

스마트한 학습을 위한 애플리케이션



20211842 정가을

목차

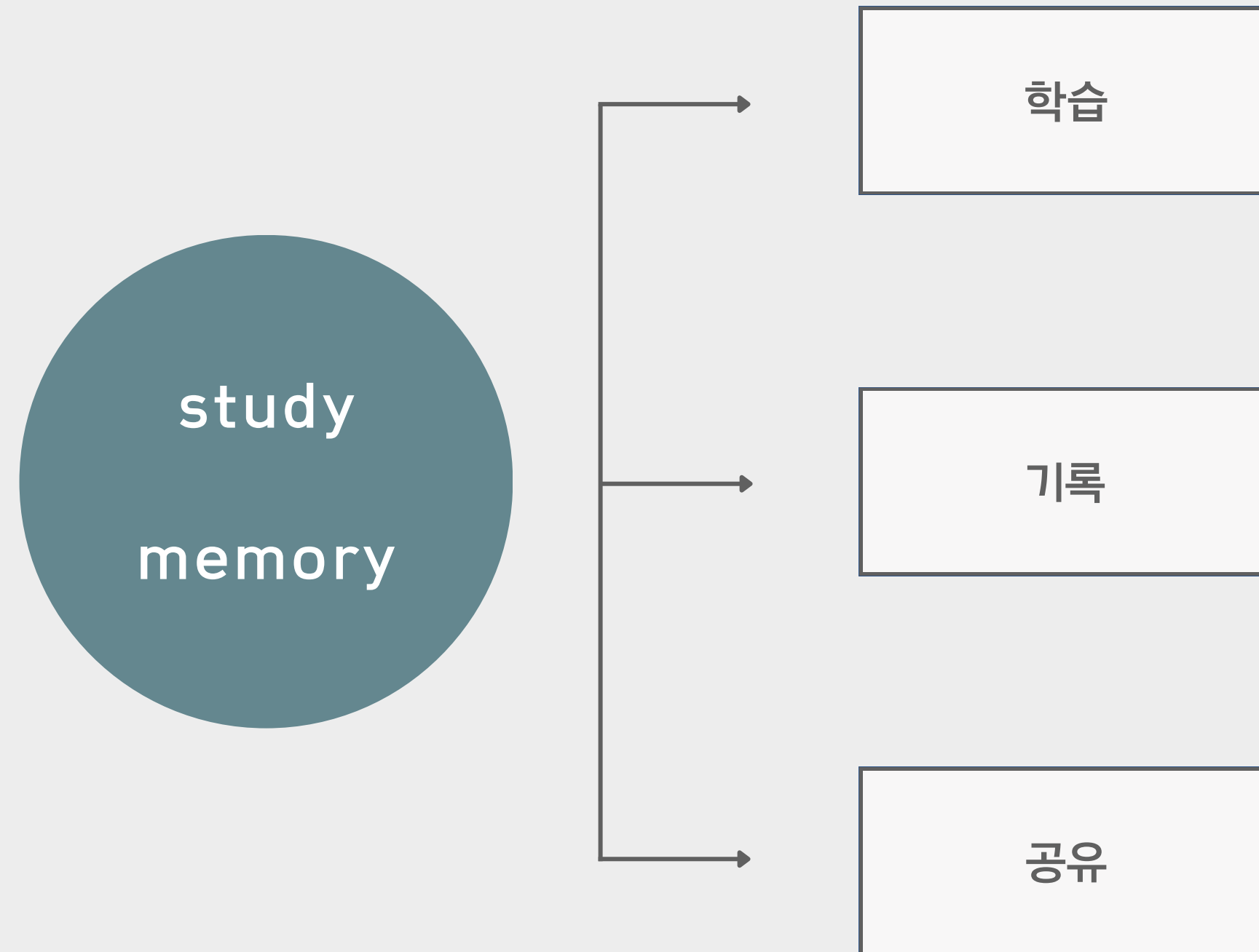
- 01 Study memory 란?
- 02 Study memory 개발 이유
- 03 Study memory 메인 기능
- 04 Study memory 구현
- 05 향후 계획, 느낀점

01

study memory란

무엇인가요?

- study & memory



02

Study memory

개발 이유

• Study memory 개발 이유



효율적인 학습 관리

공부한 내용을 체계적으로 정리하고, 복습할 수 있는 애플리케이션의 필요성



공부 내용의 공유

다른 사람들과 학습 내용을 공유하여 피드백을 받거나 함께 학습할 수 있는 수단 필요

02

Study memory

메인 기능

• Study memory 기능

01

학습 기록 및 관리

텍스트, 이미지, 링크 등 다양한
형식의 학습 자료를 입력하고 저장

02

학습 내용 공유

다른 사람의 학습 방법을
참고하거나, 함께 학습하는
동료와 자료를 공유하며 협업

03

학습 시간 측정

공부를 시작할 때 타이머를
시작하고, 학습이 끝나면 타이머를
멈춰 실제 학습 시간을 기록하는 기능

04

Study memory

구현

• Study memory MySQL

User Table

MySQL Workbench interface showing the 'user' table in the 'contacts' schema. The query 'SELECT * FROM contacts.user;' is executed, and the result grid displays the following data:

userName	userPassword	userAge	userEmail	userNum	userPhone
gaetul	1234	23	ga1398@naver.com	1	010-9513-1398

The bottom panel shows the execution log with two actions:

#	Time	Action	Message	Duration / Fetch
1	11:17:55	SELECT * FROM contacts.memory LIMIT 0, 1000	1 row(s) returned	0.000 sec / 0.000 sec
2	11:18:00	SELECT * FROM contacts.user LIMIT 0, 1000	1 row(s) returned	0.000 sec / 0.000 sec

Memory Table

MySQL Workbench interface showing the 'memory' table in the 'contacts' schema. The query 'SELECT * FROM contacts.memory;' is executed, and the result grid displays the following data:

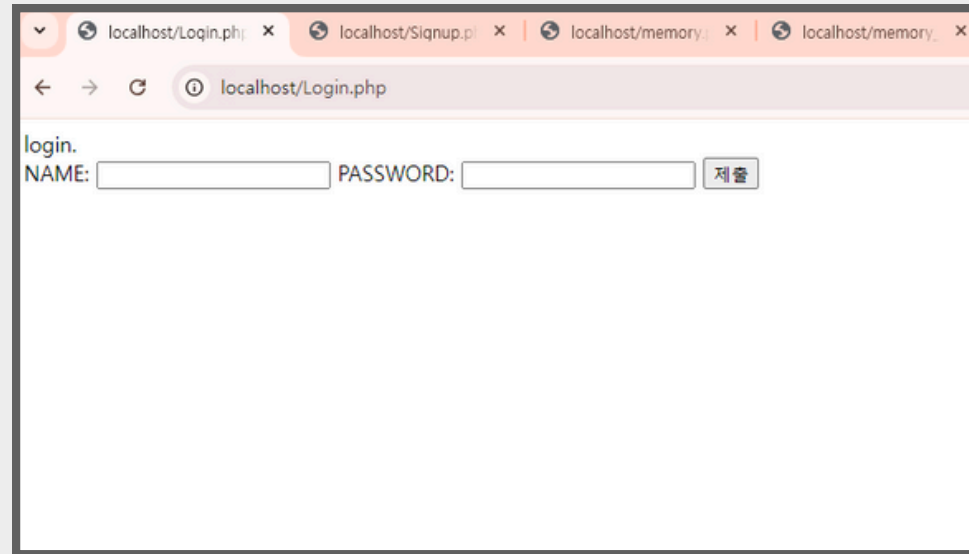
Title	Author	Memory
hi	gaetul	hi hello

The bottom panel shows the execution log with two actions:

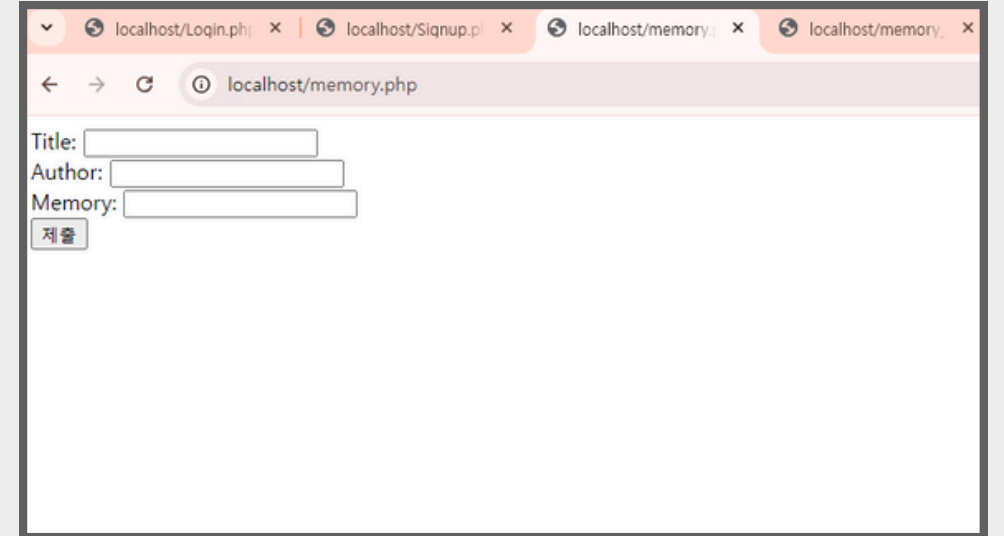
#	Time	Action	Message	Duration / Fetch
1	11:17:55	SELECT * FROM contacts.memory LIMIT 0, 1000	1 row(s) returned	0.000 sec / 0.000 sec
2	11:18:00	SELECT * FROM contacts.user LIMIT 0, 1000	1 row(s) returned	0.000 sec / 0.000 sec

• Study memory PHP 서버

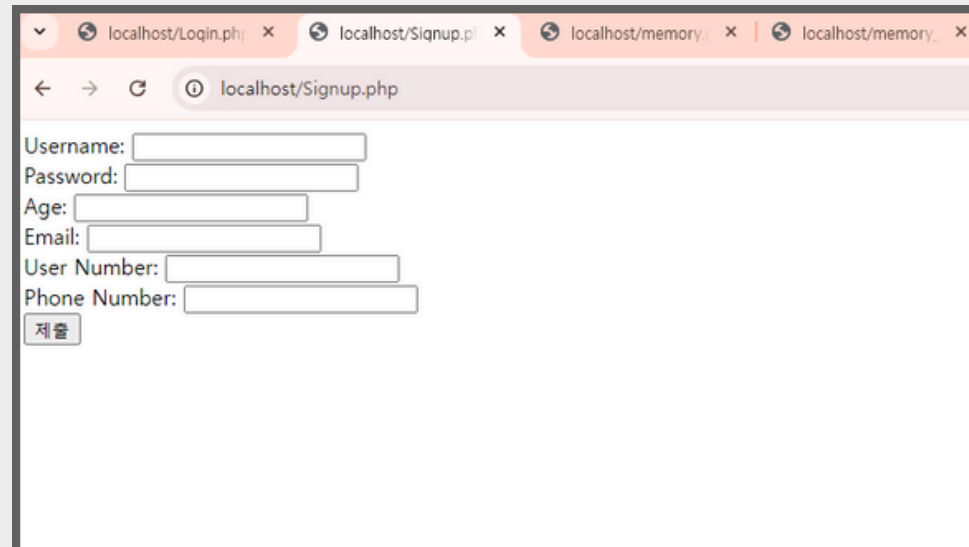
로그인 php



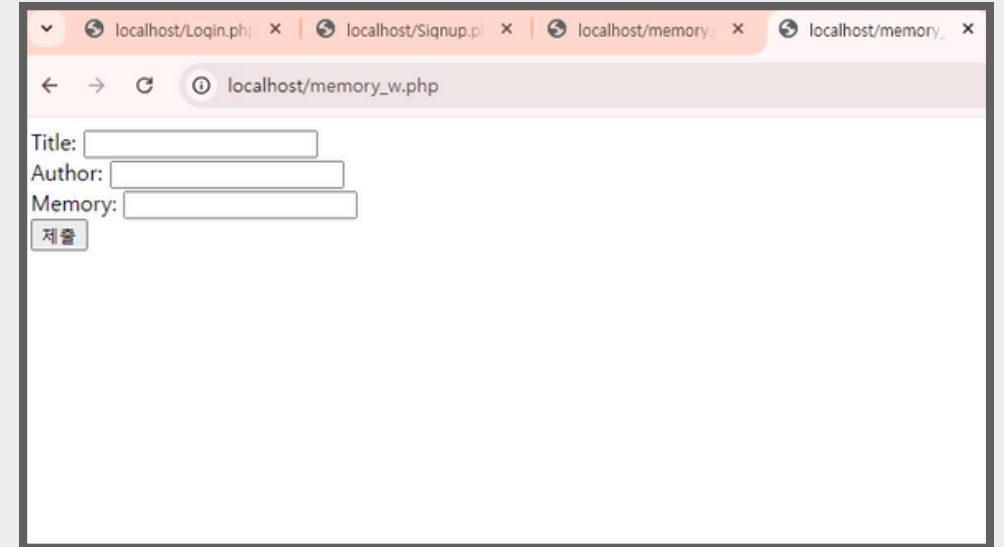
내용 기록 php



회원가입 php



기록 목록 php




• Study memory 구현

로그인화면

이름과 비밀번호를 작성 했을때
일치한다면 로그인에 성공하도록 구성

로그인



name

Password

LOGIN

처음 접속시 아래 회원가입을 눌러 회원가입 해주세요

회원가입

• Study memory 구현

로그인화면

이름과 비밀번호를 작성 했을때
일치한다면 로그인에 성공하도록 구성

```
private void performLogin() { 1 usage
    final String name = etName.getText().toString();
    final String pass = etPass.getText().toString();

    // Check if name or password fields are empty
    if (name.isEmpty() || pass.isEmpty()) {
        tvInfor.setText("이름과 비밀번호를 입력해 주세요.");
        return;
    }

    new Thread(new Runnable() {
        @Override
        public void run() {
            try {
                URL url = new URL( spec "http://10.0.2.2/Login.php");
                HttpURLConnection conn = (HttpURLConnection) url.openConnection();
                conn.setRequestMethod("POST");
                conn.setDoOutput(true);
                conn.setRequestProperty("Content-Type", "application/x-www-form-urlencoded");

                // Send POST data
                String postData = "userName=" + name + "&userPassword=" + pass;
                OutputStreamWriter writer = new OutputStreamWriter(conn.getOutputStream());
                writer.write(postData);
                writer.flush();
                writer.close();

                // Get response
                BufferedReader reader = new BufferedReader(new InputStreamReader(conn.getInputStream()));
                StringBuilder response = new StringBuilder();
                String line;
                while ((line = reader.readLine()) != null) {
                    response.append(line);
                }
                reader.close();


                final String result = response.toString();
                runOnUiThread(new Runnable() {
                    @Override
                    public void run() { handleLoginResponse(result); }
                });
            } catch (Exception e) {
                e.printStackTrace();
            }
        }
    }).start();
}
```

• Study memory 구현

회원가입화면

이름과 비밀번호, 나이, 이메일, 번호,
전화번호를 입력 받을 수 있도록 구성

회원가입



name

Password

age

email

num

phone number

JOINBACK

• Study memory 구현

회원가입화면

이름과 비밀번호, 나이, 이메일, 번호,
전화번호를 입력 받을 수 있도록 구성

```
@Override no usages
protected String doInBackground(String... params) {
    String serverURL = params[0];
    String userName = params[1];
    String userPassword = params[2];
    String userEmail = params[3];
    String userPhone = params[4];
    String userAge = params[5];
    String userNum = params[6];

    String postParameters = "userName=" + userName + "&userPassword=" + userPassword + "&userAge=" + userAge + "&userEmail=" + userEmail + "&userNum=" + userNum + "&userPhone=" + userPhone;

    try {
        URL url = new URL(serverURL);
        HttpURLConnection httpURLConnection = (HttpURLConnection) url.openConnection();

        httpURLConnection.setReadTimeout(5000);
        httpURLConnection.setConnectTimeout(5000);
        httpURLConnection.setRequestMethod("POST");
        httpURLConnection.setDoOutput(true);
        httpURLConnection.connect();

        OutputStream outputStream = httpURLConnection.getOutputStream();
        outputStream.write(postParameters.getBytes( "UTF-8"));
        outputStream.flush();
        outputStream.close();

        int responseStatusCode = httpURLConnection.getResponseCode();
        Log.d(TAG, "POST response code - " + responseStatusCode);

        InputStream inputStream;
        if (responseStatusCode == HttpURLConnection.HTTP_OK) {
            inputStream = httpURLConnection.getInputStream();
        } else {
            inputStream = httpURLConnection.getErrorStream();
        }

        InputStreamReader inputStreamReader = new InputStreamReader(inputStream, "UTF-8");
        BufferedReader bufferedReader = new BufferedReader(inputStreamReader);

        StringBuilder sb = new StringBuilder();
        String line;
        while ((line = bufferedReader.readLine()) != null) {
            sb.append(line);
        }
        bufferedReader.close();

        return sb.toString();
    } catch (Exception e) {
        Log.d(TAG, "InsertData: Error ", e);
        return "Error: " + e.getMessage();
    }
}
```

• Study memory 구현

홈화면

상단에는 회원 가입

당시 입력한 정보를 출력하고

하단에는 캘린더가 위치하도록 구성

홈

Name:

Age:

Email:

Number:

Phone:

<

1970년 1월

>

일	월	화	수	목	금	토
				1	2	3
4	5	6	7	8	9	10
11	12	13	14	15	16	17
18	19	20	21	22	23	24
25	26	27	28	29	30	31

작성

타이머

목록

• Study memory 구현

홈화면

상단에는 회원 가입

당시 입력한 정보를 출력하고

하단에는 캘린더가 위치하도록 구성

```
btnWrite = findViewById(R.id.btnWrite);
btnList = findViewById(R.id.btnList);
btnTime = findViewById(R.id.btnTime);

tvUserName = findViewById(R.id.tv_user_name);
tvUserAge = findViewById(R.id.tv_user_age);
tvUserEmail = findViewById(R.id.tv_user_email);
tvUserNum = findViewById(R.id.tv_user_num);
tvUserPhone = findViewById(R.id.tv_user_phone);

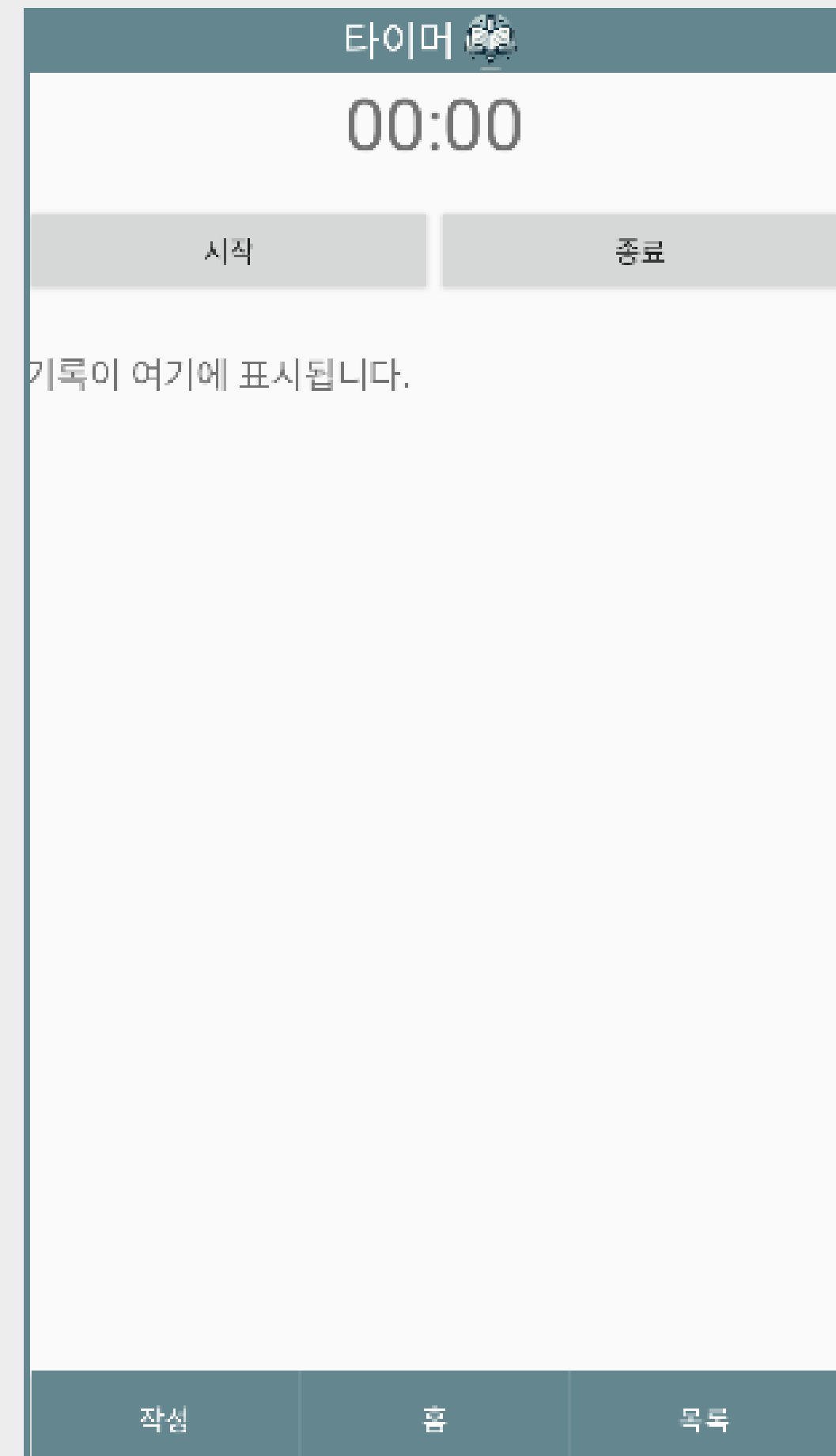
String userName = getIntent().getStringExtra( name: "userName");
String userAge = getIntent().getStringExtra( name: "userAge");
String userEmail = getIntent().getStringExtra( name: "userEmail");
String userNum = getIntent().getStringExtra( name: "userNum");
String userPhone = getIntent().getStringExtra( name: "userPhone");

tvUserName.setText("Name: " + userName);
tvUserAge.setText("Age: " + userAge);
tvUserEmail.setText("Email: " + userEmail);
tvUserNum.setText("Number: " + userNum);
tvUserPhone.setText("Phone: " + userPhone);
```

• Study memory 구현

타이머화면

공부 한 시간을 정확하게
측정하여 기록할 수 있도록 구성



• Study memory 구현

타이머화면

공부 한 시간을 정확하게
측정하여 기록할 수 있도록 구성

```
// 시간 측정을 시작하는 메서드
private void startTimer() { 1 usage
    startTime = SystemClock.elapsedRealtime(); // 현재 시간을 기록
    isRunning = true;
    handler.post(updateTimerRunnable); // 타이머 업데이트 시작
}

// 시간 측정을 종료하고 결과를 기록하는 메서드
private void stopTimer() { 1 usage
    long elapsedTime = SystemClock.elapsedRealtime() - startTime;
    isRunning = false;
    handler.removeCallbacks(updateTimerRunnable); // 타이머 업데이트 중지
    addTimeRecord(elapsedTime);
}

// 경과 시간을 TextView에 업데이트하는 메서드
private void updateTimer(long elapsedTime) { 1 usage
    int minutes = (int) (elapsedTime / 1000) / 60;
    int seconds = (int) (elapsedTime / 1000) % 60;

    String timeFormatted = String.format("%02d:%02d", minutes, seconds);
    timerTextView.setText(timeFormatted);
}

// 경과 시간을 기록하는 메서드
private void addTimeRecord(long elapsedTime) { 1 usage
    int minutes = (int) (elapsedTime / 1000) / 60;
    int seconds = (int) (elapsedTime / 1000) % 60;


    String timeFormatted = String.format("%02d:%02d", minutes, seconds);
    timeRecords.add(timeFormatted);
    updateRecordsView();
}

// 기록된 시간을 TextView에 표시하는 메서드
private void updateRecordsView() { 1 usage
    StringBuilder records = new StringBuilder();
    for (String record : timeRecords) {
        records.append(record).append("\n");
    }
    recordsTextView.setText(records.toString());
}
```

• Study memory 구현

작성화면

과목과 내용을
기록할 수 있도록 구성

작성 

제목

요약

내용

저장

타이머

홈

목록

• Study memory 구현

작성화면

과목과 내용을

기록할 수 있도록 구성

```
private class InsertData extends AsyncTask<String, Void, String> { 2 usages
    ProgressDialog progressDialog; no usages

    @Override no usages
    protected String doInBackground(String... params) {
        if (params.length < 3) {
            return "Error: Insufficient parameters";
        }

        String serverURL = "http://" + IP_ADDRESS + "/memory_w.php"; // 서버 주소
        String title = params[0];
        String Author = params[1];
        String memory = params[2];

        // Include 'submit' parameter to match PHP code expectations
        String postParameters = "Title=" + title + "&Author=" + Author + "&Memory=" + memory + "&submit=true";

        try {...} catch (Exception e) {
            Log.d(TAG, msg: "InsertData: Error ", e);
            return "Error: " + e.getMessage();
        }
    }

    @Override 5 usages
    protected void onPostExecute(String result) {
        super.onPostExecute(result);

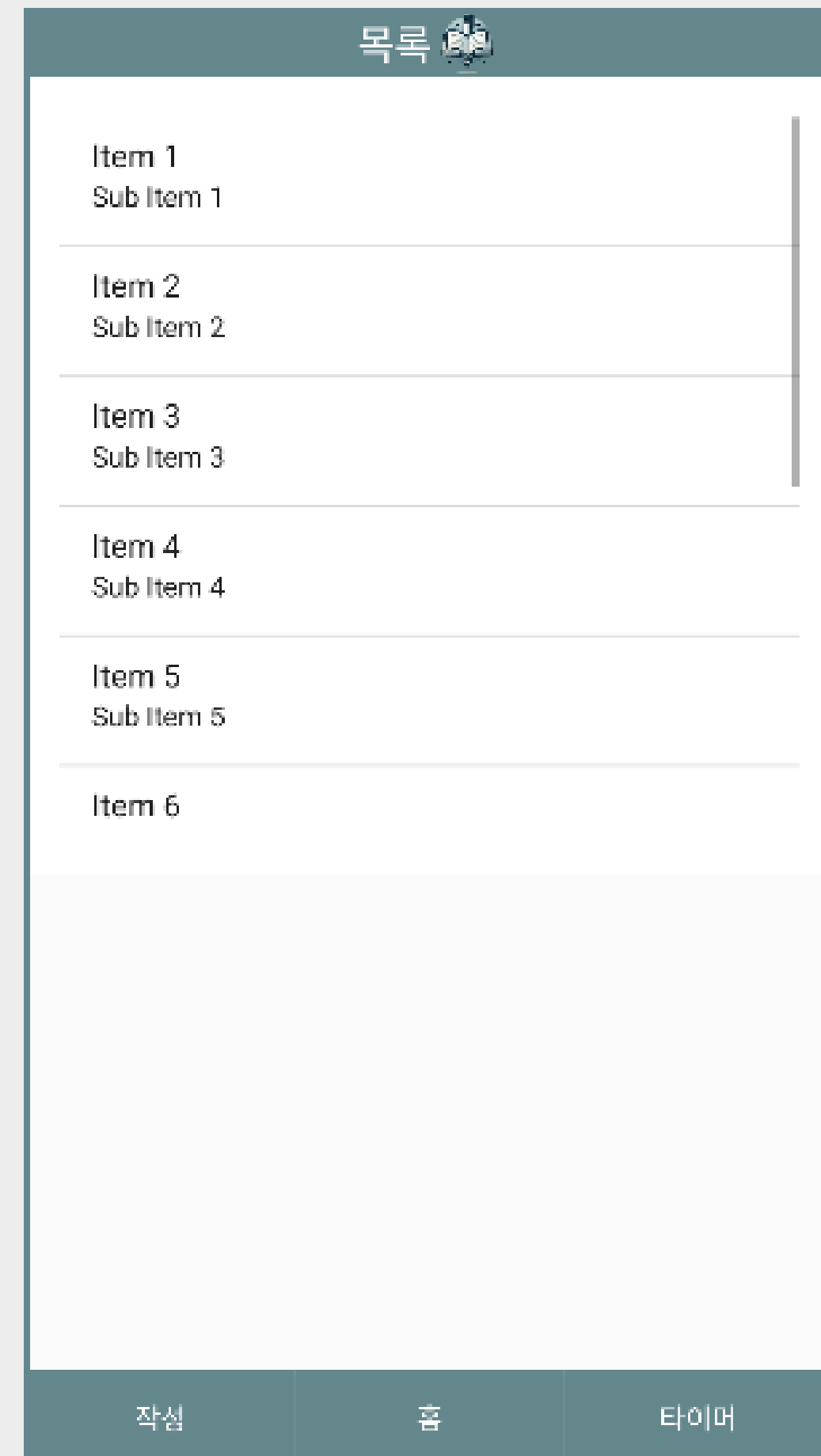
        Log.d(TAG, msg: "Server Response: " + result);

        if (result.contains("Data inserted successfully")) {
            Toast.makeText(context WriteActivity.this, text: "저장에 성공했습니다!", Toast.LENGTH_LONG).show();
        } else {
            Toast.makeText(context WriteActivity.this, text: "저장에 실패했습니다.", Toast.LENGTH_LONG).show();
        }
    }
}
```

• Study memory 구현

목록 및 공유 화면

작성한 스터디 기록
목록을 볼 수 있도록 구성



• Study memory 구현

목록 및 공유 화면

작성한 스터디 기록

목록을 볼 수 있도록 구성

```
listView.setOnItemClickListener(new AdapterView.OnItemClickListener() {  
    @Override  
    public void onItemClick(AdapterView<?> parent, View view, int position, long id) {  
        JSONObject selectedItem = listItems.get(position);  
        try {  
            String title = selectedItem.getString("title");  
            String author = selectedItem.getString("author");  
            String memory = selectedItem.optString("memory", "No memory data available");  
            textViewResult.setText("제목: " + title + "\n작성자: " + author + "\n내용: " + memory);  
        } catch (JSONException e) {  
            e.printStackTrace();  
            Log.e("ITEM_CLICK_ERROR", "Error parsing JSON object: " + selectedItem.toString());  
            Toast.makeText(context, ListActivity.this, "Error displaying details.", Toast.LENGTH_SHORT).show();  
        }  
    }  
});
```

05

향후 계획, 느낀점

- **향후 개발 과제**



계정에 따른 데이터

로그인 한 계정에 따라 기록한
데이터 들을 볼 수 있는 기능 추가

● 느낀점

학습 데이터를 저장하고 공유하기 위해서는 안정적이고 효율적인 서버 통신이 필수적이었지만, 처음 접하는 서버 개발과 연동 작업은 생각보다 복잡하고 까다로웠습니다.

이번 개발을 통해 끈기와 문제 해결 능력의 중요성을 깨달았고, 서버와 클라이언트 간의 통신 구조를 더 깊이 이해할 수 있게 되었습니다.

Q&A