

종합설계 프로젝트 최종 결과 보고서


프로젝트명	코로나 바이러스 감염증 대응 자가지킴이 출석체크 시스템
팀번호	S5-12
문서제목	수행계획서(O) 2차발표 중간보고서(O) 3차발표 중간보고서(O) 최종결과보고서(O)

2020.12.03

팀원 : 컴퓨터공학과 2016150008 김다인(팀장)

컴퓨터공학과 2014152022 안준영

컴퓨터공학과 2014152017 신현수

지도교수 : 전광일 교수 (인) 

지도교수 : 한경숙 교수 (인)

지도교수 : 최종필 교수 (인)

문서 수정 내역

작성일	대표작성자	버전(Revision)	수정내용	
2020.9.20	김다인(팀장)	1.0	수행계획서	최초작성
2020.10.23	김다인(팀장)	2.0	2차발표자료	설계서추가
2020.12.03	김다인(팀장)	3.0	최종결과보고서	추가작성및수정

문서 구성

진행단계	프로젝트 계획서 발표	중간발표1	중간발표2 (4월)	학기말발표 (6월)	최종발표 (10월)
기본양식	계획서 양식	계획서 양식	계획서 양식	계획서 양식	계획서 양식
포함되는 내용	I. 서론 (1~6)	I. 서론 (1~6)	I. 서론 (1~6)	I. 서론 (1~6)	I II III
	II. 본론 (1~3)	II. 본론 (1~4)	II. 본론 (1~5)	II. 본론 (1~7)	
	참고자료	참고자료	참고자료	참고자료	

이 문서는 한국산업기술대학교 컴퓨터공학부의
 “종합설계” 교과목에서 프로젝트 “코로나 바이러스 감염증 대응 자가지킴이
 출석체크 시스템”을 수행하는
 (S5-12, 김다인, 안준영, 신현수)들이 작성한 것으로 사용하기
 위해서는 팀원들의 허락이 필요합니다.

목 차

I. 서론

1. 작품선정 배경 및 필요성
2. 기존 연구/기술동향 분석
3. 개발 목표
4. 팀 역할 분담
5. 개발 일정
6. 개발 환경

II. 본론

1. 개발 내용
2. 문제 및 해결방안
3. 시험시나리오
4. 상세 설계
5. Prototype 구현
6. 시험/ 테스트 결과
7. Coding & DEMO

III. 결론

1. 연구 결과
2. 작품제작 소요재료 목록

참고자료

I. 서론

1. 작품선정 배경 및 필요성

2019년 12월부터 현재까지 전 세계에 지속되는 범유행전염병이 우리 생활에 깊숙하게 자리 잡음으로써 기존에 존재하는 출석체크 만으로는 코로나에 대처하기 쉽지 않다고 판단하였다. 대면 수업시 증상 유무를 매 강의 시간마다 수기로 작성하는 것은 번거롭다고 느껴지며 체온 체크와 기록으로 발열 증상 유무를 알 수 있는 것도 또한 필요하다고 여겨진다. 즉 코로나 19 감염 확산의 장기화에 따른 대면 수업을 받는 학생들에게 필요한 자가 진단, 체온 기록, 긴급 연락처, 확진자정보 등을 제공하는 맞춤형 출석 체크 어플이 필요하다.

2. 기존 연구/기술동향 분석

1. 코로나 19 경남 : 경남 코로나 19 확진자 동선 및 공적 마스크 재고 알리미
기능으로 경남 지역 속 코로나맵, 마스크맵 등 지도를 통해 한 눈에 정보를 알 수 있는 어플, 확진자 동선 정보는 2주 간의 동선 정보만 표기, 심평원 및 정보화진흥원(공공데이터포털)데이터를 활용
2. 코로나 19 지침 검색 : 신속한 현장 대응을 지원하기 위한 모바일 앱으로
유관기관과 시민들은 빠르게 변화하는 정부의 코로나 19 지침에 대해 확인 가능, 중앙방역대책본부 등 정부에서 제공하는 코로나 19 대응 관련 30여개 지침들을 바탕으로 코로나 19 대응 현장에서 필요한 대상자 분류 정의, 의사 및 확진환자 발생시 대응, 격리해제 기준, 검체 채취, 소독, 폐기물처리, 개인정보구 사용 등 주제별 대응 지침들이 통합 제공
3. 코로나 19 꼼짝마! : 인천 광역시 교육청에서 제작한 코로나 19 건강관리
앱으로 학생, 학부모, 교직원 누구나 코로나 19에 대한 건강관리, 대응, 보건교육 등 자기 건강 관리를 위해 이용 가능

3. 개발 목표

1. 수기로 작성하는 것보다 정확하고 빠른 어플리케이션을 통한 자가 체크
2. 자가진단(의심 증상)체크 시 모두 무증상이어야 출석 체크 진행 가능
3. 체온계를 카메라로 찍어 온도 입력 및 저장을 통한 발열 체크
4. 강의실 좌석에 부착된 QR코드 스캔을 이용한 빠르고 정확한 출석 체크
5. 어플 내 긴급 연락처를 이용한 신고 및 상담 가능

4. 업무 분담

	안준영	신현수	김다인
자료 탐색	Firebase 자료 탐색	DB관련 자료 탐색	출석 체크 어플 자료 탐색
설계	전체적인 어플 구성 데이터베이스 연동	서버 구축 데이터베이스 관리	자가 체크 기능 관리자 기능
구현	Firebase 연동 QR 코드 기능 연결 코드 보안	안드로이드 스튜디오 활용 서버 관리 코드 보안	관리자 기능 코드 보안
테스트	Firebase 연동 테스트, 통합 테스트/유지 보수		

5. 개발 일정

추진사항	7월	8월초	8월말	9월초	9월말	10월	11월	12월
사전 조사 및 계획서 발표	■							
자료 수집 및 분석	■	■						
서버 구축 및 설계		■	■					
자바를 이용한 s/w 설계			■	■	■			
통합 시스템 구현				■	■	■		
테스트 및 유지보수						■	■	
최종보고서 작성 및 최적화								■

6. 개발 환경

OS	Windows 10
IDE	Android Studio
Language	JAVA, Kotlin
Database	Firebase
Server	Firebase

II. 본론

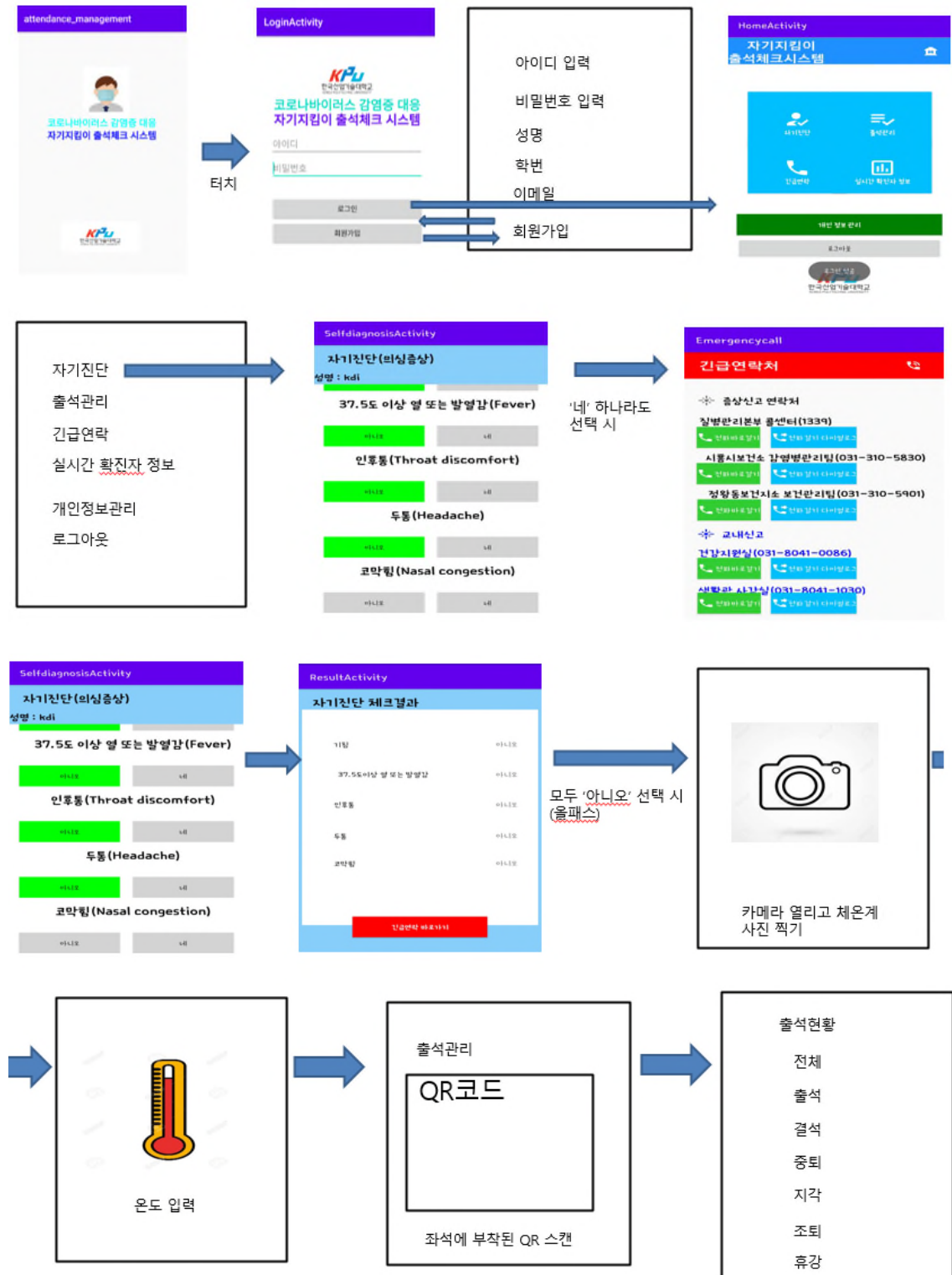
1. 개발 내용

회원가입 및 로그인을 하여 사용자 관리를 할 수 있도록 개발
사용자의 자가 진단 체크를 통해 모든 증상이 해당하지 않을 경우 출석 체크 진행
이 가능하며 증상이 하나라도 해당이 되면 긴급 연락을 할 수 있도록 개발
자가 진단이 모두 무증상일 경우 사용자의 온도를 체온계로 잰 후 카메라로 촬영한
후 구글 ml kit을 통한 영상처리를 이용해 체온을 데이터로 저장 기능 개발
체온 입력 후 좌석에 부착된 QR코드 스캔을 통해 빠르게 관리자가 좌석 확인 및 출
석 체크를 할 수 있는 기능 구현

2. 문제 및 해결방안

PHP를 통한 DB의 통합이 잘 되지 않아, Firebase 기반의 DB를 구현하였다.
체온계의 온도를 촬영한 후 단순히 사진을 저장하고 온도를 기록하는 것은 가용성
이 없다고 판단하여, OCR을 구글 ml kit을 통해 구현하였다.
자신의 정보를 QR스캔을 통해서 읽어내려고 하였으나, 실용성이 없고 의미가 없다
고 판단하여 좌석마다 부착된 QR코드를 스캔하여 관리자가 확인하기로 결정하였다.

3. 시험 시나리오



사용자는 회원 가입을 통해 아이디를 만들어 출석 체크 시스템을 사용할 수 있다. 로그인 후 자가 진단이 모두 무증상인 경우 카메라가 작동하고 체온계에 나와있는 온도를 찍고 나면 영상처리를 통해 체온을 데이터로 저장 하여 기록, 그 후에 qr코

드로 좌석을 스캔하면 최종으로 관리자가 그 내용을 확인하여 출석으로 인증한다.
자가 진단 중 하나라도 증상이 있을 경우 긴급 연락처로 넘어가면서 출석할 수 없게 한다.

체온이 38도 이상인 경우 출석 체크가 불가능한다.

4. 상세 설계

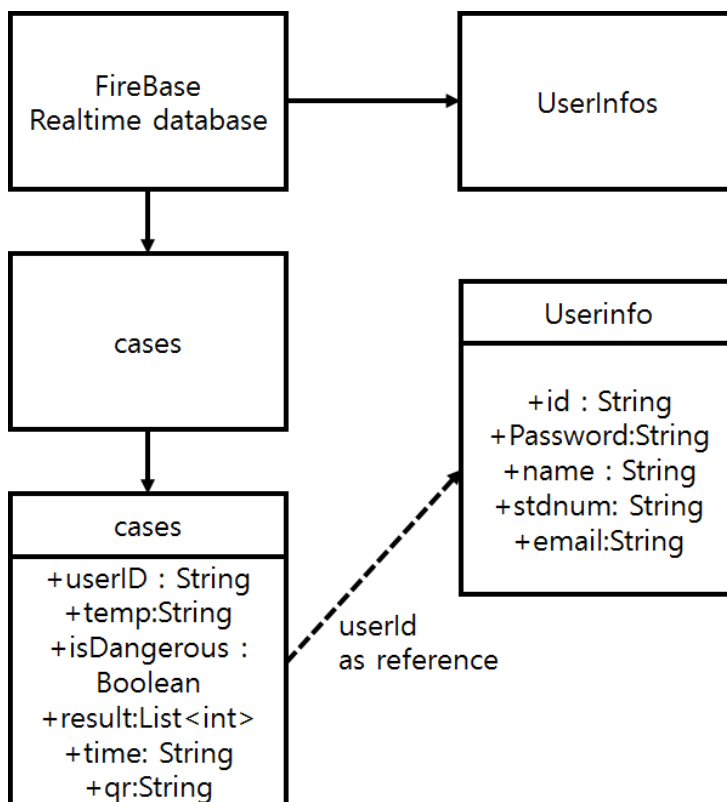
로그인 후 메인 화면 설계

```
btn_qrcodescan=findViewById(R.id.btn_qrcodescan);
```

```
btn_qrcodescan.setOnClickListener(new View.OnClickListener() {
```

버튼 클릭 시 이벤트를 발생하게 하는 setOnClickListener를 이용하여 qr코드 스캔 화면으로 넘어가는 기능, 출석현황 화면으로 넘어가는 기능, 뒤로 가기 버튼을 눌렀을 때 기능을 구현

데이터 베이스 설계



체온계의 온도 영상처리 설계

ML Kit 사용 - 텍스트 인식, 얼굴 인식, 랜드마크 인식, 바코드 스캔, 이미지 라벨 지정, 텍스트 언어 식별 등 일반적인 모바일 사용 사례에 즉시 사용 가능한 API 집합이 제공된다. ML Kit 라이브러리에 데이터를 전달하기만 하면 필요한 정보를 확인할 수 있다.

구현 경로

1. SDK 통합
2. 입력 데이터 준비
3. ML 모델을 데이터에 적용

데이터베이스 설계

관리자 계정 설계

5. Prototype 구현



The login screen features the KPSU logo at the top, followed by the text '코로나 바이러스 감염증 대응 자가지킴이 출석체크 시스템'. Below this are two input fields labeled '아이디' (ID) and '비밀번호' (Password). At the bottom are two buttons: '로그인' (Login) and '회원가입' (Sign Up).

(그림 1) 로그인 화면

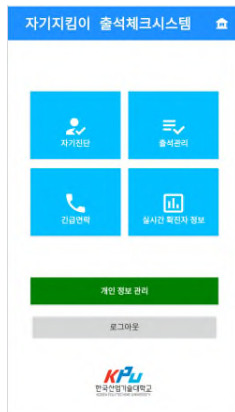
그림 1은 로그인 화면을 구성한 레이아웃 디자인이다. 아이디와 회원가입을 입력하는 텍스트 컴포넌트와 로그인, 회원가입 버튼으로 구성을 하였다.



The sign-up screen features the KPSU logo at the top, followed by the text '코로나 바이러스 감염증 대응 자가지킴이 출석체크 시스템'. Below this are five input fields labeled '아이디' (ID), '비밀번호' (Password), '성명' (Name), '학번' (Student ID), and '이메일' (Email). At the bottom is a single button labeled '회원가입' (Sign Up).

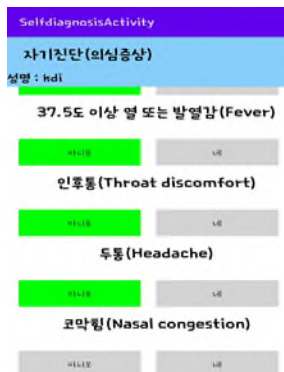
(그림 2) 회원가입 화면

그림 2는 회원가입 화면을 구성한 레이아웃 디자인이다. 아이디, 비밀번호, 성명, 학번, 이메일을 입력하는 텍스트 컴포넌트와 회원가입 버튼으로 구성을 하였다. 여기서 `implementation platform('com.google.firebase:firebase-bom:25.12.0')` 다음과 같은 코드로 firebase와 연동을 시켜 입력된 값은 사용자별로 DB에 저장되도록 구현을 하였다.



(그림 3) 메인화면

그림 3은 메인화면을 구성한 레이아웃 디자인이다. 되도록 사용하기 쉽고 보기 편하도록 간단하게 메뉴를 구성하려고 하였다. 자기진단, 출석관리, 긴급연락, 실시간 확진자 정보, 회원 정보 관리, 로그아웃 버튼으로 구성을 하였다.



(그림 4) 자기진단 화면

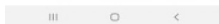
그림 4는 자기진단을 구성한 레이아웃 디자인이다. 기침, 발열, 인후통, 두통, 코막힘 5가지 항목이 있으며 각 항목별로 네, 아니오 버튼을 두어 사용자가 체크하도록 구성을 하였다. 아니오 버튼을 누르면 버튼의 색이 초록색으로 변하는 이벤트가 발생하고 네 버튼을 누르면 버튼의 색이 빨간색으로 변하는 이벤트가 발생하도록 구현을 하였다. 모두 체크를 했을 시 다음 액티비티로 넘어간다.



계좌계 인지도 확인하기

QR 코드 인식하기

필수 항목을 수형해주세요.



(그림 5) QR코드 및 채운 촬영 화면

그림 5는 QR코드 및 채운 촬영 액티비티 레이아웃 디자인이다. 채운계 인식 하러가기와 QR코드 인식 하러가기 버튼으로 구성되어 있으며 모두 했을 경우 메인화면으로 넘어간다.

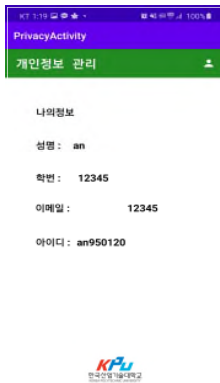


(그림 6) 긴급연락처 화면

그림 6은 긴급연락처를 구성한 레이아웃 디자인이다. 기관 이름과 해당 연락처가 표시되도록 디자인을 하였으며 각 항목에 전화 바로 걸기와 전화걸기 다이얼로그 버튼이 있도록 구성을 하였다. `<uses-permission android:name="android.permission.CALL_PHONE"/>` `<uses-permission android:name="android.permission.DIAL_PHONE"/>` 다음과 같은 코드로 앱으로 전화걸기가 가능하도록 구현을 하였으며 전화 바로 걸기 버튼을 누르면 바로 전화를 걸 수 있게 다이얼로그 버튼을 누르면 연락처가 자동으로 입력되도록 구현을 하였다.

(그림 7) 개인정보 화면

그림 7은 개인정보를 구성한 레이아웃 디자인이다. 나의정보, 성명, 학번, 이메일, 아이디 이렇게 5개의 텍스트 뷰로 구성을 하였으며 나의 정보를 제외한 텍스트 뷰 옆에 텍스트뷰를 넣어 DB에 저장된 정보가 앱으로 전달되어 표시되도록 구현을 하였다.



6. 시험/ 테스트 결과



사용자는 자가진단 페이지를 통해 테스트를 한다. 그 후 체온 인식과 QR코드 인식을 할 수 있다. 위의 과정을 마친 후, 사용자는 자가 진단 체크 결과를 확인할 수 있으며, 사용자는 자신의 정보와 관리자로부터 승인 여부를 확인할 수 있다.

7. Coding & DEMO

1. 체온 측정

1-1사진기능으로 qr코드 정보를 읽어오는 정보를 읽어오는 함수를 사용하여 성공시 qr코드 정보와 시간이 데이터를 자동으로 DB에 전달하는 기능을 수행한다.

```
override fun onActivityResult(requestCode: Int, resultCode: Int, data: Intent?) {
    super.onActivityResult(requestCode, resultCode, data)
    if (requestCode == REQUEST_IMAGE_CAPTURE && resultCode == RESULT_OK) {
        galleryAddPic()
    }
    if (requestCode == REQUEST_QR_CAPTURE && resultCode == RESULT_OK) {
        processQRResult(requestCode, resultCode, data)
    }
}
```

1-2체온 측정 후 식별이 되지 않았거나 온도에 따른 안내를 설정하여 수행한다.

```
79     val failureCallback = { ->
80         run {
81             toast("체온이 식별되지 않았습니다.")
82             return@run
83         }
84     }
85     val successCallback = { uri: Uri ->
86         // Partial application(Curring) 적용 함수
87         { s: String ->
88             run {
89                 try {
90                     val num = s.toDouble()
91                     if (25.0 > num || 45.0 < num) {
92                         throw InputMismatchException()
93                     }
94                     if(32.0 > num || 38.0 < num) {
95                         intent.putExtra("dangerous?", true)
96                     }
97                     preparedIntent.putExtra("temp", num)
98                     preparedIntent.putExtra("imgUri", uri)
99                     OCRCompleted = true
100                     makeCompleteText(OCRButton)
101                     checkCompletableAndIfEditText()
102                     OCRButton.setOnClickListener {
103                         toast("이미 완료되었습니다.")
104                     }
105                     return@run
106                 } catch (e: Exception) {
107                     failureCallback()
108                 }
109             }
110         }
111     }
112 }
```

2. 회원가입

```
32      //회원가입 버튼 클릭 시 수행
33      val btm_register = findViewById<Button>(R.id.btn_register2)
34      val database = Firebase.database
35      btm_register.setOnClickListener(View.OnClickListener {
36          val userID = et_id.text.toString()
37          val rawPassword = et_pass.text.toString()
38          val userName = et_name.text.toString()
39          val userNumber = et_number.text.toString()
40          val email = et_email.text.toString()
41          val user = UserInfoBuilder.build(userID, rawPassword, userName, userNumber, email)
42          val ref = database.getReference("userinfos/$userID")
43          val existsListener = object : ValueEventListener {
44              override fun onDataChange(dataSnapshot: DataSnapshot) {
45                  if(!dataSnapshot.exists()) {
46                      ref.setValue(user)
47                      finish()
48                  } else {
49                      toast("중복된 id입니다.")
50                  }
51              }
52              override fun onCancelled(databaseError: DatabaseError) {
53                  // Getting Post failed, log a message
54                  Log.w("FIREBASE", "existsListener:onCancelled", databaseError.toException())
55                  // ...
56              }
57          }
58          ref.addListenerForSingleValueEvent(existsListener)
59      })
60  }
61 }
```

3. 로그인

```
40         val btn_login = findViewById<Button>(R.id.btn_login)
41         val database = Firebase.database
42         btn_login.setOnClickListener(View.OnClickListener {
43             val userID = et_id.text.toString()
44             val rawUserPassword = et_pass.text.toString()
45             val encryptedPassword = Sha512.encrypt(rawUserPassword)
46             val matchListener = object : ValueEventListener {
47                 override fun onDataChange(dataSnapshot: DataSnapshot) {
48                     if(!dataSnapshot.exists()) {
49                         loginFailed()
50                         return
51                     }
52                     val user: UserInfo = dataSnapshot.getValue<UserInfo>()!!
53                     user.let {
54                         if (it.id == userID && it.password == encryptedPassword) {
55                             loginSuccess(it)
56                         } else {
57                             loginFailed()
58                         }
59                     }
60                 }
61
62                 override fun onCancelled(databaseError: DatabaseError) {
63                     // Getting Post failed, log a message
64                     Log.w("FIREBASE", "matchListener:onCancelled", databaseError.toException())
65                     // ...
66                 }
67             }
68             val userRef = database.getReference("userinfos/$userID")
69             userRef
70             userRef.addListenerForSingleValueEvent(matchListener)
71         })
72     }
73
```

4. 긴급 연락

```

    ○
9   class EmergencyCall : AppCompatActivity() {
10      private val telNumbers = listOf<String>("1339", "031-310-5830", "031-310-5901", "031-8041-0086", "031-8041-1030")
11      .map {
12          s -> "tel${s}"
13      }
14
15      override fun onCreate(savedInstanceState: Bundle?) {
16          super.onCreate(savedInstanceState)
17          setContentView(R.layout.emergencycall_xml)
18
19          for((number, idx) in telNumbers.zip(1..telNumbers.size)) {
20              val call = findViewById<Button>(resources.getIdentifier("btn_call${idx}", "id", packageName))
21              val dialogCall = findViewById<Button>(resources.getIdentifier("btn_dialogcall${idx}", "id", packageName))
22              call.setOnClickListener { _ ->
23                  startActivity(Intent("android.intent.action.CALL", Uri.parse(number)))
24              }
25              dialogCall.setOnClickListener { _ ->
26                  startActivity(Intent("android.intent.action.DIAL", Uri.parse(number)))
27              }
28          }
29      }

```

5. 자기 진단

```

42     private fun inflateSymptomsLayout() {
43         val inflater = layoutInflater
44         val root = findViewById<ViewGroup>(R.id.SymptomsLayout)
45         for (i in symptomsList.indices) {
46             inflater.inflate(R.layout.form_xml, root)
47             val currentView = root.getChildAt(root.childCount - 1)
48             val textView = currentView.findViewById<TextView>(R.id.symptom_textView)
49             textView.text = symptomsList[i].second
50             val noButton = currentView.findViewById<Button>(R.id.no_button)
51             val yesButton = currentView.findViewById<Button>(R.id.yes_button)
52             val noListener = ListenerFactory.makeClickListener(this, i, false, Color.GREEN, yesButton)
53             val yesListener = ListenerFactory.makeClickListener(this, i, true, Color.RED, noButton)
54             noListener.mutualListener = yesListener
55             yesListener.mutualListener = noListener
56             noButton.setOnClickListener(noListener)
57             yesButton.setOnClickListener(yesListener)
58         }
59     }
60
61     fun setSymptomsFlag(idx: Int, flag: Boolean) {
62         if (flagMap.size <= idx) {
63             return
64         }
65         val symptom = fullEnglishSymptoms[idx]
66         visited[idx] = true
67         flagMap[symptom] = flag
68     }
69
70     fun call() {
71         caller.call(this)
72     }
73 }

```

Ⅲ. 결론

1. 연구 결과

최근 코로나19를 치료할 백신 개발의 소식이 들려오고 있지만 아직 감염의 확산을

멈추게 하는 단계까지 하지 못하고 확진자의 수는 더욱 늘어나 꼼꼼한 대처 방법이 더욱 필요한 시점이다.

이 코로나 19 어플을 이용한 출석 체크를 통해 빠르고 정확한 출석을 하고 수업시간 속 최소한의 지장을 목표로 하여금 수월한 수업을 학생들이 들을 수 있도록 할 수 있다.

또한 관리자가 관리 업무 기능을 통해 편리하게 학생들의 수업 정보 및 출결 관리로 손쉬운 상황 관리가 가능하다.

이제 코로나19 말고도 다른 전염병의 위기가 다시 도래할 가능성이 높다. 따라서 코로나 19 뿐만 아니라 다음 확산의 위기 중에 이 어플리케이션의 활용하여 수업에 도움이 될 수 있다.

2. 작품제작 소요재료 목록

어플리케이션 제작으로 실물 소요 재료는 없다.

참고자료

Firebase 실시간 데이터베이스

<https://firebase.google.com/docs/database>

<https://github.com/DPS0340/KPU-R6SDB>

QR코드 생성

<https://blog.naver.com/cosmosjs/221157180721>

Github 연동

https://www.youtube.com/watch?v=tC8Xj_Bf8Fw

관리자 계정 생성 및 관리

<https://www.youtube.com/watch?v=Hk1thVSref4>