

2023학년도 모바일프로그래밍 프로젝트 완료 보고서

메모라이즈(memorize)

2023 년 6 월 05 일

컴퓨터정보과

팀명	양까라 찬호
팀장	정찬호(202144091)
팀원	1. 정찬호(202144091) 2. 이소은(202144104)



인하공업전문대학
INHA TECHNICAL COLLEGE
仁荷工業專門大學

2023년도
컴퓨터정보과 모바일프로그래밍 프로젝트
수행 결과보고서
(분야 : Android 기반의 앱 개발)

연구과제명 : Android 기반의 영단어 어플리케이션 개발

2023. 6. 20



인하공업전문대학
INHA TECHNICAL COLLEGE
仁荷工業專門大學

본 과제(결과물)는 인하공업전문대학 컴퓨터정보과 모바일프로그래밍 교과목의 프로젝트 완료보고서 입니다.

제 출 문

- 분 야 : Android 기반의 소프트웨어 개발
- 연구과제명 : Memorize
- 프로젝트 책임자 : 인하공업전문대학 컴퓨터정보과 정찬호

2023년도 인하공업전문대학 컴퓨터정보과
모바일프로그래밍 교과목의 프로젝트 결과물로 이 보고서를 제출합니다.




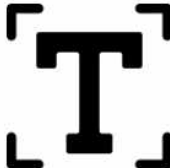
2023. 6. 05

프로젝트 책임자 : 컴퓨터정보과(202144091) 정 찬 호 (인)

공동 참여자 : 컴퓨터정보과(202144104) 아 소 은 (인)

인하공업전문대학 컴퓨터정보과 학과장 귀하

프로젝트 요약문

분 야	Android 기반의 영단어장 어플리케이션 개발
프로젝트명	Memorize
프로젝트 책임자	컴퓨터정보과(202144091) 정찬호
<div>연구 내용</div> <div><div><p>단어 암기 학습장</p></div><div><p>학습 진행률 확인</p></div><div><p>TTS</p></div><div><p>텍스트 인식</p></div></div>	

참여자 인적사항

1. 프로젝트 팀장

성 명		정 찬 호	학 번	202144091
학 과		컴퓨터정보과		
연락처	H.P	010 3158 2431		
	E-mail	uni070@naver.com		

2. 프로젝트 팀원

성 명		이 소 은	학 번	202144104
학 과		컴퓨터정보과		
연락처	H.P	010 9895 7996		
	E-mail	soeun21216@naver.com		

목 차

1. 프로젝트 목적	1
2. 프로젝트 목표와 기대 효과	1
3. 프로젝트 진행범위 및 방법	1
4. 프로젝트 주요 내용	3
5. 연구 결과물	5
5-1. 프로젝트명	5
5-2. 프로젝트 개요	5
5-3. 프로젝트 수행 세부 일정 및 내용	6
5-4. 프로젝트 결과	16
 - 부 록 -	 17
사용자 설명서	19

1. 프로젝트 목적

영어는 현재 전 세계에서 가장 널리 사용되는 언어 중 하나이며, 국제 비즈니스, 학문, 과학, 기술, 엔터테인먼트 등 여러 분야에서 중요한 역할을 한다. 따라서 영어를 학습하는 것은 다양한 이유로 매우 유용하다.

영어 학습의 가장 기본이 되는 단어를 시간과 장소의 제약 없이 받지 않는 안드로이드 앱으로 제작하여 사용자의 더욱 효율적인 학습을 도모하기 위해 본 프로젝트를 진행하고자 한다.

2. 프로젝트 목표 및 기대효과

기존의 영어 단어 암기 앱은 단순히 개발자의 편의성을 위해 사용자의 학습 수준, 관심 분야, 학습 스타일 등을 고려하지 않고 일괄적으로 학습 내용을 제공한다.

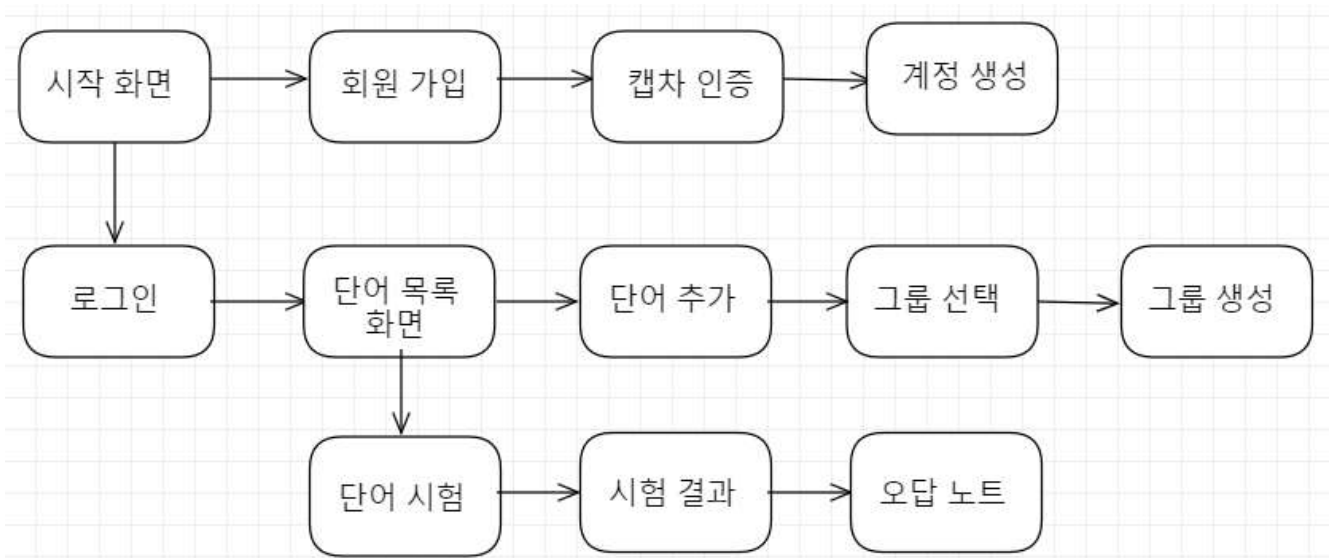
이러한 부분은 사용자의 개인적인 학습 성과와 만족도를 저하시키는 요인이 될 수 있고, 사용자가 자신의 학습 진행 상황을 파악하고, 스스로 학습 계획을 수립하는데에 어려움이 발생하게 된다.

때문에 메모라이즈에서는 사용자 맞춤형 인터페이스를 제공하여 효과적인 영어 학습을 진행할 수 있도록 하고자 한다.

3. 프로젝트 진행범위 및 방법

본 프로젝트를 통해 사용자들은 보다 효과적이고 흥미로운 방식으로 영어 단어를 학습할 수 있으며, 높은 학습 효과를 얻을 수 있다. 또한, 메모라이즈 앱은 사용자들의 만족도를 높이고 관련 분야에서의 경쟁 우위를 확보한다.

3-1. 작업 분할 구조도(WBS)



4. 프로젝트 주요 내용

4.1. 회원 가입

1. 아이디, 패스워드, 사용자 이름, 전화번호
2. 사용자 아이디 중복 방지 구현
3. Google Recaptcha 인증 구현
4. Sharedpreference를 사용해 앱 종료시 자동 로그인 구현

4.2. 영단어 입력

1. 영단어 생성, 수정, 삭제,
2. Google TTS 기능 구현
3. 단어 그룹 생성, 수정, 삭제 구현
4. 그룹별 단어 생성 구현
5. 그룹 내 단어 생성, 수정, 삭제 구현

4.3. 영단어 시험

1. 저장된 단어들로 스펠링 시험 구현
2. 모든 단어, 맞은 단어, 틀린 단어 별로 확인 구현

5. 프로젝트 결과물

5-1. 프로젝트명 : Memorize

5-2. 프로젝트 수행 세부일정 및 내용

■ 프로젝트 수행 세부 일정

프로젝트 진행 계획(2023. 5. 20. - 2023. 6. 10)											
진 행 내 용	책임자	1	2	3	4	5	6	7	8	비고	
화면 UI/UX 구성	이소은										
SQLite DB 설계	정찬호										
영단어 생성, 저장, 삭제	정찬호										
그룹 생성, 저장, 삭제	이소은										
영단어 시험	정찬호										

5-4. 프로젝트 결과

- 프로젝트 소스

```
// 시작 화면 액티비티
import androidx.appcompat.app.AppCompatActivity;

public class IntroActivity extends AppCompatActivity implements View.OnClickListener{

    private ActivityIntroBinding binding;
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        binding = ActivityIntroBinding.inflate(getLayoutInflater());
        setContentView(binding.getRoot());

        /**
         * todo 어플리케이션 시작 시 sharedPreferences에 사용자 로그인 정보가
         * todo 저장 되어 있으면 해당 계정으로 자동 로그인
         */
        getPreferences();
        binding.btnJoin.setOnClickListener(this);
        binding.btnLogin.setOnClickListener(this);
    }

    // sharedPreferences 정보를 가져오는 메소드
    private void getPreferences(){
        Map<String, String> rs = SharedPreferencesManager.getLoginInfo(this);
        String userId = rs.get("userID");
        String userPassword = rs.get("userPassword");

        if(!userId.isEmpty() && !userPassword.isEmpty()){
            Intent intent = new Intent(this, NavActivity.class);
            startActivity(intent);
        }
    }

    // 버튼 클릭 리스너
    @Override
    public void onClick(View v) {
        // 회원 가입 버튼 클릭 시 회원 가입 화면으로 이동
        if (v == binding.btnJoin){
            Intent intent = new Intent(this, JoinActivity.class);
            startActivity(intent);
        } else if (v == binding.btnLogin) { // 로그인 버튼 클릭 시 로그인 화면으로 이동
            Intent intent = new Intent(this, LoginActivity.class);
            startActivity(intent);
        }
    }
}
```

// 회원 가입 액티비티

```
public class JoinActivity extends AppCompatActivity implements View.OnClickListener{
```

```
    private ActivityJoinBinding binding;
```

```
    SQLiteHelper helper; // SQLite Helper 호출
```

```
    UserTableController sqlite;
```

```
@Override
```

```
protected void onCreate(Bundle savedInstanceState) {
```

```
    super.onCreate(savedInstanceState);
```

```
    binding = ActivityJoinBinding.inflate(getLayoutInflater());
```

```
    setContentView(binding.getRoot());
```

```
    binding.btnChap.setOnClickListener(this);
```

```
    helper = new SQLiteHelper(
```

```
        this, // context
```

```
        "MemorizeDB", // DBNAME
```

```
        null, // factory
```

```
        1 // DB 버전
```

```
    );
```

```
    sqlite = new UserTableController(helper);
```

```
    binding.btnSubmit.setOnClickListener(view -> {
```

```
        String id = binding.userId.getText().toString();
```

```
        String pw = binding.password.getText().toString();
```

```
        String name = binding.name.getText().toString();
```

```
        String phone = binding.phone.getText().toString();
```

```
        if (id.length() == 0){
```

```
            Toast.makeText(this, "아이디를 입력해 주세요", Toast.LENGTH_SHORT).show();
```

```
        } else if (pw.length() == 0) {
```

```
            Toast.makeText(this, "비밀번호를 입력해 주세요", Toast.LENGTH_SHORT).show();
```

```
        }else if (name.length() == 0) {
```

```
            Toast.makeText(this, "이름을 입력해 주세요", Toast.LENGTH_SHORT).show();
```

```
        }else if (phone.length() == 0) {
```

```
            Toast.makeText(this, "전화번호를 입력해 주세요", Toast.LENGTH_SHORT).show();
```

```
        }else{
```

```
            if (checkDupleId(id)){
```

```
                sqlite.join(new UserVO(id, pw, name, phone));
```

```
                sqlite.db_close();
```

```
                Intent intent = new Intent(this, LoginActivity.class);
```

```
                startActivity(intent);
```

```
                finish();
```

```
            }else Toast.makeText(this, "이미 사용 중인 아이디 입니다.", Toast.LENGTH_SHORT).show();
```

```
        }
```

```
    }
```

```

else{
    if (checkDupleId(id)){
        sqlite.join(new UserVO(id, pw, name, phone));
        sqlite.db_close();
        Intent intent = new Intent(this, LoginActivity.class);
        startActivity(intent);
        finish();
    }else Toast.makeText(this, "이미 사용 중인 아이디 입니다.", Toast.LENGTH_SHORT).show();
}

```

//아이디 중복 확인

```

private Boolean checkDupleId(String id){
    List<String> list = sqlite.getUserIDs();
    return !list.contains(id);
}

```

// 캡차 인증

```

@Override
public void onClick(View v) {
    SafetyNet.getClient(this).verifyWithRecaptcha(getResources().getString(R.string.site_key))
        .addOnSuccessListener(response -> {
            Toast.makeText(this, "인증 확인", Toast.LENGTH_SHORT).show();
        });
}

```

```

}

```

```

}

```

// 로그인 액티비티

```
public class LoginActivity extends AppCompatActivity {
    private ActivityLoginBinding binding;

    SQLiteHelper helper;
    UserTableController sqlite;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        binding = ActivityLoginBinding.inflate(getLayoutInflater());
        setContentView(binding.getRoot());

        helper = new SQLiteHelper(
            this, // context
            "MemorizeDB", // DBNAME
            null, // factory
            1 // DB 버전4
        );

        sqlite = new UserTableController(helper);

        binding.txtJoin.setOnClickListener(view->{
            Intent intent = new Intent(this, JoinActivity.class);
            startActivity(intent);
        });

        binding.btnSubmit.setOnClickListener(view-> {
            String userId = binding.userID.getText().toString();
            String userPw = binding.password.getText().toString();

            checkLoginProcess(userId, userPw);
        });
    }
}
```

```

private void checkLoginProcess(String userId, String userPw){
    List<String> list = sqlite.getUserIDs();

    if(userId.isEmpty()){
        Toast.makeText(this, "아이디를 입력해 주세요.", Toast.LENGTH_SHORT).show();
    }else if(userPw.isEmpty()){
        Toast.makeText(this, "비밀 번호를 입력해 주세요.", Toast.LENGTH_SHORT).show();
    } else if (!list.contains(userId)){
        Toast.makeText(this, "존재 하지 않는 아이디 입니다.", Toast.LENGTH_SHORT).show();
    } else{
        String pw = sqlite.getUserPassword(userId);

        if (!userPw.equalsIgnoreCase(pw)) {
            Toast.makeText(this, "비밀 번호가 틀립니다.", Toast.LENGTH_SHORT).show();
        } else{
            sqlite.db_close();
            // 로그인 성공 시 Sharedpreference에 로그인 정보 저장
            SharedPreferencesManager.setLoginInfo(this, userId ,pw);
            startActivity(new Intent(this, NavActivity.class));
        }
    }
}

```

//로그인 및 시험 결과를 저장하기 위한 SharedPreferences Class

```
public class SharedPreferencesManager {  
    private static final String PREFERENCES_NAME = "my_preferences";  
    private static final String EXAM_PREFERENCES_NAME = "exam_preferences";  
  
    public static SharedPreferences getPreferences(Context context){  
        return context.getSharedPreferences(PREFERENCES_NAME, Context.MODE_PRIVATE);  
    }  
}
```

// 로그인 정보 저장

```
public static void setLoginInfo(Context context, String userID, String userPassword){  
    SharedPreferences prefs = getPreferences(context);  
    SharedPreferences.Editor editor = prefs.edit();  
    editor.putString("userID", userID);  
    editor.putString("userPassword", userPassword);  
  
    editor.apply();  
}
```

// 로그인 정보 가져옴

```
public static Map<String, String> getLoginInfo(Context context){  
    SharedPreferences prefs = getPreferences(context);  
    Map<String, String> LoginInfo = new HashMap<>();  
    String userID = prefs.getString("userID", ""); // default Value = "" <- nullPointerException 방지  
    String userPassword = prefs.getString("userPassword", "");  
  
    LoginInfo.put("userID", userID);  
    LoginInfo.put("userPassword", userPassword);  
  
    return LoginInfo;  
}
```

```
public static void setGroupName(Context context, String group){  
    SharedPreferences prefs = getPreferences(context);  
    SharedPreferences.Editor editor = prefs.edit();  
    editor.putString("groupName", group);  
    editor.apply();  
}
```

```
public static String getGroupName(Context context){  
    SharedPreferences prefs = getPreferences(context);  
    prefs.getString("groupName", "");  
  
    return prefs.getString("groupName", "");  
}
```

```

public static void setAllWordsPref(Context context, String key, ArrayList<WordVO> values){
    SharedPreferences prefs = context.getSharedPreferences(EXAM_PREFERENCES_NAME,
        Context.MODE_PRIVATE);
    SharedPreferences.Editor editor = prefs.edit();

    // Json 배열
    JSONArray jsonArray = new JSONArray();

    // Gson 객체 생성
    Gson gson = new GsonBuilder().create();

    for (int i = 0; i < values.size(); i++) {
        // WordVO -> Json
        String value = gson.toJson(values.get(i), WordVO.class);
        jsonArray.put(value);
    }

    if (!values.isEmpty()) {
        editor.putString(key, jsonArray.toString());
    } else {
        editor.putString(key, null);
    }

    editor.apply();
}

public static ArrayList getAllWordsPref(Context context, String key){
    SharedPreferences prefs = context.getSharedPreferences(EXAM_PREFERENCES_NAME,
        Context.MODE_PRIVATE);
    String json = prefs.getString(key, null);

    ArrayList<WordVO> list = new ArrayList<WordVO>();
    Gson gson = new GsonBuilder().create();

    if (json != null) {
        try {
            JSONArray arr = new JSONArray(json);
            for (int i = 0; i < arr.length(); i++) {
                WordVO ww = gson.fromJson(arr.get(i).toString(), WordVO.class);
                list.add(ww);
            }
        } catch (JSONException e) {
            e.printStackTrace();
        }
    }

    return list;
}

```



```

public static void setCorrectWordsPref(Context context, String key, ArrayList<WordVO> values){
    SharedPreferences prefs =
        context.getSharedPreferences(EXAM_PREFERENCES_NAME, Context.MODE_PRIVATE);
    SharedPreferences.Editor editor = prefs.edit();

    // Json 배열
    JSONArray jsonArray = new JSONArray();

    // Gson 객체 생성
    Gson gson = new GsonBuilder().create();

    for (int i = 0; i < values.size(); i++) {
        // WordVO -> Json
        String value = gson.toJson(values.get(i), WordVO.class);
        jsonArray.put(value);
    }

    if (!values.isEmpty()) {
        editor.putString(key, jsonArray.toString());
    } else {
        editor.putString(key, null);
    }

    editor.apply();
}

public static ArrayList getCorrectWordsPref(Context context, String key){
    SharedPreferences prefs =
        context.getSharedPreferences(EXAM_PREFERENCES_NAME, Context.MODE_PRIVATE);
    String json = prefs.getString(key, null);

    ArrayList<WordVO> list = new ArrayList<WordVO>();
    Gson gson = new GsonBuilder().create();

    if (json != null) {
        try {
            JSONArray arr = new JSONArray(json);
            for (int i = 0; i < arr.length(); i++) {
                WordVO ww = gson.fromJson(arr.get(i).toString(), WordVO.class);
                list.add(ww);
            }
        } catch (JSONException e) {
            e.printStackTrace();
        }
    }
    return list;
}

```

```

public static void setIncorrectWordsPref(Context context, String key, ArrayList<WordVO> values){
    SharedPreferences prefs =
        context.getSharedPreferences(EXAM_PREFERENCES_NAME, Context.MODE_PRIVATE);
    SharedPreferences.Editor editor = prefs.edit();

    // Json 배열
    JSONArray jsonArray = new JSONArray();

    // Gson 객체 생성
    Gson gson = new GsonBuilder().create();

    for (int i = 0; i < values.size(); i++) {
        // WordVO -> Json
        String value = gson.toJson(values.get(i), WordVO.class);
        jsonArray.put(value);
    }

    if (!values.isEmpty()) {
        editor.putString(key, jsonArray.toString());
    } else {
        editor.putString(key, null);
    }

    editor.apply();
}

public static ArrayList getIncorrectWordsPref(Context context, String key){
    SharedPreferences prefs = context.getSharedPreferences(EXAM_PREFERENCES_NAME,
        Context.MODE_PRIVATE);
    String json = prefs.getString(key, null);

    ArrayList<WordVO> list = new ArrayList<WordVO>();
    Gson gson = new GsonBuilder().create();

    if (json != null) {
        try {
            JSONArray arr = new JSONArray(json);
            for (int i = 0; i < arr.length(); i++) {
                WordVO ww = gson.fromJson(arr.get(i).toString(), WordVO.class);
                list.add(ww);
            }
        } catch (JSONException e) {
            e.printStackTrace();
        }
    }

    return list;
}

```

```
public static void removeExamResult(Context context){
    SharedPreferences prefs = context.getSharedPreferences(EXAM_PREFERENCES_NAME,
Context.MODE_PRIVATE);
    SharedPreferences.Editor editor = prefs.edit();

    editor.clear();
    editor.commit();
}
}
```

// 리사이클러 뷰를 이용해 시험 결과 중 정답을 보여주는 fragment

```
public class AnswerFragment extends Fragment {
    ExamWordAdapter adapter;
    List<WordVO> list = new ArrayList<>();

    @Override
    public View onCreateView(@NonNull LayoutInflater inflater, ViewGroup container,
        Bundle savedInstanceState) {
        FragmentAnswerBinding binding = FragmentAnswerBinding.inflate(inflater, container, false);

        list = SharedPreferencesManager.getCorrectWordsPref(requireContext(), "CORRECT_WORD");

        binding.recyclerView.setLayoutManager(new LinearLayoutManager(requireActivity()));
        adapter = new ExamWordAdapter(requireContext(), list);

        binding.recyclerView.setAdapter(adapter);

        return binding.getRoot();
    }
}
```

// 리사이클러 뷰를 이용해 시험 종료 후 모든 단어를 보여주는 fragment

```
public class AllWordFragment extends Fragment {
    ExamWordAdapter adapter;
    List<WordVO> list = new ArrayList<>();

    @Override
    public View onCreateView(LayoutInflater inflater, ViewGroup container,
        Bundle savedInstanceState) {
        FragmentAllWordBinding binding = FragmentAllWordBinding.inflate(inflater,
            container, false);

        list = SharedPreferencesManager.getAllWordsPref(requireContext(), "ALL_WORD");

        binding.recyclerView.setLayoutManager(new LinearLayoutManager(requireActivity()));
        adapter = new ExamWordAdapter(requireContext(), list);

        binding.recyclerView.setAdapter(adapter);
        return binding.getRoot();
    }
}
```

```

//모든 단어를 보여주는 fragment
public class WordFragment extends Fragment {
    SQLiteHelper helper;
    WordTableController db;
    WordAdapter adapter;

    ArrayList<WordVO> list = new ArrayList<>();

    @Override
    public View onCreateView(LayoutInflater inflater, ViewGroup container,
        Bundle savedInstanceState) {

        FragmentWordBinding binding = FragmentWordBinding.inflate(inflater,container, false);

        // DB 참조 객체 생성
        helper = new SQLiteHelper(
            requireActivity(), // context
            "MemorizeDB", // DBNAME
            null, // factory
            1 // DB 버전);

        db = new WordTableController(helper);
        // 레이아웃 설정
        binding.recyclerView.setLayoutManager(new LinearLayoutManager(requireActivity()));

        // 어댑터 등록
        adapter = new WordAdapter(requireActivity(), list);
        binding.recyclerView.setAdapter(adapter);

        binding.fab.setOnClickListener(view->{
            startActivity(new Intent(requireActivity(), InputWordActivity.class));
        });

        getAllWords();

        adapter.notifyDataSetChanged();
        binding.allGroup.setOnClickListener(view->{
            startActivity(new Intent(requireActivity(), GroupListActivity.class));
        });
        return binding.getRoot();
    }
}

```

```
public void getAllWords(){
    Cursor cursor = db.readAllData();
    while (cursor.moveToNext()){
        WordVO vo = new WordVO(
            cursor.getString(0),
            cursor.getString(1),
            cursor.getString(2)
        );
        list.add(vo);
    }
}

@Override
public void onPause() {
    super.onPause();
    SharedPreferencesManager.setAllWordsPref(requireActivity(), "ALL_WORD", list);
}
}
```

// 단어 추가 액티비티

```
public class InputWordActivity extends AppCompatActivity {
    private ActivityInputWordBinding binding;
    SQLiteHelper helper;
    WordTableController db;
    String group;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        binding = ActivityInputWordBinding.inflate(getLayoutInflater());
        setContentView(binding.getRoot());

        helper = new SQLiteHelper(this, "MemorizeDB", null, 1);
        db = new WordTableController(helper);

        Intent intent = getIntent();
        String tag = intent.getStringExtra("tag");

        binding.imgX.setOnClickListener(view->{
            switch (tag){
                case "FromGroupByWord":
                    Intent exitIntent = new Intent(this, GroupByWordActivity.class);
                    exitIntent.setFlags(Intent.FLAG_ACTIVITY_SINGLE_TOP);
                    startActivity(exitIntent);

                case "FromWord":
                    Intent exitIntent2 = new Intent(this, NavActivity.class);
                    exitIntent2.setFlags(Intent.FLAG_ACTIVITY_SINGLE_TOP);
                    startActivity(exitIntent2);
            }
        });

        ActivityResultLauncher<Intent> resultLauncher = registerForActivityResult(
            new ActivityResultContracts.StartActivityForResult(),
            new ActivityResultCallback<ActivityResult>() {

                @Override
                public void onActivityResult(ActivityResult result) {
                    group = SharedPreferencesManager.getGroupName(InputWordActivity.this);
                    binding.tvGroup.setText(group);
                    SharedPreferencesManager.setGroupName(InputWordActivity.this, "");
                }
            }
        );

        binding.tvGroup.setOnClickListener(view->{
            Intent intent2 = new Intent(this, SelectGroupActivity.class);
            resultLauncher.launch(intent2);
        });
    }
}
```

```

binding.button.setOnClickListener(view ->{
String word = binding.etWord.getText().toString();
String mean = binding.etMean.getText().toString();

if (word.isEmpty()){
    Toast.makeText(this, "단어를 입력해 주세요", Toast.LENGTH_SHORT).show();
} else if (mean.isEmpty()) {
    Toast.makeText(this, "뜻을 입력해 주세요", Toast.LENGTH_SHORT).show();
}else {
    if (checkDupleWord(word)){
        db.insert(new WordVO(word, mean, binding.tvGroup.getText().toString()));
        Toast.makeText(this, "추가 완료", Toast.LENGTH_SHORT).show();
        binding.etWord.requestFocus();
        binding.etWord.setText("");
        binding.etMean.setText("");
        binding.tvGroup.setText("그룹 미지정");
    }else Toast.makeText(this, "이미 존재 하는 단어 입니다!", Toast.LENGTH_SHORT).show();
    });
}

private Boolean checkDupleWord(String word){
    List<String> list = db.getWordList();
    return !list.contains(word);
}
}

```


// 그룹 생성 액티비티

```
public class MakeGroupActivity extends AppCompatActivity {

    private ActivityMakeGroupBinding binding;
    SQLiteHelper helper;
    GroupTableController db;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        binding = ActivityMakeGroupBinding.inflate(getLayoutInflater());
        setContentView(binding.getRoot());

        helper = new SQLiteHelper(this, "MemorizeDB", null, 1);
        db = new GroupTableController(helper);

        binding.imgX.setOnClickListener(view->{
            finish();
        });

        Intent intent = getIntent();
        String tag = intent.getStringExtra("tag");

        binding.button.setOnClickListener(view->{
            String groupName = binding.etWord.getText().toString();

            if (groupName.isEmpty()){
                Toast.makeText(this, "그룹 이름을 입력해 주세요", Toast.LENGTH_SHORT).show();
            } else {
                if (checkDuple(groupName)){
                    db.insert(groupName);
                    switch (tag){
                        case "FromSelectGroup":
                            startActivity(new Intent(this, SelectGroupActivity.class));
                            finish(); break;
                        case "FromGroupList":
                            startActivity(new Intent(this, GroupListActivity.class));
                            finish(); break;
                    }
                } else Toast.makeText(this, "이미 존재 하는 그룹명 입니다.", Toast.LENGTH_SHORT).show();
            }
        });
    }

    public Boolean checkDuple(String s){
        List<String> list = db.getAllGroups();
        return !list.contains(s);
    }
}
```

// SQLite DB 생성 클래스

```
public class SQLiteHelper extends SQLiteOpenHelper {
    public final int DATABASE_VERSION = 1;
    public final String USER_TABLE_NAME = "userinfo";
    public final String USER_ID = "userID";
    public final String USER_PASSWORD = "userPassword";
    public final String USER_NAME = "username";
    public final String USER_PHONE = "userphone";
    public final String WORD_TABLE_NAME = "voca";
    public final String WORD = "word";
    public final String MEAN = "mean";
    public final String GROUP_TABLE_NAME = "groups";
    public final String GROUP_NAME = "groupName";
    public SQLiteHelper(@Nullable Context context, @Nullable String name,
        @Nullable SQLiteDatabase.CursorFactory factory, int version) {
        super(context, name, factory, version);
    }

    @Override
    public void onCreate(SQLiteDatabase db) {
        String create_query = "create table if not exists " + USER_TABLE_NAME + "("
            + USER_ID + "text not null primary key, "
            + USER_PASSWORD + "text not null, "
            + USER_NAME + "text not null, "
            + USER_PHONE + "text not null);";

        String create_query2 = "create table if not exists " + WORD_TABLE_NAME + "("
            + WORD + "text not null primary key, "
            + MEAN + "text not null, "
            + GROUP_NAME + "text not null );";

        String create_query3 = "create table if not exists " + GROUP_TABLE_NAME + "("
            + GROUP_NAME + "text not null primary key );";

        db.execSQL(create_query);
        db.execSQL(create_query2);
        db.execSQL(create_query3);
    }

    @Override
    public void onUpgrade(SQLiteDatabase db, int oldVersion, int newVersion) {
        String drop_query = "drop table " + USER_TABLE_NAME + ";";
        String drop_query2 = "drop table " + USER_TABLE_NAME + ";";
        db.execSQL(drop_query);
        db.execSQL(drop_query2);
        onCreate(db);
    }
}
```

// 그룹 관리 Query Class

```
public class GroupTableController {
    SQLiteHelper helper;
    SQLiteDatabase sqlite;
    List<String> list = new ArrayList<>();

    public GroupTableController(SQLiteHelper _helper) {
        this.helper = _helper;
    }

    public void insert(String groupName) {
        sqlite = helper.getWritableDatabase();
        ContentValues values = new ContentValues();
        values.put(helper.GROUP_NAME, groupName);
        sqlite.insert(helper.GROUP_TABLE_NAME, null, values);
    }

    public List<String> getAllGroups(){
        sqlite = helper.getReadableDatabase();
        String query = "Select * from " + helper.GROUP_TABLE_NAME;
        Cursor c = sqlite.rawQuery(query, null);
        while (c.moveToNext()) list.add(c.getString(0));
        return list;
    }

    public Cursor readAllData(){
        String query = "SELECT * FROM "+ helper.GROUP_TABLE_NAME;
        sqlite = helper.getReadableDatabase();
        Cursor cursor = null;
        if (sqlite !=null) cursor = sqlite.rawQuery(query, null);
        return cursor;
    }

    public void update(String _key, String _value, String _groupName){
        sqlite = helper.getWritableDatabase();
        ContentValues value = new ContentValues();
        value.put(_key, _value);
        sqlite.update(helper.GROUP_TABLE_NAME, value, "groupName=?", new
        String[]{_groupName});
    }

    public void delete(String _groupName){
        sqlite = helper.getWritableDatabase();
        sqlite.delete(helper.GROUP_TABLE_NAME, "groupName=?", new
        String[]{_groupName});
    }

    public void updateAllGroup(String oldName, String newName){
        String query = "UPDATE " + helper.WORD_TABLE_NAME + "SET " +
        helper.GROUP_NAME + "=" + newName + " WHERE "
        + helper.GROUP_NAME + "=" + oldName + "";
        sqlite.execSQL(query);
    }
}
```

}}

// 회원 관리 Query Class

```
public class UserTableController {
    SQLiteHelper helper;
    SQLiteDatabase sqlite;
    List<String> list = new ArrayList<>();

    public UserTableController(SQLiteHelper _helper) {
        this.helper = _helper;
    }

    public void join(UserVO vo) {
        sqlite = helper.getWritableDatabase();
        ContentValues values = new ContentValues();
        values.put(helper.USER_ID, vo.getUserID());
        values.put(helper.USER_PASSWORD, vo.getUserPassword());
        values.put(helper.USER_NAME, vo.getUserName());
        values.put(helper.USER_PHONE, vo.getUserPhone());
        sqlite.insert(helper.USER_TABLE_NAME, null, values);
    }

    public UserVO getUser(String userID){
        sqlite = helper.getReadableDatabase();
        UserVO vo = new UserVO();

        String query = "Select * from " + helper.USER_TABLE_NAME + "where userID='"+
        userID+ "'";
        Cursor c = sqlite.rawQuery(query, null);
        while (c.moveToNext()){
            vo.setUserID(c.getString(0));
            vo.setUserPassword(c.getString(1));
            vo.setUserName(c.getString(2));
            vo.setUserPhone(c.getString(3));
        }
        return vo;
    }

    public List<String> getUserIDs(){
        sqlite = helper.getReadableDatabase();
        String query = "Select userID from " + helper.USER_TABLE_NAME;
        Cursor c = sqlite.rawQuery(query, null);
        while (c.moveToNext()) list.add(c.getString(0));
        return list;
    }

    public String getUserPassword(String userID){
        sqlite = helper.getReadableDatabase();
        String query = "Select userPassword from " + helper.USER_TABLE_NAME + "where
        userID='"+ userID+ "'";
        Cursor c = sqlite.rawQuery(query, null);
        String result = "";
```

```
public void update(String _key, String _value, String _userID){
    sqlite = helper.getWritableDatabase();

    ContentValues value = new ContentValues();
    value.put(_key, _value);
    sqlite.update(helper.USER_TABLE_NAME, value, "userID=?", new String[] {_userID});
}

public void delete(String _userID){
    sqlite = helper.getWritableDatabase();
    sqlite.delete(helper.USER_TABLE_NAME, "userID=?", new String[] {_userID});
}

public void db_close(){
    sqlite.close();
    helper.close();
}
}
```

```

    public class WordTableController {
        SQLiteHelper helper;
        SQLiteDatabase sqlite;
        List<String> list = new ArrayList<>();

        public WordTableController(SQLiteHelper _helper) {
            this.helper = _helper;
        }

        public void insert(WordVO vo) {
            sqlite = helper.getWritableDatabase();
            ContentValues values = new ContentValues();
            values.put(helper.WORD, vo.getWord());
            values.put(helper.MEAN, vo.getMean());
            values.put(helper.GROUP_NAME, vo.getGroup());
            sqlite.insert(helper.WORD_TABLE_NAME, null, values);
        }

        public Cursor readAllData(){
            String query = "SELECT * FROM " + helper.WORD_TABLE_NAME;
            sqlite = helper.getReadableDatabase();
            Cursor cursor = null;
            if (sqlite !=null){
                cursor = sqlite.rawQuery(query, null);
            }
            return cursor;
        }

        public List<String> getWordList(){
            sqlite = helper.getReadableDatabase();
            String query = "Select word from " + helper.WORD_TABLE_NAME;
            Cursor c = sqlite.rawQuery(query, null);

            while (c.moveToNext()) list.add(c.getString(0));
            return list;
        }

        public void update(WordVO vo, String _word){
            sqlite = helper.getWritableDatabase();

            String query = "UPDATE " + helper.WORD_TABLE_NAME + "SET " +
                helper.WORD + "='" + vo.getWord() + "', " +
                helper.MEAN + "='" + vo.getMean() + "', " +
                helper.GROUP_NAME + "='" + vo.getGroup() + "' WHERE " +
                helper.WORD + "='" + _word + "'";

            sqlite.execSQL(query);
        }
    }

```

```

public void delete(String _word){
    sqlite = helper.getWritableDatabase();
    sqlite.delete(helper.WORD_TABLE_NAME, "word=?", new String[] {_word});
}

public List<WordVO> readGroupData(String group){
    List<WordVO> list = new ArrayList<>();
    sqlite = helper.getReadableDatabase();
    String query = "SELECT * FROM " + helper.WORD_TABLE_NAME + "where " +
        helper.GROUP_NAME + "=" + group + "" ;

    Cursor cursor = null;
    if (sqlite !=null) cursor = sqlite.rawQuery(query, null);

    while (cursor.moveToNext()){
        WordVO vo = new WordVO(
            cursor.getString(0),
            cursor.getString(1),
            cursor.getString(2)
        );

        list.add(vo);
    }

    return list;
}

public void db_close(){
    sqlite.close();
    helper.close();
}
}

```

// 영단어 시험Activity

```
public class ExamActivity extends AppCompatActivity implements TextView.OnEditorActionListener {
    ActivityExamBinding binding;
    Dialog dialog;
    SQLiteHelper helper;
    WordTableController db;
    List<WordVO> list = new ArrayList<>();
    ArrayList<WordVO> correctWordList = new ArrayList<>();
    ArrayList<WordVO> incorrectWordList = new ArrayList<>();
    int i = 0;

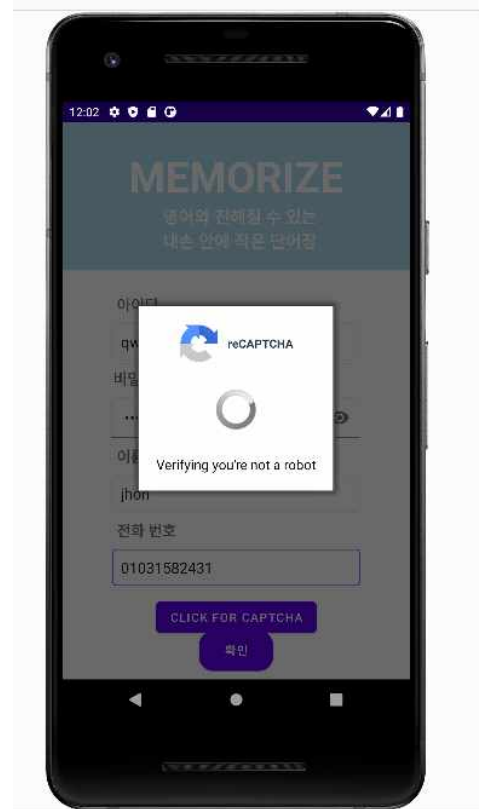
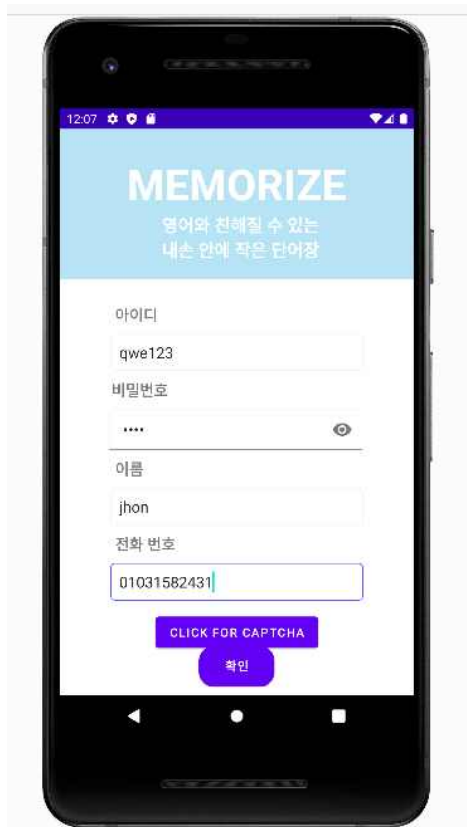
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        binding = ActivityExamBinding.inflate(getLayoutInflater());
        setContentView(binding.getRoot());

        getWindow().setSoftInputMode(WindowManager.LayoutParams.SOFT_INPUT_ADJUST_RESIZE);
        dialog = new Dialog(this);
        dialog.requestWindowFeature(Window.FEATURE_NO_TITLE);
        dialog.setContentView(R.layout.custom_alert);
        dialog.getWindow().setBackgroundDrawable(new
            ColorDrawable(Color.TRANSPARENT));

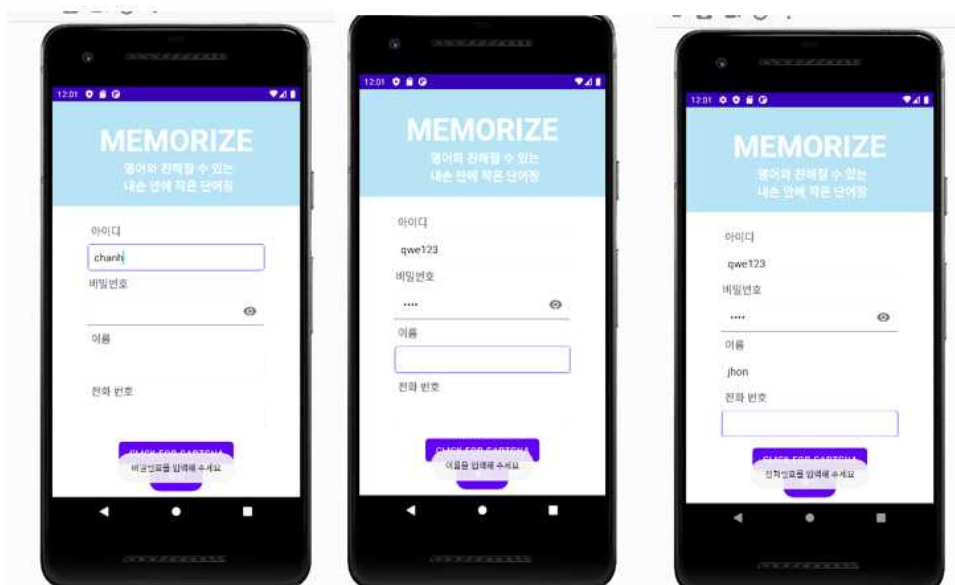
        binding.imgX.setOnClickListener(view->{
            showDialog();
        });
        // DB 참조 객체 생성
        helper = new SQLiteHelper( this,"MemorizeDB", null, 1 버전);
        db = new WordTableController(helper);
        getAllWords();
        Collections.shuffle(list);
        binding.etWord.setOnEditorActionListener(this);
        binding.tvWord.setText(list.get(0).getMean());
    }
    public void getAllWords(){
        Cursor cursor = db.readAllData();
        while (cursor.moveToNext()){
            WordVO vo = new WordVO(
                cursor.getString(0),
                cursor.getString(1),
                cursor.getString(2)
            );
            list.add(vo);
        }
    }
}
```


- 사용자 메뉴얼 (부록)

1.1 회원가입 설계 프로세스



1.2 회원가입 유효성 검사 설계 프로세스



1. 비밀번호 미 입력시

2. 이름 미 입력시

3. 전화번호 미 입력시

1.3. 회원 정보 DB INSERT 프로세스

App Inspection

Pixel 2 API 30 > com.example.memorizes

Database Inspector Network Inspector Background Task Inspector

Databases

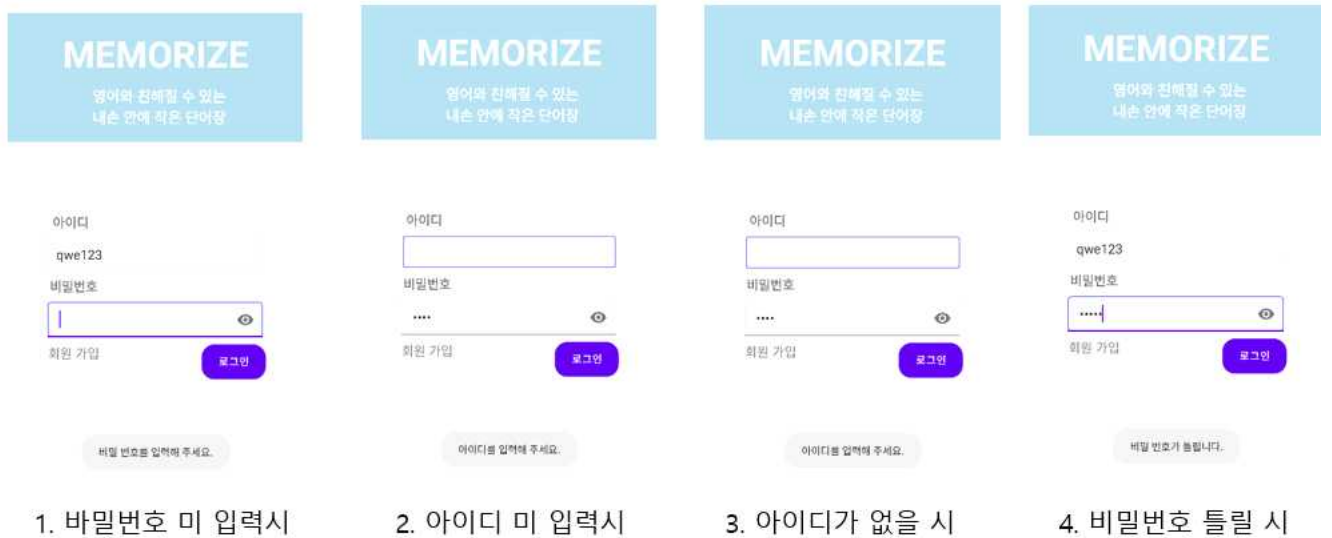
- MemorizeDB
 - groups
 - userinfo
 - voca

userinfo

Live updates

	userID	userPassword	username	userphone
1	qwe123	1234	Jhon	01031582431

2.1 로그인 프로세스



3.1 단어 추가 및 그룹 추가 프로세스

단어장

공부할 단어

모든 카테고리 >

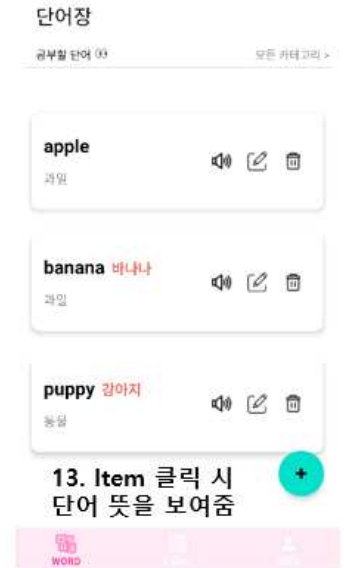
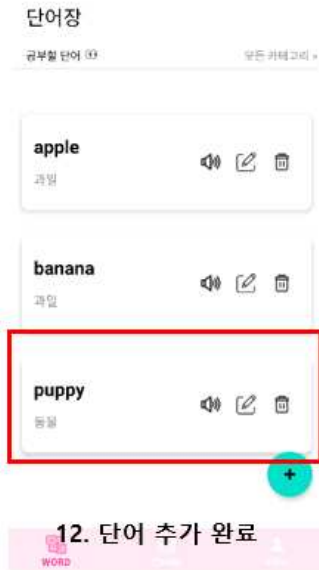


1. 단어 추가 버튼 클릭



4. 그룹 추가 버튼 클릭





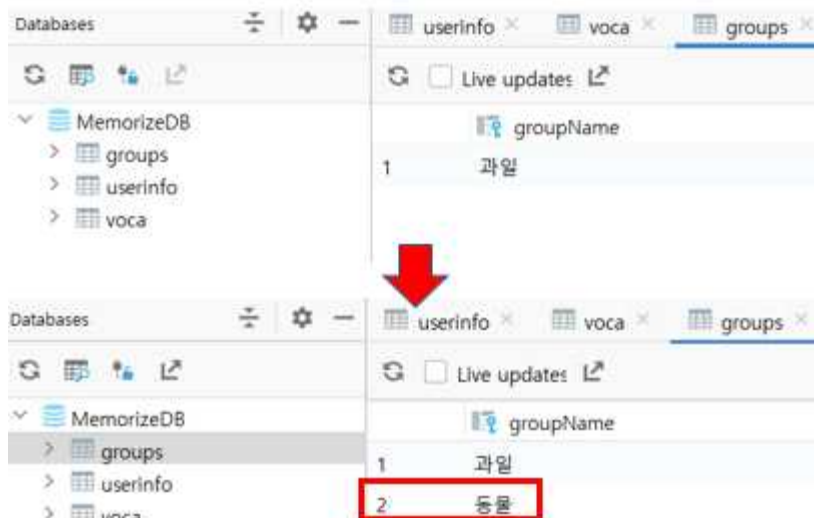
Databases

userinfo × voca × groups ×

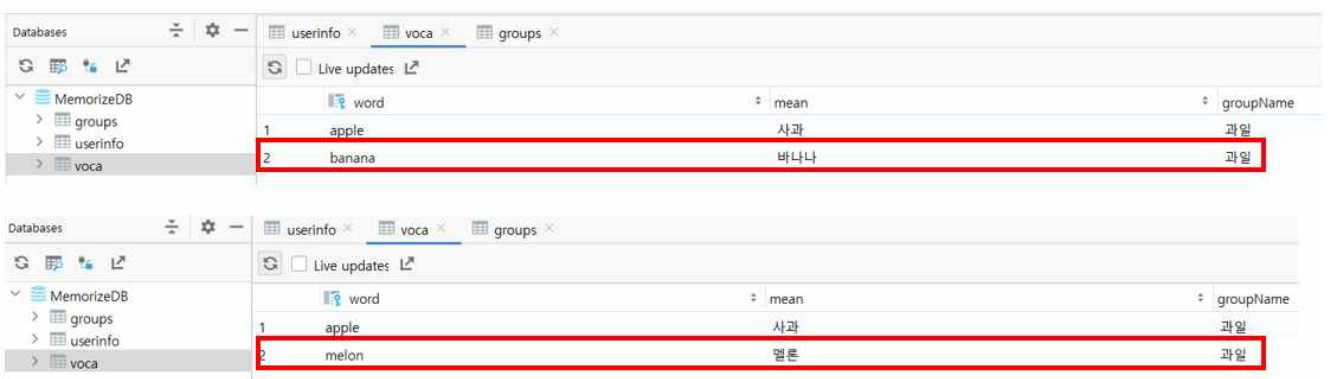
Live updates

	word	mean	groupName
1	apple	사과	과일
2	banana	바나나	과일
3	puppy	강아지	동물

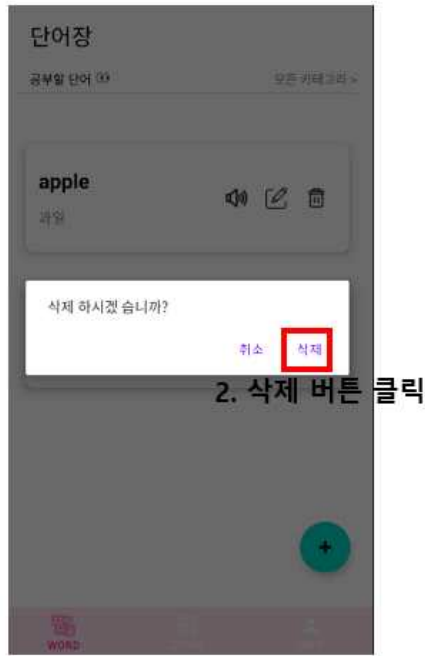
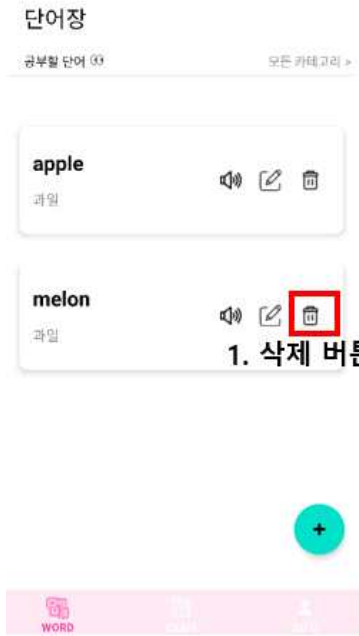
4.1 단어 수정 및 삭제 프로세스



4.1.1 단어 수정



4.1.2 단어 삭제



Databases

userinfo × voca × groups ×

Live updates

	word	mean	groupName
1	apple	사과	과일
2	melon	멜론	과일

↓

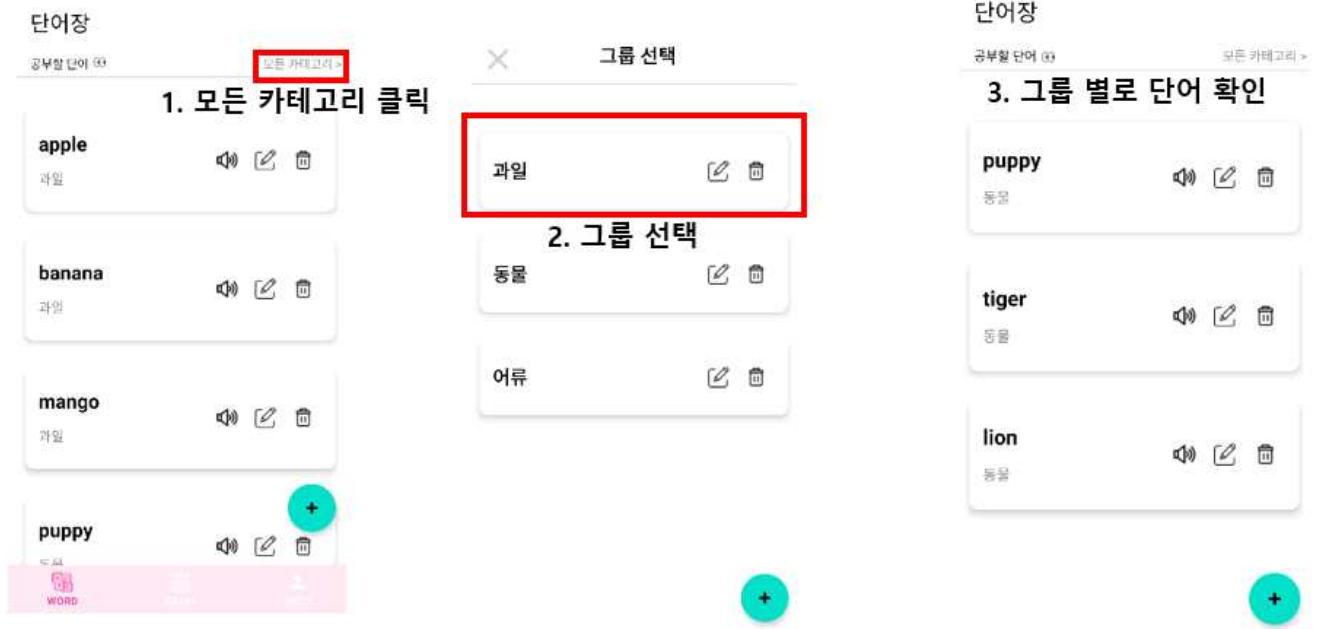
Databases

userinfo × voca × groups ×

Live updates

	word	mean	groupName
1	apple	사과	과일

5.1 카테고리 별로 단어 확인



6.1 단어 시험 프로세스

