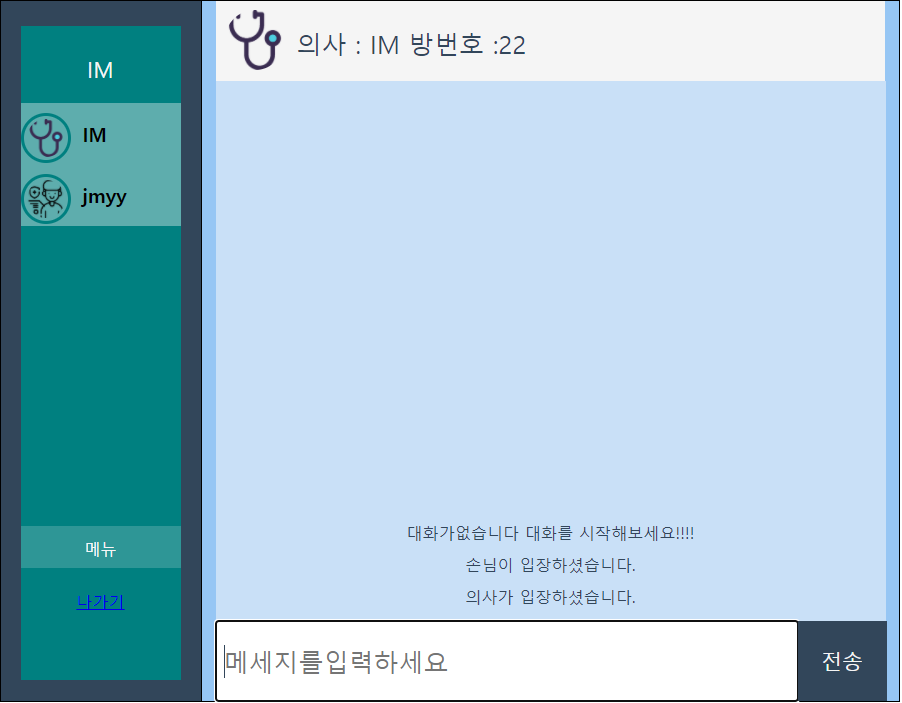
1.챗팅 시작 화면(의사/손님)



손님 시작화면(의사는 미접속) 의사의 시작화면

|  |  |  |  |
| --- | --- | --- | --- |
| 작성자 | 윤재진 | 작성일 | 2020.10.19 |
| Express.js | Express.js | | |
| 환자가 링크(http://localhost:2000/CROIR?guest=jmyy&section=IM&who=guest)를타면 해당 라우터가 실행됨  app.get("/CROIR", (req,res)=>{      let roomCode=""      // http://localhost:2000/CROIR?guest=jmyy&section=IM&who=guest     손님이 들어올 URL      //손님이 챗팅방에 들어오면 자동으로 roomcode를 시퀀스하여 방번호 생성(=방 생성) req에 손님이름 진료과 선정      StatusModel.findOne({guest:req.query.guest,section:req.query.section}).then((result)=>{          console.log("방있니?",result)          if(result==null){              //방이없으면 새로운방 생성              var status= Object.assign(req.query,{guestIO:1})              console.log("방합침",status)              StatusModel.create(status).then((result)=>{                  StatusModel.findOne({guest:req.query.guest, section:req.query.section},(err, status)=>{                      //찾는방의 룸코드                     console.log(status.roomCode)                     roomCode=status.roomCode                     res.redirect("/guest?guest="+req.query.guest+"&section="+req.query.section+"&who="+req.query.who+"&roomCode="+status.roomCode)                     //룸코드로 방이동해야한다.                 })              })          }else{              //방이있으면 그방에 guest상태값 1로 업데이트              StatusModel.findOneAndUpdate({guest:req.query.guest,section:req.query.section},{guestIO:1},(err,status)=>{                  if(!err)console.log("방에 들어오셨습니다.!")                  StatusModel.findOne({guest:req.query.guest, section:req.query.section},(err, status)=>{                      //찾는방의 룸코드                     console.log(status.roomCode)                     roomCode=status.roomCode                     res.redirect("/guest?guest="+req.query.guest+"&section="+req.query.section+"&who="+req.query.who+"&roomCode="+status.roomCode)                     //룸코드로 방이동해야한다.                 })              })          }      })  })  의사의 시작 페이지 링크를 타면 해당 라우터가 실행됨  //의사 챗팅방 목록보기  // http://localhost:2000/standBy?section=IM  app.get("/standBy", (req,res)=>{      StatusModel.find({section:req.query.section}, async function (err,List){          if(err) return console.log("dddd",err)            var roomList={rooms:List}          let lfls="";          for(var i = 0; i<roomList.rooms.length; i++){                var roomCode=roomList.rooms[i].roomCode;              lfls=(await LogModel.find({roomCode:roomCode,lfl:1})).length                Object.assign(roomList.rooms[i],{lflLength:lfls})          }              res.render("standBy",{rooms:List,section:req.query.section,roomCount:roomList.rooms.length});      })          // 룸코드를 물고나오면 그방에서 닥터 out        if(req.query.roomCode){          whoOut="doctor";          StatusModel.updateMany({roomCode:req.query.roomCode},{doctor:null},(err,status)=>{            })      }  }); | | | |

2.의사가 챗팅방에 들어가는 방법



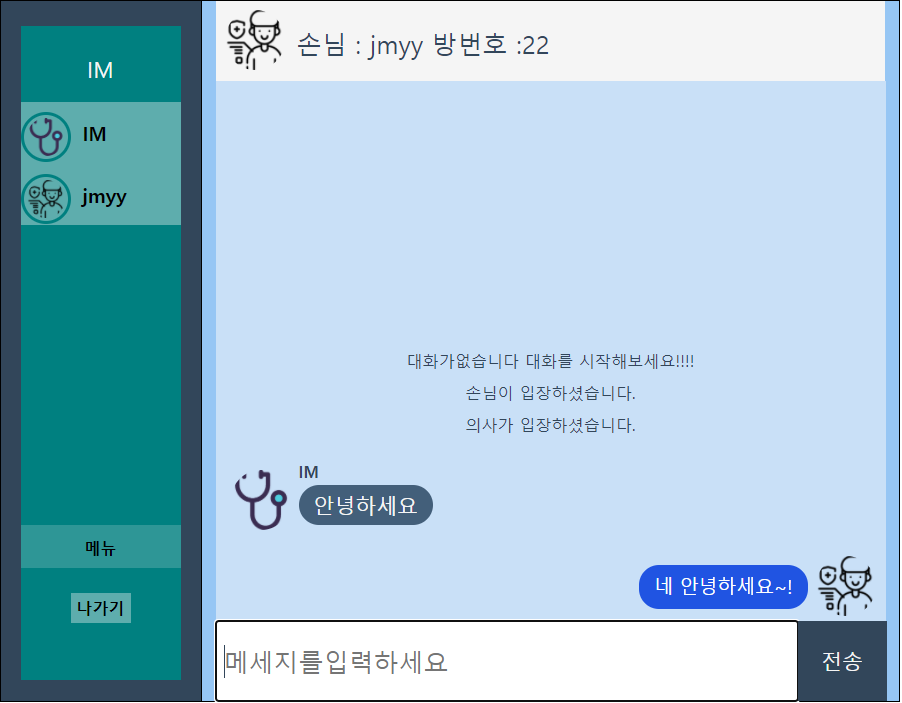
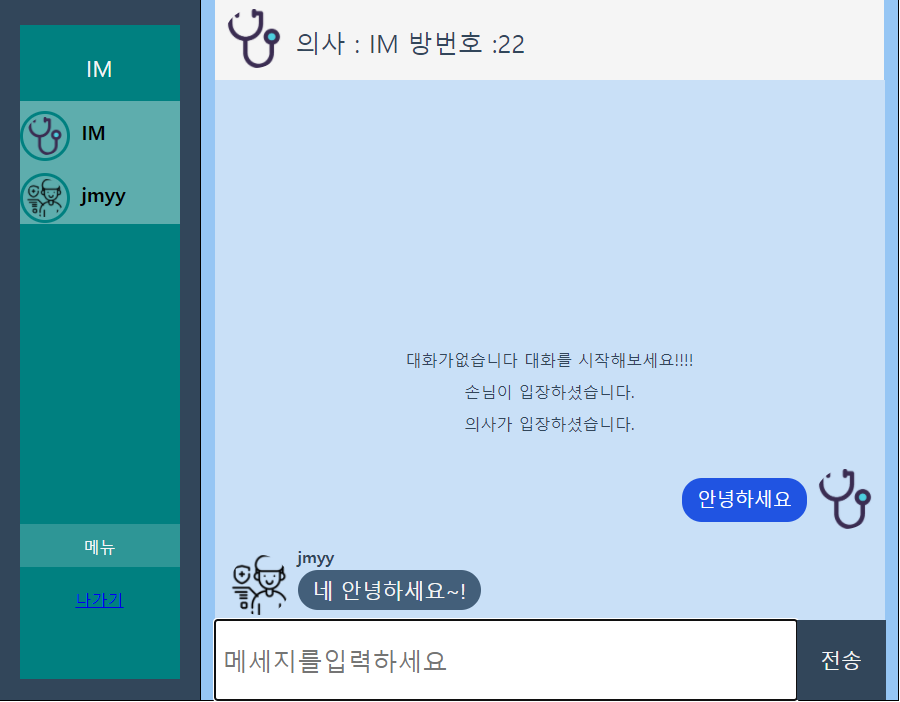
의사가 입장을 눌러주면 해당 방으로 입장이 되며 왼쪽 상태창이 빨간색에서 녹색으로 바뀌게됩니다.(Ajax활용)

|  |  |  |  |
| --- | --- | --- | --- |
| 작성자 | 윤재진 | 작성일 | 2020.10.19 |
| ChatScript.js | javaScript. ChatScript.js | | |
| //의사가 방에 들어갈떄   function joinDoctor(){      var number=""      var list=document.getElementsByName("list");      for(var i =0; i<list.length; i++){          if(list[i].checked==true){              number=list[i].value          }      }      var guest   = document.getElementsByName("guest")[number].value      var section = document.getElementsByName("section")[number].value      var who     = document.getElementsByName('who')[number].value      var doctor  = document.getElementsByName('doctor')[number].value      var roomCode= document.getElementsByName('roomCode')[number].value      //socket 방으로 이동(roomCode로 생성)      location.href="/guest?guest="+guest+"&section="+section+"&who="+who+"&doctor="+section+"&roomCode="+roomCode;  }  입장을 통해 위에 메소드가 실행되어 환자가 챗팅방에 들어가기 위해 실행했던 라우터로 이동하게 됩니다. | | | |

3.webSocket.IO을 통해 Room에 입장하는 방법

|  |  |  |  |
| --- | --- | --- | --- |
| 작성자 | 윤재진 | 작성일 | 2020.10.19 |
| Express.js/ChatScript.js | Express.js/ javaScript.ChatScript.js | | |
| ChatScript.js를 사용하는 페이지로 이동하게 되면  <script src="/socket.io/socket.io.js"></script>  가 사용되어지고있어 자동으로 소켓과 연결되게 됩니다.    //io와 연결될때 실행할 함수      socket.on('connect',()=>{          //  alert("life-care와 건강한 챗팅 되세요!")          //연결하자마자 해당 룸으로 이동          socket.emit('join',{roomCode:roomCode,who:who,doctor:doctor})      })  또한 방에 입장할 때 RoomCode 정보를 가지고 입장하여 위와 같이 Room에 입장하는 코드와 연결되어  socket.on('join', (roomCode,fn)=>{          console.log(roomCode)          if(roomCode.who=="doctor"){              StatusModel.findOneAndUpdate({roomCode:roomCode.roomCode},{doctor:roomCode.doctor},(err,status)=>{              })          }else if(roomCode.who=='guest'){              StatusModel.findOneAndUpdate({roomCode:roomCode.roomCode},{guestIO:1},(err,status)=>{              })          }          socket.join(roomCode.roomCode,()=>{              console.log(socket.id," join : ",roomCode.roomCode)          })      })  Express.js에 있는 websocket과 연결되어 소캣Room에 연결되게 됩니다.  (환자와 의사 모두 같은 로직으로 Room에입장합니다.) | | | |

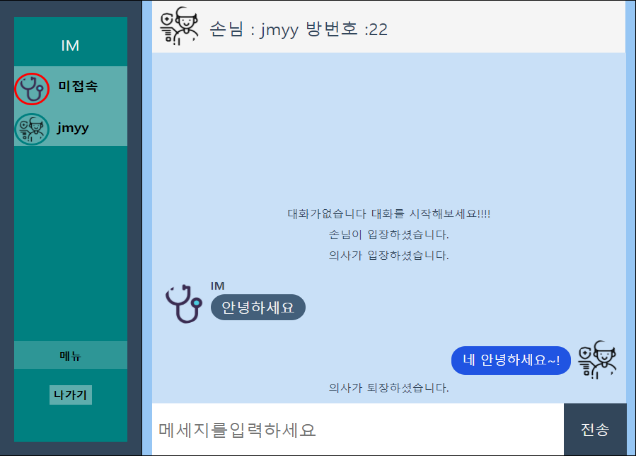
1. 메시지를 보내고 받는 방법

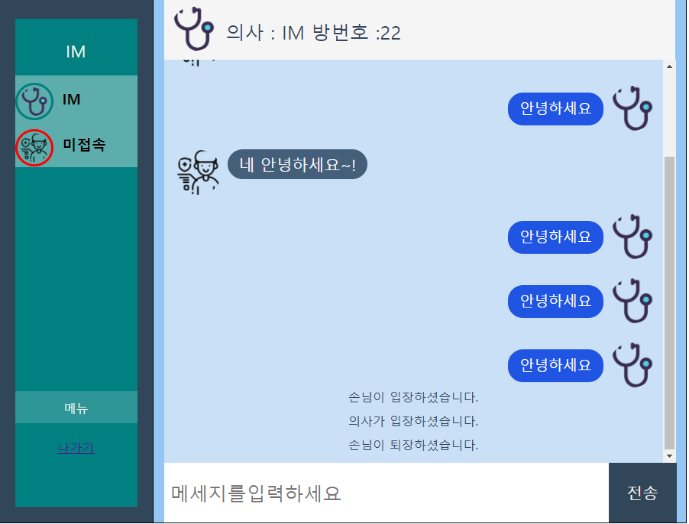


의사와 환자가 메시지를 주고받는 화면

|  |  |  |  |
| --- | --- | --- | --- |
| 작성자 | 윤재진 | 작성일 | 2020.10.19 |
| Express.js/ChatScript.js  /geust.ejs | Express.js/ javaScript.ChatScript.js/client.geust.ejs | | |
| EJS 페이지에서 해당 Form이 Submit이 되면  <form  action="/guest/message" name="chatting" method="POST" onsubmit="ajaxMessage();" >                  <input class="input"    id="m" autocomplete="off" name="message" input placeholder="메세지를입력하세요" autofocus/>                  <input class="button"   type="submit" value="전송"  readonly>                  <input type="hidden"    name="guest" value="<%=guest%>"/>                  <input type="hidden"    name="section"value="<%=section%>"/>                  <input type="hidden"    name="roomCode" value="<%=roomCode%>"/>                  <input type="hidden"    id="doc" name="doctor"value="<%=doctor%>"/>                  <input type="hidden"    name="who"value="<%=who%>"/>                  <input type="hidden"    name="guestIO" value="<%=guestIO%>">                  <input type="hidden"    name="conversation" value="<%=conversation%>">  </form>  아래 Script가 실행됩니다.(ChatScript.js)   $('form').submit(function(e){            e.preventDefault(); // prevents page reloading          var message = document.getElementById("m").value;          var guest=document.chatting.guest.value;          var section=document.chatting.section.value;          var roomCode = document.chatting.roomCode.value;          var doctor=document.chatting.doctor.value;          var who=document.chatting.who.value;          var guestIO=document.chatting.guestIO.value;            var data={              roomCode:roomCode,              message:message,              guest:guest,              doctor:doctor,              section:section,              who:who,              guestIO:guestIO          }          //매시지 보내는곳          socket.emit('message', data);          $('#m').val('');          return false;      });  소캣으로 대화를 하는 사용자의정보(roomCode, message, guest, doctor, section, who, gusetIO)를 같이 보내게 됩니다.(Express.js)   //매세지를 보낼때 실행되는곳      socket.on ('message', (msg) => {          console.log(msg)          //속한 방으로 보네기          io.to(msg.roomCode).emit ('message', msg);      });  위에 코드가 실행되면 다시 (ChatScript.js)에서 그 값을 받아 환자가 보낸 메시지 인지, 의사가 보낸 메시지인지 판단하여  화면에 뿌려주게 됩니다.  //매시지 받으면 출력시키는 곳      socket.on('message',function(msg){           var message=msg.message;         var guest=msg.guest;         var doctor=msg.doctor         var msgwho=msg.who;         var who=document.chatting.who.value;           if(who=='guest'){               if(msgwho=='guest'){                  var doctor=document.getElementById("doc").value                    var lfl="<p class='message'>"+message+"</li>"                  $('#messages').append($(" <li class='replies'>\                  <img src='../image/patient.png' />" +lfl));                  }else if(msgwho=='doctor'){                  $('#messages').append($(" <li class='sent'>\                  <img src='../image/doctor.png'/>\                  <span>"+doctor+"</span><br><p>"+message+"</p></li>"));                }          }else if(who!=guest){                if(msgwho=='doctor'){                  $('#messages').append($("<li class='replies'>\                      <img src='../image/doctor.png' />\                      <p class='message'>"+message+"\</p></li>"));              }else if(msgwho=='guest'){                  $('#messages').append($(" <li class='sent'>\                  <img src='../image/patient.png' />\                  <span>"+guest+"</span><br><p>"+message+"</p></li>"));              }          }      }); | | | |

5. 챗팅방을 나갈 때 실행되는 코드

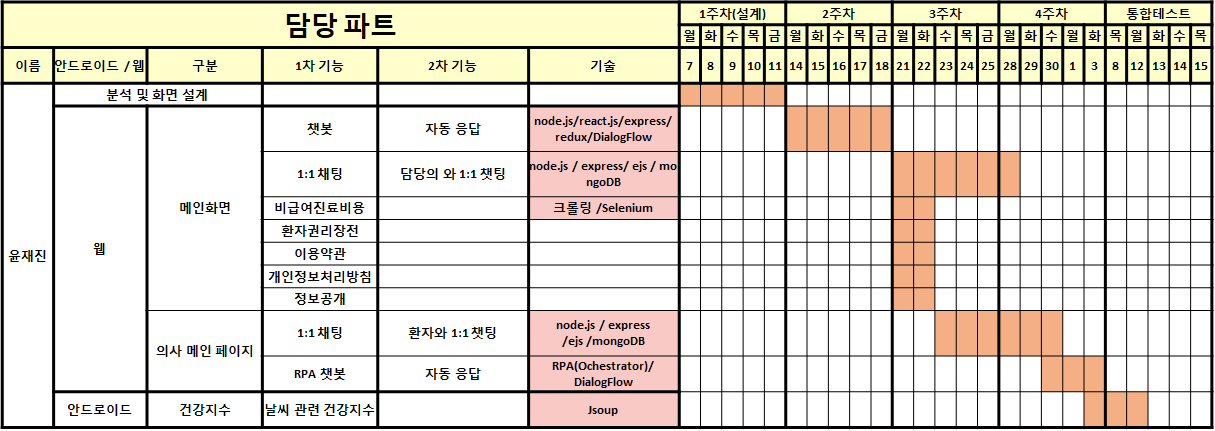




의사가 나갈 때 상태 창 손님이 나갈 때 상태 창

|  |  |  |  |
| --- | --- | --- | --- |
| 작성자 | 윤재진 | 작성일 | 2020.10.19 |
| Express.js/Status.js  /geust.ejs | Express.js/ javaScript.Status.js/client.geust.ejs | | |
| 소캣과 연결된 페이지를 끄거나 뒤로가거나 새로고침 할 경우 자동으로 Disconnect이 실행됩니대 이때  shakehand기능으로 특정 값을 전달해 mongoDB에 RoomStatus 필드에 의사와 손님의 IO값을 변경합니다.  아래는 실행 코드 입니다.  -Disconnect 될경우 실행되는 코드(스키마를통해 MongoDB에 접속하여 값을 변경해준다)    //소캣 서버에서 나갈떄 실행되는곳      socket.on('disconnecting',(roomCode)=>{          var query=socket.handshake.query;          var roomCode=query.roomCode          socket.leave(roomCode,()=>{              console.log(socket.id,"는 -> leave : ",roomCode)          })          if(query.who=='doctor'){              StatusModel.findOneAndUpdate({roomCode:query.roomCode},{doctor:null},(err,status)=>{                  if(err) console.log(err)              })          }else if(query.who=='guest'){              StatusModel.findOneAndUpdate({roomCode:query.roomCode},{guestIO:0},(err,status)=>{                  if(err) console.log(err)              })          }          console.log("끊겼습니다.",socket.id)      });    C:\Users\KOSMO_24\Desktop\mongo.png  MongoDB Field에 guestIO의 값이 0이되면 환자는 퇴장  MongoDB Field에 Doctor의 값이 null이되면 환자는 퇴장    위에서 전달된 코드를 Status.js에서 받아 ajax를 통해 MongoDB 필드에 입력된 값을 받아 상태창을 변하하고  퇴장 message를 뿌려주게됩니다.  setInterval(() => {      var roomCode=document.getElementsByName('roomCode')[0].value;      var data={roomCode:roomCode}      //데이타 준비      var localhost="http://192.168.219.123:2000"      var ajax=new XMLHttpRequest();      var url='/status'      ajax.open('POST',localhost+url)      ajax.setRequestHeader('Content-Type',"application/json")      ajax.send(JSON.stringify(data))      ajax.addEventListener('load',()=>{          var status=JSON.parse(ajax.responseText)           //의사가 들어왔는지 손님이 들어왔는지 확인          if(status.guestIO==1){              document.getElementById("guest").className="online"              document.getElementById('GN').innerHTML=status.guest              if(guest==1){                 $('#messages').append($(" <li ><p style='text-align: center;'>손님이 입장하셨습니다.</p></li> "));                 guest=2;              }          }else if(status.guestIO==0){              document.getElementById("guest").className='offline'              document.getElementById('GN').innerHTML="미접속"                if(guest==2){                  $('#messages').append($(" <li style='text-align:center;'><p style='text-align: center;'>손님이 퇴장하셨습니다.</p></li> "));                  guest=1;               }          }          if(status.doctor||status.doctor!=null){               document.getElementById("doctor").className="online";              document.getElementById('DN').innerHTML=status.doctor               if(doctor==1){                  $('#messages').append($(" <li style='text-align:center;'><p style='text-align: center;'>의사가 입장하셨습니다.</p> </li> "));                  doctor=2;               }          }else if (!status.doctor||status.doctor==null){              document.getElementById("doctor").className='offline'              document.getElementById('DN').innerHTML="미접속"              if(doctor==2){                  $('#messages').append($(" <li style='text-align:center;'><p style='text-align: center;'>의사가 퇴장하셨습니다.</p> </li> "));                  doctor=1;                }          }      })   },500); | | | |

**-개발일정\_크롤링**

****

**홈페이지>Footer>비급여 진료비용**

**-사용된 기술 :** Python, Selenium, OracleDB, Spring

**-기술을 구현한 이유**

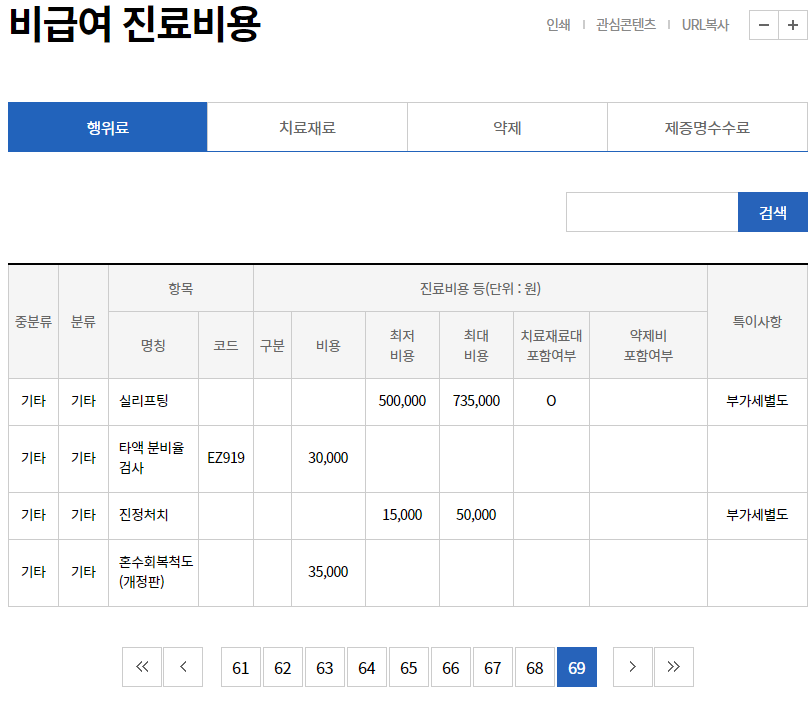
-사람이 손수 입력하기에 시간이 오래걸리는 대량의 데이터를 쉽게 가져오고 DB에 입력하기위하여

구현하였습니다.

**-구현방법**

-Beautifulsoup와 Selenium을 이용하여 페이지를 동적으로 이동시켜 크롤링을 한 뒤 데이터를 가져와 Pandas로 가공하고 OracleDB에 Insert 후 Spring을 이용하여 웹에 출력하여 사용자가 정보를 제공 받을 수 있도록 하였습니다.

1.웹크롤링 후 화면에 출력



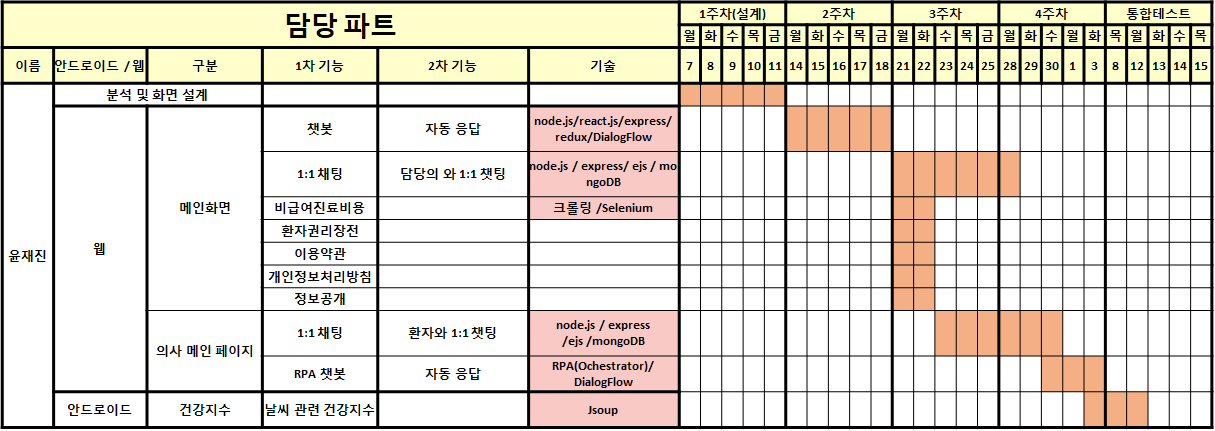
1. 

|  |  |  |  |
| --- | --- | --- | --- |
| 작성자 | 윤재진 | 작성일 | 2020.10.19 |
| life\_care\_비급여크롤링  (4개항목).ipynb | Jupyter notebook.life\_care\_비급여크롤링(4개항목).ipynb | | |
| #행위료 크롤링  from selenium import webdriver  from bs4 import BeautifulSoup  #크롬드라이버 경로  driver=webdriver.Chrome("C:/Users/KOSMO\_24/Desktop/팀플젝/크롬드라이버/chromedriver.exe")  #페이지를 켜는것(셀레니움)  driver.get("https://www.snuh.org/reservation/unPaid/list.do")  #뷰티플 슾이 크롤링할 인터넷창을 지정해줌(셀레니움이지정된 드라이버를 넣어줌)  soup=BeautifulSoup(driver.page\_source,"html.parser")  list\_td=[]#크롤링 데이터가 들어가는 어레이  DF\_list=[]#데이터프레임용 어레이  #크롤링시작  for i in range(0,7):  #시작블럭에 있는 tr을 크롤링  soup=BeautifulSoup(driver.page\_source,"html.parser")  #크롤링할 요소1  tr=soup.select("#content > div.costWrap > div.tableType01 > table > tbody > tr > td")  for i in tr:  #크롤링된 데이터를 리스트에 append  list\_td.append(i.text)  for i in range(4,13):  #셀렉터.click  driver.find\_element\_by\_css\_selector("#content > div.costWrap > div.pagination > a:nth-child(%d)"%(i)).click()  soup=BeautifulSoup(driver.page\_source,"html.parser")  #크롤링할 요소2  tr=soup.select("#content > div.costWrap > div.tableType01 > table > tbody > tr > td")  for i in tr:  #크롤링된 데이터를 리스트에append  list\_td.append(i.text)  driver.find\_element\_by\_css\_selector("#content > div.costWrap > div.pagination > a.nextBtn").click()  a = 0  b = 11  c = 11  import pandas as pd  #데이터베이스에 넣어줄 line수  line=683  for i in range(0,line):  #1개의 row 들어갈 데이터를 리스트에 append  DF\_list.append(list\_td[a:b])  a=c  b=c+11  c+=11  #넣어준 어레이를 판다스데이터프레임화시킨다.  DF=pd.DataFrame(DF\_list)  DF\_list  #데이터베이스 연결  import cx\_Oracle  con = cx\_Oracle.connect('lifecare','lifecare',cx\_Oracle.makedsn('192.168.219.156',1521,'xe'))  cur = con.cursor()  #판다스 데이터를 튜퓰화  rows = [tuple(x) for x in DF.values]  cur.executemany('''INSERT INTO non\_payment\_treatment  (medium\_div,div,name,code,repet,cost,min\_cost,max\_cost,add\_medicine,add\_drug\_cost,issue)  VALUES (:0,:1,:2,:3,:4,:5,:6,:7,:8,:9,:10)''',rows)  con.commit()  cur.close()  con.close()  최초 시작페이지에서 1회 크롤링후 셀리니움으로 화면을 이동 할 때마다 1회 크롤링 하여  Data set을 만든 후 pandas로 튜플화 하여 OracleDB에 insert | | | |

2.DB에 insert된 데이터를 Select하여 view에 출력

|  |  |  |  |
| --- | --- | --- | --- |
| 작성자 | 윤재진 | 작성일 | 2020.10.19 |
| OracleDB, Spring | | | |
| 1. OracleDB 확인(총 4종류의 정보를 4게의 테이블에 정리)  -정리된 테이이블    -긁어온 정보 확인  C:\Users\KOSMO_24\Desktop\1212.png  2.Spring을 이용하여 MVC패턴으로 게시물을 화면에 출력  -Controller  C:\Users\KOSMO_24\Desktop\257A813A58D9346201.jpg  -Models  C:\Users\KOSMO_24\Desktop\257A813A58D9346201.jpg  -View  C:\Users\KOSMO_24\Desktop\257A813A58D9346201.jpg | | | |

**-개발일정\_안드로이드 날씨 관련 건강지수**

****

**안드로이드 앱>건강정보**

**–사용된 기술 :** Android, Jsoup , 공공데이터 API, AsyncTask

**-구현이유**

사용자들에게 날씨정보와 식중독, 미세먼지, 감기, 폐질환, 뇌졸증 위험 지수를 제공하여 건강관리에 도움을 주고자 구현하였습니다.

**-구현 방법**

1.날씨/미세먼지정보 얻기

-날씨 정보는 현제 자신이 위치한 위도, 경도 값을 얻어 geocoder를 이용하여 주소로변환

-주소값을 얻으면 jsoup을 통해 구글주소+검색 쿼리스트링에 주소를 넣고 HTML 정보를얻어온다

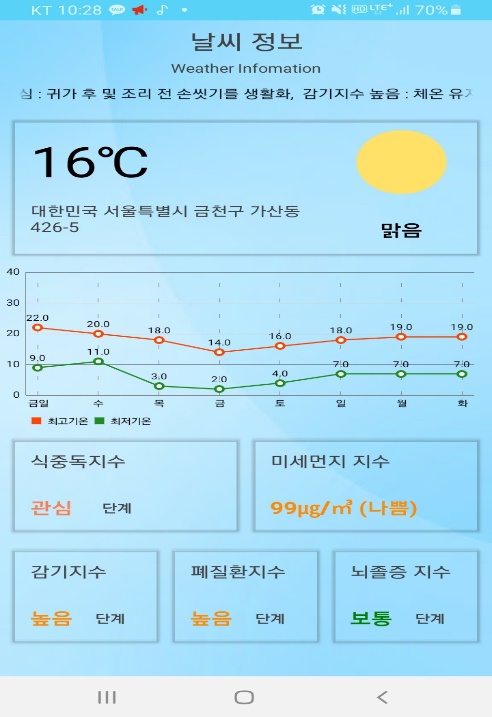
-HTML 정보중에 날씨와 요일 금일 날씨 이미지URL을 얻어와 값을 Setting한다.

-위와 같은 방법으로 네이버를 통해 미세먼지 지수를 얻어와 값을 Setting 한다.

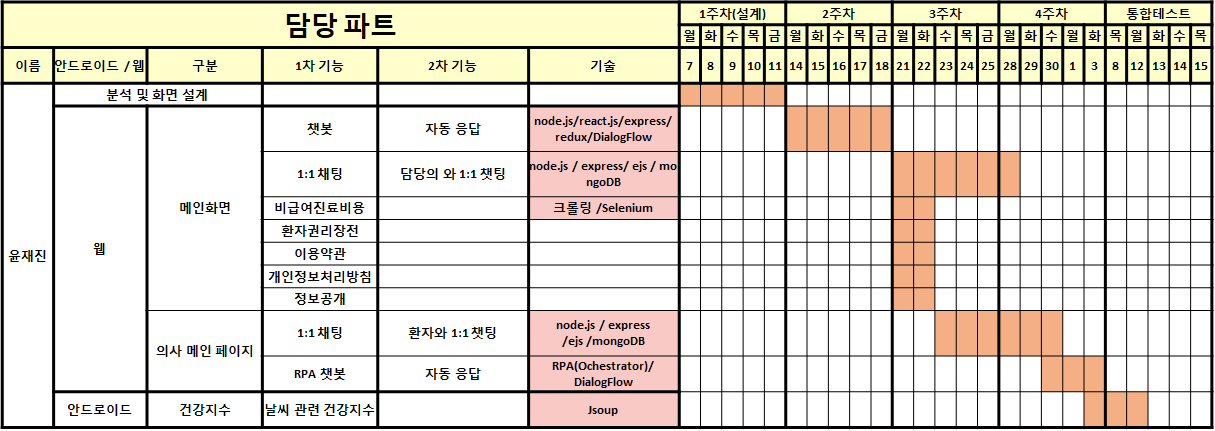
**2.건강지수 얻기**

-공공데이터 보건기상지수 API를 통해 식중독, 감기, 폐질환, 뇌졸증 지수 등의 정보를 얻어 그 값을 Setting

1.건강정보 페이지 설명



|  |  |  |  |
| --- | --- | --- | --- |
| 작성자 | 윤재진 | 작성일 | 2020.10.19 |
| health.java/  activity\_health.xml | Com.example.lifecare.health.java/  res.layout.activity\_health.xml | | |
| -위도 경도 값 얻기 : LocationManager 를 통해 위도 경도를 얻는다  final LocationManager lm = (LocationManager) getSystemService(Context.*LOCATION\_SERVICE*); Location location = lm.getLastKnownLocation(LocationManager.*GPS\_PROVIDER*);  if (location == null) {  location = lm.getLastKnownLocation(LocationManager.*NETWORK\_PROVIDER*);  try{  wido = Double.*toString*(location.getLatitude());//위도  kyungdo = Double.*toString*(location.getLongitude());//경도  }catch (Exception e){  Toast.*makeText*(getApplicationContext(), "위치서비스를 켜주세요", Toast.*LENGTH\_LONG*).show();  Intent intent = new Intent(getApplicationContext(), MainActivity.class);  startActivity(intent);  }  } final LocationListener mLocationListener = new LocationListener() {  public void onLocationChanged(Location location) {  if (location.getProvider().equals(LocationManager.*GPS\_PROVIDER*)) {  try{  wido = Double.*toString*(location.getLatitude());//위도  kyungdo = Double.*toString*(location.getLongitude());//경도  }catch (Exception e){  Toast.*makeText*(getApplicationContext(), "위치서비스를 켜주세요", Toast.*LENGTH\_LONG*).show();  Intent intent = new Intent(getApplicationContext(), MainActivity.class);  startActivity(intent);  }  }  }  public void onProviderDisabled(String provider) {}  public void onProviderEnabled(String provider) {}  public void onStatusChanged(String provider, int status, Bundle extras) {} };  -위도 경도 값으로 현재 주소 얻기  //검색할 지역 샛팅  //좌표를 통해 위치 가져오기  final Geocoder geocoder = new Geocoder(health.this);  List<Address> list = null;  try {  list = geocoder.getFromLocation(  Double.*parseDouble*(wido), // 위도  Double.*parseDouble*(kyungdo), // 경도  10); // 얻어올 값의 개수  } catch (Exception e) {  Toast.*makeText*(getApplicationContext(), "위치서비스를 켜주세요", Toast.*LENGTH\_LONG*).show();  Intent intent = new Intent(getApplicationContext(), MainActivity.class);  startActivity(intent);  }  nowWhere = list.get(0).getAddressLine(0);  -얻은 주소를 바탕으로 구글과 네이버 URL의 쿼리스트링을 이용하여 검색한 뒤 날씨 정보를 크롤링한다.  (어싱크 테스트 이용)  @Override  protected Void doInBackground(Void... params) {  try {  //구글에서 가져올 이미지  doc = Jsoup.*connect*("https://www.google.com/search?rlz=1C1OKWM\_enKR895KR895&sxsrf=ALeKk01pWao6aprfNgj5uQ\_cqBw\_GjqpCw%3A1602326693887&ei=pZCBX6DMNZiSr7wP3--L2Ak&q="+first+"+날씨").get();  System.*out*.println("검색어"+first);  Elements d=doc.select("#wob\_wc.vk\_c.nawv0d");  Elements b=d.select("#wob\_dp.wob\_dfc");  Elements c=b.select(".wob\_t");  //오전  graph1=Integer.*parseInt*(c.eq(0).text());  graph2=Integer.*parseInt*(c.eq(4).text());  graph3=Integer.*parseInt*(c.eq(8).text());  graph4=Integer.*parseInt*(c.eq(12).text());  graph5=Integer.*parseInt*(c.eq(16).text());  graph6=Integer.*parseInt*(c.eq(20).text());  graph7=Integer.*parseInt*(c.eq(24).text());  graph8=Integer.*parseInt*(c.eq(28).text());  //오후  graph9=Integer.*parseInt*(c.eq(2).text());  graph10=Integer.*parseInt*(c.eq(6).text());  graph11=Integer.*parseInt*(c.eq(10).text());  graph12=Integer.*parseInt*(c.eq(14).text());  graph13=Integer.*parseInt*(c.eq(18).text());  graph14=Integer.*parseInt*(c.eq(22).text());  graph15=Integer.*parseInt*(c.eq(26).text());  graph16=Integer.*parseInt*(c.eq(30).text());  //요일  todayp1=b.select(".QrNVmd").eq(2).text();  todayp2=b.select(".QrNVmd").eq(4).text();  todayp3=b.select(".QrNVmd").eq(6).text();  todayp4=b.select(".QrNVmd").eq(8).text();  todayp5=b.select(".QrNVmd").eq(10).text();  todayp6=b.select(".QrNVmd").eq(12).text();  todayp7=b.select(".QrNVmd").eq(14).text();  //네이버 미세먼지  doc2 = Jsoup.*connect*("https://search.naver.com/search.naver?sm=top\_hty&fbm=1&ie=utf8&query="+first+"날씨").get();  //웨더 이미지 cssQuery 는 css의 클래스 명, 아이디, 등등  Elements image = doc.select("#wob\_tci.wob\_tci").eq(0);  wImageUrl=image.attr("src");  //날씨정보 가져오기  Elements infomation = doc.select("#wob\_dc.vk\_gy.vk\_sh").eq(0);  wInfomation=infomation.text();    //온도정보 가져오기  wDgree=doc.select("#wob\_tm").eq(0).text();  //미세먼지 가져오기  dust=doc2.select(".indicator").eq(1).select(".num").eq(0).text();  } catch (IOException e) {  e.printStackTrace();  }  return null;  }  -가져온 날씨정보를 그래프화 하고 날씨정보를 setText를 통해 안드로이드 화면에 출력한다.  //날시 이미지 뿌리기  Glide.*with*(getApplicationContext()).load("https:"+wImageUrl).into(weather\_image);  //날씨정보 뿌리기  wi.setText(wInfomation);  //위치정보 뿌리기  where.setText(nowWhere);  //온도 뿌리기  dgree.setText(wDgree+"℃");  //온도그래프 뿌리기  //그래프  lineChart =(LineChart)findViewById(R.id.*chart*);  List<Entry> entries= new ArrayList<>();  //오전  entries.add(new Entry(1f, graph1));  entries.add(new Entry(2f, graph2));  entries.add(new Entry(3f, graph3));  entries.add(new Entry(4f, graph4));  entries.add(new Entry(5f, graph5));  entries.add(new Entry(6f, graph6));  entries.add(new Entry(7f, graph7));  entries.add(new Entry(8f, graph8));      List<Entry> entries1= new ArrayList<>();  entries1.add(new Entry(1f,graph9));  entries1.add(new Entry(2f,graph10));  entries1.add(new Entry(3f,graph11));  entries1.add(new Entry(4f,graph12));  entries1.add(new Entry(5f,graph13));  entries1.add(new Entry(6f,graph14));  entries1.add(new Entry(7f,graph15));  entries1.add(new Entry(8f,graph16));  -각종 건강 지수를 API를 통해 가져와 정보를 가공한다.(지수는 숫자로 얻어지며 그 숫자에 대한 단계를 글로 표현한다)  @Override  protected Void doInBackground(Void... voids) {  try {  //식중독  juso = "http://apis.data.go.kr/1360000/HealthWthrIdxService/getFoodPoisoningIdx";  fp = get.getJisu(juso, jijum);  System.*out*.println(fp);  a = gs.fromJson(fp, Map.class);  b = (Map) a.get("response");  c = (Map) b.get("body");  d = (Map) c.get("items");  e = (Map) d.get("item").get(0);  if(!e.get("today").equals("")){  f = Integer.*parseInt*(e.get("today"));  }else{  f = Integer.*parseInt*(e.get("tomorrow"));  }    //식중독  if (f < 55) {  foodp = "관심";  foodpoison.setTextColor(Color.*parseColor*(coral));  slideText=slideText+" 식중독 관심 : 귀가 후 및 조리 전 손씻기를 생활화, ";    } else if (55 <= f && f < 71) {  foodp = "주의";  foodpoison.setTextColor(Color.*parseColor*(maroon));  slideText=slideText+" 식중독 주의 : 식중독 예방 철저 조리시 반듯이 완전히 조리, ";    } else if (71 <= f && f < 86) {  foodp = "경고";  foodpoison.setTextColor(Color.*parseColor*(darkOrange));  slideText=slideText+" 식중독 경고 : 음식물 관리 도구 소독 철저히, ";  } else if (86 <= f) {  foodp = "위험";  foodpoison.setTextColor(Color.*parseColor*(crimson));  slideText=slideText+" 식중독 경고 : 식중독 의심시 의사 진료 요망 의심환자는 식품 조리 금지, ";  }  //감기  juso = "http://apis.data.go.kr/1360000/HealthWthrIdxService/getColdIdx";  fp = get.getJisu(juso, jijum);  System.*out*.println(fp);  a = gs.fromJson(fp, Map.class);  b = (Map) a.get("response");  c = (Map) b.get("body");  d = (Map) c.get("items");  e = (Map) d.get("item").get(0);    int cnt=0;    if(!e.get("today").equals("")){  cnt = Integer.*parseInt*(e.get("today"));  }else{  cnt = Integer.*parseInt*(e.get("tomorrow"));  }    if(cnt==0){  tCold="낮음";  cold.setTextColor(Color.*parseColor*(greenYellow));  slideText=slideText+" 감기지수 낮음 : 건강한 생활 습관 유지, ";  }else if(cnt==1){  tCold="보통";  cold.setTextColor(Color.*parseColor*(green));  slideText=slideText+" 감기지수 보통 : 수분 적절히 섭취, 개인위생 철저히, ";  }else if (cnt==2){  cold.setTextColor(Color.*parseColor*(darkOrange));  tCold="높음";  slideText=slideText+" 감기지수 높음 : 체온 유지, 적정 실내온도 유지, ";  }else if (cnt==3){  cold.setTextColor(Color.*parseColor*(crimson));  tCold="매우높음";  slideText=slideText+" 감기지수 매우높음 : 방한용품 사용, 외출 자제, ";  }    //폐질환  juso = "http://apis.data.go.kr/1360000/HealthWthrIdxService/getAsthmaIdx";  fp = get.getJisu(juso, jijum);  System.*out*.println(fp);  a = gs.fromJson(fp, Map.class);  b = (Map) a.get("response");  c = (Map) b.get("body");  d = (Map) c.get("items");  e = (Map) d.get("item").get(0);    if(!e.get("today").equals("")){  cnt = Integer.*parseInt*(e.get("today"));  }else{  cnt = Integer.*parseInt*(e.get("tomorrow"));  }    if(cnt==0){  lungDisease.setTextColor(Color.*parseColor*(greenYellow));  tLungDisease="낮음";  slideText=slideText+" 폐질환지수 낮음 : 건강관리 유의, ";  }else if(cnt==1){  lungDisease.setTextColor(Color.*parseColor*(green));  tLungDisease="보통";  slideText=slideText+" 폐질환지수 보통 : 만성 폐질환 환자는 주의, ";  }else if (cnt==2){  lungDisease.setTextColor(Color.*parseColor*(darkOrange));  tLungDisease="높음";  slideText=slideText+" 폐질환지수 높음 : 외부와 노출 주의 공기청정기 사용, ";  }else if (cnt==3){  lungDisease.setTextColor(Color.*parseColor*(crimson));  tLungDisease="매우높음";  slideText=slideText+" 폐질환지수 매우높음 : 천식 환자는 각별히 주의, ";  }  //뇌졸증  juso = "http://apis.data.go.kr/1360000/HealthWthrIdxService/getStrokeIdx";  fp = get.getJisu(juso, jijum);  System.*out*.println(fp);  a = gs.fromJson(fp, Map.class);  b = (Map) a.get("response");  c = (Map) b.get("body");  d = (Map) c.get("items");  e = (Map) d.get("item").get(0);    if(!e.get("today").equals("")){  cnt = Integer.*parseInt*(e.get("today"));  }else{  cnt = Integer.*parseInt*(e.get("tomorrow"));  }    if(cnt==0){  tBrain="낮음";  brain.setTextColor(Color.*parseColor*(greenYellow));  slideText=slideText+" 뇌졸증지수 낮음 : 평소 뇌졸증 예방 관리, ";  }else if(cnt==1){  brain.setTextColor(Color.*parseColor*(green));  tBrain="보통";  slideText=slideText+" 뇌졸증지수 보통 : 갑작스러운 체온변화 주의, ";  }else if (cnt==2){  brain.setTextColor(Color.*parseColor*(darkOrange));  tBrain="높음";  slideText=slideText+" 뇌졸증지수 높음 : 외부 온도가 낮을경우 외출 금지, ";  }else if (cnt==3){  brain.setTextColor(Color.*parseColor*(crimson));  tBrain="매우높음";  slideText=slideText+" 뇌졸증지수 매우높음 : 기왕력 환자들은 각결히 주의, ";  }    } catch (IOException e) {  e.printStackTrace();  }      return null;  }  -모든 AsyncTask의 셋팅이 완료되면  //지수  sync s = new sync();  s.execute();    //온도  JsoupAsyncTask j = new JsoupAsyncTask();  j.execute();  excute하여 실행시켜준다. | | | |

****

1. **홈페이지 > 의사페이지 > RPA챗봇**

* **사용된 기술 Uipath ,Uipath Ochastrator , Dialog flow**
* **기술을 구현한 이유**

**의사가 반복적으로 해야하는 업무를 액티비티화 하였고 이 액티비티를 RPA 챗봇 과 연계하면서 의사의 반복업무 시간을 줄이고 좀 더 쉬운 인터페이스를 제공하기 위하여 만듦**

|  |  |  |  |
| --- | --- | --- | --- |
| 작성자 | 윤재진 | 작성일 | 2020.10.19 |
| **Uipath (프로세스 작성)** | | | |
| 사진설명    네이버에서 코로나 정보를 크롤링 하여 Dialog flow 에 인수로(방향을 출력) 값을 전달하여 챗봇에서 dialogflow 인수로 전달한다. | | | |

|  |
| --- |
| **Uipath Ochastrator (프로세스 등록 및 머신, 로봇 생성)** |
| 1.머신등록(오케스트레이터의 머신명으로 등록해야한다)    2. 로봇 생성(환경설정 후 로봇을 등록한다)    3.환경설정    4.Uipath Process 등록    Uipath 에서 작성된 프로세스를 Uipath ochastrator 에서 등록하여 dialog flow와 연동을 할 준비 작업을  해준다 준비작업으로 는 머신생성 -> 로봇 생성 ->환경설정을 해야한다. |

|  |
| --- |
| Dialog Flow(자연어 학습) |
| 인텐트를 생성하여 학습을 시킨다.    질문 목록을 등록한다.    변수 생성하여 어떤 인수를 받을지 엔티티를 정하여준다.  Uipath 에서 보낸 인수를 받기 위해 <> 안에 인수 명을 입력해준다.    생성된 인텐트는 Uipath Chatbot 에서 Mapped 해준다. |

|  |
| --- |
| **ChatBot(Uipath Assistant실행화면)** |
| Uipath Assistant를 통해 상태를 확인할수있다. |