

음식점 내 QR 간편 주문 및 결제 애플리케이션 (찍고 먹자)

Simple QR code order and payment
application in the restaurant

본 보고서를 Project 프로젝트의 최종보고서로 제출합니다.

학과(전공): 컴퓨터소프트웨어공학

담당 교수: 박유현

팀 명: 초코파이썬칩

제 출 일: 2021년 12월 8일

팀 장 : 김기태 20173152

팀원1 : 구본영 20173219

팀원2 : 안대현 20173217

팀원3 : 정순범 20173212

동 의 대 학 교

컴퓨터소프트웨어공학과(전공)장 귀하

〈 요약 문 〉

| | | | |
|--|--|--------------|-------------|
| <p>프로젝트 목표 (300자내외)</p> | <p>찍고 먹자는 매장 내 무인 결제 시스템의 발전이 가속화되고 있는 가운데, 이미 많은 매장에서 사용되고 있는 무인 주문 및 결제 시스템인 키오스크의 단점을 해결하는 것을 목적으로 하는 서비스이다. 사용자들은 한정된 수량과 복잡한 사용법의 키오스크를 대신하여 애플리케이션을 사용하여 매장 내 각 테이블에 부착된 QR코드를 통하여 핸드폰으로 주문 및 결제를 진행함으로써 사용자의 편의성과 주문 대기시간 감소를 목표하며, 등록된 가맹점의 실시간 잔여 좌석 확인을 통해 식사 가능한 매장인지 애플리케이션을 통하여 확인할 수 있다.</p> | | |
| <p>내용 (500자내외)</p> | <p>본 프로젝트의 이름인 ‘찍고 먹자’의 사용자는 일반 회원과 가맹점주로 나뉜다. 최초 회원가입 시 DB에 사용자의 ID와 가맹점주의 ID를 구분하여 저장한 후 로그인 시 각 사용자에게 따른 화면을 출력한다.</p> <p>먼저 일반 회원은 지도 API를 기반으로 매장 검색이 가능하며 해당 매장의 정보와 메뉴, 실시간 좌석 현황을 파악할 수 있다. 이후 매장에서 식사하는 경우 테이블에 부착된 QR코드를 이용하여 애플리케이션 내에서 주문 및 결제할 수 있다. 한번 이용한 매장의 쿠폰 및 스탬프를 마이 페이지에서 통합 관리하여 사용자의 편의성을 중점으로 구상하였다.</p> <p>가맹점 주는 가맹점의 메뉴를 등록 수정 삭제가 가능하며, 애플리케이션을 이용하여 주문 명세를 확인할 수 있다. 또한, 실시간으로 좌석의 현황을 파악하며 관리해 줌으로써 일반회원에게 보이는 좌석 현황을 관리한다. 마지막으로 총 매출을 시간 별, 메뉴별로 파악하여 가맹점 주가 애플리케이션을 통해 매장 관리하기 쉽게 구상하였다.</p> | | |
| <p>기대효과 (200자내외)</p> | <p>본 프로젝트는 현재 매장에서 사용하는 키오스크를 대신하는 무인 주문 및 결제 시스템으로, 키오스크의 설치 비용을 절감할 수 있다.</p> <p>또한, 한정된 개수의 키오스크로 인해 발생하는 대기시간을 각 테이블에 배치된 QR코드를 통해 대기시간을 감소할 수 있다. 이를 통해 사용자는 시간적 이익과 손쉬운 접근성을, 가맹점주는 금전적 이익을 취할 수 있다.</p> | | |
| <p>Keywords</p> | <p>QR</p> | <p>무인 매장</p> | <p>키오스크</p> |
| | <p>접근성</p> | <p>찍고 먹자</p> | <p>API</p> |

목 차

| | |
|----------------------|----|
| I . 프로젝트 개요 | 1 |
| 1.1 프로젝트 배경 및 필요성 | 1 |
| II . 프로젝트 설계 내용 | 2 |
| 2.1 프로젝트 설계 | 2 |
| 2.1.1 디자인 설계 | 2 |
| 2.1.2 모듈 설계 | 4 |
| 2.1.2.1 액티비티 UML | 5 |
| 2.1.2.2 프래그먼트 UML | 5 |
| 2.1.2.2 서버 아키텍처 | 5 |
| 2.1.3 DB 설계 | 6 |
| 2.1.3 세부 기능 명시 | 6 |
| 2.2 구현환경 및 시스템 스펙 | 7 |
| III . 프로젝트 구현 내용 | 8 |
| 3.1 기능별 모듈 설명 | 8 |
| 3.1.1 가맹점 기능 | 8 |
| 3.1.2 사용자 기능 | 22 |
| 3.2 기능 동작 화면 | 33 |
| 3.2.1 가맹점 기능 | 33 |
| 3.2.2 사용자 기능 | 43 |
| 3.4 팀원별 역할 분석 | 51 |
| IV . 프로젝트 결론 | 52 |
| 4.1 프로젝트 성과 내용 | 52 |
| 4.1.1 활용방안 | 52 |
| 4.1.2 기대효과 | 52 |
| 4.2 설계 목표의 중요도 및 달성도 | 53 |
| 4.3 개발 일정 | 54 |
| 4.4 소감 | 54 |

1. 프로젝트 개요

1.1 프로젝트 배경 및 필요성

코로나-19 바이러스로 인해 무인 시스템의 발전이 가속화되고 있는 가운데, 이미 많은 매장에서는 키오스크와 같은 무인 시스템을 도입해 사용 중이다. 하지만 키오스크는 고정층의 사용법 미숙과 더불어 한정된 수량으로 인한 대기시간 증가문제가 잇달아 나타나고 있다.



이에 반해, QR코드를 사용한 주문 및 결제 방식은 현대 사회에서 널리 보급된 스마트폰에서 이루어지기 때문에 사용자 입장에서는 대기시간 없이 좀 더 간편하고 빠르게 서비스를 이용할 수 있다. 또한, 키오스크와 같은 별도의 기기가 필요하지 않기 때문에 초기 비용에 대한 부담을 덜 수 있다.

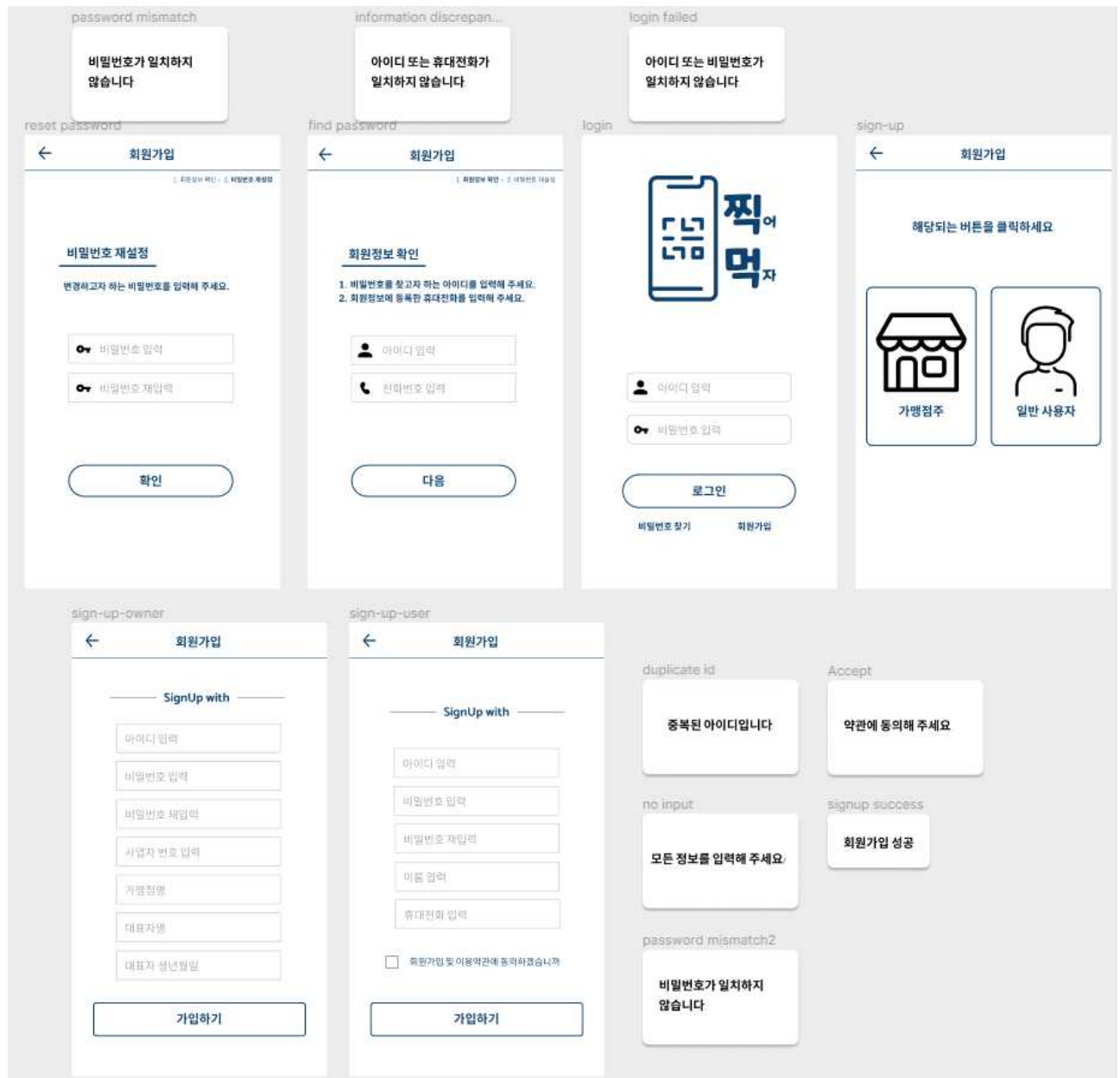
따라서 본 프로젝트에서는 가속화되고 있는 비대면 서비스 시장에 대한 기술현황, 적용사례 등을 종합적으로 살펴보고 실제 매장에 QR 주문 및 결제 시스템이 도입될 수 있도록 애플리케이션을 개발해보고자 한다.

2. 프로젝트 설계 내용

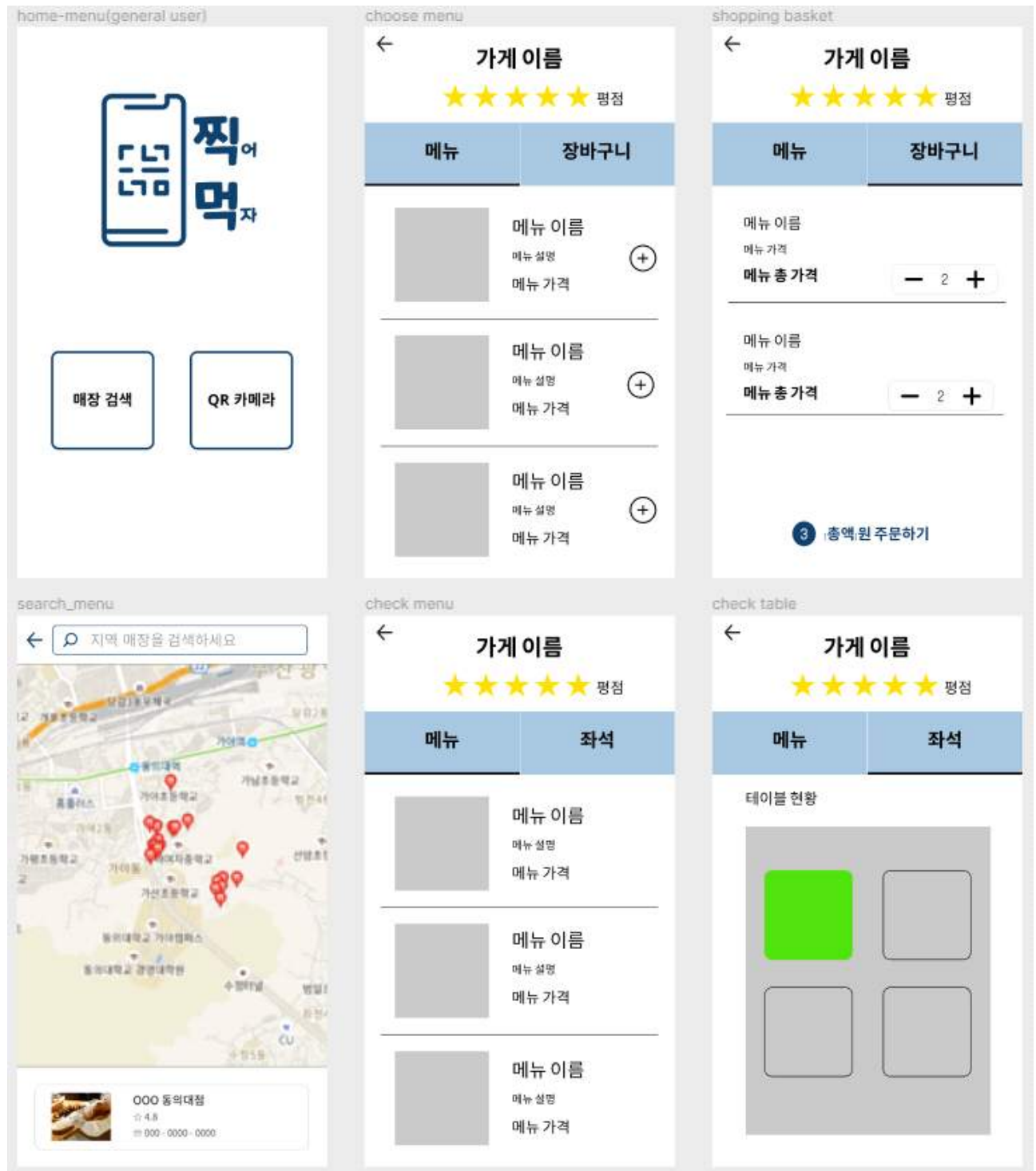
2.1 프로젝트 설계

2.1.1 디자인 설계

2.1.1.1 공통 기능



2.1.1.2 사용자 기능

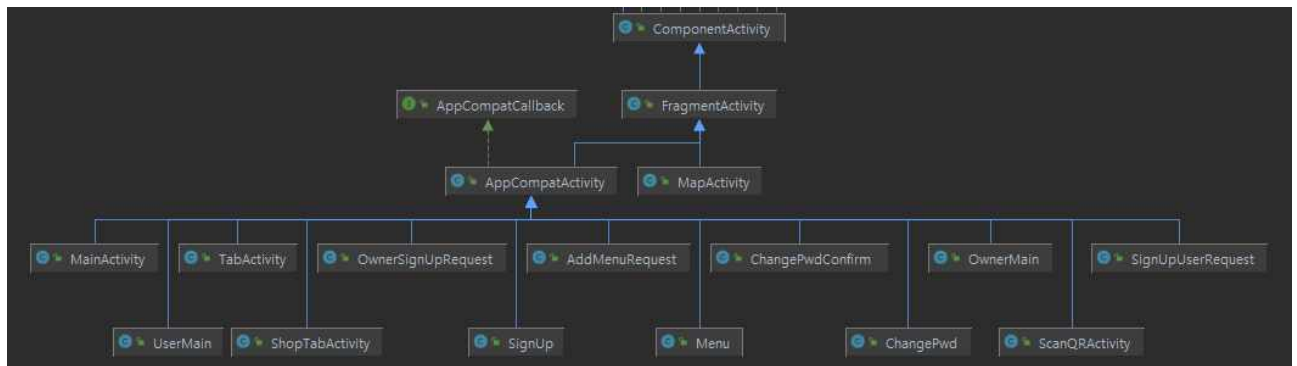


2.1.1.3 가맹점 기능

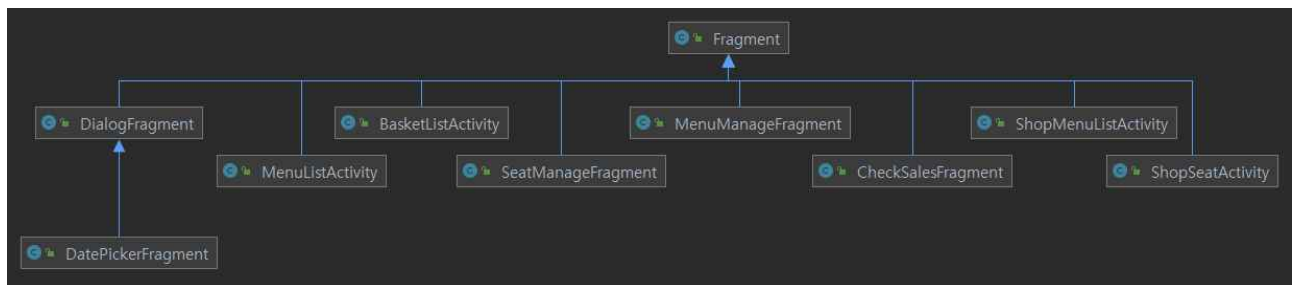


2.1.2 모듈 설계

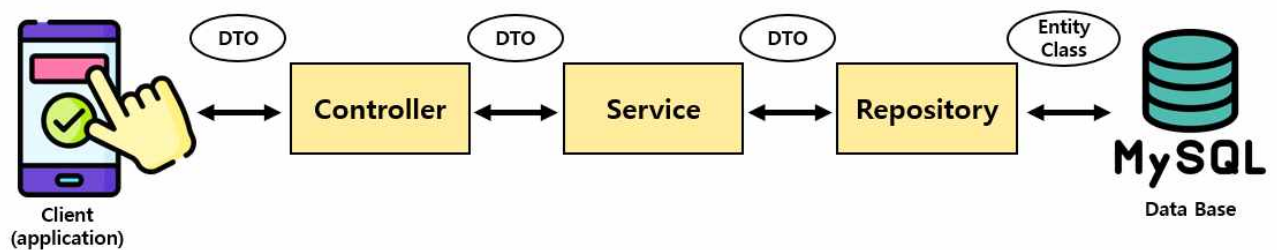
2.1.2.1 액티비티 UML



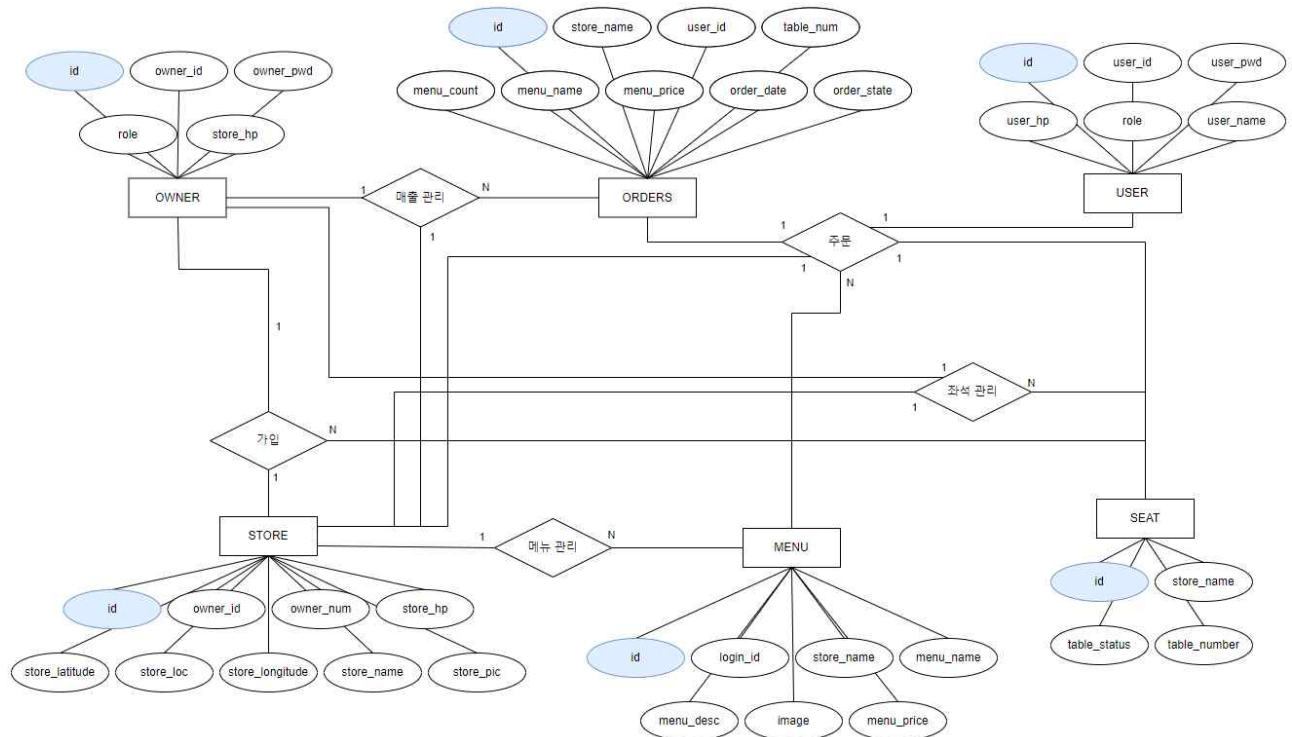
2.1.2.2 프래그먼트 UML



2.1.2.3 서버 아키텍처



2.1.3 DB 설계



2.1.4 세부 기능 명시

가. 공통기능

1) 회원가입

- 서비스를 사용할 사용자와 서비스에 가맹점을 등록하는 가맹점주가 회원가입을 진행한다.
- 사용자와 가맹점주는 아이디, 비밀번호, 전화번호, 이름은 공통으로 입력한다.
- 가맹점주는 사업자번호, 가맹점명, 가맹점 위치를 추가로 입력한다.

2) 로그인

- 사용자의 구분에 따라 가맹점주, 일반 사용자를 선택할 수 있다.
- 선택한 구분에 따라 아이디와 비밀번호를 입력하여 로그인할 수 있다.

3) 비밀번호 재설정

- 비밀번호를 잊어버렸을 때 재설정 할 수 있는 기능이다.
- 사용자와 가맹점주는 아이디와 전화번호를 입력하여 회원임을 인증한다.
- 재설정할 비밀번호와 재확인 비밀번호를 입력한다.

나. 가맹점주 기능

1) 좌석 관리

- 사용자가 해당 테이블에 주문한 메뉴이름, 메뉴수량, 가격을 확인할 수 있다.
- 사용자가 테이블 사용이 끝나면 가맹점주는 토글버튼으로 테이블이 비었음을 알린다.

2) 메뉴 관리

- 가맹점주는 메뉴 추가할 때 메뉴 이미지, 메뉴명, 메뉴 설명, 가격을 기입한다.
- 추가한 메뉴를 확인할 수 있으며 삭제도 가능하다.

3) 매출 확인

- 날짜를 선택하여 해당 날짜에 매출을 확인할 수 있다.
- 매출 확인은 메뉴명, 판매수량, 총 합계 확인이 가능하다.

다. 사용자 기능

1) 매장 검색

- 사용자는 검색창을 통해 매장 검색이 가능하다.
- 사용자의 주변에 존재하는 매장을 마커를 통해 화면에 표시한다.

2) 메뉴 확인

- 사용자는 선택한 매장의 메뉴를 확인한다.
- 메뉴 이름, 메뉴 설명, 메뉴 가격을 확인할 수 있다.

3) 좌석 확인

- 사용자는 선택한 매장의 좌석현황을 확인할 수 있다.

4) QR 주문 및 결제

- QR코드를 이용하여 해당 가맹점의 주문으로 이동할 수 있다.
- 원하는 음식을 선택 및 취소가 가능하다.
- 선택한(장바구니) 메뉴에 대해 결제가 가능하다.

2.2 구현환경 및 시스템 스펙

| Project Specification | |
|-----------------------|--|
| Platform | Android (minSDK : 26, targetSDK : 31) |
| RDBMS | MySQL 8.0 |
| Prototype | Figma |
| IDE | Android Studio 2020.3.1 + IntelliJ 2021.2.3 Ultimate Edition |
| Web ApplicationServer | Spring Boot 2.5.5 |
| Library | Lombok, json, Glidle ... etc |
| Api | KAKAO API, NAVER MAP API |
| Open Data | 도로명 공공데이터 |
| OS | Windows 10 |
| AVD | Pixel 2 API 30 |

3. 프로젝트 구현 내용

3.1 기능별 모듈 설명

3.1.1 가맹점 기능

| 클래스 | OwnerSignUpRequest |
|------|--|
| 기능 | 가맹점주 회원가입 기능 |
| 담당자 | 정순범 |
| 중요코드 | <pre> public class OwnerSignUpRequest extends AppCompatActivity { new Thread(new ApiRunner()).start(); Thread.sleep(1000); new Thread(new ConnectRunner()).start(); } public class ApiRunner implements Runnable { @SneakyThrows @Override public void run() { String storeLocEncode = URLEncoder.encode(storeLoc, "UTF-8"); String url = "https://dapi.kakao.com/v2/local/search/address.json?analyze_type=similar&page=1& size=10&query="+storeLocEncode; String authorizationKey = "KakaoAKa85dda23e2cf591d865b03498582c750"; URL urlObject = null; HttpURLConnection con = null; StringBuffer response = new StringBuffer(); String result=""; try { urlObject =new URL(url); con = (HttpURLConnection) urlObject.openConnection(); con.setRequestMethod("GET"); con.setRequestProperty("Authorization", authorizationKey); int responseCode = con.getResponseCode(); BufferedReader iny =new BufferedReader(new InputStreamReader(con.getInputStream())); String output; while ((output = iny.readLine()) !=null) { response.append(output); } iny.close(); result = response.toString(); System.out.println(result); } catch (Exception e) { e.printStackTrace(); } JSONObject jsonObject =new JSONObject(result); JSONArray jArray = jsonObject.getJSONArray("documents"); JSONObject jsonObject2 = jArray.getJSONObject(0); JSONObject jsonObject3 = jsonObject2.getJSONObject("address"); storeLatitude=jsonObject3.getDouble("y"); storeLongitude=jsonObject3.getDouble("x"); } } </pre> |
| 설명 | <p>가맹점주 회원가입 시 입력 필드에 가맹점 위치가 들어가기 때문에 도로명 주소를 가져오기 위해 Kakao Api를 사용하였다.</p> <p>request 보낼 때 도로명 주소를 입력받아 보내면 response로 json Array를 받는다. json Array를 json 라이브러리를 사용해서 파싱한 다음 서버로 전송한다.</p> |
| 중요코드 | <pre> public class ConnectRunner implements Runnable { @Override </pre> |

| | |
|----|---|
| | <pre> public void run() { HttpService httpService = HttpClient.getApiService(); try { OwnerSignUpDto ownerSignUpDto =new OwnerSignUpDto(ownerId, ownerPwd,storeHP,Role.ROLE_OWNER); StoreSignUpDto storeSignUpDto=new StoreSignUpDto(ownerId,storeName,storeHP,storeLoc,storeLatitude,storeLongitude,null,ownerNum); Response<Message> loginResponse = httpService.OwnerSignUpRequest(ownerSignUpDto).execute(); Response<Message> StoreSignUpResponse= httpService.StoreSignUpRequest(storeSignUpDto).execute(); Intent intent =new Intent(getApplicationContext(), MainActivity.class); startActivity(intent); }catch (IOException e){ Handler handler=new Handler(Looper.getMainLooper()); handler.postDelayed(new Runnable() { @Override public void run() { AlertDialog.Builder builder =new AlertDialog.Builder(OwnerSignUpRequest.this); builder.setTitle("SignUP Failed").setMessage("중복된 회원 정보 입니다."+"\\n"+"다시 입력 해주세요."); AlertDialog alertDialog = builder.create(); alertDialog.show(); } },0); e.printStackTrace(); } } </pre> |
| 설명 | <p>가맹점주 회원가입시 owner테이블과 store테이블에 분할하여 저장된다. owner테이블은 가맹점주 개인정보가 저장되며 store테이블은 가맹점에 관련된 정보가 저장된다.</p> <p>kakao api를 사용해 도로명 주소의 경도 위도를 가져와 저장하며 사용자가 매장 검색 할 때 지도에 마커로 표시된다.</p> |

| 클래스 | MainActivity |
|------|---|
| 기능 | 로그인 기능 |
| 담당자 | 정순범 |
| 중요코드 | <pre> public class MainActivity extends AppCompatActivity { RadioGroup radioGroup; EditText IdText; EditText PwdText; Role loginUserType = Role.ROLE_OWNER; public static String loginId; private String loginPwd; //메뉴 저장할때 로그인한 사용자 아이디 넘기기 위해 public static Context context_main; // context 변수 선언 public String var; // 다른 Activity에서 접근할 변수 @Override protected void onCreate(Bundle savedInstanceState) { super.onCreate(savedInstanceState); setContentView(R.layout.main_screen); //메인 화면 context_main=this; IdText = (EditText) findViewById(R.id.login_Id); radioGroup = (RadioGroup) findViewById(R.id.radioGroup); radioGroup.setOnCheckedChangeListener(new </pre> |

| | |
|------|---|
| | <pre> RadioGroup.OnCheckedChangeListener() { @Override public void onCheckedChanged(RadioGroup radioGroup, int checkedId) { switch (checkedId){ case R.id.owner_choice: IdText.setHint("가맹점주 아이디 입력"); loginUserType = Role.ROLE_OWNER; break; case R.id.user_choice: IdText.setHint("일반사용자 아이디 입력"); loginUserType = Role.ROLE_USER; break; } } } public void go_login(View v) throws JSONException { IdText = (EditText) findViewById(R.id.login_Id); PwdText = (EditText) findViewById(R.id.login_Pwd); loginId = IdText.getText().toString(); loginPwd = PwdText.getText().toString(); new Thread(new ConnectRunner()).start(); } </pre> |
| 설명 | <p>로그인 시 라디오 버튼을 통해 사용자, 가맹점주 선택이 가능하다. 선택에 따라 데이터 베이스에서 조회하는 아이디가 다르다.</p> <p>go_login 메소드를 통해 로그인이 이루어진다.</p> |
| 중요코드 | <pre> public class ConnectRunner implements Runnable { @Override public void run() { HttpService httpService = HttpClient.getApiService(); LoginRequestDto loginRequestDto =new LoginRequestDto(loginId, loginPwd, loginUserType); try { Response<Message> loginResponse = httpService.login(loginRequestDto).execute(); Message message=loginResponse.body(); if (message.getRole()== Role.ROLE_OWNER){ var=loginId; Intent intent =new Intent(getApplicationContext(), OwnerMain.class); startActivity(intent); } else if (message.getRole()== Role.ROLE_USER){ Intent intent =new Intent(getApplicationContext(), UserMain.class); startActivity(intent); } }catch (Exception e){ Handler handler=new Handler(Looper.getMainLooper()); handler.postDelayed(new Runnable() { @Override public void run() { AlertDialog.Builder builder =new AlertDialog.Builder(MainActivity.this); builder.setTitle("LoginFailed").setMessage("올바르지 않은 회원정보 입니다."+ "\n\n" + "다시 입력 해주세요."); AlertDialog alertDialog = builder.create(); alertDialog.show(); } },0); } } } </pre> |

| | |
|----|---|
| 설명 | 로그인 버튼을 누르면 Thread 통신이 이루어져 서버에 있는 데이터베이스에 로그인 아이디와 로그인 비밀번호를 전송한다. 서버에서 이루어진 응답을 통해 로그인 성공/실패 여부를 판단하고 로그인 성공시 사용자 역할에 맞는 화면으로 전환한다. |
|----|---|

| 클래스 | ChangePwd, ChangePwdConfirm |
|------|---|
| 기능 | 비밀번호 재설정 기능 |
| 담당자 | 정순범 |
| 중요코드 | <pre> public class ConfirmRunner implements Runnable{ @Override public void run() { HttpService httpService = HttpClient.getApiService(); try { Response<Boolean> message=httpService.getId_Phone (changeConfirmId,changeConfirmPhone,loginUserType).execute(); if (message.body() == true){ Intent intent =new Intent(getApplicationContext(), ChangePwd.class); startActivity(intent); } else{ Handler handler=new Handler(Looper.getMainLooper()); handler.postDelayed(new Runnable() { @Override public void run() { AlertDialog.Builder builder =new AlertDialog.Builder(ChangePwdConfirm.this); builder.setTitle("Failed").setMessage("올바르지 않은 회원정보 입 니다."+"Wn"+"다시 입력 해주세요."); AlertDialog alertDialog = builder.create(); alertDialog.show(); } },0); } } catch (IOException e) { e.printStackTrace(); } } } </pre> |
| 설명 | 입력한 아이디와 전화번호가 서버에 전송해 저장된 회원정보와 일치하는지 검사한다. 일치하면 비밀번호 변경 페이지로 전환되고, 일치하지 않으면 다이얼로그 창을 보여준다. |
| 중요코드 | <pre> if (change_pwd.equals(change_pwd2)){ new Thread(new ChangePwdRunner()).start(); }else{ AlertDialog.Builder builder =new AlertDialog.Builder(ChangePwd.this); builder.setTitle("Failed").setMessage("변경할 비밀번호와 재입력한 비밀번호가 다릅니다."+"Wn"+"다시 입력 해주세요."); AlertDialog alertDialog = builder.create(); alertDialog.show(); } private class ChangePwdRunner implements Runnable { @Override public void run() { HttpService httpService = HttpClient.getApiService(); try { Response<Message> message= httpService.changePwdRequest (changeId,change_pwd,varRole).execute(); </pre> |

| | |
|----|---|
| | <pre> Handler handler=new Handler(Looper.getMainLooper()); handler.postDelayed(new Runnable() { @Override public void run() { AlertDialog.Builder builder = new AlertDialog.Builder(ChangePwd.this); builder.setTitle("Change Password").setMessage("비밀번호 가 변경되었습니다."); AlertDialog alertDialog = builder.create(); alertDialog.show(); } },0); } catch (IOException e) { e.printStackTrace(); } } } </pre> |
| 설명 | <p>변경할 비밀번호와 확인 비밀번호를 입력 받는다. 변경할 비밀번호와 재입력한 비밀번호가 다르면 경고 다이얼로그 창을 보여준다.</p> <p>경할 비밀번호와 재입력한 비밀번호가 같으면 서버에 새로 입력받은 비밀번호를 전송해 update 쿼리를 전송해 비밀번호를 변경한다.</p> |

| 클래스 | MenuManageFragment |
|------|---|
| 기능 | 가맹점주 메뉴 조회 기능 |
| 담당자 | 정순범 |
| 중요코드 | <pre> public class MenuManageFragment extends Fragment implements View.OnClickListener{ public static ArrayList<Menu> InsertMenuList =new ArrayList<>(); private RecyclerView recyclerView; private MenuAdapter mAdapter; public static Bitmap globalBitmap; String loginId=((MainActivity)MainActivity.context_main).var; //로그인 아이디 가 저 오기 private MenuManageViewModel menuManageViewModel; private FragmentOwnerMenuManageBinding binding; public View onCreateView(@NonNull LayoutInflater inflater, ViewGroup container, Bundle savedInstanceState){ menuManageViewModel = new ViewModelProvider(this).get(MenuManageViewModel.class); binding = FragmentOwnerMenuManageBinding.inflate(inflater, container, false); View root = binding.getRoot(); Button add_menu=root.findViewById(R.id.bt_go_menu_page); add_menu.setOnClickListener(new View.OnClickListener(){ //프레그먼트에서 액티비티 호출할때 @Override public void onClick(View view) { Intent intent=new Intent(getActivity(), AddMenuRequest.class); startActivity(intent); } }); recyclerView = (RecyclerView) root.findViewById(R.id.recyclerView_menu); recyclerView.setHasFixedSize(true); mAdapter =new MenuAdapter(InsertMenuList); RecyclerView.LayoutManager mLayoutManager = new LinearLayoutManager(getActivity()); recyclerView.setLayoutManager(mLayoutManager); recyclerView.setAdapter(mAdapter); return root; } public class ConnectGetRunner implements Runnable { @Override public void run() { InsertMenuList.clear(); HttpService httpService= HttpClient.getApiService(); try { Response<List<MenuDto>> menu= httpService.getMenu(loginId).execute(); List<MenuDto> menuList= menu.body(); for(int i=0;i<menuList.size();i++) { String menuName=menuList.get(i).getImage(); int start=menuName.lastIndexOf("www"); int finish=menuName.lastIndexOf("."); menuName=menuName.substring(start+1,finish); InsertMenuList.add(new Menu(menuList.get(i).getMenuName(),menuList.get(i).getMenuPrice(), menuList.get(i).getMenuDesc(),"http://10.0.2.2:8080/img?loginId="+ loginId +"&menuName="+ menuName)); } } } } </pre> |
| 설명 | <p>가맹점주 에게 메뉴를 보여줄 때 리사이클러 뷰에 커스텀 리스트뷰를 사용하여 화면에 보여준다.</p> <p>Thread를 통해서 서버에 저장된 메뉴를 가져오기 위해 이미지 이름과 로그인 아이디를</p> |

| | |
|------|---|
| | 전송한다. 가져온 메뉴를 ArrayList에 저장해 어댑터에게 전달한다. |
| 중요코드 | <pre> public class MenuAdapter extends RecyclerView.Adapter<MenuAdapter.MyViewHolder>{ private ArrayList<Menu> mDataset; public static String selectMenuName; String loginId=((MainActivity)MainActivity.context_main).var; //로그인 아이디 가져오기 public class MyViewHolder extends RecyclerView.ViewHolder { public TextView menuName, menuPrice, menuDesc; public ImageView menuImage; //ViewHolder public MyViewHolder(View view) { super(view); menuName = (TextView) view.findViewById(R.id.menuName); menuPrice = (TextView) view.findViewById(R.id.menuPrice); menuDesc = (TextView) view.findViewById(R.id.menuDesc); menuImage= (ImageView) view.findViewById(R.id.ListMenuImage); } } public MenuAdapter(ArrayList<Menu> myData){ this.mDataset = myData; } @NonNull @Override public MyViewHolder onCreateViewHolder(@NonNull ViewGroup parent, int viewType) { View view = LayoutInflater.from(parent.getContext()) .inflate(R.layout.list, parent, false); return new MyViewHolder(view); } @Override public void onBindViewHolder(@NonNull MenuAdapter.MyViewHolder holder, int position) { Glide.with(holder.itemView.getContext()) .load(mDataset.get(position).getMenuImage()) .into(holder.menuImage); holder.menuName.setText(mDataset.get(position).getMenuName()); holder.menuPrice.setText(mDataset.get(position).getMenuPrice()); holder.menuDesc.setText(mDataset.get(position).getMenuDesc()); holder.itemView.setOnClickListener(new View.OnClickListener() { @Override public void onClick(View view) { String menuName=holder.menuName.getText().toString(); selectMenuName=menuName; new Thread(new removeMenuRunner()).start(); } }); } } </pre> |
| 설명 | <p>커스텀 리스트뷰에 사용할 어댑터 이다. ViewHolder에 메뉴이름, 메뉴 가격, 메뉴 설명, 이미지를 설정하고 onBindViewHolder 메소드에서- Glide 라이브러리를 사용해 이미지를 넣고 각 필드에 값을 넣어 리스트 형태로 가맹점주에게 보여준다.</p> |

| 클래스 | AddMenuRequest |
|------|--|
| 기능 | 가맹점주 메뉴 추가 기능 |
| 담당자 | 정순범 |
| 중요코드 | <pre> public class ConnectRunner implements Runnable { @Override public void run() { HttpService httpService = HttpClient.getApiService(); HashMap<String, RequestBody> map =new HashMap<>(); RequestBody sMenuName = RequestBody.create(MediaType.parse("text/plain"), menuName); RequestBody sMenuPrice = RequestBody.create(MediaType.parse("text/plain"), menuPrice); RequestBody sMenuDesc = RequestBody.create(MediaType.parse("text/plain"), menuDesc); RequestBody sLoginId = RequestBody.create(MediaType.parse("text/plain"), loginId); map.put("menuName",sMenuName); map.put("menuPrice",sMenuPrice); map.put("menuDesc",sMenuDesc); map.put("loginId",sLoginId); File file =new File(selectedImageUri.getPath()); // 파일 확장자 불러와서 ext에 저장 String mimeType = getContentResolver().getType(selectedImageUri); int pos = mimeType.lastIndexOf("/"); String ext = mimeType.substring(pos +1); InputStream inputStream =null; try { inputStream = imageView.getContext().getContentResolver(). openInputStream(selectedImageUri); }catch(IOException e) {e.printStackTrace();} Bitmap bitmap = BitmapFactory.decodeStream(inputStream); ByteArrayOutputStream byteArrayOutputStream =new ByteArrayOutputStream(); bitmap.compress(Bitmap.CompressFormat.JPEG, 20, byteArrayOutputStream); RequestBody requestBody = RequestBody.create(MediaType.parse(mimeType), byteArrayOutputStream.toByteArray()); MultipartBody.Part uploadFile = MultipartBody.Part.createFormData("fileName", file.getName()+"."+ext, requestBody); try { Response<Message> menuResponse = httpService.menuRequest(uploadFile, map).execute(); }catch (IOException e){ e.printStackTrace(); } } } </pre> |
| 설명 | 메뉴추가 기능은 AVD에 존재하는 이미지와 추가할 메뉴 이름, 메뉴 가격, 메뉴 설명을 입력받아 서버에 전송한다. 이미지를 Bitmap으로 변환 뒤 ByteArray 변환 후 multipart 로 변환 해서 전송한다. |

| 클래스 | OwnerMain.java, DatePickerFragment.java |
|------|--|
| 기능 | 사이드바 메뉴 기능 및 날짜 선택 기능 |
| 담당자 | 구본영 |
| 중요코드 | <pre> public class OwnerMain extends AppCompatActivity { private AppBarConfiguration mAppBarConfiguration; private OwnerMainBinding binding; @Override protected void onCreate(Bundle savedInstanceState) { super.onCreate(savedInstanceState); </pre> |

| | |
|------|---|
| | <pre> binding = OwnerMainBinding.inflate(getLayoutInflater()); setContentView(binding.getRoot()); setSupportActionBar(binding.appBarMain.toolbar); DrawerLayout drawer = binding.drawerLayout; NavigationView navigationView = binding.navView; mAppBarConfiguration =new AppBarConfiguration.Builder(R.id.nav_seat, R.id.nav_menu, R.id.nav_sales) .setOpenableLayout(drawer). build(); NavController navController = Navigation.findNavController(this, R.id.nav_host_fragment_content_main); NavigationUI.setupActionBarWithNavController(this, navController, mAppBarConfiguration); NavigationUI.setupWithNavController(navigationView, navController); } @Override public boolean onSupportNavigateUp() { NavController navController = Navigation.findNavController(this,R.id.nav_host_fragment_content_main); return NavigationUI.navigateUp(navController, mAppBarConfiguration) super.onSupportNavigateUp(); } </pre> |
| 설명 | <p>데이터 바인딩을 사용하여 코드를 간단하게 관리할 수 있도록 하였고 사이드 바 네비게이션을 통해 손쉽게 다른 프래그먼트로 이동이 가능하게 하였다.</p> |
| 중요코드 | <pre> public class DatePickerFragment extends DialogFragment implements DatePickerDialog.OnDateSetListener { @NonNull @Override public Dialog onCreateDialog(Bundle savedInstanceState) { final Calendar c = Calendar.getInstance(); int year = c.get(Calendar.YEAR); int month = c.get(Calendar.MONTH); int day = c.get(Calendar.DAY_OF_MONTH); return new DatePickerDialog(getActivity(),this,year,month,day); } @Override public void onDateSet(DatePicker datePicker, int year, int month, int day) { OwnerMain activity = (OwnerMain)getActivity(); activity.processDatePickerResult(year,month,day); } } </pre> |
| 설명 | <p>DatePickerDialog를 사용하여 원하는 날짜를 선택할 수 있도록 하였다.</p> |
| 중요코드 | <pre> public class OwnerMain extends AppCompatActivity { ... public void showDatePicker(View view) { DialogFragment newFragment =new DatePickerFragment(); newFragment.show(getSupportFragmentManager(),"datePicker"); } @SuppressWarnings("DefaultLocale") public void processDatePickerResult(int year, int month, int day){ String year_string = Integer.toString(year); String day_string =String.format("%02d", day); String month_string =String.format("%02d", month+1); String dateMessage = (year_string+"-"+month_string+"-"+day_string); EditText text = (EditText) findViewById(R.id.check_date); text.setText(dateMessage); } } </pre> |

| | |
|----|---|
| | } |
| 설명 | CheckSalesFragment에서 날짜선택 버튼을 클릭 시 DatePickerDialog를 통해 선택한 날짜를 EditText에 설정한다. |

| | |
|------|---|
| 클래스 | CheckSalesFragment.java |
| 기능 | 매출 조회 기능 |
| 담당자 | 구본영 |
| 중요코드 | <pre> public class CheckSalesFragment extends Fragment implements View.OnClickListener{ ... @Nullable @Override public View onCreateView(@NonNull LayoutInflater inflater, ViewGroup container, Bundle savedInstanceState) { checkSalesViewModel = new ViewModelProvider(this).get(CheckSalesViewModel.class); binding = FragmentOwnerCheckSalesBinding.inflate(inflater, container, false); root = binding.getRoot(); Button showSale_button=binding.checkSales; showSale_button.setOnClickListener(new View.OnClickListener() { @SneakyThrows @Override public void onClick(View view) { EditText text = (EditText) binding.checkDate; date = text.getText().toString(); new Thread(new ConnectDBRunner()).start(); oAdapter.notifyDataSetChanged(); Thread.sleep(500); onAttach(root.getContext()); } }); recyclerView = (RecyclerView) binding.recyclerViewSales; recyclerView.setHasFixedSize(true); RecyclerView.LayoutManager oLayoutManager = new LinearLayoutManager(getActivity()); recyclerView.setLayoutManager(oLayoutManager); oAdapter =new OrdersAdapter(InsertOrderList); recyclerView.setAdapter(oAdapter); return root; } } </pre> |
| 설명 | 버튼을 클릭하면 서버에서 해당 가게의 매출 정보를 가져와 커스텀 리스트뷰에 추가하고 notifyDataSetChanged()를 통해 화면에 커스텀 리스트뷰에 추가한 항목을 표시한다. |
| 중요코드 | <pre> public class CheckSalesFragment extends Fragment implements View.OnClickListener{ String loginId=((MainActivity)MainActivity.context_main).var; ... public class ConnectDBRunner implements Runnable { @Override public void run() { InsertOrderList.clear(); HttpService httpService = HttpClient.getApiService(); try { Response<StoreSignUpDto> storenames = httpService.getStoreName(loginId).execute(); StoreSignUpDto storeSignUpDto = storenames.body(); } } } } </pre> |

| | |
|----|---|
| | <pre> String storeName = storeSignUpDto.getStoreName(); Response<List<OrdersDTO>> order = httpService.getOrder(storeName).execute(); List<OrdersDTO> orderList = order.body(); SimpleDateFormat sdf =new SimpleDateFormat("yyyy-MM-dd"); sdf.setTimeZone(TimeZone.getTimeZone("Asia/Seoul")); Response<List<MenuDto>> menu= httpService.getMenu(loginId).execute(); List<MenuDto> menuList= menu.body(); for(int i=0; i<menuList.size();i++){ InsertOrderList.add(i, new Sales(menuList.get(i).getMenuName(), 0, 0)); } for(int i=0; i<orderList.size(); i++){ if (sdf.format(orderList.get(i).getOrderDate()).equals(date)){ for(int j=0; j<menuList.size(); j++){ if (orderList.get(i).getMenuName(). equals(menuList.get(j).getMenuName())){ InsertOrderList.set(j, new Sales(orderList.get(i).getMenuName(), InsertOrderList.get(j).getMenuPrice() +orderList.get(i).getMenuCount(), orderList.get(i).getMenuPrice())); } } } } for(int i=0; i< InsertOrderList.size();i++){ InsertOrderList.set(i, new Sales(InsertOrderList.get(i).getMenuName(), InsertOrderList.get(i).getMenuPrice(), InsertOrderList.get(i).getMenuPrice() *InsertOrderList.get(i).menuCount)); } } catch (IOException e) { e.printStackTrace(); } } } </pre> |
| 설명 | 로그인 한 가맹점주의 아이디를 가져와 서버에서 가게명 정보와 메뉴정보를 가져오고 가게명 정보를 통해 해당 가게의 주문목록을 서버에서 가져와, 이 정보를 메뉴 정보와 날짜 선택버튼을 통해 EditText에 설정한 날짜를 통해 해당 날짜에 대한 매출정보를 가져오고 반복문을 통해 총 매출을 계산한 정보를 가져온다. |

| 클래스 | SeatManageFragment.java |
|------|---|
| 기능 | 좌석관리 기능 |
| 담당자 | 구본영, 정순범 |
| 중요코드 | <pre> public View onCreateView(@NonNull LayoutInflater inflater, ViewGroup container, Bundle savedInstanceState) { ... @SuppressWarnings("UseSwitchCompatOrMaterialCode") Switch switch1=root.findViewById(R.id.seat_switch1); if(table1==0){ switch1.setChecked(true); Button btn1 = root.findViewById(R.id.seat1); btn1.setBackgroundResource(R.drawable.active_button); } } </pre> |

| | |
|------|--|
| | <pre> } ... public void onCreate(@Nullable Bundle savedInstanceState) { //가맹점주 로그인 후 주문 상태에 따라 테이블 스위치 토글 super.onCreate(savedInstanceState); new Thread(new ConnetGetTableStatus()).start(); } private class ConnetGetTableStatus implements Runnable { @Override public void run() { HttpService httpService= HttpClient.getApiService(); try { Response<StoreSignUpDto> storenames = httpService.getStoreName(loginId).execute(); String storename=storenames.body().getStoreName(); storeName=storename; Response<List<OrdersDTO>> seatOrders= httpService.getTableStatus(storeName,0).execute(); List<OrdersDTO> seatOrder=seatOrders.body(); for(int i=0;i<seatOrder.size();i++){ if(seatOrder.get(i).getOrderState()==0){ if (seatOrder.get(i).getTableNum()==1){table1=0; } if (seatOrder.get(i).getTableNum()==2){table2=0; } if (seatOrder.get(i).getTableNum()==3){table3=0; } if (seatOrder.get(i).getTableNum()==4){table4=0; } } } } catch (IOException e) { e.printStackTrace(); } } } } </pre> |
| 설명 | 좌석 관리화면 진입 시 서버에서 주문 목록을 가져와 현재 사용 중인 테이블을 활성화 배경으로 변경한다. |
| 중요코드 | <pre> public View onCreateView(@NonNull LayoutInflater inflater, ViewGroup container, Bundle savedInstanceState) { ... switch1.setOnCheckedChangeListener (new CompoundButton.OnCheckedChangeListener() { @Override public void onCheckedChanged(CompoundButton buttonView, boolean isChecked) { if (isChecked) { //status Button btn1 = root.findViewById(R.id.seat1); btn1.setBackgroundResource(R.drawable.active_button); } else { Button btn1 = root.findViewById(R.id.seat1); btn1.setBackgroundResource(R.drawable.inactive_button); tableNum =1; new Thread(new ConnectSetState()).start(); new Thread(new ConnectSeatState()).start(); table1 =1; } } }); ... } } </pre> |

| | |
|------|--|
| | <pre> public class ConnectSetState implements Runnable { @Override public void run() { HttpService httpService = HttpClient.getApiService(); try { Response<Message>changeResponse = httpService.updateOrderState(storeName, 1, tableNum).execute(); } catch (IOException e) { e.printStackTrace(); } } } public class ConnectSeatState implements Runnable { @Override public void run() { HttpService httpService = HttpClient.getApiService(); try { Response<Message> changeResponses = httpService.updateSeatState(storeName,1, tableNum).execute(); } catch (IOException e) { e.printStackTrace(); } } } </pre> |
| 설명 | 스위치 버튼을 토글시 테이블을 비활성화 배경으로 변경하며 서버로 현재 테이블이 사용 가능함을 전송한다. |
| 중요코드 | <pre> public View onCreateView(@NonNull LayoutInflater inflater, ViewGroup container, Bundle savedInstanceState) { ... seat1.setOnClickListener(new View.OnClickListener(){ @Override public void onClick(View view) { tableNum=1; new Thread(new ConnectGetOrders()).start(); recyclerView=dialog.findViewById(R.id.recyclerView_table); recyclerView.setHasFixedSize(true); mAdapter =new SeatOrderAdapter(TableOrderList); RecyclerView.LayoutManager mLayoutManager = new LinearLayoutManager(getActivity()); recyclerView.setLayoutManager(mLayoutManager); recyclerView.setAdapter(mAdapter); dialog.show(); } }); ... } public class ConnectGetOrders implements Runnable { @Override public void run() { TableOrderList.clear(); HttpService httpService= HttpClient.getApiService(); try { Response<List<OrdersDTO>> seatOrders= httpService.getSeatOrders(tableNum,0,storeName).execute(); List<OrdersDTO> seatOrder=seatOrders.body(); System.out.println(seatOrder.size()); for(int i=0;i<seatOrder.size();i++){ TableOrderList.add(new SeatOrder(seatOrder.get(i).getMenuName(), seatOrder.get(i).getMenuPrice(), </pre> |

| | |
|----|---|
| | <pre> seatOrder.get(i).getMenuCount()); } } catch (IOException e) { e.printStackTrace(); } } } </pre> |
| 설명 | 테이블 버튼을 터치 시 터치한 테이블의 주문 정보를 서버로부터 가져와 커스텀한 다이얼로그의 커스텀 리스트뷰를 통하여 화면에 표시한다. |

3.1.2 사용자 기능

| 클래스 | ScanQRActivirty.java |
|------|--|
| 기능 | QR 코드 인식 |
| 담당자 | 김기태 |
| 중요코드 | <pre> public class ScanQRActivity extends AppCompatActivity { private IntentIntegrator qrScan; @Override protected void onCreate(Bundle savedInstanceState) { super.onCreate(savedInstanceState); setContentView(R.layout.user_main); qrScan =new IntentIntegrator(this); qrScan.setPrompt("Scanning..."); qrScan.setOrientationLocked(false); qrScan.setCaptureActivity(CaptureActivity.class); qrScan.initiateScan(); } //Getting the scan results @Override protected void onActivityResult(int requestCode, int resultCode, @Nullable Intent data){ IntentResult result = IntentIntegrator.parseActivityResult(requestCode, resultCode, data); if (result !=null) { //qrcode 가 없으면 if (result.getContents() ==null) { Toast.makeText(ScanQRActivity.this, "취소!", Toast.LENGTH_SHORT).show(); ScanQRActivity.this.finish(); } else { Toast.makeText(ScanQRActivity.this, "스캔완료!", Toast.LENGTH_SHORT).show(); //액티비티 이동 및 데이터 전송 Intent intents =new Intent(getApplicationContext(), TabActivity.class); intents.putExtra("store_name", result.getContents().split(",")[0]); intents.putExtra("table_num", result.getContents().split(",")[1]); startActivity(intents); ScanQRActivity.this.finish(); } } else { super.onActivityResult(requestCode, resultCode, data); } } @Override public void onBackPressed() { super.onBackPressed(); finish(); } } </pre> |
| 설명 | <p>QR 코드를 인식하기 위해 카메라 화면으로 이동한다.</p> <p>QR 코드를 인식하면 '스캔완료!'를 Toast로 출력하며, QR에 담긴 데이터를 읽어 매장 화면으로 이동한다.</p> |

| 클래스 | TabActivity.java & FragmentAdapter.java |
|------|--|
| 기능 | 탭 화면 구현 |
| 담당자 | 김기태 |
| 중요코드 | <pre> public class TabActivity extends AppCompatActivity { private TabLayout tabLayout; </pre> |

| | |
|------|---|
| | <pre> private ViewPager viewPager; private FragmentAdapter adapter; private String table_num; private String storeName; String loginId=((MainActivity)MainActivity.context_main).var; @Override protected void onCreate(Bundle savedInstanceState) { super.onCreate(savedInstanceState); setContentView(R.layout.show_menu); tabLayout = findViewById(R.id.tabs); viewPager = findViewById(R.id.pager); TextView tx = (TextView) findViewById(R.id.set_store_name); Intent intent = getIntent(); storeName = intent.getStringExtra("store_name"); tx.setText(storeName); table_num = intent.getStringExtra("table_num"); adapter =new FragmentAdapter(getSupportFragmentManager(), 1); //FragmentAdapter에 컬렉션 담기 adapter.addFragment(new MenuListActivity()); adapter.addFragment(new BasketListActivity()); //ViewPager Fragment 연결 viewPager.setAdapter(adapter); //ViewPager과 TabLayout 연결 tabLayout.setupWithViewPager(viewPager); tabLayout.getTabAt(0).setText("메뉴"); tabLayout.getTabAt(1).setText("장바구니"); } </pre> |
| 설명 | 가게 내 메뉴의 리스트와 선택한 메뉴를 담은 장바구니를 구현하기 위해 뷰 페이지를 이용하여 탭 화면을 구성하였다. QR 코드 인식 시 데이터를 넘겨받아 가게 이름을 확인하며 앉은 테이블의 번호를 결제 시 확인하기 위해 변수로 저장한다. |
| 중요코드 | <pre> public class FragmentAdapter extends FragmentPagerAdapter { private List<Fragment> fragmentList =new ArrayList<>(); public FragmentAdapter(@NonNull FragmentManager fm, int behavior) { super(fm, behavior); } public void addFragment(Fragment fragment){ fragmentList.add(fragment); } @NonNull @Override public Fragment getItem(int position) { return fragmentList.get(position); } @Override public int getCount() { return fragmentList.size(); } @Override public int getItemPosition(@NonNull Object object) { return POSITION_NONE; } } </pre> |
| 설명 | 뷰 페이지에서 프래그먼트를 설정하기 위해 Adapter를 구현하여 선택한 프래그먼트가 뷰 페이지에 삽입할 수 있게 구현한다. |

| | |
|-----|----------------------|
| 클래스 | MenuListAdapter.java |
| 기능 | 메뉴 리스트 뷰 어댑터 |
| 담당자 | 김기태 |

| | |
|------|---|
| 중요코드 | <pre> public class MenuListAdapter extends BaseAdapter { private ArrayList<Menuitem> menuitems =new ArrayList<>(); private Button listButton; @Override public View getView(int i, View view, ViewGroup viewGroup) { Context context = viewGroup.getContext(); if(view ==null) { LayoutInflater inflater = (LayoutInflater) context.getSystemService(Context.LAYOUT_INFLATER_SERVICE); view = inflater.inflate(R.layout.menu_list_item, viewGroup, false); } ImageView iv_image = (ImageView) view.findViewById(R.id.iv_menu_image); TextView tv_name = (TextView) view.findViewById(R.id.tv_menu_name); TextView tv_contents = (TextView) view.findViewById(R.id.tv_menu_contents); TextView tv_cost = (TextView) view.findViewById(R.id.tv_menu_cost); Menuitem menuitem = (Menuitem) getItem(i); iv_image.setImageBitmap(menuitem.getMenuPicture()); tv_name.setText(menuitem.getMenuName()); tv_contents.setText(menuitem.getMenuDesc()); tv_cost.setText(menuitem.getMenuPrice()); listButton = (Button) view.findViewById(R.id.menu_list_button); listButton.setTag(menuitem); return view; } public void addItem(Menuitem item){ menuitems.add(item); } } </pre> |
| 설명 | 가게 메뉴를 아이템으로 넣기 위해 커스텀 리스트 뷰를 구현하며, 아이템을 리스트뷰에 매칭하기 위해 어댑터를 구현하여 각 내용이 리스트 아이템에 들어갈 수 있게 매핑 시키는 역할을 한다. |

| 클래스 | MenuListActivity.java |
|------|---|
| 기능 | 메뉴 리스트 뷰 구현 |
| 담당자 | 김기태 |
| 중요코드 | <pre> public class ConnectGetRunner implements Runnable { TextView tv = ((TabActivity) getActivity()).findViewById(R.id.set_store_name); String storeName = tv.getText().toString(); @Override public void run() { InsertMenuList.clear(); HttpService httpService= HttpClient.getApiService(); try { Response<List<ShopMenuitem>> menu=httpService.getMenus(storeName).execute(); List<ShopMenuitem> menuList= menu.body(); Bitmap bitmap =null; for(int i=0;i<menuList.size();i++) { String menuName=menuList.get(i).getMenuName(); String menuCost=menuList.get(i).getMenuPrice(); String menuDesc=menuList.get(i).getMenuDesc(); try{ URL myFileUrl =new URL("http://10.0.2.2:8080/img?storeName="+ storeName +"&menuName="+ menuName); HttpURLConnection conn = (HttpURLConnection)myFileUrl.openConnection(); </pre> |

| | |
|------|--|
| | <pre> conn.setDoInput(true); conn.connect(); InputStream is = conn.getInputStream(); bitmap = BitmapFactory.decodeStream(is); is.close(); } catch (IOException e) { e.printStackTrace(); } InsertMenuList.add(new MenuItem(bitmap, menuName, menuDesc, menuCost)); } } catch (IOException e) { e.printStackTrace(); } } </pre> |
| 설명 | 서버로부터 MENU 테이블의 정보를 받아오며 데이터를 담은 리스트를 구현하여 안드로이드 로이드에서 사용하게 구현하였다. |
| 중요코드 | <pre> private void dataSetting(){ MenuListAdapter adapter =new MenuListAdapter(); for (int i =0; i < InsertMenuList.size(); i++) { adapter.addItem(InsertMenuList.get(i)); } listView.setAdapter(adapter); } </pre> |
| 설명 | 서버에서 받아온 데이터의 리스트로부터 리스트 뷰 어댑터의 add함수를 이용하여 각 아이템을 리스트에 삽입하는 역할을 담당한다. |
| 중요코드 | <pre> public void click(View v) { if (v.getId() == R.id.menu_list_button){ Bundle bundle =new Bundle(); MenuItem dto = (MenuItem) v.getTag(); bundle.putString("Name", dto.getMenuName()); bundle.putString("Cost", dto.getMenuPrice()); getSupportFragmentManager().setFragmentResult("requestKey", bundle); } } </pre> |
| 설명 | 메뉴 리스트의 '+' 버튼을 누르면 장바구니 프래그먼트에 값을 보내주기 위해 bundle로 구현하여 데이터를 전송한다. |

| 클래스 | BasketListActivity.java |
|------|--|
| 기능 | 장바구니 리스트 뷰 구현 |
| 담당자 | 김기태 |
| 중요코드 | <pre> @Override public void onCreateView(@NonNull View view, @Nullable Bundle savedInstanceState) { super.onCreateView(view, savedInstanceState); // 메뉴 리스트에서 전송한 값을 받아옴 getParentFragmentManager().setFragmentResultListener("requestKey", this, new FragmentResultListener() { @Override public void onFragmentResult(@NonNull String requestKey, @NonNull Bundle result) { String name = result.getString("Name"); String cost = result.getString("Cost"); int len = listView.getCount(); if (len >0) { for (int i =0; i < len; i++) { </pre> |

| | |
|------|--|
| | <pre> BasketItem item = (BasketItem) listView.getItemAtPosition(i); // 장바구니에 이미 존재하는 메뉴면 i f (item.getMenuName().equals(name)) { int num = Integer.parseInt(item.getMenuNum()); int mcost = Integer.parseInt(item.getMenuCost()); num +=1; int total = mcost * num; item.setMenuNum(Integer.toString(num)); item.setMenuTotal(Integer.toString(total)); listView.setAdapter(adapter); break; }else if (i == len-1){ dataSetting(name, cost, "1"); } } }else{ dataSetting(name, cost, "1"); } } }); } </pre> |
| 설명 | 메뉴 탭에서 원하는 메뉴를 '+' 버튼을 눌러 장바구니로 보낼 때 해당 메뉴의 값을 전 송받아 장바구니 리스트에 아이템을 생성하는 역할을 한다. 이미 존재하는 메뉴면 개수 와 총 가격을 더해준다. |
| 중요코드 | <pre> basketHolder.add_btn.setOnClickListener(new View.OnClickListener() { @Override public void onClick(View v) { int num = Integer.parseInt(basketHolder.menuNum.getText().toString()+1; int cost = Integer.parseInt(basketHolder.menuCost.getText().toString()); basketItem.setMenuNum(Integer.toString(num)); basketItem.setMenuTotal(Integer.toString(cost * num)); basketHolder.menuNum.setText(Integer.toString(num)); basketHolder.menuTotal.setText(Integer.toString(cost * num)); click.onClick(Integer.toString(cost*num)); } }); </pre> |
| 설명 | 장바구니 리스트 아이템 내 '+' 버튼을 누르면 아이템의 개수와 가격이 추가될 수 있게 구현하였다. |
| 중요코드 | <pre> basketHolder.minus_btn.setOnClickListener(new View.OnClickListener() { @Override public void onClick(View v) { int num = Integer.parseInt(basketHolder.menuNum.getText().toString()-1; int cost = Integer.parseInt(basketHolder.menuCost.getText().toString()); if(num ==0){ removeItem(basketItem); notifyDataSetChanged(); } basketItem.setMenuNum(Integer.toString(num)); basketItem.setMenuTotal(Integer.toString(cost * num)); basketHolder.menuNum.setText(Integer.toString(num)); basketHolder.menuTotal.setText(Integer.toString(cost * num)); click.onClick(Integer.toString(cost*num)); } }); </pre> |
| 설명 | 장바구니 리스트 아이템 내 '-' 버튼을 누르면 아이템의 개수와 가격이 차감될 수 있게 |

| | |
|--|---|
| | 구현하였다. 아이템의 개수가 0이 되면 아이템이 삭제되어 장바구니에서 빠지게 구현하였다. |
|--|---|

| 클래스 | TabActivity.java |
|------|---|
| 기능 | 결제 기능 |
| 담당자 | 김기태 |
| 중요코드 | <pre> public void pay(View v){ TextView tv = v.findViewById(R.id.Total); String pay = tv.getText().toString().split(" ")[0]; AlertDialog.Builder oDialog = new AlertDialog.Builder(this, R.style.Base_Theme_AppCompat_Light_Dialog); oDialog.setMessage("결제 요청 금액 : "+pay) .setTitle("결제 요청") .setPositiveButton("아니오", new DialogInterface.OnClickListener(){ @Override public void onClick(DialogInterface dialog, int which) { Log.i("Dialog", "취소"); } }) .setNegativeButton("예", new DialogInterface.OnClickListener() { @Override public void onClick(DialogInterface dialog, int which) { new Thread(new ConnectGetRunner()).start(); TabActivity.this.finish(); } }) .setCancelable(false) .show(); } </pre> |
| 설명 | 장바구니의 결제 버튼을 클릭하면 다이얼로그 창이 화면위에 떠오르며 결제 여부를 확인한다. 결제를 취소하면 다시 화면으로, 결제 승인하면 유저의 기본 화면으로 이동된다. |
| 중요코드 | <pre> public class ConnectGetRunner implements Runnable { @SneakyThrows @Override public void run() { ListView listView = findViewById(R.id.basket_list); HttpService httpService= HttpClient.getApiService(); for (int i =0; i<listView.getCount();i++){ BasketItem dto = (BasketItem) listView.getItemAtPosition(i); Date today =new Date(); String temp = today.toString(); OrdersDTO ordersDTO =new OrdersDTO(loginId, storeName, dto.getMenuName(), Integer.parseInt(dto.getMenuCost()), Integer.parseInt(dto.getMenuNum()), null, Integer.parseInt(table_num), 1); try { Response<Message> order = httpService.addOrder(temp, ordersDTO).execute(); Response<Message> seat = httpService.updateSeatStateToUse(storeName, Integer.parseInt(table_num)).execute(); }catch (IOException e){ e.printStackTrace(); } } } } </pre> |

| | |
|----|---|
| | <pre> } } </pre> |
| 설명 | 장바구니의 결제 버튼을 클릭한 뒤 결제 승인을 누르면 유저 정보와 메뉴, 가격, 테이블 번호 등 서버로 주문 정보를 담은 객체를 전송하여 ORDERS 테이블에 넣는 역할을 담당한다. |

| | |
|------|---|
| 클래스 | SignUpUserRequest.java |
| 기능 | 일반 사용자 회원가입 기능 |
| 담당자 | 안대현 |
| 중요코드 | <pre> public class ConnectRunner implements Runnable { @Override public void run() { HttpService httpService = HttpClient.getApiService(); UserSignUpDto userSignUpDto =new UserSignUpDto(userId, userPwd, userPwd2, userName, userHP, Role.ROLE_U SER); Response<Message> loginResponse =null; try { loginResponse = httpService.UserSignUpRequest(userSignUpDto).e xecute(); }catch (IOException e){ e.printStackTrace(); } String status = loginResponse.body().getMessage(); if (!status.equals("가입 성공")) { Handler handler=new Handler(Looper.getMainLooper()); handler.postDelayed(new Runnable() { @Override public void run() { AlertDialog.Builder builder =new AlertDialog.Builder(SignUp UserRequest.this); builder.setTitle("SignUP Failed").setMessage(status+"\\n"+" 다시 입력 해주세요."); AlertDialog alertDialog = builder.create(); alertDialog.show(); } },0); } else { Intent intent =new Intent(getApplicationContext(), MainActivity.clas s); startActivity(intent); } } } </pre> |
| 설명 | 일반 사용자로 회원가입 할 경우, 사용자가 입력한 정보를 UserSignUpDto 객체로 만들어 서버에 전송한다. 반환 메시지에 따라 가입에 성공했는지 실패했는지를 판단하고, 가입에 실패한 경우 AlertDialog로 실패한 이유를 사용자에게 보여주도록 한다. |

| | |
|------|---|
| 클래스 | MapActivity.java |
| 기능 | 매장 검색 기능 |
| 담당자 | 안대현 |
| 중요코드 | <pre> @Override public void onMapReady(@NonNull NaverMap naverMap) { this.naverMap = naverMap; } </pre> |

| | |
|------|---|
| | <pre> naverMap.setOnMapClickListener((coord, point) -> { if (storeInfo_layout !=null && storeInfo_layout.getVisibility() == View.VIS IBLE) { storeInfo_layout.setVisibility(View.INVISIBLE); } }); for (int i =0; i < storeList.size(); i++) { Marker mk =new Marker(); mk.setPosition(new LatLng(storeList.get(i).storeLatitude, storeList.get(i). storeLongitude)); mk.setMap(naverMap); mk.setCaptionText(storeList.get(i).storeName); mk.setIcon(OverlayImage.fromResource(R.drawable.marker_resize)); marker.add(mk); } Overlay.OnClickListener listener = overlay -> { Marker overlay_marker = (Marker)overlay; if (storeInfo_layout !=null && storeInfo_layout.getVisibility() == View.VIS IBLE) { storeInfo_layout.setVisibility(View.INVISIBLE); } else if (storeInfo_layout ==null storeInfo_layout.getVisibility() == Vie w.INVISIBLE) { for (int i =0; i < storeList.size(); i++) { if (marker.get(i) == overlay_marker) { showStoreInfo(storeList.get(i).storeName, storeList.get(i).st oreLoc, storeList.get(i).storeHp); } } } return true; }; </pre> |
| 설명 | <p>서버에서 불러온 가맹점들의 위도, 경도를 기반으로 가맹점들의 위치를 마커로 표시하는 코드이다.</p> <p>마커의 클릭 이벤트로 showStoreInfo 함수를 연결하여 가맹점 정보가 오버레이 될 수 있도록 하였다.</p> |
| 중요코드 | <pre> public void showStoreInfo(String selectStoreName, String selectStoreLoc, String selectStoreHP) { LayoutInflater inflater = (LayoutInflater) getSystemService(Context.LAYOUT_I NFLATER_SERVICE); // store_info 레이아웃 객체 생성 storeInfo_layout = (LinearLayout)inflater.inflate(R.layout.store_info, null); TextView storeName = (TextView) storeInfo_layout.findViewById(R.id.storeN ame); storeName.setText(selectStoreName); TextView storeLoc = (TextView) storeInfo_layout.findViewById(R.id.storeLo c); storeLoc.setText(selectStoreLoc); TextView storeHP = (TextView) storeInfo_layout.findViewById(R.id.storeHP); storeHP.setText(selectStoreHP); // 현재 레이아웃 위에 store_info 레이아웃 겹치기 LinearLayout.LayoutParams paramll =new LinearLayout.LayoutParams (LinearLayout.LayoutParams.FILL_PARENT,LinearLayout.LayoutPara ms.FILL_PARENT); addContentView(storeInfo_layout, paramll); } </pre> |
| 설명 | <p>마커를 클릭하면 화면 하단부에 매장 정보가 표시되는 레이아웃이 오버레이 되도록 하는 코드이다. 해당 레이아웃에 매장 이름과 위치, 전화번호가 표시된다.</p> |
| 중요코드 | @Override |

| | |
|------|---|
| | <pre> protected void onCreate(Bundle savedInstanceState) { super.onCreate(savedInstanceState); setContentView(R.layout.search_shop); searchView = findViewById(R.id.searchView); searchView.setOnQueryTextListener(new SearchView.OnQueryTextListener() { @Override public boolean onQueryTextSubmit(String s) { for (int i =0; i < storeList.size(); i++) { // 검색창이 공백인 경우 if (s.equals("")) { naverMap.setLocationTrackingMode(LocationTrackingMode.Follow); searchView.clearFocus(); storeInfo_layout.setVisibility(View.INVISIBLE); break; } // 검색 값과 일치하는 가게명이 있는 경우 if (storeList.get(i).storeName.equals(s)) { CameraPosition cameraPosition = new CameraPosition(new LatLng(storeList.get(i).storeLatitude, storeList.get (i).storeLongitude), 17); naverMap.moveCamera(CameraUpdate.toCameraPosition(c ameraPosition)); for (int j =0; j < storeList.size(); j++) { marker.get(j).setMap(null); } marker.get(i).setMap(naverMap); searchView.clearFocus(); showStoreInfo(storeList.get(i).storeName, storeList.get(i).st oreLoc, storeList.get(i).storeHp); break; } } } } </pre> |
| 설명 | 매장 이름을 검색한 경우, 해당 매장으로 카메라를 포커스하고, 매장 정보를 표시하기 위한 코드이다. 매장 정보는 showStoreInfo 함수를 통해 표시한다. |
| 중요코드 | <pre> public class ConnectRunner implements Runnable { @Override public void run() { HttpService httpService = HttpClient.getApiService(); try { Response<List<Stores>> store = httpService.getStore().execute(); storeList = store.body(); } catch (IOException e) { e.printStackTrace(); } } } </pre> |
| 설명 | 가맹점들의 가게 이름, 위치 정보, 전화번호를 서버로부터 받아오는 코드이다. 가맹 점들의 정보를 Stores 객체의 리스트로 받아와 storeList에 저장한다. |

| | |
|------|--|
| 클래스 | ShopMenuListActivity.java |
| 기능 | 매장 메뉴 확인 기능 |
| 담당자 | 안대현 |
| 중요코드 | <pre> public class ConnectGetRunner implements Runnable { TextView tv = ((ShopTabActivity) getActivity()).findViewById(R.id.set_store_n ame); String storeName = tv.getText().toString(); </pre> |


| | |
|------|---|
| | <pre> @Override public void run() { InsertMenuList.clear(); HttpService httpService= HttpClient.getApiService(); try { Response<List<ShopMenuItem>> menu=httpService.getMenus(store Name).execute(); List<ShopMenuItem> menuList= menu.body(); Bitmap bitmap =null; for(int i=0;i<menuList.size();i++) { String menuName=menuList.get(i).getMenuName(); String menuCost=menuList.get(i).getMenuPrice(); String menuDesc=menuList.get(i).getMenuDesc(); try{ URL myFileUrl =new URL("http://10.0.2.2:8080/img?storeN ame="+ storeName +"&menuName="+ menuName); HttpURLConnection conn = (HttpURLConnection)myFileUrl. openConnection(); conn.setDoInput(true); conn.connect(); InputStream is = conn.getInputStream(); bitmap = BitmapFactory.decodeStream(is); is.close(); }catch(IOException e){ e.printStackTrace(); } InsertMenuList.add(new ShopMenuItem(bitmap, menuName, m enuDesc, menuCost)); } </pre> |
| 설명 | 서버로부터 메뉴 정보를 불러오는 코드이다. 선택한 매장의 메뉴 목록을 ShoMenuItem 객체에 담아 리스트로 받아온 뒤, 이미지와 함께 InsertMenuList에 저장한다. |
| 중요코드 | <pre> @Nullable @Override public View onCreateView(@NonNull LayoutInflater inflater, @Nullable ViewGrou p container, @Nullable Bundle savedInstanceState) { View v = inflater.inflate(R.layout.shop_menu_list, container, false); listView = (ListView) v.findViewById(R.id.menu_shop_list); dataSetSetting(); return v; } private void dataSetSetting(){ ShopMenuListAdapter adapter =new ShopMenuListAdapter(); for (int i =0; i < InsertMenuList.size(); i++) { adapter.addItem(InsertMenuList.get(i)); } listView.setAdapter(adapter); } </pre> |
| 설명 | 어댑터를 사용하여 각각의 메뉴 이미지, 메뉴명, 설명, 가격을 리스트 뷰에 매핑한다. |

| 클래스 | ShopSeatActivity.java |
|------|--|
| 기능 | 매장 좌석 확인 기능 |
| 담당자 | 안대현 |
| 중요코드 | <pre> @Override public void onResume() { View v = getView(); TextView[] seat = {v.findViewById(R.id.seat1), v.findViewById(R.id.seat2), v.findViewById(R.id.seat3), v.findViewById(R.id.seat4)}; for(int i =0; i <4; i++) { if(table[i]==1){ </pre> |

| | |
|------|---|
| | <pre> seat[i].setBackgroundResource(R.drawable.active_button); } else{ seat[i].setText("사용중"); } } super.onResume(); } </pre> |
| 설명 | 서버에서 불러온 좌석 현황에 따라 좌석의 사용 여부를 변경하는 코드이다. |
| 중요코드 | <pre> private class ConnetGetTableStatus implements Runnable { TextView tv = ((ShopTabActivity) getActivity()).findViewById(R.id.set_store_n ame); String storeName = tv.getText().toString(); @Override public void run() { HttpService httpService= HttpClient.getApiService(); try { Response<List<SeatDTO>> seatOrders=httpService.getSeat(storeNa me).execute(); List<SeatDTO> seatOrder=seatOrders.body(); for(int i=0;i<seatOrder.size();i++){ if(seatOrder.get(i).tableStatus ==0){ table[i] =0; } else { table[i] =1; } } } catch (IOException e) { e.printStackTrace(); } } } </pre> |
| 설명 | 서버로부터 매장의 좌석 현황을 불러오는 코드이다. 좌석 정보를 SeatDTO 객체의 리스트로 받아와 좌석이 사용 가능한지 불가능한지의 상태 값을 조회한다. |

3.2 기능 동작 화면

3.2.1 가맹점 기능

| 가맹점주 회원가입 기능 | | | | | | | | | | | | | | | | | | | | | | | | | | | |
|--------------|--|---|----------|--------------------------|------------------|-----------------|--------------------------|-----------|-----------------|------------|---|---|--------|----------|---------------|------------------|---------|--------------------------|----|----------|-----------|------|---------|---|---|--------|---------|
| 실행결과 |  | <p>위 화면은 가맹점주 회원가입 페이지 화면으로 아이디, 비밀번호, 확인 비밀번호, 사업자 번호, 가맹점 명, 가맹점 위치, 가맹점주 이름, 가맹점 전화번호를 입력한다.</p> <p>회원가입 버튼을 누르면 owner 테이블과 store 테이블에 정보가 나뉘어져 들어간다. owner 테이블에는 가맹점주 정보가 들어가고 store 테이블에는 가맹점 정보가 들어간다.</p> | | | | | | | | | | | | | | | | | | | | | | | | | |
| | DB | <div>store</div> <table><tr><th>id</th><th>owner_id</th><th>owner_num</th><th>store_hp</th><th>store_latitude</th><th>store_loc</th><th>store_longitude</th><th>store_name</th></tr><tr><td>1</td><td>2</td><td>testId</td><td>20173212</td><td>010-8888-7777</td><td>35.1443184545334</td><td>엄광로 176</td><td>129.036309884005 모바일 맛있닭</td></tr></table> <div>owner</div> <table><tr><th>id</th><th>owner_id</th><th>owner_pwd</th><th>role</th><th>storehp</th></tr><tr><td>1</td><td>2</td><td>testId</td><td>testPwd</td><td>ROLE_OWNER 010-8888-7777</td></tr></table> | id | owner_id | owner_num | store_hp | store_latitude | store_loc | store_longitude | store_name | 1 | 2 | testId | 20173212 | 010-8888-7777 | 35.1443184545334 | 엄광로 176 | 129.036309884005 모바일 맛있닭 | id | owner_id | owner_pwd | role | storehp | 1 | 2 | testId | testPwd |
| id | owner_id | owner_num | store_hp | store_latitude | store_loc | store_longitude | store_name | | | | | | | | | | | | | | | | | | | | |
| 1 | 2 | testId | 20173212 | 010-8888-7777 | 35.1443184545334 | 엄광로 176 | 129.036309884005 모바일 맛있닭 | | | | | | | | | | | | | | | | | | | | |
| id | owner_id | owner_pwd | role | storehp | | | | | | | | | | | | | | | | | | | | | | | |
| 1 | 2 | testId | testPwd | ROLE_OWNER 010-8888-7777 | | | | | | | | | | | | | | | | | | | | | | | |

로그인 기능


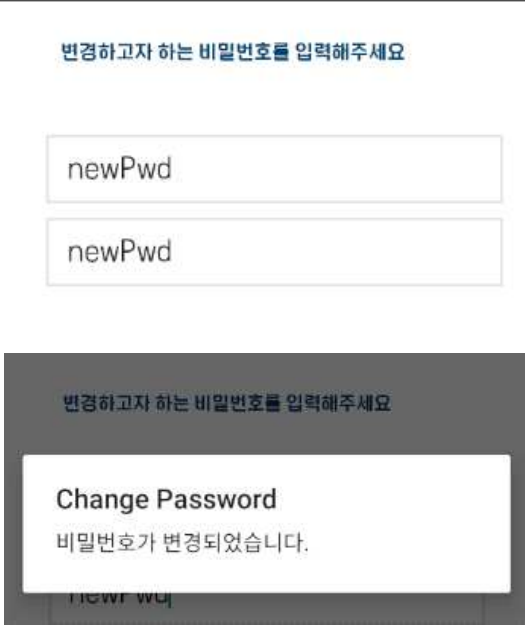
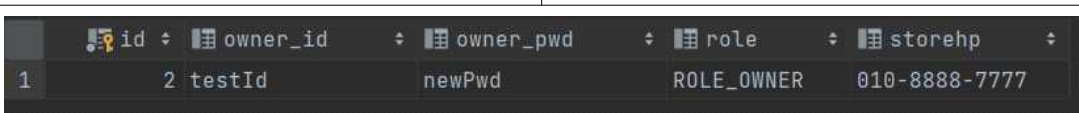
실행결과



위 화면은 어플을 처음 실행시켰을 때 보이는 로그인 화면으로 가맹점주, 일반 사용자를 선택해 로그인 가능하다. 서버에 저장된 회원정보 중 역할에 따라 로그인 성공시 전환되는 화면이 다르다.



위 화면은 로그인 실패시 보이는 화면이다. 서버에 저장된 회원정보 중 아이디와 비밀번호를 조회후 일치할 때 로그인이 진행되며 그렇지 않으면 로그인이 되지 않고 커스텀 다이얼로그 화면이 보인다.

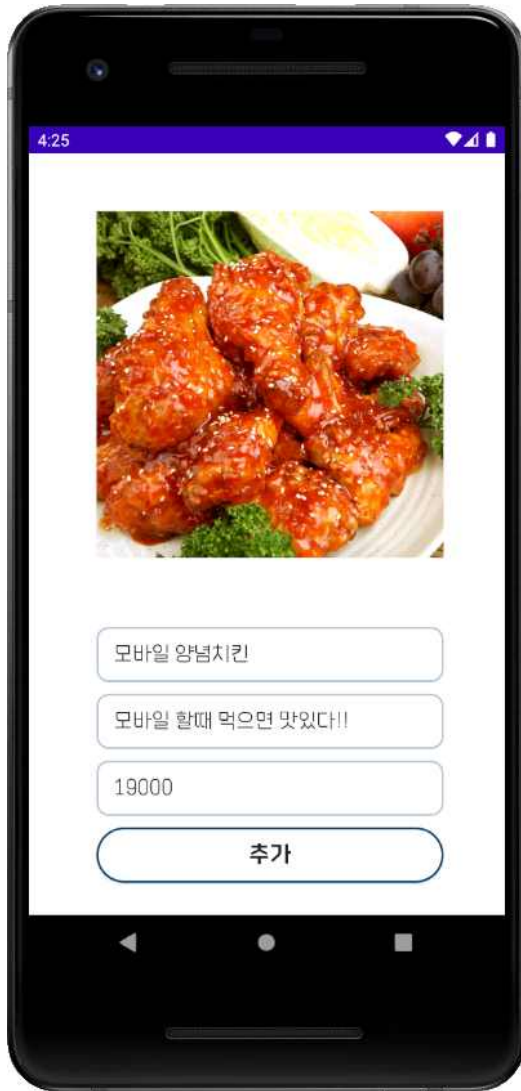
| 비밀번호 재설정 기능 | | | | | | | | | | | | |
|-------------|--|--|------------|---------------|-----------|------|---------|---|----------|--------|------------|---------------|
| 실행결과 |  | <p>위 화면은 비밀번호 재설정 화면이다. 변경할 비밀번호와 비밀번호 재확인 비밀번호가 같아야 변경이 가능하다.</p> | | | | | | | | | | |
| |  | <p>위 화면은 새로 변경할 비밀번호를 입력 후 변경하기 버튼을 누른 화면이다. 입력한 두 필드의 값이 같으면 정상적으로 비밀번호가 변경되었다고 다이얼로그 창이 보인다.</p> | | | | | | | | | | |
| DB |  <table><tr><th>id</th><th>owner_id</th><th>owner_pwd</th><th>role</th><th>storehp</th></tr><tr><td>1</td><td>2 testId</td><td>newPwd</td><td>ROLE_OWNER</td><td>010-8888-7777</td></tr></table> | | id | owner_id | owner_pwd | role | storehp | 1 | 2 testId | newPwd | ROLE_OWNER | 010-8888-7777 |
| id | owner_id | owner_pwd | role | storehp | | | | | | | | |
| 1 | 2 testId | newPwd | ROLE_OWNER | 010-8888-7777 | | | | | | | | |

| 사이드바 메뉴 | |
|---------|--|
| 실행결과 | <ul style="list-style-type: none"> - 가맹점주로 로그인시 원활한 화면 전환을 위해 사이드바 네비게이션 메뉴를 사용한다. |
| | <ol style="list-style-type: none"> 1. 앱바의 메뉴 아이콘 터치 시 사이드바 네비게이션 메뉴가 좌측에서 나와 원하는 항목으로 손쉽게 이동할 수 있다. |



가맹점주 메뉴 추가 기능

실행결과



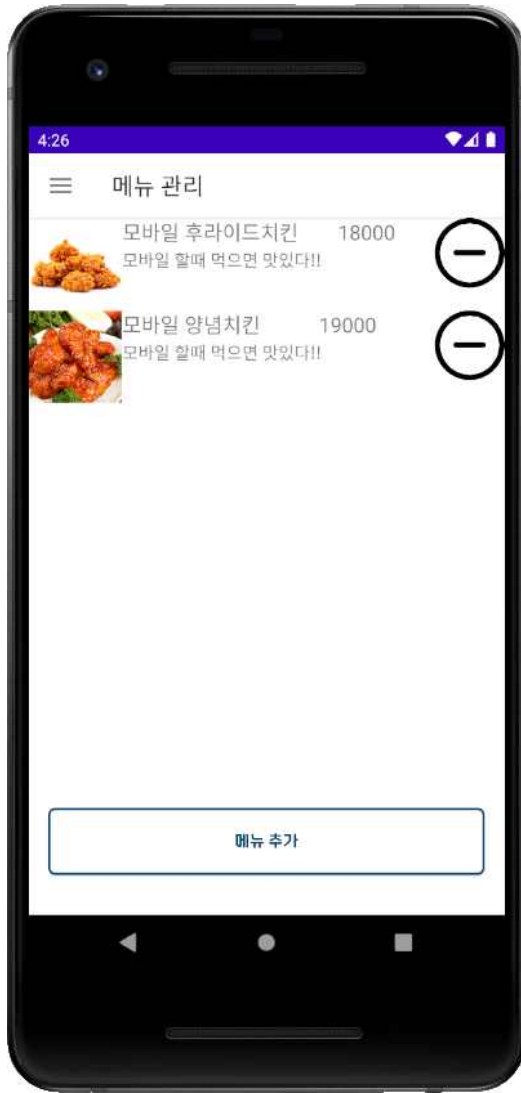
위 화면은 메뉴 추가 화면이다. ImageView를 클릭하면 AVD 내부에 존재하는 이미지가 보이며 선택시 이미지가 올라온다. 메뉴 이름, 메뉴 설명, 가격을 입력하고 추가 버튼을 누르면 메뉴가 추가된다. 서버에는 메뉴 이미지가 저장된 주소가 저장된다.

DB

| | Image | id | login_id | menu_desc | menu_name | menu_p... | store_name |
|---|-------------------------------|----|----------|----------------|-----------|-----------|------------|
| 1 | C:\menu\testId\149423224.jpeg | 3 | testId | 모바일 할때 먹으면 맛있다 | 모바일 양념치킨 | 19000 | 모바일 맛있다 |

가맹점주 메뉴 조회 기능

실행결과



위 화면은 추가된 메뉴 목록을 보여준다. 리사이클러뷰 안에 커스텀 리스트뷰로 구성되어 있다. 왼쪽에 마이너스 버튼을 누르면 메뉴가 삭제된다.

DB

| | id | image | login_id | menu_desc | menu_name | menu_price | store_name |
|---|----|--------------------------------|----------|--------------|------------|------------|------------|
| 1 | 1 | C:\menu\testId\1576756365.jpeg | testId | 모바일 할때 먹으면 맛 | 모바일 후라이드치킨 | 18000 | 모바일 맛있는 |
| 2 | 2 | C:\menu\testId\55944774.jpeg | testId | 모바일 할때 먹으면 맛 | 모바일 양념치킨 | 19000 | 모바일 맛있는 |

| 좌석 관리 메뉴 | |
|----------|---|
| 실행결과 |  <p>onCreate()에서 서버에서 주문 정보를 받아와 onCreateView에서 고객이 이용중인 테이블 버튼의 색상과 스위치 버튼을 활성화 시킨다.</p> |
| DB | <pre> id : menu_count : menu_name : menu_price : order_date : order_state : store_name : table_num : user_id : 1 1 3 모바일 후라이드치킨 18000 2021-12-06 01:28:43 0 모바일 맛있달 1 testUser 2 2 2 모바일 양념치킨 19000 2021-12-06 01:30:21 0 모바일 맛있달 2 testUser </pre> |

| 실행결과 |  <p>테이블 버튼 터치 시 해당 테이블의 주문 목록을 확인할 수 있다.</p> | | | | | | | | | | | | | | | | | | |
|------|---|-----------|------------|---------------------|-------------|------------|-------------|------------|-----------|---------|---|---|----------|-------|---------------------|---|---------|--|------------|
| DB | <table><tr><th>id</th><th>menu_count</th><th>menu_name</th><th>menu_price</th><th>order_date</th><th>order_state</th><th>store_name</th><th>table_num</th><th>user_id</th></tr><tr><td>1</td><td>2</td><td>모바일 양념치킨</td><td>19000</td><td>2021-12-06 01:30:21</td><td>0</td><td>모바일 맛있답</td><td></td><td>2 testUser</td></tr></table> | id | menu_count | menu_name | menu_price | order_date | order_state | store_name | table_num | user_id | 1 | 2 | 모바일 양념치킨 | 19000 | 2021-12-06 01:30:21 | 0 | 모바일 맛있답 | | 2 testUser |
| id | menu_count | menu_name | menu_price | order_date | order_state | store_name | table_num | user_id | | | | | | | | | | | |
| 1 | 2 | 모바일 양념치킨 | 19000 | 2021-12-06 01:30:21 | 0 | 모바일 맛있답 | | 2 testUser | | | | | | | | | | | |

실행결과



활성화된 테이블의 스위치 버튼을 터치 시 테이블 버튼의 색상이 변한다. 또한 서버로 주문 상태를 변경시키며, 해당 테이블이 주문을 다시 받을 수 있는 상태임을 서버로 전송한다.

DB

| id | menu_count | menu_name | menu_price | order_date | order_state | store_name | table_num | user_id |
|----|------------|--------------|--------------|---------------------|-------------|------------|-----------|----------|
| 1 | 2 | 2 모바일 양념치킨 | 19000 | 2021-12-06 01:30:21 | 1 | 모바일 맛있달 | 2 | testUser |
| id | store_name | table_number | table_status | | | | | |
| 1 | 1 모바일 맛있달 | 1 | 0 | | | | | |
| 2 | 2 모바일 맛있달 | 2 | 1 | | | | | |
| 3 | 3 모바일 맛있달 | 3 | 1 | | | | | |
| 4 | 4 모바일 맛있달 | 4 | 1 | | | | | |

매출 확인 메뉴

실행결과

The top screenshot shows the '매출 확인' (Sales Confirmation) screen with a date picker dialog for '2021 Mon, Dec 6'. The bottom screenshot shows the sales data for '2021-12-06' with a table of menu items, quantities, and prices.

| 메뉴 | 판매수량 | 총 합계 |
|------------|------|-------|
| 모바일 후라이드치킨 | 3 | 54000 |
| 모바일 양념치킨 | 2 | 38000 |



가맹 점주가 매출을 확인하고 싶은 날짜를 날짜 선택 버튼 터치하여 나오는 DatePickerDialog에서 선택하고 매출 조회 버튼을 터치하면 해당하는 날짜에 대한 매출 정보를 서버에서 가져와 화면의 커스텀 리스트뷰에 추가하여 가맹 점주가 볼 수 있도록 표시해준다.

DB

| id | menu_count | menu_name | menu_price | order_date | order_state | store_name | table_num | user_id |
|----|------------|--------------|------------|---------------------|-------------|------------|-----------|------------|
| 1 | 1 | 3 모바일 후라이드치킨 | 18000 | 2021-12-06 01:28:43 | 0 | 모바일 맛있달 | | 1 testUser |
| 2 | 2 | 2 모바일 양념치킨 | 19000 | 2021-12-06 01:30:21 | 0 | 모바일 맛있달 | | 2 testUser |

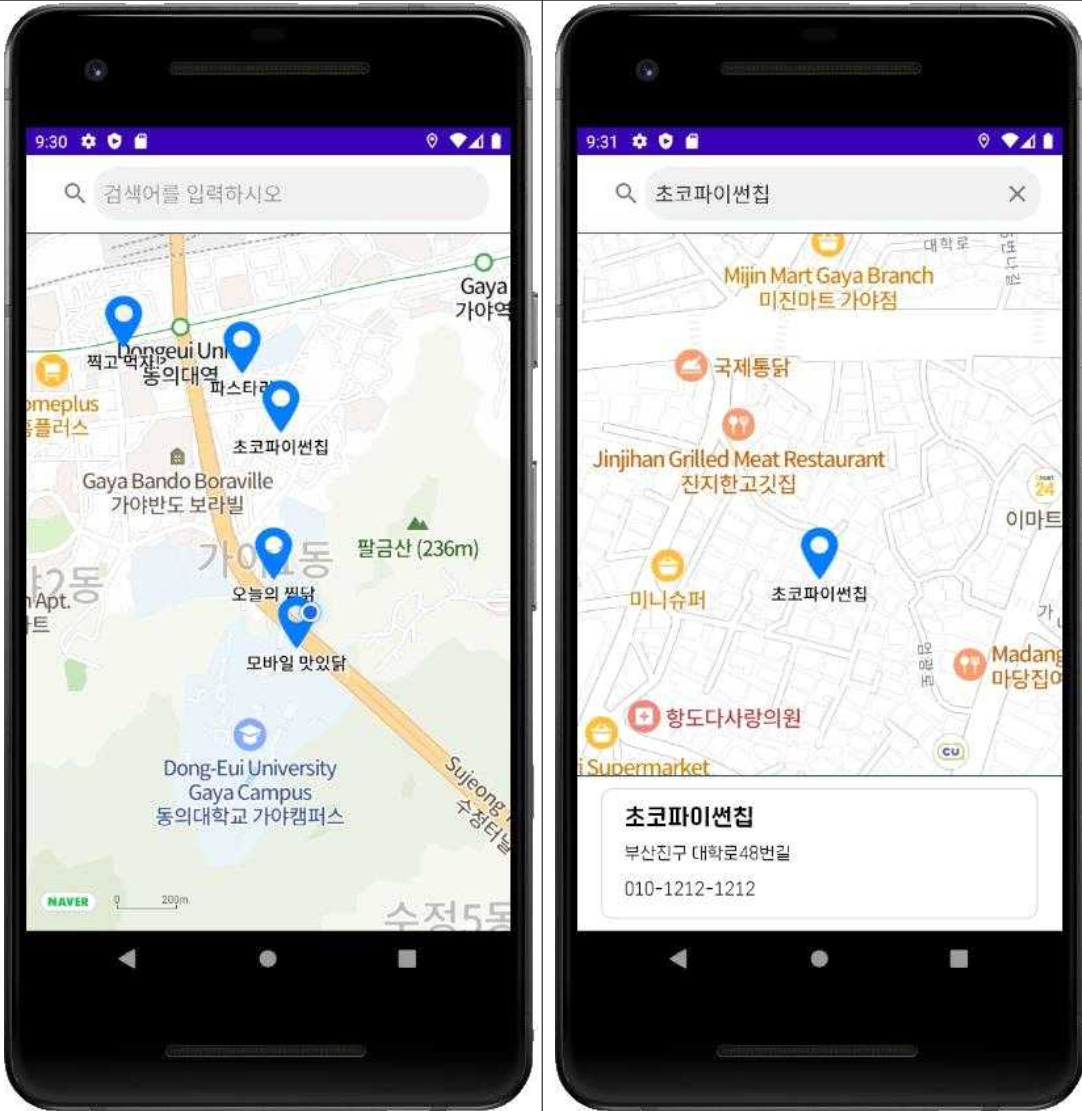
3.2.2 사용자 기능

| 일반 사용자 회원가입 기능 | |
|----------------|--|
| 실행결과 | <div data-bbox="341 241 874 1352"> </div> <p>해당 화면은 일반 사용자 회원가입 페이지 화면으로 아이디, 비밀번호, 확인 비밀번호, 사용자 이름, 전화번호를 입력한다.</p> <p>회원가입 버튼을 누르면 DB의 user 테이블에 일반 사용자 회원 정보가 저장된다.</p> |
| DB | <pre> id ÷ role ÷ userhp ÷ user_id ÷ user_name ÷ user_pwd ÷ 1 1 ROLE_USER 010-1234-5678 deu2021 안대현 qwer0070 </pre> |

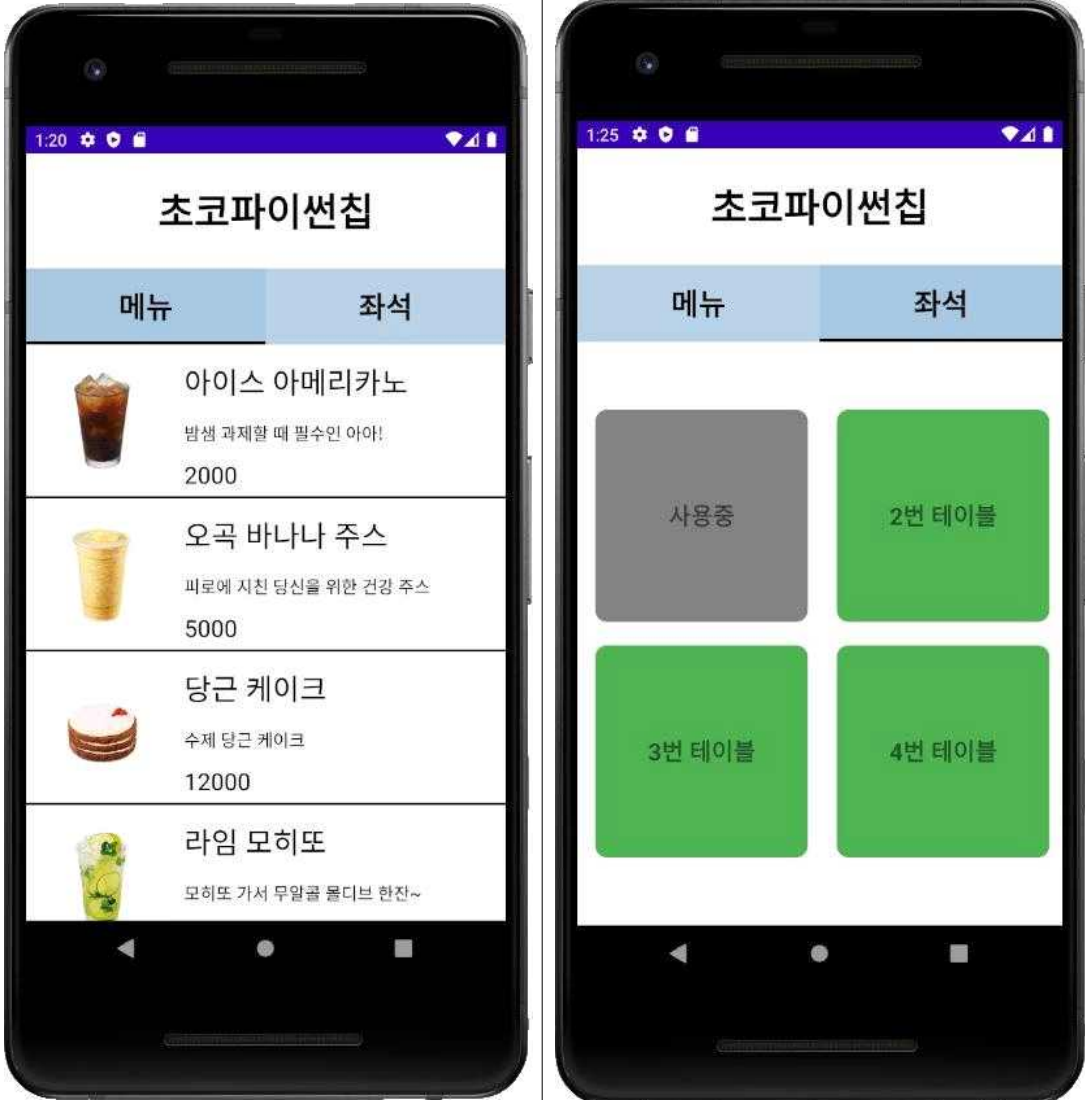
| 일반 사용자 로그인 기능 | |
|---------------|---|
| 실행결과 |  <p>해당 화면은 일반 사용자가 아이디, 비밀번호를 입력하여 로그인을 하는 화면이다.</p> <p>로그인 버튼을 누르면 입력한 아이디와 비밀번호가 user 테이블의 사용자 정보와 일치하는지 확인한다. 만일, 정보가 일치한다면 로그인에 성공한다.</p> |
| |  <p>해당 화면은 로그인 실패 시 보이는 화면이다.</p> <p>사용자가 입력한 아이디와 비밀번호가 user 테이블의 사용자 정보와 일치하지 않으면 해당 커스텀 다이얼로그 화면을 표시하고, 로그인에 실패한다.</p> |

| 일반 사용자 홈 화면 | |
|-------------|---|
| 실행결과 | <div data-bbox="336 188 869 1288" data-label="Image"> </div> <p>해당 화면은 일반 사용자가 로그인하면 나타나는 홈 화면이다.</p> <p>매장 검색 기능을 통해 어플에 등록되어 있는 매장의 메뉴와 좌석 정보를 확인할 수 있다.</p> <p>QR 카메라 기능을 사용하면 사용자가 매장의 메뉴를 확인하고 주문을 할 수 있다.</p> |

매장 검색 기능

| | |
|-------------|---|
| <p>실행결과</p> |  |
| <p>설명</p> | <p>해당 화면은 홈 화면에서 매장 검색 버튼을 눌렀을 때 표시되는 화면이다. GPS 기능을 사용하기 위해 사용자에게 GPS 권한 설정을 요청한 뒤, 카메라를 사용자의 위치로 이동시킨다.</p> <p>지도에는 등록된 가맹점들의 위치가 마커로 표시되고, 마커를 클릭하거나 찾으려는 매장 이름을 검색하면 하단에 매장 리스트 뷰가 표시된다. 하단부의 리스트 뷰를 클릭하면 해당 매장 화면으로 이동한다.</p> |

매장 정보 확인 기능

| | |
|-------------|--|
| <p>실행결과</p> |  |
| <p>설명</p> | <p>해당 화면은 지도에서 매장 리스트 뷰를 클릭하면 표시되는 화면이다.</p> <p>메뉴 탭에서는 메뉴 이름, 설명, 가격과 같은 메뉴 정보를 확인할 수 있고, 좌석 탭에서는 해당 매장에서 사용 가능한 좌석 정보를 확인할 수 있다.</p> |

| QR 카메라 기능 | |
|-------------|---|
| <p>실행결과</p> |  <p>해당 화면은 일반 사용자 홈 화면에서 QR 카메라 버튼을 누른 경우 표시되는 화면이다.</p> <p>카메라를 사용하여 매장 테이블의 QR코드를 인식하고, QR 정보를 통해 해당 매장의 화면으로 이동할 수 있다.</p> <p>본 테스트의 QR코드에는 매장 이름인 “초코파이썬칩”과 테이블 번호가 담겨있다.</p> |

실행결과



해당 화면은 QR코드를 인식하면 표시되는 화면으로, 원하는 메뉴를 장바구니에 추가할 수 있다.

장바구니 기능

실행결과



설명

해당 화면은 매장 화면의 장바구니 탭으로, 메뉴 탭에서 추가한 목록을 확인할 수 있다. 메뉴 수량을 늘리거나 줄이고 싶은 경우는 -와 + 버튼을 사용하여 조절할 수 있다.

결제하기 버튼을 클릭하면 결제 요청 다이얼로그가 표시되고, “예”를 클릭하면 DB에 해당 주문 정보가 저장된다.

DB

| | id | menu_count | menu_name | menu_price | order_date | order_state | store_name | table_num | user_id |
|---|----|------------|-----------|------------|----------------------------|-------------|------------|-----------|-----------|
| 1 | 1 | 1 | 당근 케이크 | 12000 | 2021-12-07 10:25:19.960000 | 0 | 초코파이썬칩 | | 1 deu2021 |
| 2 | 2 | 4 | 아이스 아메리카노 | 2000 | 2021-12-07 10:25:20.261000 | 0 | 초코파이썬칩 | | 1 deu2021 |
| 3 | 3 | 1 | 오곡 바나나 주스 | 5000 | 2021-12-07 10:25:20.357000 | 0 | 초코파이썬칩 | | 1 deu2021 |

3.3 팀원별 역할 분석

| 성명 / 학번 | 수행분야 | 역할 |
|----------|---|----|
| 김기태 | <ul style="list-style-type: none"> 카메라를 통한 QR코드 인식 기능 서버 DB 및 서버 구축 매장 검색 및 지도 API 결제 시스템 회의록 등 문서 작성 | 팀장 |
| 20173152 | | |
| 구본영 | <ul style="list-style-type: none"> 좌석 상태 변화 및 좌석 관리 기능 구현 서버 DB 및 서버 구축 네비게이션 메뉴 구현 매출 확인 기능 구현 회의록 등 문서 작성 | 팀원 |
| 20173219 | | |
| 정순범 | <ul style="list-style-type: none"> 로그인 및 회원가입 실시간 좌석 현황 및 좌석 관리 기능 구현 서버 DB 및 서버 구축 가맹점 등록 및 메뉴 관리 구현 회의록 등 문서 작성 | 팀원 |
| 20173212 | | |
| 안대현 | <ul style="list-style-type: none"> 카메라를 통한 QR코드 인식 기능 구현 로그인 및 회원가입 매장 검색 및 지도 API 서버 DB 및 서버 구축 결제 시스템 회의록 등 문서 작성 | 팀원 |
| 20173217 | | |

4. 결론

4.1 프로젝트 성과 내용

4.1.1 활용방안

- 식사가 가능한 모든 매장의 키오스크를 대신하여 사용할 수 있다. 이는 정해진 수량만 있는 키오스크와 다르게 모든 테이블에 부착되어 사용자의 접근성이 쉽다. 또한, 키오스크 기계의 설치 비용을 절감할 수 있다.
- 매장의 좌석 현황을 실시간으로 확인하여 대면 예약이나, 대기 시스템을 대체할 수 있다.
- 애플리케이션에 등록된 가맹점들은 애플리케이션 지도를 통해 사용자들에게 자주 노출되어 가게 홍보 효과를 가진다.
- 코로나가 활발한 요즈음 비대면 주문을 통해 감염을 줄일 수 있다.

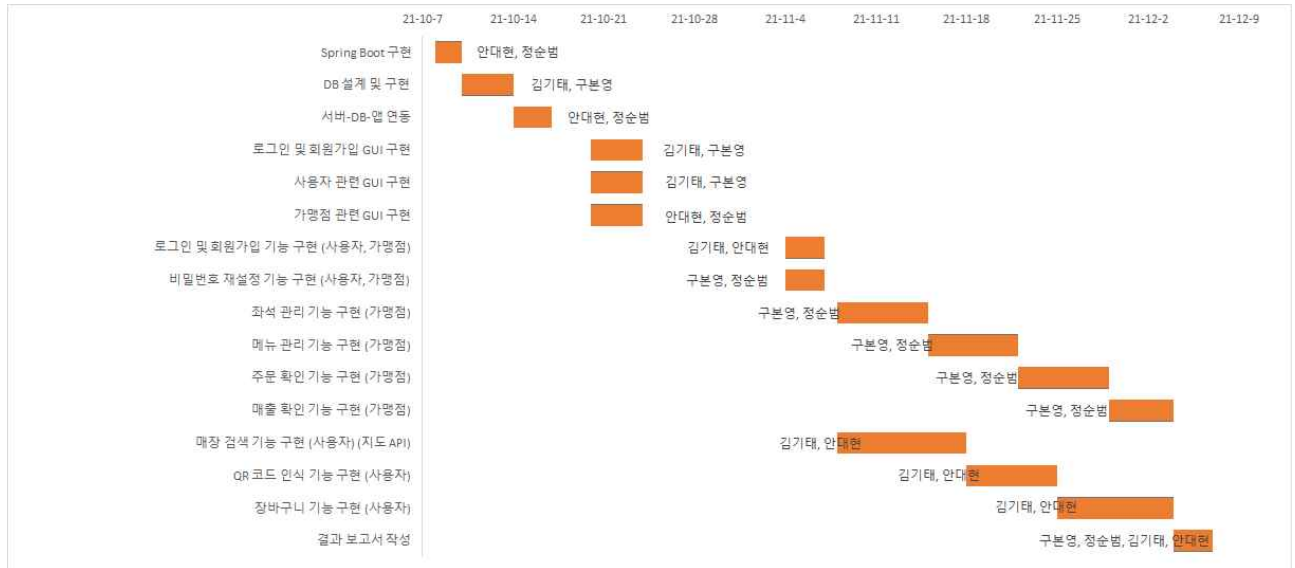
4.1.2 기대효과

| | |
|--------|---|
| 기술적 측면 | <ul style="list-style-type: none">● Kakao 도로명 주소 API를 사용하여 손쉽게 가게의 좌표값을 알 수 있다.● Naver 지도 API를 사용하여 손쉽게 가게 위치를 알 수 있다. |
| 경제적 측면 | <ul style="list-style-type: none">● 매장 내 키오스크를 대신하여 이용할 수 있으므로, 키오스크 설치 비용을 절감할 수 있다.● 정해진 수량의 키오스크 이용을 대신에 하여 모든 테이블에서 사용할 수 있으므로 사용자의 시간적 이익을 창출할 수 있다.● 매장의 좌석 현황을 실시간으로 이용하여 매장 방문시 대기 시간을 줄일 수 있다.● 애플리케이션을 사용하는 가맹점주들은 적은 비용으로 가게를 홍보할 수 있다. |

4.2 설계 목표의 중요도 및 달성도

| 목표 | 중요도(%) | 달성도(%) | 수행내용 |
|---|--------|--------|---|
| 로그인 | 3 | 100 | 사용자와 가맹점주를 구분해 로그인을 진행하여 각 이용자에게 알맞은 화면을 보여준다. |
| 회원 가입 | 3 | 100 | 로그인 시 중복아이디, 전화번호 등 입력값에 대해 유효성 검사를 진행한다. |
| 비밀번호 재설정 | 2 | 100 | 서버에 저장된 회원정보를 조회 후 Update 쿼리를 날려 비밀번호를 수정한다. |
| 메뉴 추가 | 12 | 100 | 이미지와 메뉴 정보를 입력받아 retrofit2 라이브러리를 사용하여 서버에 request를 보낸다. |
| 추가된 메뉴 조회 | 10 | 80 | RecyclerView에 커스텀 리스트 뷰를 사용하여 서버에 저장된 메뉴 정보 리스트를 출력한다. |
| 매출 확인 | 12 | 100 | RecyclerView에 커스텀 리스트 뷰를 사용하여 서버에 저장된 주문 정보 리스트를 선택한 날짜에 맞게 출력한다. |
| QR 코드 인식 | 12 | 100 | 카메라 접근 권한을 설정하여 QR코드 내 데이터를 불러와 해당 매점의 페이지로 이동한다. |
| 지도 매장 검색 | 12 | 100 | Naver 지도 API를 이용하여 가맹점들의 위치를 마커로 표시하며 선택한 가맹점의 페이지로 이동한다. |
| 매장 메뉴 조회 | 10 | 100 | 지정한 매장의 메뉴를 불러와 CustomListView로 메뉴 정보를 표시한다. |
| 매장 좌석 조회 | 10 | 100 | 지정한 매장의 좌석 현황을 서버통신을 통해 실시간으로 확인 가능하다. |
| 주문 및 결제 | 11 | 80 | 장바구니에 담은 메뉴를 주문할 수 있다. |
| 쿠폰 적립 및 관리 | 3 | 0 | - |
| 총평 | | | |
| <ul style="list-style-type: none"> ● 메뉴 조회에서 커스텀 리스트뷰에 추가 및 삭제되어도 바로 갱신이 되지 않는 점이 아쉽다. ● 결제 API를 적용하여 구현하려고 계획하였으나, 기간 조정실패로 인해 구현하지 못하였다. ● 쿠폰 적립 및 관리 기능을 기간 조정실패로 인해 구현하지 못하였다. | | | |

4.3 개발일정



4.4 소감

| 이름 | 느낀점 |
|-----|---|
| 김기태 | 기술이 발전해가며 사람과 가장 오래 붙어있는 기기는 스마트폰이라 생각한다. 평소 스마트폰을 이용하기만 했을 뿐 직접 어플을 개발하게 될 줄 몰랐다. 어플의 레이아웃부터 각 기능, 또한 서버와 DB 설계부터 구현까지 진행해가며 각 액티비티와 프래그먼트의 역할, 생명주기 등 많은 것을 알 수 있었다. 또한 팀원들과 역할 분담을 하여 기능해가며 협업에 대해 많은 것을 깨달을 수 있었다. Git을 통해 버전관리 및 형상관리를 해서 많은 사고 없이 분담한 업무를 성공적으로 끝낼 수 있었다. |
| 구본영 | 기능 설계부터 업무 분담까지 적절히 이루어져 프로젝트 진행하는 데 큰 어려움이 없었고, 안드로이드에서 제공하는 새로운 기능들을 배울 수 있었다. 초기 설계를 잘 해 두어서 프론트 제작은 금방 처리할 수 있었다. 팀원 모두 제 역할을 잘 해주어서 완성도 높은 앱을 만들 수 있었다. |
| 안대현 | 본 프로젝트를 통해 커스텀 리스트 뷰와 프래그먼트, 데이터 매핑 등 다양한 기능을 활용해 볼 수 있었고, 그 과정에서 안드로이드의 생명주기, 동작 원리 등을 익힐 수 있었다. 처음으로 서버를 연동하여 데이터를 관리하면서 새로운 기술들을 활용해 볼 수 있어서 좋은 공부가 되었다고 생각한다. 팀원들과의 협력이 원활히 이루어졌기 때문에 정해진 기간 내에 만족할 만한 결과물을 만들어낼 수 있었다. |
| 정순범 | 팀 프로젝트를 진행하면서 안드로이드 어플리케이션의 동작 과정, 생명주기, 내부 구조 등을 상세하게 알 수 있었다. 특히 안드로이드가 서버와 통신하기 위해 retrofit2 라이브러리를 사용해보고 객체 매핑, json 통신을 직접 구현해 보고 Kakao API를 사용해 데이터를 받아와 사용해볼 수 있었다. Git을 사용해 형상관리를 진행하며 체계적인 협업 환경을 구성하였다. 다만 메뉴 추가할시 메뉴 조회 리스트가 자동으로 갱신되지 않는점이 아쉬웠다. |