모바일 복지카드 서비스

팀　　명 :	Blockchallenge
지도 교수 :	이병천 교수님
팀　　장 :	김현서
팀　　원 :	이성욱
	조은경
	최경라

2021. 11.

중부대학교 정보보호학과

목　　차

1. 서론

1.1 연구 배경

플라스틱 카드로 된 주민등록증, 운전면허증 등의 신분증에는 사진을 비롯한 주민등록번호, 주소지 등 개인정보가 모두 나타나 정보 유출 위험이 있다. 또한 기존의 신분증은 플라스틱이기 때문에 환경문제가 야기될 수 있다. 이러한 피해를 최소화하기 위해서 모바일 신분증을 기획하게 되었다.
장애인등록증은 몸이 불편한 장애인들을 위한 혜택을 누리게 할 수 있는 신분증이다. 장애인 복지 혜택을 받기 위해선 반드시 있어야 하는 것으로 잃어버리면 재발급이 가능하지만 몸이 불편한 분들에겐 이것 또한 쉽지 않은 일이다. 그렇기 때문에 언제 어디서나 간편하게 이용하고, 쉽게 잃어버릴 수 없는 모바일 복지 카드를 고안해보았다.

1.2 연구 필요성

코로나19 확산으로 비대면 서비스 수요가 증가함에 따라 디지털상에서 물리적 신분증을 대체할 새로운 인증 수단의 필요성이 점차 대두했다. 기존의 온라인 금융 서비스의 경우 실명 확인이 필요하며, 신원 확인을 위해 별도의 인증을 거처야 하는 등의 한계점을 가졌다. 기존 신분증은 중요 개인정보를 포함하고 있어 확인 과정에서 민감 정보가 노출되고, 분실 시 정보 유출 및 도용 가능성이 꾸준히 제기되기도 했다. 아울러, 각국에서 개인정보 공개 및 자기 주권 보장 필요성이 대두해 국가, 신뢰 기관 등의 중개자 없이 스스로가 자신을 인증하고 데이터를 관리할 수 있도록 하는 요구가 늘어났다. 이에 주민등록증, 운전면허증, 여권 등의 국가 신분증을 디지털화한 상태로 모바일 기 기에 저장한 모바일 신분증을 고안해보았다.

1.3 연구 목적 및 주제 선정

이번 연구는 위변조 및 도용 우려가 있는 기존 카드의 문제점과 안전성과 활용 편의성의 향상을 위해 모바일 복지 카드 애플리케이션을 구현하고자 했다. 복지 카드를 인증할 때 장애 진단서와 같은 서류도 같이 인증받아 이름, 주민등록번호, 등록번호가 같으면 인증 확인을 해주어 간편하고 휴대가 편리한 모바일 복지 카드를 만들기 위해 주제로 선정하였다.

2. 관련 연구

2.1 Android Studio

안드로이드 스튜디오는 안드로이드 및 안드로이드 전용 어플 제작을 위한 공식 통합 개발 환경(IDE)이다. 언어는 자바와 코틀린을 지원하며 과거 이클립스 기반의 안드로이드 개발 Tool인 Android Development Tool의 주요 빌드 시스템은 아파치 앤트였으나 공식 안드로이드 스튜디오는 gradle빌드 시스템을 사용하고 있다.

2.2 JAVA

1991년 Sun Microsystem사의 James Gosling Patrick Naughton, Chris Warth, Ed Frank Mike Sheridna에 의해서 고안된 언어로 플랫폼에 독립적인 프로그램을 작성할 수 있고 완벽한 객체 지향적 언어이다. 플랫폼 중 하나인 Java EE(Java Platform – Enterprise Edition)는 Java SE를 기반으로 대규모 기업용 서버를 구축하고, 실행할 수 있는 환경을 제공하고Web Application Server(GlassFish)와 Servlet, JSP, JDBC, DataSource, JPA, JTA, JNDI, RMI, EJB, JMS 등 다수의 API를 제공한다.

2.3 XML

마크업 언어를 정의하기 위한 확장성 마크업 언어이다. 인터넷 웹페이지를 만드는 HTML을 획기적으로 개선하여 만든 언어이다. 홈페이지 구축 기능, 검색 기능 등이 향상되었고, 클라이언트 시스템의 복잡한 데이터 처리를 쉽게 하며 웹페이지의 추가와 작성이 편리해졌다. 또한 HTML의 경우 웹페이지에서 데이터베이스처럼 구조화된 데이터를 지원할 수 없지만 XML은 사용자가 구조화된 데이터베이스를 뜻대로 조작할 수 있다. 구조적으로 XML은 SGML문서 형식을 따르고 있다. XMK은 SGML의 부분집합이라고도 할 수 있기때문에 응용판 또는 축양된 형식의 SGML이라고 볼 수 있다.

2.4 HTML

HTML은 WWW 문서를 작성하는 Markup 언어로써 여러 태그들로 구성되어 있으며 각 태그들을 사용하여 원하는 형태의 문서를 만들 수 있다. 또한 인터넷 문서는 HyperText의 원리를 이용하여 여러 문서를 링크시킬 수 있고 다양한 정보를 손쉽게 검색하여 볼 수 있게 만들어져 있다.

2.5 Firebase

파이어베이스에는 2011년 파이어베이스사가 개발하고 2014년 구글에 인수된 모바일 및 웹 어플리케이션 개발 플랫폼으로 구글 애널리스틱과 구글 패브릭에서 제공하는 기능들을 포함한 다양한 기능들을 제공한다. 한마디로 운영체제에 상관없이 앱 혹은 웹을 만들 수 있도록 해주는 개발 Tool이라고 할 수 있다. 구글 드라이브와 애널리스틱을 적용함으로써 어떠한 기기에서나 개발할 수 있고 사용자들의 방문 수, 이용횟수, 문제 발생 빈도 등을 알려줘서 개발자들이 쉽게 활용할 수 있도록 지원한다.

2.6 Block Chain

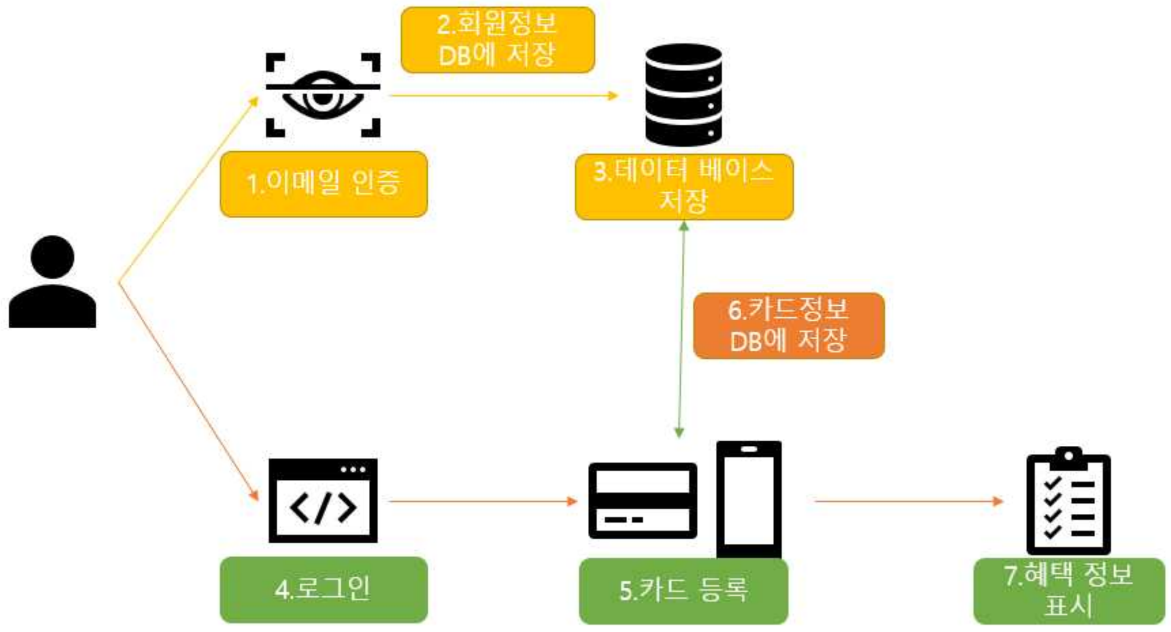
블록체인(Blockchain)이란 데이터를 거래할 때 중앙집중형 서버에 기록을 보관하는 기존 방식과 달리 거래 참가자 모두에게 내용을 공개하는 분산형 디지털 장부를 말한다. 블록체인에 참여한 모든 구성원이 네트워크를 통해 서로 데이터를 검증하고 저장함으로써 특정인의 임의적인 조작이 어렵도록 설계된 저장 플랫폼이라고 할 수 있다. 이러한 상호 분산원장을 통하여 기존 중앙 집중형 네트워크 기반의 인프라를 뛰어넘은 높은 보안성,

확장성, 투명성 등을 보장한다. 이를 위해 해시 함수, 머클 트리, 비대칭 암호, 디지털 서명, 합의 알고리즘 등 다양한 기술이 사용된다.

3. 본론

3.1 시스템 구성

복지 카드를 스마트폰의 Android(OS) 환경에서 프로그램이 사용될 수 있도록 제작하였다.

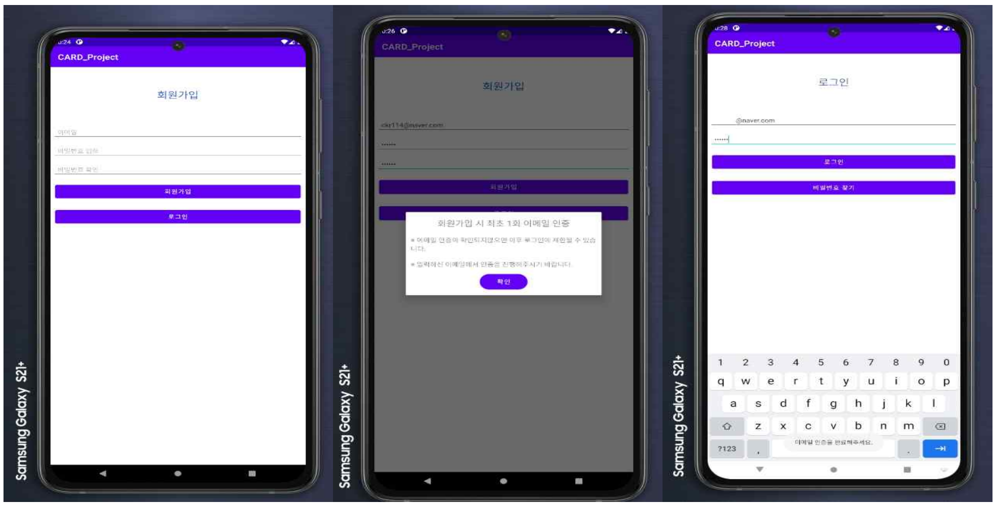


[그림 1. 프로그램 제작 구상도]

3.2 프로그램 구성

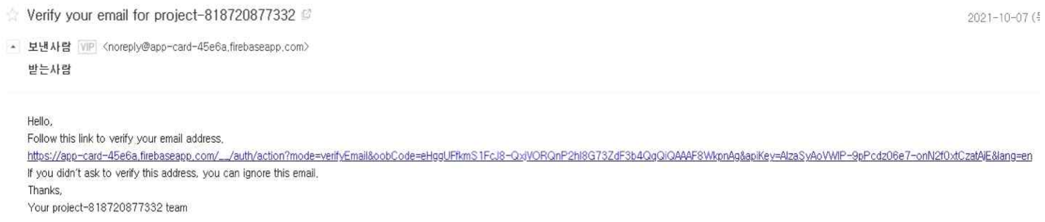
3.2.1 안드로이드 애플리케이션

Java를 기반으로 Android Studio 애플리케이션을 이용하여 모바일 복지 카드를 제작하였다.



[그림 2. 안드로이드 애플리케이션 회원가입, 로그인 화면]

먼저 프로그램을 실행하면 회원가입, 이메일 인증, 로그인 과정을 통하여 복지 카드를 등록할 수 있는 페이지로 넘어간다.



Verify your email for project-818720877332 2021-10-07 (

보낸사람 VIP <noreply@app-card-45e6a.firebaseapp.com>

받는사람

Hello,

Follow this link to verify your email address.

https://app-card-45e6a.firebaseapp.com/__/auth/action?mode=verifyEmail&oobCode=eHggUFfkmS1FcJ8-QxIVORQnP2hI8G73ZdF3b4QqQiQAAAF8WkpnAg&apiKey=AIzaSyAoVWIP-9pPcdz06e7-onN2f0xtCzatAjE&lang=en

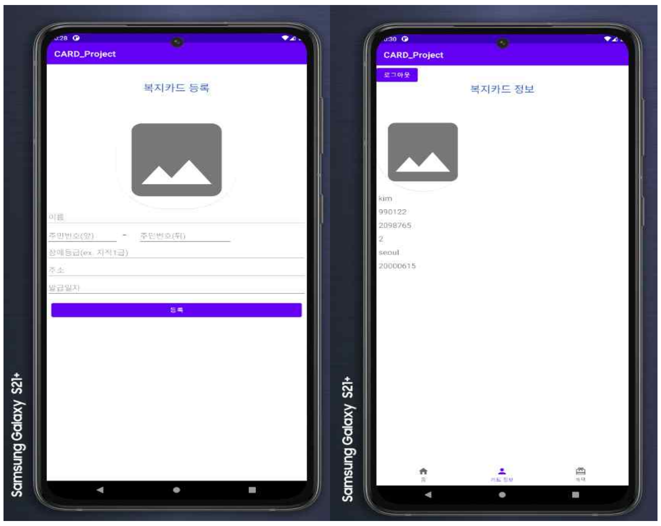
If you didn't ask to verify this address, you can ignore this email.

Thanks,

Your project-818720877332 team

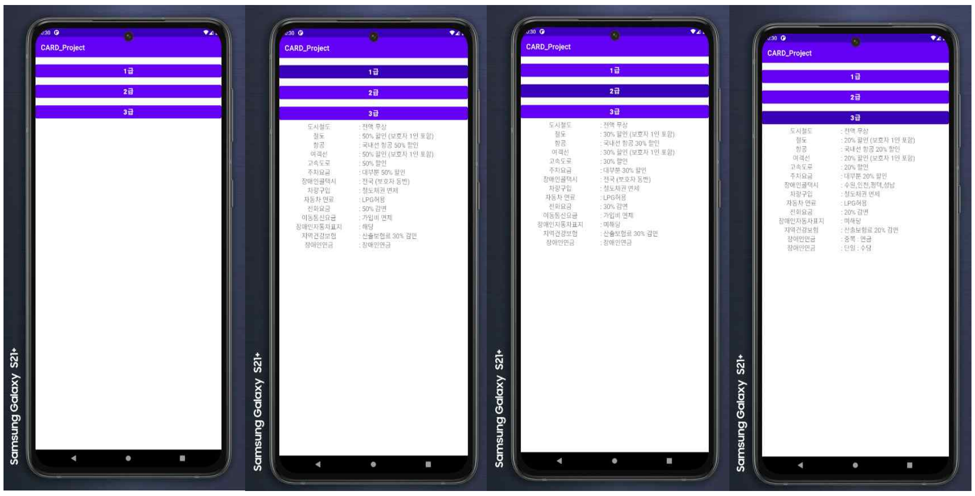
[그림 3. 이메일 인증]

이메일을 인증하기 위해 이메일 창으로 넘어가서 링크를 눌러준다.



[그림 4. 복지 카드 등록]

인증이 완료되면 복지 카드를 정보를 입력하여 복지 카드 정보를 볼 수 있다.



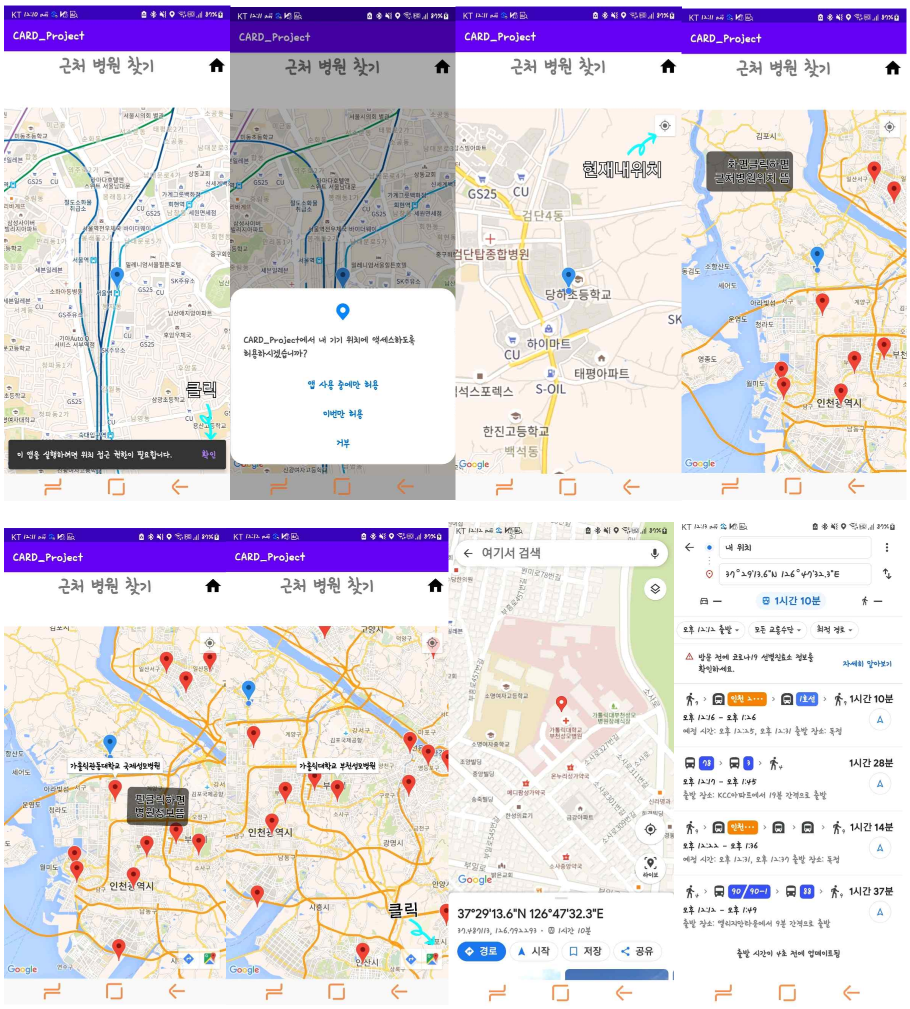
도시철도	: 전액 무상
철도	: 50% 할인 (보호자 1인 포함)
항공	: 국내선 항공 50% 할인
여객선	: 50% 할인 (보호자 1인 포함)
고속도로	: 50% 할인
주차요금	: 대부분 50% 할인
장애인콜택시	: 전국 (보호자 동반)
차량구입	: 철도채권 면제
자동차 연료	: LPG허용
전화요금	: 50% 감면
이동통신요금	: 가입비 면제
장애인자동차표지	: 해당
지역건강보험	: 산출보험료 30% 감면
장애인연금	: 장애인연금

도시철도	: 전액 무상
철도	: 30% 할인 (보호자 1인 포함)
항공	: 국내선 항공 30% 할인
여객선	: 30% 할인 (보호자 1인 포함)
고속도로	: 30% 할인
주차요금	: 대부분 30% 할인
장애인콜택시	: 전국 (보호자 동반)
차량구입	: 철도채권 면제
자동차 연료	: LPG허용
전화요금	: 30% 감면
이동통신요금	: 가입비 면제
장애인자동차표지	: 미해당
지역건강보험	: 산출보험료 30% 감면
장애인연금	: 장애인연금

도시철도	: 전액 무상
철도	: 20% 할인 (보호자 1인 포함)
항공	: 국내선 항공 20% 할인
여객선	: 20% 할인 (보호자 1인 포함)
고속도로	: 20% 할인
주차요금	: 대부분 20% 할인
장애인콜택시	: 수원,인천,평택,성남
차량구입	: 철도채권 면제
자동차 연료	: LPG허용
전화요금	: 20% 감면
장애인자동차표지	: 미해당
지역건강보험	: 산출보험료 20% 감면
장애인연금	: 중복 : 연금
장애인연금	: 단일 : 수당

[그림 5. 혜택 화면]

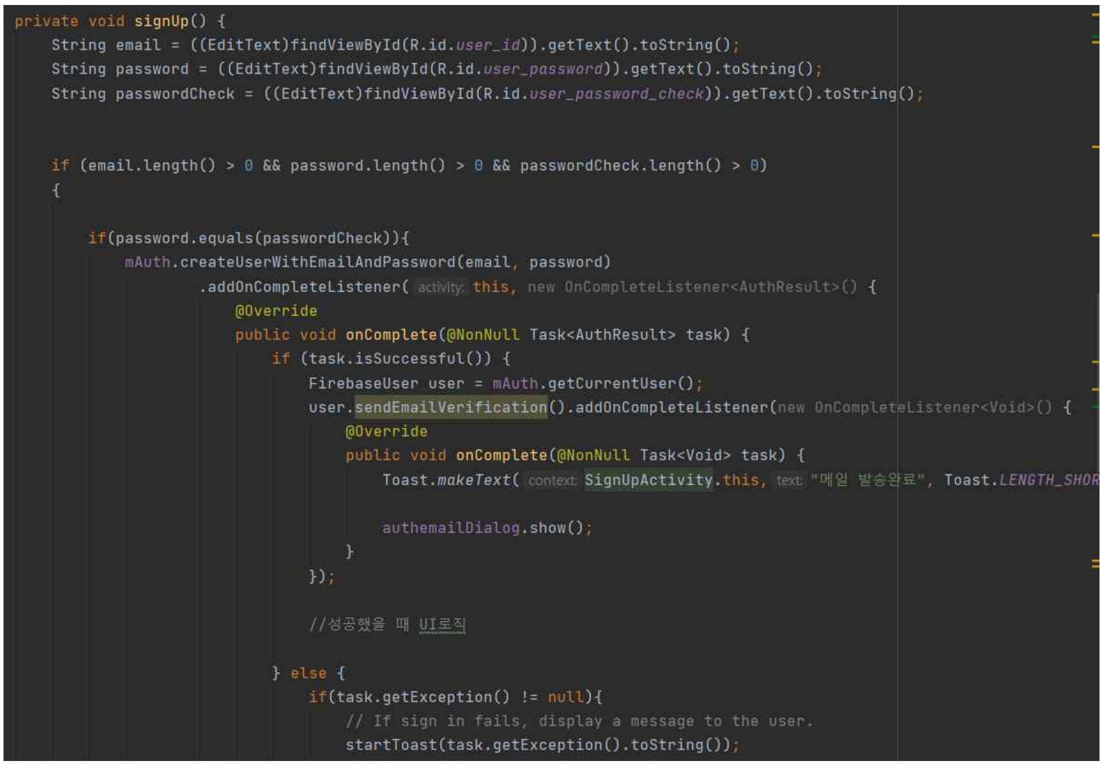
하단 메뉴의 혜택 버튼을 누르면 등급별 혜택을 볼 수 있다.


CARD_Project
근처 병원 찾기
클릭
이 앱을 실행하려면 위치 접근 권한이 필요합니다.
확인
CARD_Project에서 내 기기 위치에 액세스하도록 허용하시겠습니까?
앱 사용 중에만 허용
이번만 허용
거부
현재내위치
검단탑종합병원
당하초등학교
화면클릭하면 근처병원위치 뜸
인천역시
가톨릭관동대학교 국제성모병원
핀클릭하면 병원정보뜸
가톨릭대학교 부천성모병원
여기서 검색
37°29'13.6"N 126°47'32.3"E
경로
시작
저장
공유
Google

내 위치
37°29'13.6"N 126°47'32.3"E
1시간 10분
오후 12:12 출발 · 모든 교통수단 · 최적 경로
방문 전에 코로나19 선별진료소 정보를 확인하세요. 자세히 알아보기
인천 2··· › 1호선 · 1시간 10분
오후 12:16 – 오후 1:26
예정 시간: 오후 12:25, 오후 12:31 출발 장소: 독정
78 › 3 · 1시간 28분
오후 12:17 – 오후 1:45
출발 장소: KCC아파트에서 19분 간격으로 출발
인천··· · 1시간 14분
오후 12:22 – 오후 1:36
예정 시간: 오후 12:31, 오후 12:37 출발 장소: 독정
90/90-1 › 88 · 1시간 37분
오후 12:12 – 오후 1:49
출발 장소: 엘리지안타운에서 9분 간격으로 출발
출발 시간이 4초 전에 업데이트됨

[그림 6. 혜택 활용 가능 장소 지도에 표시]

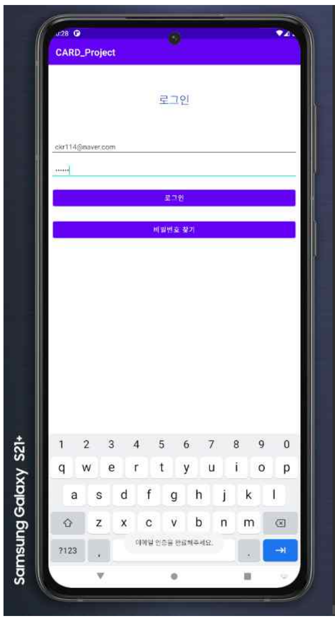
혜택을 받을 수 있는 병원이 지도에 뜨기 때문에 확인하기 편하며 가는데 걸리는 시간을 확인 할 수 있다.



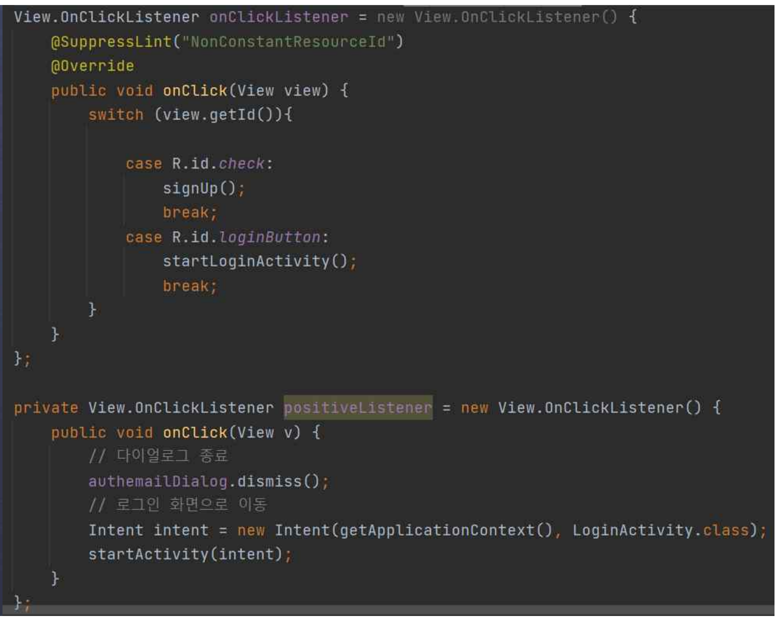
```
private void signUp() {
    String email = ((EditText)findViewById(R.id.user_id)).getText().toString();
    String password = ((EditText)findViewById(R.id.user_password)).getText().toString();
    String passwordCheck = ((EditText)findViewById(R.id.user_password_check)).getText().toString();


    if (email.length() > 0 && password.length() > 0 && passwordCheck.length() > 0)
    {

        if(password.equals(passwordCheck)){
            mAuth.createUserWithEmailAndPassword(email, password)
                    .addOnCompleteListener( activity: this, new OnCompleteListener<AuthResult>() {
                        @Override
                        public void onComplete(@NonNull Task<AuthResult> task) {
                            if (task.isSuccessful()) {
                                FirebaseUser user = mAuth.getCurrentUser();
                                user.sendEmailVerification().addOnCompleteListener(new OnCompleteListener<Void>() {
                                    @Override
                                    public void onComplete(@NonNull Task<Void> task) {
                                        Toast.makeText( context: SignUpActivity.this, text: "메일 발송완료", Toast.LENGTH_SHOR

                                        authemailDialog.show();
                                    }
                                });

                                //성공했을 때 UI로직

                            } else {
                                if(task.getException() != null){
                                    // If sign in fails, display a message to the user.
                                    startToast(task.getException().toString());
```

[그림 7. 이메일 인증]

그림 6은 이메일 인증 코드이다. 회원가입에 성공하면 이메일 인증 메일을 발송하고 다이얼로그 종료로 넘어간다.



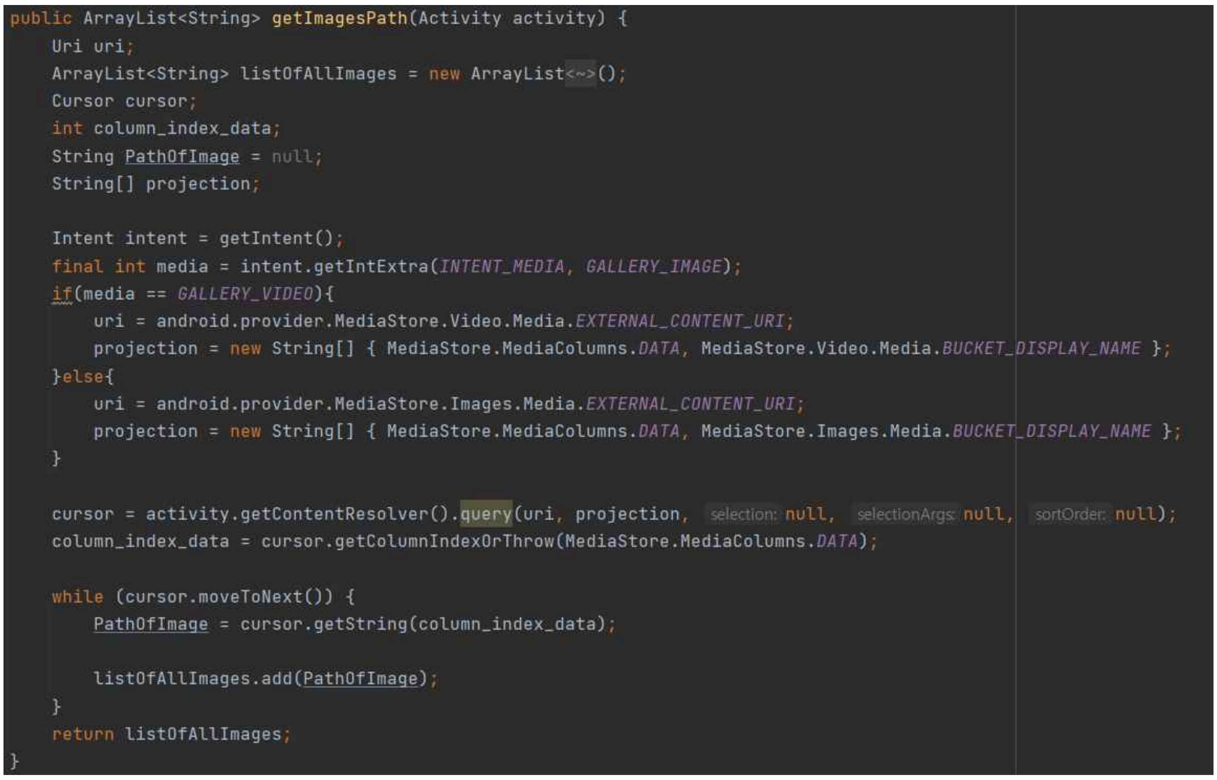
이메일 인증을 완료해주세요.



```
View.OnClickListener onClickListener = new View.OnClickListener() {
    @SuppressLint("NonConstantResourceId")
    @Override
    public void onClick(View view) {
        switch (view.getId()){

            case R.id.check:
                signUp();
                break;
            case R.id.loginButton:
                startLoginActivity();
                break;
        }
    }
};

private View.OnClickListener positiveListener = new View.OnClickListener() {
    public void onClick(View v) {
        // 다이얼로그 종료
        authemailDialog.dismiss();
        // 로그인 화면으로 이동
        Intent intent = new Intent(getApplicationContext(), LoginActivity.class);
        startActivity(intent);
    }
};
```

[그림 8. 인증 확인 후 로그인]

이메일 인증을 전송하면 다이얼로그를 종료해 로그인 화면으로 이동하게 한다. 이메일 인증이 완료되지 않으면 로그인을 할 수 없고 인증이 완료되면 로그인을 해 복지 카드를 등록할 수 있다.



```java
public ArrayList<String> getImagesPath(Activity activity) {
    Uri uri;
    ArrayList<String> listOfAllImages = new ArrayList<~>();
    Cursor cursor;
    int column_index_data;
    String PathOfImage = null;
    String[] projection;

    Intent intent = getIntent();
    final int media = intent.getIntExtra(INTENT_MEDIA, GALLERY_IMAGE);
    if(media == GALLERY_VIDEO){
        uri = android.provider.MediaStore.Video.Media.EXTERNAL_CONTENT_URI;
        projection = new String[] { MediaStore.MediaColumns.DATA, MediaStore.Video.Media.BUCKET_DISPLAY_NAME };
    }else{
        uri = android.provider.MediaStore.Images.Media.EXTERNAL_CONTENT_URI;
        projection = new String[] { MediaStore.MediaColumns.DATA, MediaStore.Images.Media.BUCKET_DISPLAY_NAME };
    }

    cursor = activity.getContentResolver().query(uri, projection, selection: null, selectionArgs: null, sortOrder: null);
    column_index_data = cursor.getColumnIndexOrThrow(MediaStore.MediaColumns.DATA);

    while (cursor.moveToNext()) {
        PathOfImage = cursor.getString(column_index_data);

        listOfAllImages.add(PathOfImage);
    }
    return listOfAllImages;
}
```

[그림 9. 사용자 사진 등록]

복지 카드를 등록할 때 사진첩에서 사용자의 사진을 가져와 등록한다.

4. 결론

4.1 결론

모바일 복지 카드를 구현함으로써 분실 위험이 있는 플라스틱 카드 대신 항상 휴대하고 있는 핸드폰 하나로 편리하고 안전하게 자신을 증명하고 사용 가능한 혜택을 받을 수 있다. 또한 지갑에서 카드를 찾아 꺼내는 등의 번거로움 없이 편리한 사용을 할 수 있다. 이로 인해 사회적 비용을 줄이고 장애인들이 사회에 조금 더 편리하게 다가갈 수 있다.

4.2 기대효과

현재 만들어진 애플리케이션은 카드를 수동으로 등록하고 사용하는 프로그램이다. 하지만 향후 카드를 등록할 때 OCR을 이용해 카드 스캔을 하여 등록하고 QR코드를 이용해 복지 카드 혜택을 받을 수 있는 기관에서 QR코드를 스캔하면 사용자의 복지 카드 정보를 띄워주는 등 더욱더 편리하게 사용할 수 있을 것이다. 또한 회원가입을 할 때 장애인 진단서와 같은 서류로 장애인임을 확인하고 이를 블록체인을 이용하면 보다 안전하게 이

용할 수 있을 것이다.

5. 별첨

5.1 소스코드

(Android) MainActivity.java

```
package com.example.card_project;

import androidx.annotation.NonNull;
import androidx.appcompat.app.AppCompatActivity;

import android.content.Intent;
import android.content.pm.ActivityInfo;
import android.os.Bundle;
import android.util.Log;
import android.view.MenuItem;
import android.view.View;

import com.google.android.material.bottomnavigation.BottomNavigationView;
import com.google.firebase.auth.FirebaseAuth;
import com.google.firebase.auth.FirebaseUser;
import com.google.firebase.firestore.DocumentReference;
import com.google.firebase.firestore.DocumentSnapshot;
import com.google.firebase.firestore.FirebaseFirestore;

public class MainActivity extends AppCompatActivity {
    private static final String TAG = "MainActivity";


    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        FirebaseUser user = FirebaseAuth.getInstance().getCurrentUser();

        if (user == null) {
            myStartActivity(SignUpActivity.class);

        } else {
            FirebaseFirestore db = FirebaseFirestore.getInstance();
            DocumentReference docRef = db.collection("users").document(user.getUid());
            docRef.get().addOnCompleteListener((task) -> {
                if (task.isSuccessful()) {
                    DocumentSnapshot document = task.getResult();
                    if (document != null) {
                        if (document.exists()) {
                            Log.e(TAG, "DocumentSnapshot data: " + document.getData());
                        } else {
                            Log.d(TAG, "No such document");
                            myStartActivity(MemberInitActivity.class);


                        }
                    }
                } else {
                    Log.d(TAG, "get failed with", task.getException());

                }
            });
            BottomNavigationView bottomNavigationView =
findViewById(R.id.bottomNavigationView);
            bottomNavigationView.setOnNavigationItemSelectedListener(new
BottomNavigationView.OnNavigationItemSelectedListener() {
                @Override
                public boolean onNavigationItemSelected(@NonNull MenuItem item) {
                    switch (item.getItemId()) {

                        case R.id.myInfo:
                            UserInfoFragment userInfoFragment = new UserInfoFragment();
                            getSupportFragmentManager().beginTransaction()
                                    .replace(R.id.container, userInfoFragment)
                                    .commit();
                            return true;

                        case R.id.userList:
                            myStartActivity(MainAct.class);
                            break;
                    }
                    return false;
                }
            });
        }



        findViewById(R.id.logoutButton).setOnClickListener(onClickListener);

    }
    View.OnClickListener onClickListener = (v) -> {

        switch (v.getId()) {
            case R.id.logoutButton:
                FirebaseAuth.getInstance().signOut();
                myStartActivity(SignUpActivity.class);
                break;



        }
    };





    private void myStartActivity(Class c) {
        Intent intent = new Intent(this, c);
        startActivity(intent);
    }
}
```

(Android) AutoFitTextureView.java

```
/*
 * Copyright 2014 The Android Open Source Project
 *
 * Licensed under the Apache License, Version 2.0 (the "License");
 * you may not use this file except in compliance with the License.
 * You may obtain a copy of the License at
 *
 *       http://www.apache.org/licenses/LICENSE-2.0
 *
 * Unless required by applicable law or agreed to in writing, software
 * distributed under the License is distributed on an "AS IS" BASIS,
 * WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
 * See the License for the specific language governing permissions and
 * limitations under the License.
 */

package com.example.card_project;

import android.content.Context;
import android.util.AttributeSet;
import android.view.TextureView;

/**
 * A {@link TextureView} that can be adjusted to a specified aspect ratio.
 */
public class AutoFitTextureView extends TextureView {

    private int mRatioWidth = 0;
    private int mRatioHeight = 0;

    public AutoFitTextureView(Context context) {
        this(context, null);
    }

    public AutoFitTextureView(Context context, AttributeSet attrs) {
        this(context, attrs, 0);
    }

    public AutoFitTextureView(Context context, AttributeSet attrs, int defStyle) {
        super(context, attrs, defStyle);
    }

    /**
     * Sets the aspect ratio for this view. The size of the view will be measured
based on the ratio
     * calculated from the parameters. Note that the actual sizes of parameters don't
matter, that
     * is, calling setAspectRatio(2, 3) and setAspectRatio(4, 6) make the same result.
     *
     * @param width  Relative horizontal size
     * @param height Relative vertical size
     */
    public void setAspectRatio(int width, int height) {
        if (width < 0 || height < 0) {
            throw new IllegalArgumentException("Size cannot be negative.");
        }
        mRatioWidth = width;
        mRatioHeight = height;
        requestLayout();
    }

    @Override
    protected void onMeasure(int widthMeasureSpec, int heightMeasureSpec) {
        super.onMeasure(widthMeasureSpec, heightMeasureSpec);
        int width = MeasureSpec.getSize(widthMeasureSpec);
        int height = MeasureSpec.getSize(heightMeasureSpec);
        if (0 == mRatioWidth || 0 == mRatioHeight) {
            setMeasuredDimension(width, height);
        } else {
            if (width < height * mRatioWidth / mRatioHeight) {
                setMeasuredDimension(width, width * mRatioHeight / mRatioWidth);
            } else {
                setMeasuredDimension(height * mRatioWidth / mRatioHeight, height);
            }
        }
    }

}
```

(Android) Camera2BasicFragment

```
/*
 * Copyright 2017 The Android Open Source Project
 *
 * Licensed under the Apache License, Version 2.0 (the "License");
 * you may not use this file except in compliance with the License.
 * You may obtain a copy of the License at
 *
 *        http://www.apache.org/licenses/LICENSE-2.0
 *
 * Unless required by applicable law or agreed to in writing, software
 * distributed under the License is distributed on an "AS IS" BASIS,
 * WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
 * See the License for the specific language governing permissions and
 * limitations under the License.
 */

package com.example.card_project;

import android.Manifest;
import android.app.Activity;
import android.app.AlertDialog;
import android.app.Dialog;
import android.content.Context;
import android.content.DialogInterface;
import android.content.pm.PackageManager;
import android.content.res.Configuration;
import android.graphics.ImageFormat;
import android.graphics.Matrix;
import android.graphics.Point;
import android.graphics.RectF;
import android.graphics.SurfaceTexture;
import android.hardware.camera2.CameraAccessException;
import android.hardware.camera2.CameraCaptureSession;
import android.hardware.camera2.CameraCharacteristics;
import android.hardware.camera2.CameraDevice;
import android.hardware.camera2.CameraManager;
import android.hardware.camera2.CameraMetadata;
import android.hardware.camera2.CaptureRequest;
import android.hardware.camera2.CaptureResult;
import android.hardware.camera2.TotalCaptureResult;
import android.hardware.camera2.params.StreamConfigurationMap;
import android.media.Image;
import android.media.ImageReader;
import android.os.Bundle;
import android.os.Handler;
import android.os.HandlerThread;



import android.util.Log;
import android.util.Size;
import android.util.SparseIntArray;
import android.view.LayoutInflater;
import android.view.Surface;
import android.view.TextureView;
import android.view.View;
import android.view.ViewGroup;
import android.widget.Toast;

import androidx.annotation.NonNull;
import androidx.core.app.ActivityCompat;
import androidx.core.content.ContextCompat;
import androidx.fragment.app.DialogFragment;
import androidx.fragment.app.Fragment;

import java.io.File;
import java.io.FileOutputStream;
import java.io.IOException;
import java.nio.ByteBuffer;
import java.util.ArrayList;
import java.util.Arrays;
import java.util.Collections;
import java.util.Comparator;
import java.util.List;
import java.util.concurrent.Semaphore;
import java.util.concurrent.TimeUnit;

public class Camera2BasicFragment extends Fragment
        implements                                                  View.OnClickListener,
ActivityCompat.OnRequestPermissionsResultCallback {

    /**
     * Conversion from screen rotation to JPEG orientation.
     */
    private static final SparseIntArray ORIENTATIONS = new SparseIntArray();
    private static final int REQUEST_CAMERA_PERMISSION = 1;
    private static final String FRAGMENT_DIALOG = "dialog";

    static {
        ORIENTATIONS.append(Surface.ROTATION_0, 90);
        ORIENTATIONS.append(Surface.ROTATION_90, 0);
        ORIENTATIONS.append(Surface.ROTATION_180, 270);
        ORIENTATIONS.append(Surface.ROTATION_270, 180);
    }

    /**
     * Tag for the {@link Log}.
     */
    private static final String TAG = "Camera2BasicFragment";

    /**
     * Camera state: Showing camera preview.
     */
    private static final int STATE_PREVIEW = 0;

    /**
     * Camera state: Waiting for the focus to be locked.
     */
    private static final int STATE_WAITING_LOCK = 1;

    /**
     * Camera state: Waiting for the exposure to be precapture state.
     */
    private static final int STATE_WAITING_PRECAPTURE = 2;

    /**
     * Camera state: Waiting for the exposure state to be something other than
precapture.
     */
    private static final int STATE_WAITING_NON_PRECAPTURE = 3;

    /**
     * Camera state: Picture was taken.
     */
    private static final int STATE_PICTURE_TAKEN = 4;

    /**
     * Max preview width that is guaranteed by Camera2 API
     */
    private static final int MAX_PREVIEW_WIDTH = 1920;

    /**
     * Max preview height that is guaranteed by Camera2 API
     */
    private static final int MAX_PREVIEW_HEIGHT = 1080;

    /**
     * {@link TextureView.SurfaceTextureListener} handles several lifecycle events on
a
     * {@link TextureView}.
     */
    private final TextureView.SurfaceTextureListener mSurfaceTextureListener
            = new TextureView.SurfaceTextureListener() {

        @Override
        public void onSurfaceTextureAvailable(SurfaceTexture texture, int width, int
height) {
            openCamera(width, height);
        }

        @Override
        public void onSurfaceTextureSizeChanged(SurfaceTexture texture, int width,
int height) {
            configureTransform(width, height);
        }

        @Override
        public boolean onSurfaceTextureDestroyed(SurfaceTexture texture) {
            return true;
        }

        @Override
        public void onSurfaceTextureUpdated(SurfaceTexture texture) {
        }

    };

    /**
     * ID of the current {@link CameraDevice}.
     */
    private String mCameraId;

    /**
     * An {@link AutoFitTextureView} for camera preview.
     */
    private AutoFitTextureView mTextureView;
    /**
     * A {@link CameraCaptureSession } for camera preview.
     */
    private CameraCaptureSession mCaptureSession;

    /**
     * A reference to the opened {@link CameraDevice}.
     */
    private CameraDevice mCameraDevice;

    /**
     * The {@link Size} of camera preview.
     */
    private Size mPreviewSize;

    /**
     * 카메라 전면 후면 아이디 값
     */
    private int facingId = 0;

    /**
     * 자동포커스 체크
     */
    private boolean mAutoFocusSupported ;


    /**
     * {@link CameraDevice.StateCallback} is called when {@link CameraDevice}
changes its state.
     */
    private final CameraDevice.StateCallback mStateCallback = new
CameraDevice.StateCallback() {

        @Override
        public void onOpened(@NonNull CameraDevice cameraDevice) {
            // This method is called when the camera is opened.  We start camera
preview here.
            mCameraOpenCloseLock.release();
            mCameraDevice = cameraDevice;
            createCameraPreviewSession();
        }
        @Override
        public void onDisconnected(@NonNull CameraDevice cameraDevice) {
            mCameraOpenCloseLock.release();
            cameraDevice.close();
            mCameraDevice = null;
        }

        @Override
        public void onError(@NonNull CameraDevice cameraDevice, int error) {
            mCameraOpenCloseLock.release();
            cameraDevice.close();
            mCameraDevice = null;
            Activity activity = getActivity();
            if (null != activity) {
                activity.finish();
            }
        }

    };

    /**
     * An additional thread for running tasks that shouldn't block the UI.
     */
    private HandlerThread mBackgroundThread;

    /**
     * A {@link Handler} for running tasks in the background.
     */
    private Handler mBackgroundHandler;

    /**
     * An {@link ImageReader} that handles still image capture.
     */
    private ImageReader mImageReader;

    /**
     * This is the output file for our picture.
     */

    private File mFile;
    /**
     * This a callback object for the {@link ImageReader}. "onImageAvailable" will be
called when a
     * still image is ready to be saved.
     */
    /*
    private final ImageReader.OnImageAvailableListener mOnImageAvailableListener
            = new ImageReader.OnImageAvailableListener() {

        @Override
        public void onImageAvailable(ImageReader reader) {
            mBackgroundHandler.post(new ImageUploader(reader.acquireNextImage()));
        }

    };
*/
    private ImageReader.OnImageAvailableListener mOnImageAvailableListener;

    public    void    setOnImageAvailableListener(ImageReader.OnImageAvailableListener
mOnImageAvailableListener){
        this.mOnImageAvailableListener = mOnImageAvailableListener;
    }
    /**
     * {@link CaptureRequest.Builder} for the camera preview
     */
    private CaptureRequest.Builder mPreviewRequestBuilder;

    /**
     * {@link CaptureRequest} generated by {@link #mPreviewRequestBuilder}
     */
    private CaptureRequest mPreviewRequest;

    /**
     * The current state of camera state for taking pictures.
     *
     * @see #mCaptureCallback
     */
    private int mState = STATE_PREVIEW;

    /**
     * A {@link Semaphore} to prevent the app from exiting before closing the
camera.
```

```
     */
    private Semaphore mCameraOpenCloseLock = new Semaphore(1);

    /**
     * Whether the current camera device supports Flash or not.
     */
    private boolean mFlashSupported;

    /**
     * Orientation of the camera sensor
     */
    private int mSensorOrientation;

    /**
     * A {@link CameraCaptureSession.CaptureCallback} that handles events related to
JPEG capture.
     */
    private CameraCaptureSession.CaptureCallback mCaptureCallback
            = new CameraCaptureSession.CaptureCallback() {

        private void process(CaptureResult result) {
            switch (mState) {
                case STATE_PREVIEW: {
                    // We have nothing to do when the camera preview is working
normally.
                    break;
                }
                case STATE_WAITING_LOCK: {
                    Integer afState = result.get(CaptureResult.CONTROL_AF_STATE);
                    if (afState == null) {
                        captureStillPicture();
                    } else if (CaptureResult.CONTROL_AF_STATE_FOCUSED_LOCKED
== afState ||

CaptureResult.CONTROL_AF_STATE_NOT_FOCUSED_LOCKED == afState) {
                        // CONTROL_AE_STATE can be null on some devices
                        Integer                        aeState                        =
result.get(CaptureResult.CONTROL_AE_STATE);
                        if (aeState == null ||
                                aeState                                        ==
CaptureResult.CONTROL_AE_STATE_CONVERGED) {
                            mState = STATE_PICTURE_TAKEN;
                            captureStillPicture();
                        } else {
                            runPrecaptureSequence();
                        }
                    }
                    break;
                }
                case STATE_WAITING_PRECAPTURE: {
                    // CONTROL_AE_STATE can be null on some devices
                    Integer aeState = result.get(CaptureResult.CONTROL_AE_STATE);
                    if (aeState == null ||
                            aeState                                                ==
CaptureResult.CONTROL_AE_STATE_PRECAPTURE ||
                            aeState                                                ==
CaptureRequest.CONTROL_AE_STATE_FLASH_REQUIRED) {
                        mState = STATE_WAITING_NON_PRECAPTURE;
                    }
                    break;
                }
                case STATE_WAITING_NON_PRECAPTURE: {
                    // CONTROL_AE_STATE can be null on some devices
                    Integer aeState = result.get(CaptureResult.CONTROL_AE_STATE);
                    if      (aeState      ==      null      ||      aeState      !=
CaptureResult.CONTROL_AE_STATE_PRECAPTURE) {
                        mState = STATE_PICTURE_TAKEN;
                        captureStillPicture();
                    }
                    break;
                }
            }
        }

        @Override
        public void onCaptureProgressed(@NonNull CameraCaptureSession session,
                                        @NonNull CaptureRequest request,
                                        @NonNull CaptureResult partialResult) {
            process(partialResult);
        }

        @Override
        public void onCaptureCompleted(@NonNull CameraCaptureSession session,
                                       @NonNull CaptureRequest request,
                                        @NonNull TotalCaptureResult result) {
            process(result);
        }

    };

    /**
     * Shows a {@link Toast} on the UI thread.
     *
     * @param text The message to show
     */
    private void showToast(final String text) {
        final Activity activity = getActivity();
        if (activity != null) {
            activity.runOnUiThread(new Runnable() {
                @Override
                public void run() {
                    Toast.makeText(activity, text, Toast.LENGTH_SHORT).show();
                }
            });
        }
    }

    /**
     * Given {@code choices} of {@code Size}s supported by a camera, choose the
smallest one that
     * is at least as large as the respective texture view size, and that is at most as
large as the
     * respective max size, and whose aspect ratio matches with the specified value.
If such size
     * doesn't exist, choose the largest one that is at most as large as the respective
max size,
     * and whose aspect ratio matches with the specified value.
     *
     * @param choices           The list of sizes that the camera supports for the
intended output
     *                          class
     * @param textureViewWidth  The width of the texture view relative to sensor
coordinate
     * @param textureViewHeight The height of the texture view relative to sensor
coordinate
     * @param maxWidth          The maximum width that can be chosen
     * @param maxHeight          The maximum height that can be chosen
     * @param aspectRatio        The aspect ratio
     * @return The optimal {@code Size}, or an arbitrary one if none were big
enough
     */
    private static Size chooseOptimalSize(Size[] choices, int textureViewWidth,
                                          int textureViewHeight, int maxWidth, int
maxHeight, Size aspectRatio) {

        // Collect the supported resolutions that are at least as big as the preview
Surface
        List<Size> bigEnough = new ArrayList<>();
        // Collect the supported resolutions that are smaller than the preview
Surface
        List<Size> notBigEnough = new ArrayList<>();
        int w = aspectRatio.getWidth();
        int h = aspectRatio.getHeight();
        for (Size option : choices) {
            if (option.getWidth() <= maxWidth && option.getHeight() <= maxHeight &&
                    option.getHeight() == option.getWidth() * h / w) {
                if (option.getWidth() >= textureViewWidth &&
                        option.getHeight() >= textureViewHeight) {
                    bigEnough.add(option);
                } else {
                    notBigEnough.add(option);
                }
            }
        }

        // Pick the smallest of those big enough. If there is no one big enough, pick
the
        // largest of those not big enough.
        if (bigEnough.size() > 0) {
            return Collections.min(bigEnough, new CompareSizesByArea());
        } else if (notBigEnough.size() > 0) {
            return Collections.max(notBigEnough, new CompareSizesByArea());
        } else {
            Log.e(TAG, "Couldn't find any suitable preview size");
            return choices[0];
        }
    }
    public static Camera2BasicFragment newInstance() {
        return new Camera2BasicFragment();
    }

    @Override
    public View onCreateView(LayoutInflater inflater, ViewGroup container,
                             Bundle savedInstanceState) {
        return inflater.inflate(R.layout.fragment_camera2_basic, container, false);
    }

    @Override
    public void onViewCreated(final View view, Bundle savedInstanceState) {
        view.findViewById(R.id.picture).setOnClickListener(this);
        view.findViewById(R.id.change).setOnClickListener(this);
        mTextureView = (AutoFitTextureView) view.findViewById(R.id.texture);
    }

    @Override
    public void onActivityCreated(Bundle savedInstanceState) {
        super.onActivityCreated(savedInstanceState);
        mFile = new File(getActivity().getExternalFilesDir(null), "pic.jpg");
    }

    @Override
    public void onResume() {
        super.onResume();
        startBackgroundThread();

        // When the screen is turned off and turned back on, the SurfaceTexture is
already
        // available, and "onSurfaceTextureAvailable" will not be called. In that case,
we can open
        // a camera and start preview from here (otherwise, we wait until the
surface is ready in
        // the SurfaceTextureListener).
        if (mTextureView.isAvailable()) {
            openCamera(mTextureView.getWidth(), mTextureView.getHeight());
        } else {
            mTextureView.setSurfaceTextureListener(mSurfaceTextureListener);
        }
    }
    @Override
    public void onPause() {
        closeCamera();
        stopBackgroundThread();
        super.onPause();
    }

    private void requestCameraPermission() {
        if (shouldShowRequestPermissionRationale(Manifest.permission.CAMERA)) {
            new                   ConfirmationDialog().show(getChildFragmentManager(),
FRAGMENT_DIALOG);
        } else {
            requestPermissions(new                String[]{Manifest.permission.CAMERA},
REQUEST_CAMERA_PERMISSION);
        }
    }



    @Override
    public   void   onRequestPermissionsResult(int   requestCode,   @NonNull   String
permissions[],
                                           @NonNull int[] grantResults) {
        if (requestCode == REQUEST_CAMERA_PERMISSION) {
            if      (grantResults.length      !=      1      ||      grantResults[0]      !=
PackageManager.PERMISSION_GRANTED) {
                ErrorDialog.newInstance(getString(R.string.request_permission))
                        .show(getChildFragmentManager(), FRAGMENT_DIALOG);
            }
        } else {
            super.onRequestPermissionsResult(requestCode, permissions, grantResults);
        }
    }



    /**
     * Sets up member variables related to camera.
     *
     * @param width  The width of available size for camera preview
     * @param height The height of available size for camera preview
     */
    @SuppressWarnings("SuspiciousNameCombination")
    private void setUpCameraOutputs(int width, int height) {
        Activity activity = getActivity();
        CameraManager              manager             =             (CameraManager)
activity.getSystemService(Context.CAMERA_SERVICE);
        try {
            for (String cameraId : manager.getCameraIdList()) {
                CameraCharacteristics characteristics
                        = manager.getCameraCharacteristics(cameraId);

                int[]                           afAvailableModes                          =
characteristics.get(CameraCharacteristics.CONTROL_AF_AVAILABLE_MODES);

                if(afAvailableModes.length==0 || (afAvailableModes.length ==1
                        &&                    afAvailableModes[0]                        ==
CameraMetadata.CONTROL_AF_MODE_OFF)){
                    mAutoFocusSupported = false;
                }else{
                    mAutoFocusSupported = true;
                }
                Integer                              facing                              =
characteristics.get(CameraCharacteristics.LENS_FACING);
                if (facing != null && facing == facingId) {
                    StreamConfigurationMap map = characteristics.get(

CameraCharacteristics.SCALER_STREAM_CONFIGURATION_MAP);
                    if (map == null) {
                        continue;
                    }
                    // For still image captures, we use the largest available size.
                    Size largest = Collections.max(
                            Arrays.asList(map.getOutputSizes(ImageFormat.JPEG)),
                            new CompareSizesByArea());
                    mImageReader     =     ImageReader.newInstance(largest.getWidth(),
largest.getHeight(),
                            ImageFormat.JPEG, /*maxImages*/2);
                    mImageReader.setOnImageAvailableListener(
                            mOnImageAvailableListener, mBackgroundHandler);

                    // Find out if we need to swap dimension to get the preview size
relative to sensor
                    // coordinate.
                    int                           displayRotation                           =
activity.getWindowManager().getDefaultDisplay().getRotation();
```

```
                //noinspection ConstantConditions
                mSensorOrientation = characteristics.get(CameraCharacteristics.SENSOR_ORIENTATION);
                boolean swappedDimensions = false;
                switch (displayRotation) {
                    case Surface.ROTATION_0:
                    case Surface.ROTATION_180:
                        if (mSensorOrientation == 90 || mSensorOrientation == 270) {
                            swappedDimensions = true;
                        }
                        break;
                    case Surface.ROTATION_90:
                    case Surface.ROTATION_270:
                        if (mSensorOrientation == 0 || mSensorOrientation == 180) {
                            swappedDimensions = true;
                        }
                        break;
                    default:
                        Log.e(TAG, "Display rotation is invalid: " + displayRotation);
                }

                Point displaySize = new Point();
                activity.getWindowManager().getDefaultDisplay().getSize(displaySize);
                int rotatedPreviewWidth = width;
                int rotatedPreviewHeight = height;
                int maxPreviewWidth = displaySize.x;
                int maxPreviewHeight = displaySize.y;

                if (swappedDimensions) {
                    rotatedPreviewWidth = height;
                    rotatedPreviewHeight = width;
                    maxPreviewWidth = displaySize.y;
                    maxPreviewHeight = displaySize.x;
                }

                if (maxPreviewWidth > MAX_PREVIEW_WIDTH) {
                    maxPreviewWidth = MAX_PREVIEW_WIDTH;
                }

                if (maxPreviewHeight > MAX_PREVIEW_HEIGHT) {
                    maxPreviewHeight = MAX_PREVIEW_HEIGHT;
                }

                // Danger, W.R.! Attempting to use too large a preview size could
exceed the camera
                // bus' bandwidth limitation, resulting in gorgeous previews but
the storage of
                // garbage capture data.
                mPreviewSize                                                    =
chooseOptimalSize(map.getOutputSizes(SurfaceTexture.class),
                        rotatedPreviewWidth,            rotatedPreviewHeight,
maxPreviewWidth,
                        maxPreviewHeight, largest);

                // We fit the aspect ratio of TextureView to the size of preview
we picked.
                int orientation = getResources().getConfiguration().orientation;
                if (orientation == Configuration.ORIENTATION_LANDSCAPE) {
                    mTextureView.setAspectRatio(
                            mPreviewSize.getWidth(), mPreviewSize.getHeight());
                } else {
                    mTextureView.setAspectRatio(
                            mPreviewSize.getHeight(), mPreviewSize.getWidth());
                }

                // Check if the flash is supported.
                Boolean                         available                       =
characteristics.get(CameraCharacteristics.FLASH_INFO_AVAILABLE);
                mFlashSupported = available == null ? false : available;

                mCameraId = cameraId;
                return;
            }
        }


    } catch (CameraAccessException e) {
        e.printStackTrace();
    } catch (NullPointerException e) {
        // Currently an NPE is thrown when the Camera2API is used but not
supported on the
        // device this code runs.
```

```
            ErrorDialog.newInstance(getString(R.string.camera_error))
                    .show(getChildFragmentManager(), FRAGMENT_DIALOG);
        }
    }

    /**
     * Opens the camera specified by {@link Camera2BasicFragment#mCameraId}.
     */
    private void openCamera(int width, int height) {
        if                              (ContextCompat.checkSelfPermission(getActivity(),
Manifest.permission.CAMERA)
                != PackageManager.PERMISSION_GRANTED) {
            requestCameraPermission();
            return;
        }
        setUpCameraOutputs(width, height);
        configureTransform(width, height);
        Activity activity = getActivity();
        CameraManager             manager             =             (CameraManager)
activity.getSystemService(Context.CAMERA_SERVICE);
        try {
            if (!mCameraOpenCloseLock.tryAcquire(2500, TimeUnit.MILLISECONDS)) {
                throw   new   RuntimeException("Time   out   waiting   to   lock   camera
opening.");
            }
            manager.openCamera(mCameraId, mStateCallback, mBackgroundHandler);
        } catch (CameraAccessException e) {
            e.printStackTrace();
        } catch (InterruptedException e) {
            throw  new  RuntimeException("Interrupted  while  trying  to  lock  camera
opening.", e);
        }
    }


    /**
     * Closes the current {@link CameraDevice}.
     */
    public void closeCamera() {
        try {
            mCameraOpenCloseLock.acquire();
            if (null != mCaptureSession) {
                mCaptureSession.close();
                mCaptureSession = null;
            }
            if (null != mCameraDevice) {
                mCameraDevice.close();
                mCameraDevice = null;
            }
            if (null != mImageReader) {
                mImageReader.close();
                mImageReader = null;
            }
        } catch (InterruptedException e) {
            throw new RuntimeException("Interrupted while trying to lock camera
closing.", e);
        } finally {
            mCameraOpenCloseLock.release();
        }
    }

    /**
     * Starts a background thread and its {@link Handler}.
     */
    private void startBackgroundThread() {
        mBackgroundThread = new HandlerThread("CameraBackground");
        mBackgroundThread.start();
        mBackgroundHandler = new Handler(mBackgroundThread.getLooper());
    }

    /**
     * Stops the background thread and its {@link Handler}.
     */
    private void stopBackgroundThread() {
        mBackgroundThread.quitSafely();
        try {
            mBackgroundThread.join();
            mBackgroundThread = null;
            mBackgroundHandler = null;
        } catch (InterruptedException e) {
            e.printStackTrace();
        }
    }
    /**
     * Creates a new {@link CameraCaptureSession} for camera preview.
     */
    private void createCameraPreviewSession() {
        try {
            SurfaceTexture texture = mTextureView.getSurfaceTexture();
            assert texture != null;

            // We configure the size of default buffer to be the size of camera
preview we want.
            texture.setDefaultBufferSize(mPreviewSize.getWidth(),
mPreviewSize.getHeight());

            // This is the output Surface we need to start preview.
            Surface surface = new Surface(texture);

            // We set up a CaptureRequest.Builder with the output Surface.
            mPreviewRequestBuilder
                    =
mCameraDevice.createCaptureRequest(CameraDevice.TEMPLATE_PREVIEW);
            mPreviewRequestBuilder.addTarget(surface);

            // Here, we create a CameraCaptureSession for camera preview.
            mCameraDevice.createCaptureSession(Arrays.asList(surface,
mImageReader.getSurface()),
                    new CameraCaptureSession.StateCallback() {

                        @Override
                        public void onConfigured(@NonNull CameraCaptureSession
cameraCaptureSession) {
                            // The camera is already closed
                            if (null == mCameraDevice) {
                                return;
                            }

                            // When the session is ready, we start displaying the
preview.
                            mCaptureSession = cameraCaptureSession;
                            try {
                                // Auto focus should be continuous for camera
preview.
mPreviewRequestBuilder.set(CaptureRequest.CONTROL_AF_MODE,

CaptureRequest.CONTROL_AF_MODE_CONTINUOUS_PICTURE);
                                // Flash is automatically enabled when necessary.
                                setAutoFlash(mPreviewRequestBuilder);

                                // Finally, we start displaying the camera preview.
                                mPreviewRequest = mPreviewRequestBuilder.build();
                                mCaptureSession.setRepeatingRequest(mPreviewRequest,
                                        mCaptureCallback, mBackgroundHandler);
                            } catch (CameraAccessException e) {
                                e.printStackTrace();
                            }
                        }

                        @Override
                        public void onConfigureFailed(
                                @NonNull                        CameraCaptureSession
cameraCaptureSession) {
                            showToast("Failed");
                        }
                    }, null
            );
        } catch (CameraAccessException e) {
            e.printStackTrace();
        }
    }

    /**
     * Configures the necessary {@link Matrix} transformation to `mTextureView`.
     * This method should be called after the camera preview size is determined in
     * setUpCameraOutputs and also the size of `mTextureView` is fixed.
     *
     * @param viewWidth  The width of `mTextureView`
     * @param viewHeight The height of `mTextureView`
     */
    private void configureTransform(int viewWidth, int viewHeight) {
        Activity activity = getActivity();
        if (null == mTextureView || null == mPreviewSize || null == activity) {
            return;
        }
        int rotation = activity.getWindowManager().getDefaultDisplay().getRotation();
        Matrix matrix = new Matrix();
        RectF viewRect = new RectF(0, 0, viewWidth, viewHeight);
        RectF    bufferRect    =    new    RectF(0,    0,    mPreviewSize.getHeight(),
mPreviewSize.getWidth());
        float centerX = viewRect.centerX();
        float centerY = viewRect.centerY();
        if (Surface.ROTATION_90 == rotation || Surface.ROTATION_270 == rotation) {
            bufferRect.offset(centerX    -    bufferRect.centerX(),    centerY    -
bufferRect.centerY());
            matrix.setRectToRect(viewRect, bufferRect, Matrix.ScaleToFit.FILL);
            float scale = Math.max(
                    (float) viewHeight / mPreviewSize.getHeight(),
                    (float) viewWidth / mPreviewSize.getWidth());
            matrix.postScale(scale, scale, centerX, centerY);
            matrix.postRotate(90 * (rotation - 2), centerX, centerY);
        } else if (Surface.ROTATION_180 == rotation) {
            matrix.postRotate(180, centerX, centerY);
        }
        mTextureView.setTransform(matrix);
    }

    /**
     * Initiate a still image capture.
     */
    private void takePicture() {
        if(mAutoFocusSupported){
            lockFocus();
        }else{
            captureStillPicture();
        }
    }

    /**
     * Lock the focus as the first step for a still image capture.
     */
    private void lockFocus() {
        try {
            // This is how to tell the camera to lock focus.
            mPreviewRequestBuilder.set(CaptureRequest.CONTROL_AF_TRIGGER,
                    CameraMetadata.CONTROL_AF_TRIGGER_START);
            // Tell #mCaptureCallback to wait for the lock.
            mState = STATE_WAITING_LOCK;
            mCaptureSession.capture(mPreviewRequestBuilder.build(),
mCaptureCallback,
                    mBackgroundHandler);
        } catch (CameraAccessException e) {
            e.printStackTrace();
        }
    }

    /**
     * Run the precapture sequence for capturing a still image. This method should
be called when
     * we get a response in {@link #mCaptureCallback} from {@link #lockFocus()}.
     */
    private void runPrecaptureSequence() {
        try {
            // This is how to tell the camera to trigger.

mPreviewRequestBuilder.set(CaptureRequest.CONTROL_AE_PRECAPTURE_TRIGGER,
                    CaptureRequest.CONTROL_AE_PRECAPTURE_TRIGGER_START);
            // Tell #mCaptureCallback to wait for the precapture sequence to be set.
            mState = STATE_WAITING_PRECAPTURE;
            mCaptureSession.capture(mPreviewRequestBuilder.build(),
mCaptureCallback,
                    mBackgroundHandler);
        } catch (CameraAccessException e) {
            e.printStackTrace();
        }
    }

    /**
     * Capture a still picture. This method should be called when we get a response
in
     * {@link #mCaptureCallback} from both {@link #lockFocus()}.
     */
    private void captureStillPicture() {
        try {
            final Activity activity = getActivity();
            if (null == activity || null == mCameraDevice) {
                return;
            }
            // This is the CaptureRequest.Builder that we use to take a picture.
            final CaptureRequest.Builder captureBuilder =
mCameraDevice.createCaptureRequest(CameraDevice.TEMPLATE_STILL_CAPTURE);
            captureBuilder.addTarget(mImageReader.getSurface());

            // Use the same AE and AF modes as the preview.
            captureBuilder.set(CaptureRequest.CONTROL_AF_MODE,
                    CaptureRequest.CONTROL_AF_MODE_CONTINUOUS_PICTURE);
            setAutoFlash(captureBuilder);

            // Orientation
            int rotation = activity.getWindowManager().getDefaultDisplay().getRotation();
            captureBuilder.set(CaptureRequest.JPEG_ORIENTATION,
getOrientation(rotation));

            CameraCaptureSession.CaptureCallback CaptureCallback
                    = new CameraCaptureSession.CaptureCallback() {

                @Override
                public void onCaptureCompleted(@NonNull CameraCaptureSession
session,
                                               @NonNull CaptureRequest request,
                                               @NonNull TotalCaptureResult result) {
                    //showToast("Saved: " + mFile);
                    Log.d(TAG, mFile.toString());
                    unlockFocus();
                }
            };

            mCaptureSession.stopRepeating();
            mCaptureSession.abortCaptures();
            mCaptureSession.capture(captureBuilder.build(), CaptureCallback, null);
        } catch (CameraAccessException e) {
            e.printStackTrace();
        }
    }

    /**
     * Retrieves the JPEG orientation from the specified screen rotation.
     *
     * @param rotation The screen rotation.
     * @return The JPEG orientation (one of 0, 90, 270, and 360)
     */
    private int getOrientation(int rotation) {
        // Sensor orientation is 90 for most devices, or 270 for some devices (eg.
Nexus 5X)
        // We have to take that into account and rotate JPEG properly.
        // For devices with orientation of 90, we simply return our mapping from
ORIENTATIONS.
        // For devices with orientation of 270, we need to rotate the JPEG 180
degrees.
        return (ORIENTATIONS.get(rotation) + mSensorOrientation + 270) % 360;
    }

    /**
     * Unlock the focus. This method should be called when still image capture
sequence is
     * finished.
     */
    private void unlockFocus() {
        try {
            // Reset the auto-focus trigger
            mPreviewRequestBuilder.set(CaptureRequest.CONTROL_AF_TRIGGER,
                    CameraMetadata.CONTROL_AF_TRIGGER_CANCEL);
            setAutoFlash(mPreviewRequestBuilder);
            mCaptureSession.capture(mPreviewRequestBuilder.build(),
mCaptureCallback,
                    mBackgroundHandler);
            // After this, the camera will go back to the normal state of preview.
            mState = STATE_PREVIEW;
            mCaptureSession.setRepeatingRequest(mPreviewRequest, mCaptureCallback,
                    mBackgroundHandler);
        } catch (CameraAccessException e) {
            e.printStackTrace();
        }
    }

    @Override
    public void onClick(View view) {
        switch (view.getId()) {
            case R.id.picture:
                takePicture();
                break;

            case R.id.change:
                if(facingId == CameraCharacteristics.LENS_FACING_BACK){
                    facingId = CameraCharacteristics.LENS_FACING_FRONT;
                }else{
                    facingId = CameraCharacteristics.LENS_FACING_BACK;

                }
                closeCamera();
                openCamera(mTextureView.getWidth(), mTextureView.getHeight());
                break;

        }
    }

    private void setAutoFlash(CaptureRequest.Builder requestBuilder) {
        if (mFlashSupported) {
            requestBuilder.set(CaptureRequest.CONTROL_AE_MODE,
                    CaptureRequest.CONTROL_AE_MODE_ON_AUTO_FLASH);
        }
    }
    /**
     * Saves a JPEG {@link Image} into the specified {@link File}.
     */



    /**
     * Saves a JPEG {@link Image} into the specified {@link File}.
     */
    private static class ImageSaver implements Runnable {

        /**
         * The JPEG image
         */
        private final Image mImage;
        /**
         * The file we save the image into.
         */
        private final File mFile;

        ImageSaver(Image image, File file) {
            mImage = image;
            mFile = file;
        }
        @Override
        public void run() {
            ByteBuffer buffer = mImage.getPlanes()[0].getBuffer();
            byte[] bytes = new byte[buffer.remaining()];
            buffer.get(bytes);
            FileOutputStream output = null;
            try {
                output = new FileOutputStream(mFile);
                output.write(bytes);
            } catch (IOException e) {
                e.printStackTrace();
            } finally {
                mImage.close();
                if (null != output) {
                    try {
                        output.close();
                    } catch (IOException e) {
                        e.printStackTrace();
                    }
                }
            }
        }

    }


    /**
     * Compares two {@code Size}s based on their areas.
     */
    static class CompareSizesByArea implements Comparator<Size> {

        @Override
        public int compare(Size lhs, Size rhs) {
            // We cast here to ensure the multiplications won't overflow
            return Long.signum((long) lhs.getWidth() * lhs.getHeight() -
                    (long) rhs.getWidth() * rhs.getHeight());
        }

    }

    /**
```

```
     * Shows an error message dialog.
     */
    public static class ErrorDialog extends DialogFragment {

        private static final String ARG_MESSAGE = "message";

        public static ErrorDialog newInstance(String message) {
            ErrorDialog dialog = new ErrorDialog();
            Bundle args = new Bundle();
            args.putString(ARG_MESSAGE, message);
            dialog.setArguments(args);
            return dialog;
        }

        @NonNull
        @Override
        public Dialog onCreateDialog(Bundle savedInstanceState) {
            final Activity activity = getActivity();
            return new AlertDialog.Builder(activity)
                    .setMessage(getArguments().getString(ARG_MESSAGE))
                    .setPositiveButton(android.R.string.ok,                    new
DialogInterface.OnClickListener() {
                        @Override
                        public void onClick(DialogInterface dialogInterface, int i) {
                            activity.finish();
                        }
                    })
                    .create();
        }

    }

    /**
     * Shows OK/Cancel confirmation dialog about camera permission.
     */
    public static class ConfirmationDialog extends DialogFragment {

        @NonNull
        @Override
        public Dialog onCreateDialog(Bundle savedInstanceState) {
            final Fragment parent = getParentFragment();
            return new AlertDialog.Builder(getActivity())
```

```
                    .setMessage(R.string.request_permission)
                    .setPositiveButton(android.R.string.ok,                          new
DialogInterface.OnClickListener() {
                        @Override
                        public void onClick(DialogInterface dialog, int which) {
                            p a r e n t . r e q u e s t P e r m i s s i o n s ( n e w
String[]{Manifest.permission.CAMERA},
                                    REQUEST_CAMERA_PERMISSION);
                        }
                    })
                    .setNegativeButton(android.R.string.cancel,
                            new DialogInterface.OnClickListener() {
                                @Override
                                public void onClick(DialogInterface dialog, int which) {
                                    Activity activity = parent.getActivity();
                                    if (activity != null) {
                                        activity.finish();
                                    }
                                }
                            })
                    .create();
        }
    }

}
```

(Android) CourseFragment.java

```
package com.example.card_project;

import android.os.Bundle;

import androidx.fragment.app.Fragment;

import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;

import com.example.card_project.R;
```

```
/**
 * A simple {@link Fragment} subclass.
 * Use the {@link CourseFragment#newInstance} factory method to
 * create an instance of this fragment.
 */
public class CourseFragment extends Fragment {

    // TODO: Rename parameter arguments, choose names that match
    // the fragment initialization parameters, e.g. ARG_ITEM_NUMBER
    private static final String ARG_PARAM1 = "param1";
    private static final String ARG_PARAM2 = "param2";

    // TODO: Rename and change types of parameters
    private String mParam1;
    private String mParam2;

    public CourseFragment() {
        // Required empty public constructor
    }

    /**
     * Use this factory method to create a new instance of
     * this fragment using the provided parameters.
     *
     * @param param1 Parameter 1.
     * @param param2 Parameter 2.
     * @return A new instance of fragment CourseFragment.
     */
    // TODO: Rename and change types and number of parameters
    public static CourseFragment newInstance(String param1, String param2) {
        CourseFragment fragment = new CourseFragment();
        Bundle args = new Bundle();
        args.putString(ARG_PARAM1, param1);
        args.putString(ARG_PARAM2, param2);
        fragment.setArguments(args);
        return fragment;
    }

    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        if (getArguments() != null) {
            mParam1 = getArguments().getString(ARG_PARAM1);
            mParam2 = getArguments().getString(ARG_PARAM2);
        }
    }

    @Override
    public View onCreateView(LayoutInflater inflater, ViewGroup container,
                             Bundle savedInstanceState) {
        // Inflate the layout for this fragment
        return inflater.inflate(R.layout.fragment_course, container, false);
    }
}
```

(Android) GalleryActivity.java

```
package com.example.card_project;

import android.Manifest;
import android.app.Activity;
import android.content.Intent;
import android.content.pm.PackageManager;
import android.database.Cursor;
import android.graphics.Bitmap;
import android.graphics.BitmapFactory;
import android.graphics.Matrix;
import android.media.ExifInterface;
import android.net.Uri;
import android.os.Build;
import android.os.Bundle;
import android.provider.MediaStore;



import androidx.annotation.NonNull;
import androidx.annotation.RequiresApi;
import androidx.appcompat.app.AppCompatActivity;
import androidx.core.app.ActivityCompat;
import androidx.core.content.ContextCompat;
import androidx.recyclerview.widget.GridLayoutManager;
import androidx.recyclerview.widget.RecyclerView;
import java.util.ArrayList;

import static com.example.card_project.Util.GALLERY_IMAGE;
import static com.example.card_project.Util.GALLERY_VIDEO;
import static com.example.card_project.Util.INTENT_MEDIA;
import static com.example.card_project.Util.showToast;

public class GalleryActivity extends AppCompatActivity {
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_gallery);

        if (ContextCompat.checkSelfPermission(GalleryActivity.this,
                Manifest.permission.READ_EXTERNAL_STORAGE)
                != PackageManager.PERMISSION_GRANTED) {
            ActivityCompat.requestPermissions(GalleryActivity.this,
                    new String[]{Manifest.permission.READ_EXTERNAL_STORAGE},
                    1);
            if
(ActivityCompat.shouldShowRequestPermissionRationale(GalleryActivity.this,
                    Manifest.permission.READ_EXTERNAL_STORAGE)) {

            } else {
                showToast(GalleryActivity.this,
getResources().getString(R.string.please_grant_permission));
            }
        } else {
            recyclerInit();
        }
    }

    @Override
    public void onRequestPermissionsResult(int requestCode, @NonNull String
permissions[], @NonNull int[] grantResults) {
        super.onRequestPermissionsResult(requestCode, permissions, grantResults);
        switch (requestCode) {
            case 1: {
                if (grantResults.length > 0 && grantResults[0] ==
PackageManager.PERMISSION_GRANTED) {
                    recyclerInit();
                } else {
                    finish();
                    showToast(GalleryActivity.this,
getResources().getString(R.string.please_grant_permission));
                }
            }
        }
    }



    private void recyclerInit(){
        final int numberOfColumns = 3;

        RecyclerView recyclerView = findViewById(R.id.recyclerView);
        recyclerView.setHasFixedSize(true);
        recyclerView.setLayoutManager(new GridLayoutManager(this,
numberOfColumns));

        RecyclerView.Adapter mAdapter = new GalleryAdapter(this,
getImagesPath(this));
        recyclerView.setAdapter(mAdapter);
    }

    public ArrayList<String> getImagesPath(Activity activity) {
        Uri uri;
        ArrayList<String> listOfAllImages = new ArrayList<String>();
        Cursor cursor;
        int column_index_data;
        String PathOfImage = null;
        String[] projection;

        Intent intent = getIntent();
        final int media = intent.getIntExtra(INTENT_MEDIA, GALLERY_IMAGE);
        if(media == GALLERY_VIDEO){
            uri = android.provider.MediaStore.Video.Media.EXTERNAL_CONTENT_URI;
            projection = new String[] { MediaStore.MediaColumns.DATA,
MediaStore.Video.Media.BUCKET_DISPLAY_NAME };
        }else{
            uri = android.provider.MediaStore.Images.Media.EXTERNAL_CONTENT_URI;
            projection = new String[] { MediaStore.MediaColumns.DATA,
MediaStore.Images.Media.BUCKET_DISPLAY_NAME };
```

```
        }

        cursor = activity.getContentResolver().query(uri, projection, null, null, null);
        column_index_data                                                                     =
cursor.getColumnIndexOrThrow(MediaStore.MediaColumns.DATA);

        while (cursor.moveToNext()) {
            PathOfImage = cursor.getString(column_index_data);

            listOfAllImages.add(PathOfImage);
        }
        return listOfAllImages;
    }
}
```

(Android) GalleryAdapter.java

```
package com.example.card_project;



import android.app.Activity;
import android.content.Intent;
import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;
import android.widget.ImageView;

import androidx.annotation.NonNull;
import androidx.cardview.widget.CardView;
import androidx.recyclerview.widget.RecyclerView;

import com.bumptech.glide.Glide;



import java.util.ArrayList;

import static com.example.card_project.Util.INTENT_PATH;

public                  class                  GalleryAdapter                  extends
RecyclerView.Adapter<GalleryAdapter.GalleryViewHolder> {
    private ArrayList<String> mDataset;
    private Activity activity;


    public static class GalleryViewHolder extends RecyclerView.ViewHolder {
        public CardView cardView;
        public GalleryViewHolder(CardView v) {
            super(v);
            cardView = v;
        }
    }

    public GalleryAdapter(Activity activity, ArrayList<String> myDataset) {
        mDataset = myDataset;
        this.activity = activity;
    }


    @NonNull
    @Override
    public GalleryAdapter.GalleryViewHolder onCreateViewHolder(@NonNull ViewGroup
parent, int viewType) {
        CardView                    cardView                    =                    (CardView)
LayoutInflater.from(parent.getContext()).inflate(R.layout.item_gallery, parent, false);
        final GalleryViewHolder galleryViewHolder = new GalleryViewHolder(cardView);
        cardView.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                Intent resultIntent = new Intent();
                resultIntent.putExtra(INTENT_PATH,
mDataset.get(galleryViewHolder.getAdapterPosition()));
                activity.setResult(Activity.RESULT_OK, resultIntent);
                activity.finish();
            }
        });

        return galleryViewHolder;
    }

    @Override
    public void onBindViewHolder(@NonNull GalleryViewHolder holder, int position) {
        CardView cardView = holder.cardView;
        ImageView imageView = cardView.findViewById(R.id.imageView);

Glide.with(activity).load(mDataset.get(position)).centerCrop().override(500).into(imageView)
;



    }

    @Override
    public int getItemCount() {
        return mDataset.size();
    }
}
```

(Android) LoginActivity.java

```
package com.example.card_project;

import android.annotation.SuppressLint;
import android.app.Dialog;
import android.content.Intent;
import android.graphics.Matrix;
import android.os.Bundle;
import android.view.View;
import android.widget.EditText;
import android.widget.Toast;

import androidx.annotation.NonNull;
import androidx.appcompat.app.AppCompatActivity;

import com.google.android.gms.tasks.OnCompleteListener;
import com.google.android.gms.tasks.Task;
import com.google.firebase.auth.AuthResult;
import com.google.firebase.auth.FirebaseAuth;
import com.google.firebase.auth.FirebaseUser;

public class LoginActivity extends AppCompatActivity {
    private static final String TAG = "LoginActivity";
    private FirebaseAuth mAuth;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_login);

        // Initialize Firebase Auth
        mAuth = FirebaseAuth.getInstance();

        findViewById(R.id.loginButton).setOnClickListener(onClickListener);
        findViewById(R.id.gotoPassword).setOnClickListener(onClickListener);
    }

    @Override
    public void onStart() {
        super.onStart();
        // Check if user is signed in (non-null) and update UI accordingly.
        FirebaseUser currentUser = mAuth.getCurrentUser();
    }

    View.OnClickListener onClickListener = new View.OnClickListener() {
        @SuppressLint("NonConstantResourceId")
        @Override
        public void onClick(View view) {
            switch (view.getId()) {
                case R.id.loginButton:
                    login();
                    break;

                case R.id.gotoPassword:
                    myStartActivity(PasswordResetActivity.class);
                    break;
            }
        }
    };



    private void login() { //로그인 로직 처리
        String email = ((EditText) findViewById(R.id.user_id)).getText().toString(); //입력
한 텍스트를 얻어오는 작업
```

```
        String                    password                    =                    ((EditText)
findViewById(R.id.user_password)).getText().toString();

        if (email.length() > 0 && password.length() > 0) {
            mAuth.signInWithEmailAndPassword(email, password)
                    .addOnCompleteListener(this,                                        new
OnCompleteListener<AuthResult>() {
                        @Override
                        public void onComplete(@NonNull Task<AuthResult> task) {
                            if (task.isSuccessful()) {
                                if(mAuth.getCurrentUser().isEmailVerified()){
                                    // 로그인에 성공하면 "로그인 성공" 토스트를 보여줌
                                    Toast.makeText(LoginActivity.this, "로그인에 성공했
습니다.", Toast.LENGTH_SHORT).show();
                                    FirebaseUser user = mAuth.getCurrentUser();

                                    Intent intent = new Intent(getApplicationContext(),
MainActivity.class);
                                    intent.addFlags(Intent.FLAG_ACTIVITY_CLEAR_TOP
| Intent.FLAG_ACTIVITY_NEW_TASK);
                                    startActivity(intent);
                                    finish();

                                } else {
                                    Toast.makeText(LoginActivity.this, "이메일 인증을 완
료해주세요.", Toast.LENGTH_SHORT).show();
                                }
                            }else {
                                // 로그인에 실패하면 "로그인 실패" 토스트를 보여줌
                                Toast.makeText(LoginActivity.this, "로그인에 실패했습니
다.", Toast.LENGTH_SHORT).show();
                            }
                        }

                    });

        }
    }



    private void startToast(String msg){
        Toast.makeText(this, msg, Toast.LENGTH_SHORT).show();
    }

    private void myStartActivity(Class c) {
        Intent intent = new Intent(LoginActivity.this, c);
        startActivity(intent);
    }

}
```

(Android) MainAct

```
package com.example.card_project;

import androidx.appcompat.app.AppCompatActivity;
import androidx.fragment.app.FragmentManager;
import androidx.fragment.app.FragmentTransaction;

import android.annotation.SuppressLint;
import android.os.Bundle;
import android.view.View;
import android.widget.Button;
import android.widget.LinearLayout;
import android.widget.RelativeLayout;

public class MainAct extends AppCompatActivity {

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.act_main);

        final Button courseButton = (Button) findViewById(R.id.courseButton);
        final Button staticsButton = (Button) findViewById(R.id.statisticButton);
        final Button scheduleButton = (Button) findViewById(R.id.scheduleButton);
        final RelativeLayout notice = (RelativeLayout) findViewById(R.id.notice);

        courseButton.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View view) {
                notice.setVisibility(View.GONE);
```

```
courseButton.setBackgroundColor(getResources().getColor(R.color.design_default_color_p
rimary_dark));

staticsButton.setBackgroundColor(getResources().getColor(R.color.design_default_color_p
rimary));

scheduleButton.setBackgroundColor(getResources().getColor(R.color.design_default_color
_primary));
                FragmentManager fragmentManager = getSupportFragmentManager();
                FragmentTransaction              fragmentTransaction              =
fragmentManager.beginTransaction();
                fragmentTransaction.replace(R.id.fragment, new CourseFragment());
                fragmentTransaction.commit();
            }
        });


        staticsButton.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View view) {
                notice.setVisibility(View.GONE);

courseButton.setBackgroundColor(getResources().getColor(R.color.design_default_color_p
rimary));

staticsButton.setBackgroundColor(getResources().getColor(R.color.design_default_color_p
rimary_dark));

scheduleButton.setBackgroundColor(getResources().getColor(R.color.design_default_color
_primary));
                FragmentManager fragmentManager = getSupportFragmentManager();
                FragmentTransaction              fragmentTransaction              =
fragmentManager.beginTransaction();
                fragmentTransaction.replace(R.id.fragment, new StatisticFragment());
                fragmentTransaction.commit();
            }
        });

        scheduleButton.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View view) {
                    notice.setVisibility(View.GONE);

courseButton.setBackgroundColor(getResources().getColor(R.color.design_default_color_p
rimary));

staticsButton.setBackgroundColor(getResources().getColor(R.color.design_default_color_p
rimary));

scheduleButton.setBackgroundColor(getResources().getColor(R.color.design_default_color
_primary_dark));
                    FragmentManager fragmentManager = getSupportFragmentManager();
                    FragmentTransaction                  fragmentTransaction                  =
fragmentManager.beginTransaction();
                    fragmentTransaction.replace(R.id.fragment, new ScheduleFragment());
                    fragmentTransaction.commit();
                }
            });
        }
}
```

(Android) MemberInfo.java

```
package com.example.card_project;



import android.widget.ScrollView;

public class MemberInfo {
    private String name;
    private String num1;
    private String num2;
    private String rank;
    private String address;
    private String date1;
    private String photoUrl;





    public MemberInfo(String name, String num1, String num2, String rank, String
```

}

```
address, String date1, String photoUrl){
        this.name = name;
        this.num1 = num1;
        this.num2 = num2;
        this.rank = rank;
        this.address = address;
        this.date1 = date1;
        this.photoUrl = photoUrl;


    }

    public MemberInfo(String name, String num1, String num2, String rank, String
address, String date1){
        this.name = name;
        this.num1 = num1;
        this.num2 = num2;
        this.rank = rank;
        this.address = address;
        this.date1 = date1;


    }




    public String getName(){
        return name;
    }
    public void setName(String name){
        this.name = name;
    }
    public String getNum1(){
        return num1;
    }
    public void setNum1(String num1){
        this.num1 = num1;
    }
    public String getNum2(){
        return num2;
    }
    public void setNum2(String num2){
        this.num2 = num2;
    }
    public String getRank(){
        return rank;
    }
    public void setRank(String rank){
        this.rank = rank;
    }
    public String getAddress(){
        return address;
    }
    public void setAddress(String address){
        this.address = address;
    }
    public String getDate1(){
        return date1;
    }
    public void setDate1(String date1){
        this.date1 = date1;
    }
    public String getPhotoUrl(){
        return photoUrl;
    }
    public void setPhotoUrl(String photoUrl){
        this.photoUrl = photoUrl;
    }


}
```

(Android) MemberInitActivity.java

```
package com.example.card_project;

import android.app.Activity;
import android.content.Intent;

import android.graphics.Bitmap;
import android.graphics.BitmapFactory;
import android.net.Uri;
import android.os.Bundle;
import androidx.annotation.NonNull;
import androidx.appcompat.app.AppCompatActivity;


import android.provider.MediaStore;
import android.util.Log;
import android.view.View;
import android.widget.EditText;
import android.widget.ImageView;
import android.widget.RelativeLayout;
import android.widget.Toast;


import com.bumptech.glide.Glide;
import com.example.card_project.MemberInfo;
import com.example.card_project.R;

import com.google.android.gms.tasks.Continuation;
import com.google.android.gms.tasks.OnCompleteListener;
import com.google.android.gms.tasks.OnFailureListener;

import com.google.android.gms.tasks.OnSuccessListener;
import com.google.android.gms.tasks.Task;
import com.google.firebase.auth.FirebaseAuth;
import com.google.firebase.auth.FirebaseUser;

import com.google.firebase.auth.UserProfileChangeRequest;
import com.google.firebase.firestore.DocumentReference;
import com.google.firebase.firestore.DocumentSnapshot;
import com.google.firebase.firestore.FirebaseFirestore;
import com.google.firebase.firestore.auth.User;
import com.google.firebase.storage.FirebaseStorage;
import com.google.firebase.storage.StorageReference;
import com.google.firebase.storage.UploadTask;

import static com.example.card_project.Util.INTENT_PATH;
import static com.example.card_project.Util.showToast;

import java.io.File;
import java.io.FileInputStream;
import java.io.FileNotFoundException;
import java.io.InputStream;



public class MemberInitActivity extends AppCompatActivity {
    private static final String TAG = "MemberInitActivity";

    private ImageView profileImageView;
    private FirebaseStorage storage;

    private String profilePath;
    private FirebaseUser user;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_member);
        profileImageView = findViewById(R.id.profileImageView);

        findViewById(R.id.profileImageView).setOnClickListener(onClickListener);

        profileImageView = findViewById(R.id.profileImageView);
        storage = FirebaseStorage.getInstance();

        profileImageView.setOnClickListener(onClickListener);
        findViewById(R.id.OK).setOnClickListener(onClickListener);

    }

    @Override
    public void onBackPressed() {
        super.onBackPressed();
        finish();
    }
    @Override
    public void onActivityResult(int requestCode, int resultCode, Intent data) {
        super.onActivityResult(requestCode, resultCode, data);
        switch (requestCode) {
            case 0: {
                if (resultCode == Activity.RESULT_OK) {
                    profilePath = data.getStringExtra(INTENT_PATH);
Glide.with(this).load(profilePath).centerCrop().override(500).into(profileImageView);
                }
                break;
            }
        }
    }



    View.OnClickListener onClickListener = v -> {
        switch (v.getId()) {
            case R.id.profileImageView:
                myStartActivity(GalleryActivity.class);
                break;
            case R.id.OK:
                profileUpdate();
                break;
        }

    };




    private void profileUpdate() {
        final String name = ((EditText) findViewById(R.id.username)).getText().toString();
        final String num1 = ((EditText) findViewById(R.id.usernum1)).getText().toString();
        final String num2 = ((EditText) findViewById(R.id.usernum2)).getText().toString();
        final String rank = ((EditText) findViewById(R.id.rankG)).getText().toString();
        final String address = ((EditText) findViewById(R.id.address)).getText().toString();
        final String date1 = ((EditText) findViewById(R.id.date)).getText().toString();




        if (name.length() > 0 && num1.length() > 0 && num2.length() > 0 &&
rank.length() > 0 && address.length() >0 && date1.length()>0) {
            FirebaseStorage storage = FirebaseStorage.getInstance();
            StorageReference storageRef = storage.getReference();
            user = FirebaseAuth.getInstance().getCurrentUser();
            final StorageReference mountainImagesRef = storageRef.child("users/" +
user.getUid() + "/profileImage.jpg");

            if (profilePath == null) {
                MemberInfo memberInfo = new MemberInfo(name, num1, num2, rank,
address, date1);
                storeUploader(memberInfo);
            } else {
                try {
                    InputStream stream = new FileInputStream(new File(profilePath));
                    UploadTask uploadTask = mountainImagesRef.putStream(stream);
                    uploadTask.continueWithTask(new
Continuation<UploadTask.TaskSnapshot, Task<Uri>>() {
                        @Override
                        public                 Task<Uri>                 then(@NonNull
Task<UploadTask.TaskSnapshot> task) throws Exception {
                            if (!task.isSuccessful()) {
                                throw task.getException();
                            }
                            return mountainImagesRef.getDownloadUrl();
                        }
                    }).addOnCompleteListener(new OnCompleteListener<Uri>() {
                        @Override
                        public void onComplete(@NonNull Task<Uri> task) {
                            if (task.isSuccessful()) {
                                Uri downloadUri = task.getResult();

                                MemberInfo    memberInfo    =    new    MemberInfo(name,
num1, num2, rank, address, date1, downloadUri.toString());
                                storeUploader(memberInfo);
                            } else {
                                showToast(MemberInitActivity.this, "회원정보를 보내는데
실패하였습니다.");
                            }
                        }
                    });
                } catch (FileNotFoundException e) {
                    Log.e("로그", "에러: " + e.toString());
                }
            }
        } else {
            showToast(MemberInitActivity.this, "복지카드정보를 입력해주세요.");
        }
    }

    private void storeUploader(MemberInfo memberInfo) {
        FirebaseFirestore db = FirebaseFirestore.getInstance();
        db.collection("users").document(user.getUid()).set(memberInfo)
                .addOnSuccessListener(new OnSuccessListener<Void>() {
                    @Override
                    public void onSuccess(Void aVoid) {
                        showToast(MemberInitActivity.this, "회원정보 등록을 성공하였습니
다.");
                        finish();
                    }
                })
                .addOnFailureListener(new OnFailureListener() {
                    @Override
                    public void onFailure(@NonNull Exception e) {
                        showToast(MemberInitActivity.this, "회원정보 등록에 실패하였습니
다.");
                        Log.w(TAG, "Error writing document", e);
                    }
                });
    }


    private void startToast(String msg){
        Toast.makeText(this, msg,Toast.LENGTH_SHORT).show();
    }

    private void myStartActivity(Class c) {
        Intent intent = new Intent(this, c);
        startActivityForResult(intent, 0);

    }

}
```

(Android) PasswordResetActivity.java

```
package com.example.card_project;


import android.content.Intent;
import android.os.Bundle;
import android.util.Log;
import android.view.View;
import android.widget.EditText;
import android.widget.Toast;

import androidx.annotation.NonNull;
import androidx.appcompat.app.AppCompatActivity;

import com.example.card_project.R;
import com.google.android.gms.tasks.OnCompleteListener;
import com.google.android.gms.tasks.Task;
import com.google.firebase.auth.AuthResult;
import com.google.firebase.auth.FirebaseAuth;
import com.google.firebase.auth.FirebaseUser;

public class PasswordResetActivity extends AppCompatActivity {
    private static final String TAG = "PasswordResetActivity";
    private FirebaseAuth mAuth;



    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_password_reset);
        mAuth = FirebaseAuth.getInstance();
        findViewById(R.id.sendButton).setOnClickListener(onClickListener);

    }



    View.OnClickListener onClickListener = v -> {
        switch (v.getId()) {
            case R.id.sendButton:
                send();
                break;
        }
    };



    private void send() {
        String email = ((EditText) findViewById(R.id.user_id)).getText().toString();

        if (email.length() > 0) {
            mAuth.sendPasswordResetEmail(email)
                    .addOnCompleteListener(new OnCompleteListener<Void>() {
                        @Override
                        public void onComplete(@NonNull Task<Void> task) {
                            if (task.isSuccessful()) {
                                startToast("이메일을 보냈습니다.");

                            }
                        }

                    });

        } else {
            startToast("이메일 또는 비밀번호를 입력해 주세요.");
        }


    }

    private void startToast(String msg) {
        Toast.makeText(this, msg, Toast.LENGTH_SHORT).show();
    }


}
```

(Android) ScheduleFragment.java

```
package com.example.card_project;

import android.os.Bundle;

import androidx.fragment.app.Fragment;

import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;
```

```
/**
 * A simple {@link Fragment} subclass.
 * Use the {@link ScheduleFragment#newInstance} factory method to
 * create an instance of this fragment.
 */
public class ScheduleFragment extends Fragment {

    // TODO: Rename parameter arguments, choose names that match
    // the fragment initialization parameters, e.g. ARG_ITEM_NUMBER
    private static final String ARG_PARAM1 = "param1";
    private static final String ARG_PARAM2 = "param2";

    // TODO: Rename and change types of parameters
    private String mParam1;
    private String mParam2;

    public ScheduleFragment() {
        // Required empty public constructor
    }

    /**
     * Use this factory method to create a new instance of
     * this fragment using the provided parameters.
     *
     * @param param1 Parameter 1.
     * @param param2 Parameter 2.
     * @return A new instance of fragment ScheduleFragment.
     */
    // TODO: Rename and change types and number of parameters
    public static ScheduleFragment newInstance(String param1, String param2) {
        ScheduleFragment fragment = new ScheduleFragment();
        Bundle args = new Bundle();
        args.putString(ARG_PARAM1, param1);
        args.putString(ARG_PARAM2, param2);
        fragment.setArguments(args);
        return fragment;
    }

    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        if (getArguments() != null) {
            mParam1 = getArguments().getString(ARG_PARAM1);
            mParam2 = getArguments().getString(ARG_PARAM2);
        }
    }

    @Override
    public View onCreateView(LayoutInflater inflater, ViewGroup container,
                             Bundle savedInstanceState) {
        // Inflate the layout for this fragment
        return inflater.inflate(R.layout.fragment_schedule, container, false);
    }
}
```

(Android) SignUpActivity.java

```
package com.example.card_project;

import android.annotation.SuppressLint;
import android.app.Dialog;
import android.content.Context;
import android.content.Intent;
import android.os.Bundle;
import android.util.Log;
import android.view.View;
import android.view.WindowManager;
import android.widget.Button;
import android.widget.EditText;
import android.widget.Toast;

import androidx.annotation.NonNull;
import androidx.appcompat.app.AppCompatActivity;

import com.google.android.gms.tasks.OnCompleteListener;
import com.google.android.gms.tasks.Task;
import com.google.firebase.auth.AuthResult;
import com.google.firebase.auth.FirebaseAuth;
import com.google.firebase.auth.FirebaseUser;
```

```
public class SignUpActivity extends AppCompatActivity {
    private static final String TAG = "SignUpActivity";
    private FirebaseAuth mAuth;
    private AuthemailDialog authemailDialog;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_sign_up);

        // Initialize Firebase Auth
        mAuth = FirebaseAuth.getInstance();
        authemailDialog = new AuthemailDialog(this, positiveListener);
        findViewById(R.id.check).setOnClickListener(onClickListener);
        findViewById(R.id.loginButton).setOnClickListener(onClickListener);
    }




    View.OnClickListener onClickListener = new View.OnClickListener() {
        @SuppressLint("NonConstantResourceId")
        @Override
        public void onClick(View view) {
            switch (view.getId()){

                case R.id.check:
                    signUp();
                    break;
                case R.id.loginButton:
                    startLoginActivity();
                    break;
            }
        }
    };

    private View.OnClickListener positiveListener = new View.OnClickListener() {
        public void onClick(View v) {
            // 다이얼로그 종료
            authemailDialog.dismiss();
            // 로그인 화면으로 이동
            Intent intent = new Intent(getApplicationContext(), LoginActivity.class);
```

```
                startActivity(intent);
            }
        };



        private void signUp() {
            String email = ((EditText)findViewById(R.id.user_id)).getText().toString();
            String                                password                                =
((EditText)findViewById(R.id.user_password)).getText().toString();
            String                             passwordCheck                              =
((EditText)findViewById(R.id.user_password_check)).getText().toString();


            if (email.length() > 0 && password.length() > 0 && passwordCheck.length() > 0)
            {

                if(password.equals(passwordCheck)){
                    mAuth.createUserWithEmailAndPassword(email, password)
                            .addOnCompleteListener(this,                              new
OnCompleteListener<AuthResult>() {
                                @Override
                                public void onComplete(@NonNull Task<AuthResult> task) {
                                    if (task.isSuccessful()) {
                                        FirebaseUser user = mAuth.getCurrentUser();

user.sendEmailVerification().addOnCompleteListener(new OnCompleteListener<Void>() {
                                            @Override
                                            public   void   onComplete(@NonNull   Task<Void>
task) {
                                                Toast.makeText(SignUpActivity.this,"메일  발
송완료", Toast.LENGTH_SHORT).show();

                                                authemailDialog.show();
                                            }
                                        });

                                        //성공했을 때 UI로직

                                    } else {
                                        if(task.getException() != null){
                                            // If sign in fails, display a message to the
user.
                                            startToast(task.getException().toString());
                                            //실패했을 때 UI로직
                                        }
                                    }
                                }
                            });
                }else{
                    startToast("비밀번호가 일치하지 않습니다.");
                }
            }else{
                startToast("입력한 정보를 다시 한 번 확인해주세요.");
            }

        }

        public static class AuthemailDialog extends Dialog {

            private Button positivebutton;
            private View.OnClickListener positiveListener;

            public    AuthemailDialog(@NonNull    Context    context,    View.OnClickListener
positiveListener) {

                super(context);
                this.positiveListener = positiveListener;
            }
            @Override
            protected void onCreate(Bundle savedInstanceState) {
                super.onCreate(savedInstanceState);

                // 다이얼로그 밖의 화면은 흐리게 함
                WindowManager.LayoutParams          layoutParams          =          new
WindowManager.LayoutParams();
                layoutParams.flags = WindowManager.LayoutParams.FLAG_DIM_BEHIND;
                layoutParams.dimAmount = 0.8f;
                getWindow().setAttributes(layoutParams);
                setContentView(R.layout.authemail_dialog);

                // 확인 버튼
                positivebutton = findViewById(R.id.positivebutton);
                // 클릭 리스너
```

```
                positivebutton.setOnClickListener(positiveListener);
            }
        }

        private void startToast(String msg){
            Toast.makeText(this, msg, Toast.LENGTH_SHORT).show();
        }
        private void startLoginActivity(){
            Intent intent = new Intent(this, LoginActivity.class);
            startActivity(intent);
        }
}
```

(Android) StatisticFragment.java

```
package com.example.card_project;



import android.os.Bundle;

import androidx.fragment.app.Fragment;

import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;

/**
 * A simple {@link Fragment} subclass.
 * Use the {@link StatisticFragment#newInstance} factory method to
 * create an instance of this fragment.
 */
public class StatisticFragment extends Fragment {

    // TODO: Rename parameter arguments, choose names that match
    // the fragment initialization parameters, e.g. ARG_ITEM_NUMBER
    private static final String ARG_PARAM1 = "param1";
    private static final String ARG_PARAM2 = "param2";

    // TODO: Rename and change types of parameters
    private String mParam1;
    private String mParam2;

    public StatisticFragment() {
        // Required empty public constructor
    }

    /**
     * Use this factory method to create a new instance of
     * this fragment using the provided parameters.
     *
     * @param param1 Parameter 1.
     * @param param2 Parameter 2.
     * @return A new instance of fragment StatisticFragment.
     */
    // TODO: Rename and change types and number of parameters
    public static StatisticFragment newInstance(String param1, String param2) {
        StatisticFragment fragment = new StatisticFragment();
        Bundle args = new Bundle();
        args.putString(ARG_PARAM1, param1);
        args.putString(ARG_PARAM2, param2);
        fragment.setArguments(args);
        return fragment;
    }

    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        if (getArguments() != null) {
            mParam1 = getArguments().getString(ARG_PARAM1);
            mParam2 = getArguments().getString(ARG_PARAM2);
        }
    }

    @Override
    public View onCreateView(LayoutInflater inflater, ViewGroup container,
                             Bundle savedInstanceState) {
        // Inflate the layout for this fragment
        return inflater.inflate(R.layout.fragment_statistic, container, false);
    }
}
```

(Android) UserInfoFragment.java

```
package com.example.card_project;

import android.content.Context;
import android.net.Uri;
import android.os.Bundle;
import android.util.Log;
import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;
import android.widget.ImageView;
import android.widget.TextView;
import android.widget.Toast;

import androidx.annotation.NonNull;
import androidx.fragment.app.Fragment;

import com.bumptech.glide.Glide;
import com.google.android.gms.tasks.OnCompleteListener;
import com.google.android.gms.tasks.OnFailureListener;
import com.google.android.gms.tasks.OnSuccessListener;
import com.google.android.gms.tasks.Task;
import com.google.firebase.auth.FirebaseAuth;
import com.google.firebase.firestore.DocumentReference;
import com.google.firebase.firestore.DocumentSnapshot;
import com.google.firebase.firestore.FirebaseFirestore;
import com.google.firebase.storage.FirebaseStorage;
import com.google.firebase.storage.StorageReference;

import org.w3c.dom.Text;

public class UserInfoFragment extends Fragment {
    private static final String TAG ="UserInfoFragment";

    public UserInfoFragment(){

    }
    @Override
    public void onCreate(Bundle savedInstanceState){
        super.onCreate(savedInstanceState);
    }
    @Override
    public View onCreateView(LayoutInflater inflater, ViewGroup container, Bundle
savedInstanceState){
        View view = inflater.inflate(R.layout.fragment_user_info,container, false);

        final ImageView profileImageView = view.findViewById(R.id.profileImageView);
        final TextView nametxt = view.findViewById(R.id.nameText);
        final TextView num1txt = view.findViewById(R.id.num1Text);
        final TextView num2txt = view.findViewById(R.id.num2Text);
        final TextView ranktxt = view.findViewById(R.id.rankText);
        final TextView addresstxt = view.findViewById(R.id.addressText);
        final TextView datetxt = view.findViewById(R.id.dateText);


        DocumentReference documentReference =
FirebaseFirestore.getInstance().collection("users").document(FirebaseAuth.getInstance().ge
tCurrentUser().getUid());
        documentReference.get().addOnCompleteListener(new
OnCompleteListener<DocumentSnapshot>() {


            @Override
            public void onComplete(@NonNull Task<DocumentSnapshot> task) {

                if (task.isSuccessful()) {
                    DocumentSnapshot document = task.getResult();
                    if (document != null) {
                        if (document.exists()) {
                            Log.d(TAG, "DocumentSnapshot data: " +
document.getData());
                            if(document.getData().get("photoUrl") != null){

Glide.with(getActivity()).load(document.getData().get("photoUrl")).centerCrop().override(500
).into(profileImageView);
                            }
                            nametxt.setText(document.getData().get("name").toString());
                            num1txt.setText(document.getData().get("num1").toString());
                            num2txt.setText(document.getData().get("num2").toString());
                            ranktxt.setText(document.getData().get("rank").toString());

addresstxt.setText(document.getData().get("address").toString());
```

```
                                datetxt.setText(document.getData().get("date1").toString());

                            } else {
                                Log.d(TAG, "No such document");
                            }
                        }
                    } else {
                        Log.d(TAG, "get failed with ", task.getException());
                    }
                }
        });

        return view;
    }

    @Override
    public void onAttach(Context context) {
        super.onAttach(context);
    }

    @Override
    public void onDetach() {
        super.onDetach();
    }

    @Override
    public void onPause(){
        super.onPause();
    }

}
```

(Android) Util.java

```
package com.example.card_project;


import android.app.Activity;
import android.util.Patterns;
import android.widget.Toast;
import java.net.URLConnection;

public class Util {
    public Util(){/* */}

    public static final String INTENT_PATH = "path";
    public static final String INTENT_MEDIA = "media";

    public static final int GALLERY_IMAGE = 0;
    public static final int GALLERY_VIDEO = 1;

    public static void showToast(Activity activity, String msg){
        Toast.makeText(activity, msg, Toast.LENGTH_SHORT).show();
    }

    public static boolean isStorageUrl(String url){
        return                    Patterns.WEB_URL.matcher(url).matches()                    &&
url.contains("https://firebasestorage.googleapis.com/v0/b/sns-project-3e2c2.appspot.co
m/o/post");
    }

    public static String storageUrlToName(String url){
        return url.split("\\?")[0].split("%2F")[url.split("\\?")[0].split("%2F").length - 1];
    }

    public static boolean isImageFile(String path) {
        String mimeType = URLConnection.guessContentTypeFromName(path);
        return mimeType != null && mimeType.startsWith("image");
    }

    public static boolean isVideoFile(String path) {
        String mimeType = URLConnection.guessContentTypeFromName(path);
        return mimeType != null && mimeType.startsWith("video");
    }
}
```

(Android) MapsActivity

```
package com.example.card_project;

import androidx.fragment.app.FragmentActivity;
import android.os.Bundle;

import com.google.android.gms.maps.CameraUpdateFactory;
import com.google.android.gms.maps.GoogleMap;
import com.google.android.gms.maps.OnMapReadyCallback;
import com.google.android.gms.maps.SupportMapFragment;
import com.google.android.gms.maps.model.LatLng;
import com.google.android.gms.maps.model.MarkerOptions;
import com.example.card_project.databinding.ActivityMapsBinding;

public class MapsActivity extends FragmentActivity implements OnMapReadyCallback {

    private GoogleMap mMap;
    private ActivityMapsBinding binding;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);

        binding = ActivityMapsBinding.inflate(getLayoutInflater());
        setContentView(binding.getRoot());

        // Obtain the SupportMapFragment and get notified when the map is ready
to be used.
        SupportMapFragment mapFragment = (SupportMapFragment)
getSupportFragmentManager()
                .findFragmentById(R.id.map);
        mapFragment.getMapAsync(this);
    }

    /**
     * Manipulates the map once available.
     * This callback is triggered when the map is ready to be used.
     * This is where we can add markers or lines, add listeners or move the
camera. In this case,
     * we just add a marker near Sydney, Australia.
     * If Google Play services is not installed on the device, the user will be
prompted to install
     * it inside the SupportMapFragment. This method will only be triggered once the
user has
     * installed Google Play services and returned to the app.
     */
    @Override
    public void onMapReady(GoogleMap googleMap) {
        mMap = googleMap;

        // Add a marker in Sydney and move the camera
        LatLng sydney = new LatLng(-34, 151);
        mMap.addMarker(new      MarkerOptions().position(sydney).title("Marker      in
Sydney"));
        mMap.moveCamera(CameraUpdateFactory.newLatLng(sydney));
    }
}
```

5.2 발표자료

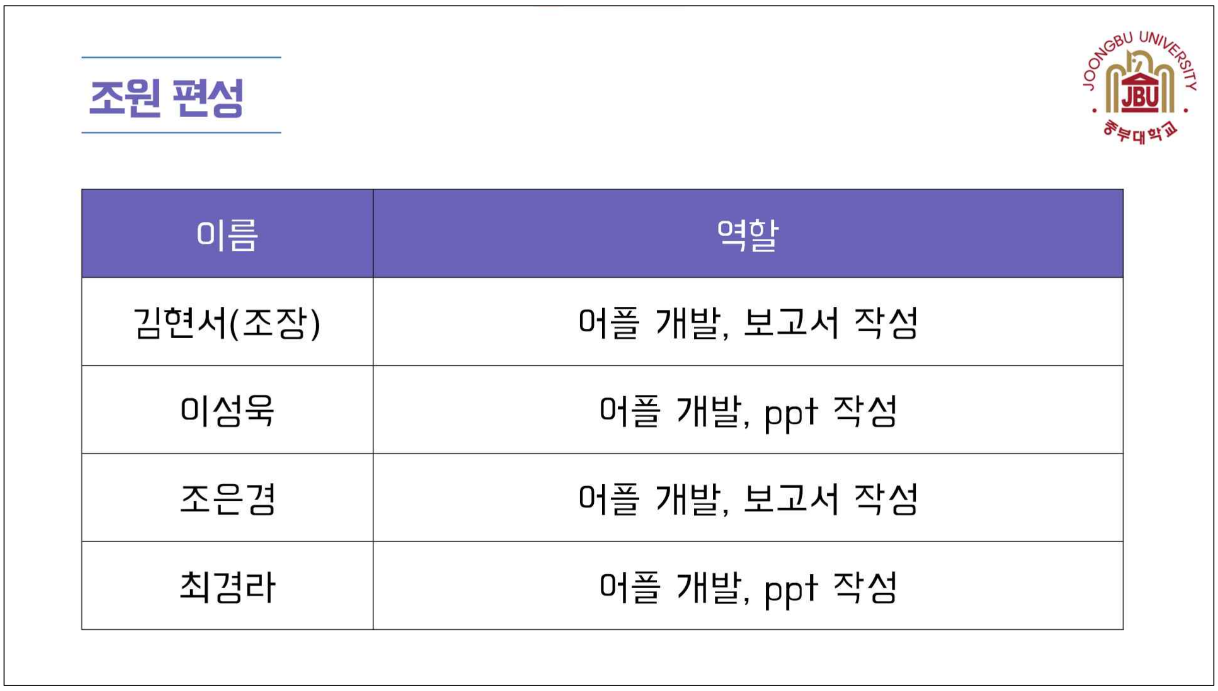
모바일 복지카드 서비스

2021.11.03

중부대학교 정보보호학과
지도교수 : 이병천 교수님

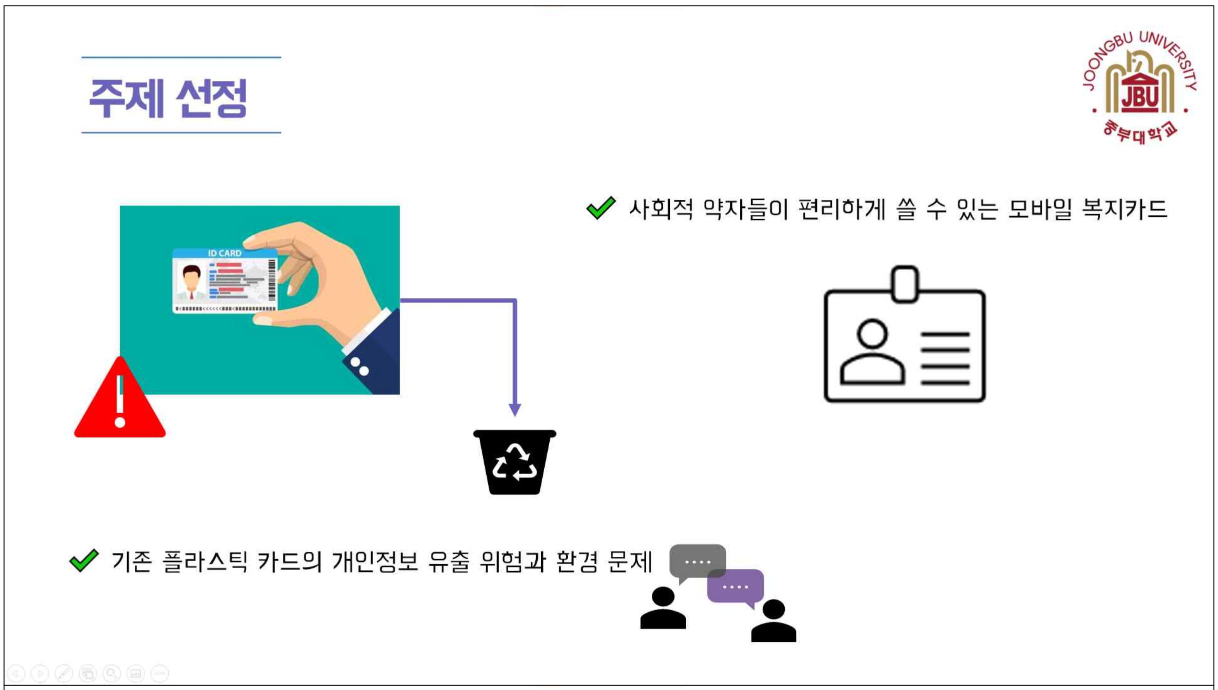
Blockchallenge : 김현서
이성욱
조은경
최경라

CONTENTS



조원 편성

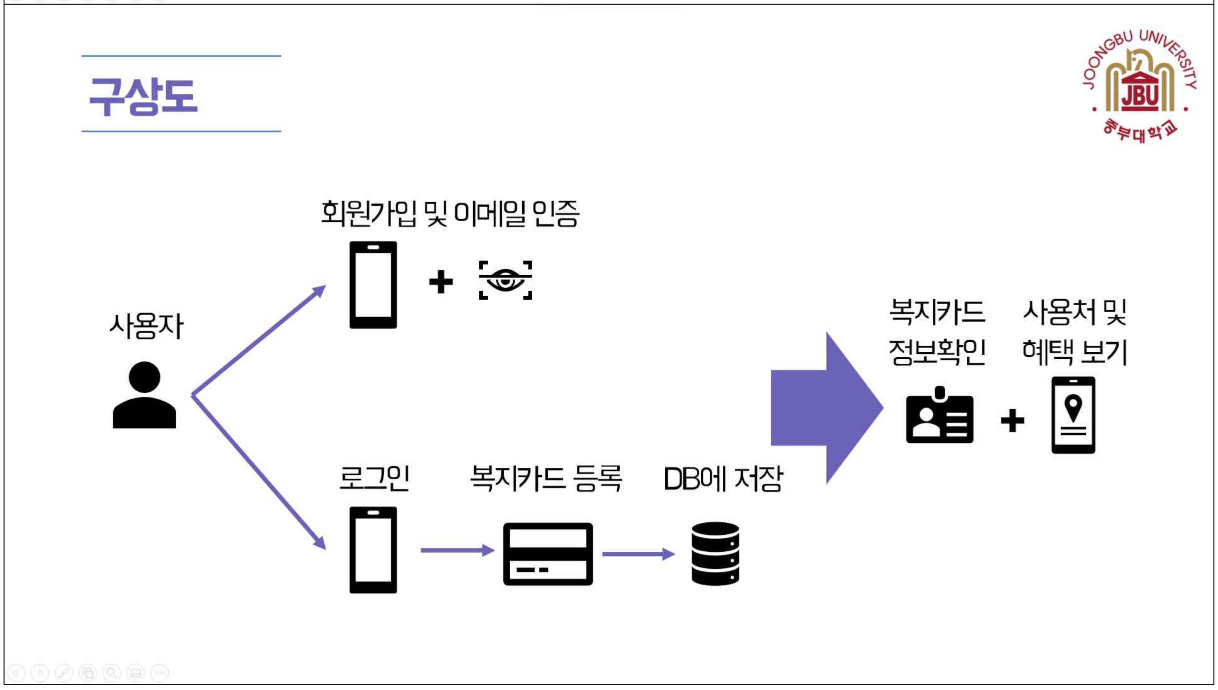
이름	역할
김현서(조장)	어플 개발, 보고서 작성
이성욱	어플 개발, ppt 작성
조은경	어플 개발, 보고서 작성
최경라	어플 개발, ppt 작성



주제 선정

✓ 사회적 약자들이 편리하게 쓸 수 있는 모바일 복지카드

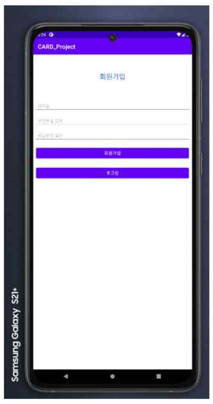
✓ 기존 플라스틱 카드의 개인정보 유출 위험과 환경 문제

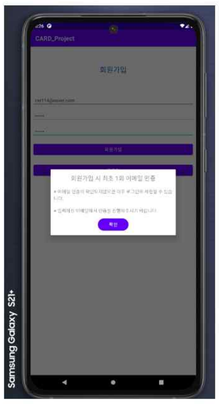


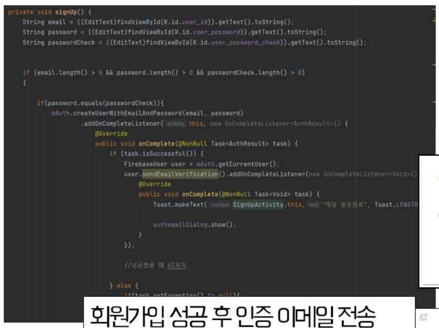
구상도

개발 내용

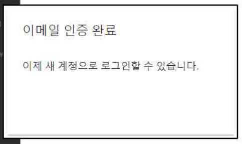
안드로이드 애플리케이션 회원가입, 이메일 인증







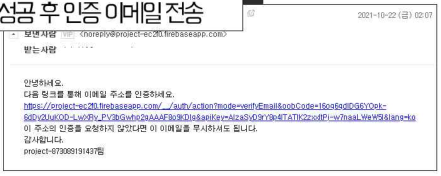
```
private void signUp() {
    String email = ((EditText)findViewById(R.id.user_id)).getText().toString();
    String password = ((EditText)findViewById(R.id.user_password)).getText().toString();
    String passwordCheck = ((EditText)findViewById(R.id.user_password_check)).getText().toString();


    if (email.length() > 0 && password.length() > 0 && passwordCheck.length() > 0)
    {

        if(password.equals(passwordCheck)){
            mAuth.createUserWithEmailAndPassword(email, password)
                    .addOnCompleteListener( activity: this, new OnCompleteListener<AuthResult>() {
                        @Override
                        public void onComplete(@NonNull Task<AuthResult> task) {
                            if (task.isSuccessful()) {
                                FirebaseUser user = mAuth.getCurrentUser();
                                user.sendEmailVerification().addOnCompleteListener(new OnCompleteListener<Void>()
                                    @Override
                                    public void onComplete(@NonNull Task<Void> task) {
                                        Toast.makeText( context: SignUpActivity.this, text: "메일 발송완료", Toast.LENGTH

                                        authemailDialog.show();
                                    }
                                });

                                //성공했을 때 UI로직

                            } else {
                                if(task.getException() != null){
```



이메일 인증 완료

이제 새 계정으로 로그인할 수 있습니다.

회원가입 성공 후 인증 이메일 전송



2021-10-22 (금) 02:07

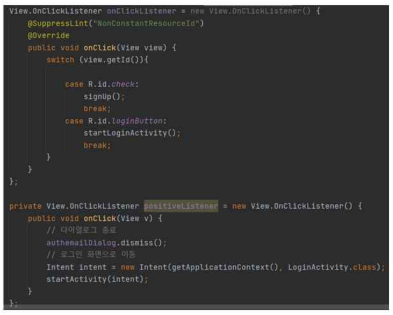
보낸사람 VIP <noreply@project-ec2f0.firebaseapp.com>

받는사람

안녕하세요.
다음 링크를 통해 이메일 주소를 인증하세요.
https://project-ec2f0.firebaseapp.com/__/auth/action?mode=verifyEmail&oobCode=16oq6qdIDG6YOpk-6dDy2UuKOD-LwXRy_PV3bGwhp2gAAAF8o9KDIg&apiKey=AIzaSyD9rY8p4ITATIK2zxxltPi-w7naaLWeW5I&lang=ko
이 주소의 인증을 요청하지 않았다면 이 이메일을 무시하셔도 됩니다.
감사합니다.
project-873089191437팀

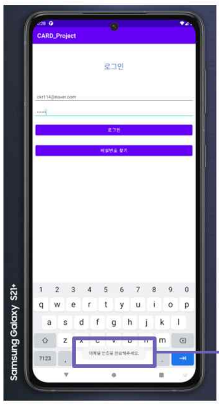
개발 내용

안드로이드 애플리케이션 회원가입, 이메일 인증



```
View.OnClickListener onClickListener = new View.OnClickListener() {
    @SuppressLint("NonConstantResourceId")
    @Override
    public void onClick(View view) {
        switch (view.getId()){

            case R.id.check:
                signUp();
                break;
            case R.id.loginButton:
                startLoginActivity();
                break;
        }
    }
};

private View.OnClickListener positiveListener = new View.OnClickListener() {
    public void onClick(View v) {
        // 다이얼로그 종료
        authemailDialog.dismiss();
        // 로그인 화면으로 이동
        Intent intent = new Intent(getApplicationContext(), LoginActivity.class);
        startActivity(intent);
    }
};
```

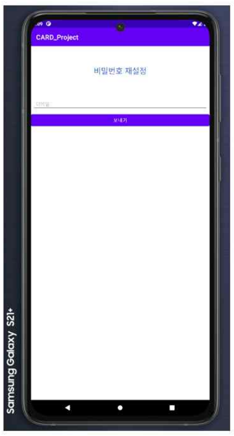
인증메일 전송 후 다이얼로그 종료, 로그인 화면으로 이동

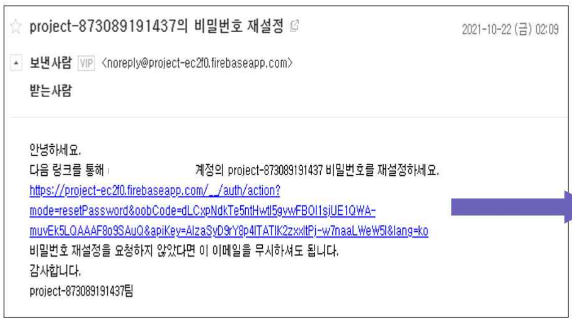


회원가입 후 이메일 인증, 인증이 완료되지 않으면 로그인이 안됨

개발 내용

비밀번호 재설정

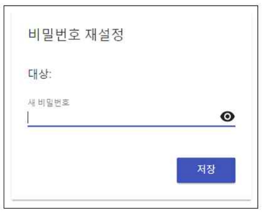



☆ project-873089191437의 비밀번호 재설정 2021-10-22 (금) 02:09

보낸사람 VIP <noreply@project-ec2f0.firebaseapp.com>

받는사람

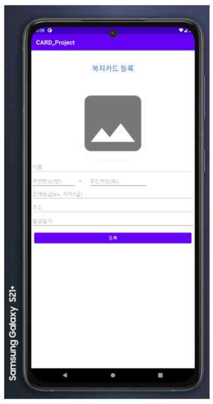
안녕하세요.
다음 링크를 통해 계정의 project-873089191437 비밀번호를 재설정하세요.
https://project-ec2f0.firebaseapp.com/__/auth/action?mode=resetPassword&oobCode=dLCxpNdkTe5ntHwtl5gvwFBOI1sjUE1QWA-muyEk5LQAAAF8o9SAuQ&apiKey=AIzaSyD9rY8p4lTATIK2zxdtPj-w7naaLWeW5I&lang=ko
비밀번호 재설정을 요청하지 않았다면 이 이메일을 무시하셔도 됩니다.
감사합니다.
project-873089191437팀

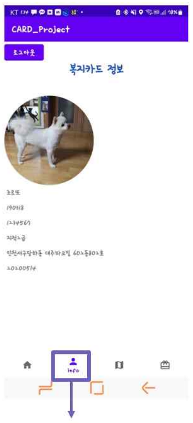


링크를 통해 비밀번호 재설정

개발 내용

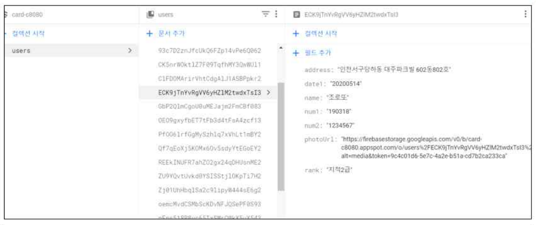
복지 카드 등록





조로또
190318
1234567
지적2급
인천서구당하동 대주파크빌 602동802호
20200514

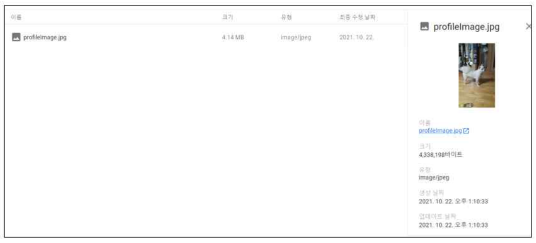
복지 카드 등록 후 카드 정보를 클릭하면 카드 정보가 보인다



users

93c7D2znJfcUkQ6FZp14vPe6Q062
CK5nrW0kt1Z7F09TqfhMY3QwWU11
C1FDOMArirVhtCdgA1J1ASBPpkr2
ECK9jTnYvRgVV6yHZ1M2twdxTsI3
GbP2Q1mCgoU0uMEJajm2FmCBf083
OE09gxyfbET7tFb3d4tFaA4zcf13
PfOO61rfGgMySzh1q7xVhLt1mBY2
Qf7qEoXj5KOMx60v5sdyYtEGoEY2
REEkINUFR7ahZO2gx24qDHUsnME2
ZU9YQvtUvkd0YSISStj1OKpTi7H2
Zj01UhHbq1Sa2c91ipyW444sE6g2
oemcMvdCSMbScKDvNFJQSePF0S93

address: "인천서구당하동 대주파크빌 602동802호"
date1: "20200514"
name: "조로또"
num1: "190318"
num2: "1234567"
photoUrl: "https://firebasestorage.googleapis.com/v0/b/card-c8080.appspot.com/o/users%2FECK9jTnYvRgVV6yHZIM2twdxTsI3%2 alt=media&token=9c4c01d6-5e7c-4a2e-b51a-cd7b2ca233ca"
rank: "지적2급"



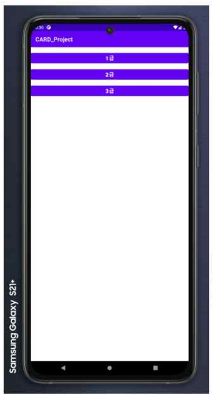
이름	크기	유형	최종 수정 날짜
profileImage.jpg	4.14 MB	image/jpeg	2021. 10. 22.

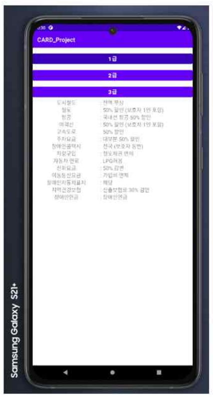
profileImage.jpg
이름
profileImage.jpg
크기
4,338,198바이트
유형
image/jpeg
생성 날짜
2021. 10. 22. 오후 1:10:33
업데이트 날짜
2021. 10. 22. 오후 1:10:33

DB에 복지카드 정보가 저장된 것을 확인

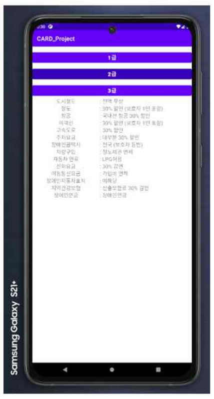
개발 내용

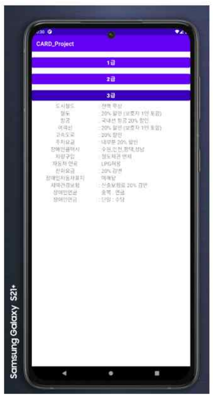
혜택 확인





도시철도	전액 무상
철도	50% 할인 (보호자 1인 포함)
항공	국내선 항공 50% 할인
여객선	50% 할인 (보호자 1인 포함)
고속도로	50% 할인
주차요금	대부분 50% 할인
장애인콜택시	전국 (보호자 동반)
차량구입	철도채권 면제
자동차 연료	LPG허용
전화요금	50% 감면
이동통신요금	가입비 면제
장애인자동차표지	해당
치아건강보험	산출보험료 30% 감면
장애인연금	장애인연금

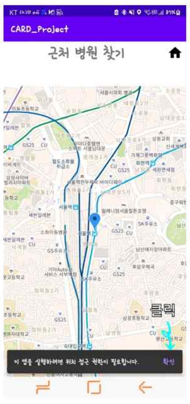




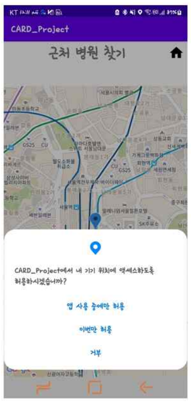
혜택을 클릭하면 등급별로 받을 수 있는 혜택 정보가 보인다

개발 내용

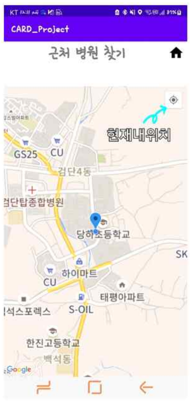
복지 카드 사용가능한 병원 찾기

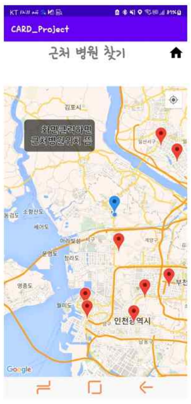


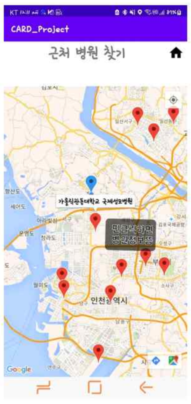
이 앱을 실행하려면 위치 접근 권한이 필요합니다.



CARD_Project에서 내 기기 위치에 액세스하도록 허용하시겠습니까?



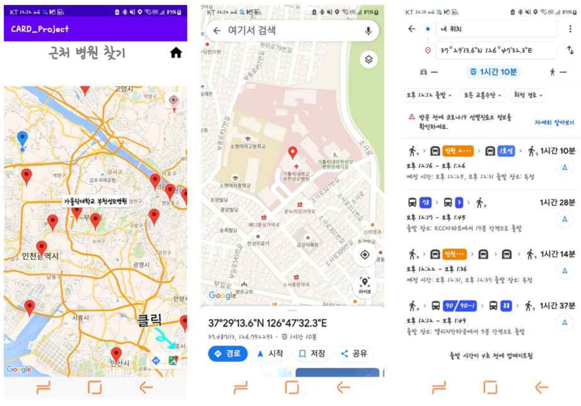




지도로 복지카드를 사용할 수 있는 병원 위치를 확인할 수 있다

개발 내용

복지 카드 사용가능한 병원 찾기



병원까지의 거리와 걸리는 시간을 확인할 수 있다

결과

결론

✓ 항상 휴대하고 있는 핸드폰 하나로 편리하고 카드 분실 위험 없이 사용할 수 있으며
혜택, 지도와 같은 서비스로 복지 카드를 더욱 효율적으로 사용할 수 있다.

기대효과

✓ 기존 카드 등록하는 방법을 보안해 OCR 방식을 이용해 더욱 편리하게 등록을 할 수 있을 것이다.

✓ 혜택을 받을 수 있는 기관에서 QR코드를 스캔하면 사용자의 복지 카드 정보를 띄워주는 형식으로 더욱 편리하게 이용할 수 있을 것이다.

✓ 장애인 진단서와 같은 서류들로 장애인임을 확인하고 이를 블록체인에 응용하면 보다 안전하게 이용할 수 있을 것이다.

Q & A

감사합니다