private async void networkCheckQuiz()

{

string url = "http://58.145.23.13:5555/dotnet/app.do?kind=checkQuiz&quizId=0001&answer=김밥";

string data = "{datas~~~~}";

string result = await Task.Run(() =>

{

var request = (HttpWebRequest)WebRequest.Create(url);

request.Method = "POST";

request.ContentType = "application/json";

request.Timeout = 10000;

byte[] bytes = Encoding.ASCII.GetBytes(data);

request.ContentLength = bytes.Length;

using (var stream = request.GetRequestStream())

stream.Write(bytes, 0, bytes.Length);

using (var response = (HttpWebResponse)request.GetResponse())

using (var reader = new StreamReader(response.GetResponseStream()))

return reader.ReadToEnd();

});

dynamic json = JObject.Parse(result);

string isConfirm = json["isConfirm"];

this.Invoke((MethodInvoker)(() =>

{

MessageBox.Show(isConfirm == "Y" ? "정답" : "오답");

}));

}

네트워크 비동기 통신, 스레드 통신 왜 별도 스레드 사용하는지 서술형 문제.

메인 스레드는 UI를 담당하며 항상 사용자 입력에 빠르게 반응해야 합니다. 하지만 네트워크 통신은 시간이 오래 걸릴 수 있기 때문에, 메인 스레드에서 처리할 경우 수업 시간에 경험했던 것처럼 UI가 멈추거나 응답하지 않는 현상이 발생할 수 있습니다. 이러한 문제를 방지하기 위해, 네트워크 작업은 별도의 스레드나 비동기 처리 방식(Thread 등)을 통해 처리합니다. 이를 통해 UI의 반응성을 유지하면서 성능 문제도 해결할 수 있습니다.

서버 통신

정답 혹은 로그인 정보(아이디, 비밀번호) -> 서버 전송 -> 정답확인 -> 로그인 및 결과

타이머 사용, 라벨, 버튼, 텍스트 박스, 이미지(애니메이션 구현)















