Cloud Computing

Final Report

<Receipt Mate>

박찬호 32181928
박현찬 32191936
정우섭 32184140
하승원 32184801
허찬용 32184939

https://github.com/Receipt-Information-Parse

Contents

Ⅰ Overview

Project Introduction

Receipt Mate is an application that helps users manage their offline consumption. It utilizes OCR technology to convert paper receipts into tabular data, allowing users to track their expenses by period and by product. Thus, users can understand their consumption patterns.

Background

According to Gallup Korea (Gallup Korea, 2022), the penetration rate of smartphones in Korea reached 97.1% in 2022. With this trend, spending management using smartphones and computers is increasing. For example, there is a ‘MyData’ business that helps manage data by being delegated the authority to use personal data, with 56 ‘MyData’ operators selected and 48 services in operation as of May 2022, with 30.25 million service subscribers and 36.88 billion information transmissions (박주석, 2023). The services including those of ‘MyData’ record and provide expenditures to users based on financial data such as credit card usage history occurring in users' online and offline transactions, and users can perform expenditure management.

Target Users

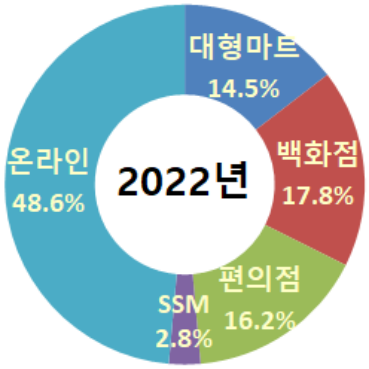


Figure 1

According to the composition ratio of distributor sales in 2021 and 2022 (산업통상자원부, 2023), more than half of the transactions are still made offline (Figure 1).

Of course, the transition from offline consumption to online consumption is ongoing, but as the importance of experience marketing that values experience in offline emerges, online retailers are also opening offline stores and trying to increase offline sales (중앙선데이, 2022). Therefore, offline consumption will still account for a large proportion in the future, and offline consumption management by consumers is important according to this trend. According to a survey (트렌드모니터, 2014), 60.4% of consumers thought it was helpful to write a household account. However, only 33.6 percent of consumers wrote household accounts, and the main reason for not writing was that it is inconvinient and unfamiliar. Thus, our service <RIP> focuses on helping consumers managing offline consumption history.

Problems

Existing services that use computerized consumption records can manage online consumption well but not offline consumption. This is caused by two reasons related with two methods supporting offline consumption management : electronic receipts and card usage history.

First, due to the low utilization of electronic receipts, many offline consumption records are still not managed in detail. According to a news article (사례뉴스, 2022), 60% of receipts for offline consumption are still issued as paper receipts. Thus, there is a problem that offline consumption with paper receipts is managed depending on users' manual record.

By considering the eco-friendly trend, it is expected that usage of online receipts will increase and that of paper receipts decrease. However, since online receipts are issued using different apps for each brand (대한민국 정책브리핑, it is difficult to manage receipts and personal analysis through the details of receipts is not being carried out.

Second, the card history is only recording total cost, so it is difficult to properly manage the purchase details of offline consumers. Services that provide expenditure management with importing card history also holds the same problem.

Solutions & Expected Effects

The following services are being used to solve problems above.

1. 뱅크샐러드(BankSalad)

   BankSalad records and analyzes users' spending and income integrating with financial database and MyData. BankSalad has the advantage of easily analyzing expenditure through data integration, but it has the disadvantage of not being able to record the user's offline consumption details because it uses only summary information such as total cost.

2. 오늘 뭐샀니(CashCow)

   CashCow is a service that register receipts by taking pictures of them, and accumulates points. However, the focus is only on accumulating points and there is no function to record and manage detailed consumption details.

3. 편한 가계부

   편한 가계부 manages expenditures based on payment history text messages. The user can check the expenditure details and statistics, and can save the income and expenditure details as an Excel file. However, payment details can only be applied to card transactions that are sent by text, and there is a disadvantage that details cannot be analyzed due to the property of the card history that only saves total cost.

4. BuBoo 부부

   Buboo is a service that supports multiple users' shared household accounts. Users can manually add and categorize their expenditures. It also supports the function of copying and registering card text details, but it has the disadvantage that it is inconvenient because most records rely on manual input.

Existing services provide advanced functions and various conveniences, but they are not suitable for offline record management because they focus on online transactions or computerized records. In addition, services that assist offline record management have the disadvantage of using only summary information such as card transactions or relying on manual user input. Therefore, our service RIP aims to solve the problem by simply registering paper receipts as correctable data.

Simplifying the management of offline consumption details through receipt image has the following expected effects. First, as it is possible to record and manage offline consumption that existing services could not manage, there is an effect by datafication of offline consumption. Such data is a source of various value creation such as marketing and advertising. Second, it can preoccupy potential users who are not familiar with writing household accounts with easy usage. Initially, only core functions that are not provided by existing services are developed and provided, but in the future, additional businesses such as online receipt development can be attempted based on the preoccupied user base.

Proposing Feature

Our service RIP proposes the following two functions to manage offline consumption records by registering paper receipts.

1. RIP service converts paper receipts to tabular data using OCR.

   The user adds a receipt image from the service, and the service extracts strings from the image using a pre-trained machine learning OCR model. The strings are converted to tabular data by parser and then returned to the service.

2. RIP service provides a sheet function that help users can modify tabular data.

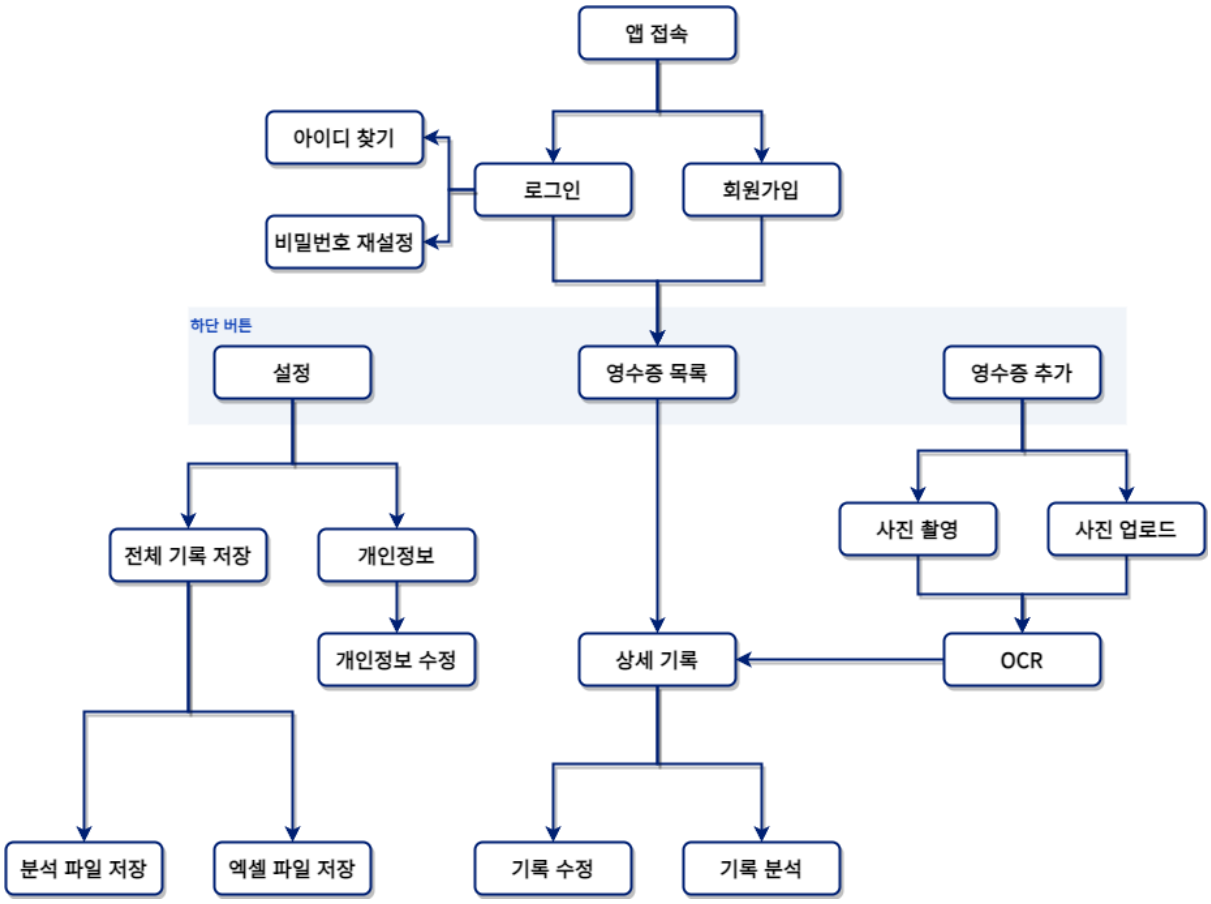
   The user may modify tabular data extracted from the receipt and record additional contents. Tabular data can be stored in Excel or csv format if the user wants.

3. Visualize tabular data with histograms or line plots.

   Users can check consumption summary by period or price changes by item with visualized plots of tabular data. This enables easier management and analysis of consumption.

Ⅱ Requirements

Flow Diagram



Prior to specifying the functions to be provided by RIP through the requirements specification, changes in tasks and functions according to the user's behavior are visualized as Flow Diagram.

First, the user executes the mobile app and goes through authentication through 로그인(login) or 회원가입(signup). If there is a problem with the login process, you can use 아이디 찾기(ID search) or 비밀번호 재설정(password reset).

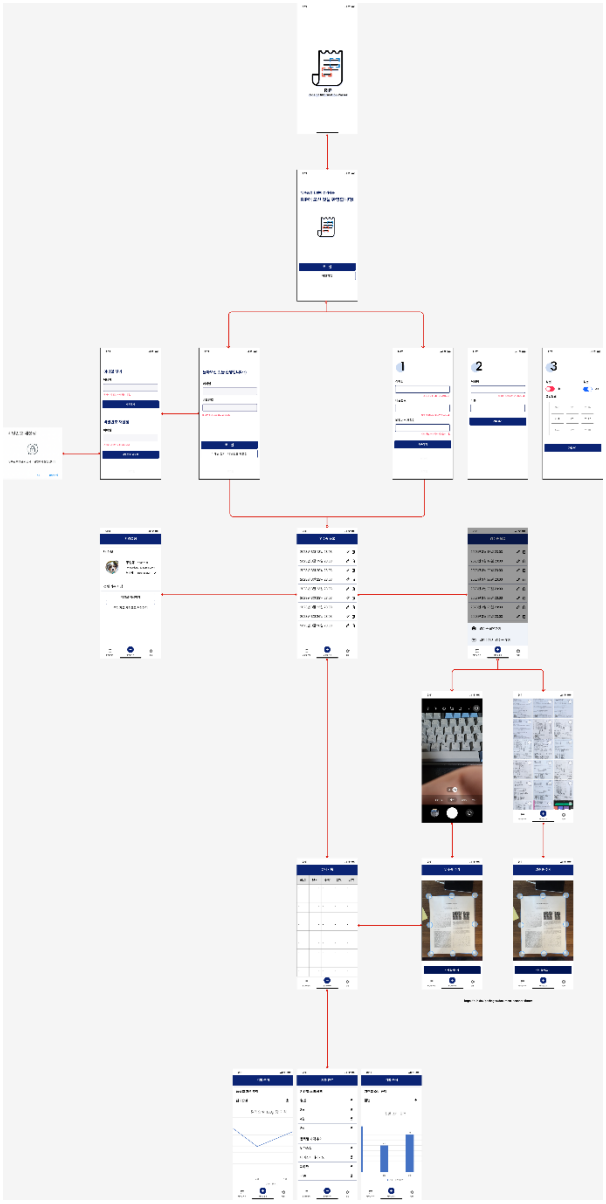
After completing authentication, the 영수증 목록(receipt list) screen is loaded for the first time. However, there is no global main screen, and the application follows a structure that can use the bottom navigation bar with 설정(setting), 영수증 목록(receipt list), and 영수증 추가(add receipts). In the 영수증 목록(receipt list), you can check the 상세 기록(detailed record) of each receipt, and can use 기록 수정(modify record) or 기록 분석(analyze record). 기록 수정(modify record) uses sheet UI, and 기록 분석(analyze record) generates histograms or line plots based on tabular data.

In the 영수증 추가(add receipts), photos are sent to the OCR model through 사진 촬영(taking photo) or 사진 업로드(photo upload), and tabular data extracted as OCR results are received and displayed as 상세 기록(detailed record).

The 설정(setting) supports 개인정보(personal information) and the 전체 기록 저장(save entire record) function, and personal information can be modified by 개인정보 수정(personal information modification). 전체 기록 저장(save entire record) allows the analysis file to be saved as an image or tabular data to be saved as Excel.

User Interface

The user interface flow based on flow diagram is as follows



Requirements

Requirements of each endpoint is as written below.

Front-End

- UI/UX

서비스(메뉴)	기능	상세기능
UI/UX 구현	스플래시	앱 접속시 RIP로고와 함께 앱 시작화면을 띄운다.
	로그인	회원별 저장된 아이디와 비밀번호로 로그인이 가능해야 한다. 입력된 아이디와 비밀번호가 서버에 저장된 정보와 같은지 비교하여, 로그인 여부를 판단한다.
	회원가입	아이디, 비밀번호, 닉네임 입력으로 신규 회원으로 등록하여, 서버에 데이터를 저장한다. 단, 아이디와 닉네임은 중복 불가능하게 설정하고 비밀번호는 숫자,영문,특수문자 8자 이상의 제약이 존재한다.
	아이디 찾기	회원가입시 입력한 닉네임으로 해당하는 회원 아이디 찾는다.
	비밀번호 재설정	회원가입시 설정한 비밀번호를 재설정한다.
	영수증 목록	유저가 입력한 영수증 전체 목록과 각 영수증의 날짜 및 시간이 리스트로 나타난다. 이때, 영수증의 정보가 수정 가능한 형태여야하고, 영수증 한 섹션 자체를 삭제 가능한 형태여야한다.
	영수증 추가	새 영수증을 등록한다. 기존 사진을 등록하는 형태와 카메라를 이용하여 영수증을 촬영해 그 파일을 저장하는 형태 2가지로 구현한다. 사진이 무사히 등록되면, ML을 이용하여 영수증 데이터를 수집한다.
	상세 기록	영수증 추가 기능에서 수집된 영수증의 정보를 표 형태로 나타낸다. 이때, 표에 있는 정보는 수정 및 삭제가 가능해야한다.
	환경설정	개인정보 수정이 가능해야한다. 이메일, 닉네임, 비밀번호가 수정 가능해야한다. 또한, 상세기록 정보를 사진(이미지)파일 혹은 엑셀 파일로 저장하는 기능이 있어야한다.
	기록 분석 (future work)	영수증의 상세기록을 바탕으로 품목별 가격 변화, 소비 패턴 분석, 기간별 소비 금액 등을 그래프나 표 형태로 나타낸다.

- Merging Screens

서비스(메뉴)	기능	상세기능
화면 병합 (Merge Screens)	State 관리방식 단일화	각 담당자들의 코드를 바탕으로 Provider를 이용하여 state 관리방식을 단일화하여 진행한다.
	단순 화면 넘김으로 화면 연결	Navigator을 이용하여 각 담당자가 완성한 화면간의 전환을 만든다.

- DataFlow

서비스(메뉴)	기능	상세기능
DataFlow 추가	DataFlow 확인	Data Flow가 올바른지 확인한다.
	Provider 재설정 (DataFlow 기준으로)	확인한 Data Flow를 바탕으로 Data Flow와 맞지않는 Provider를 수정한다.

- Quality Control

서비스(메뉴)	기능	상세기능
품질관리	성능테스트 및 디버깅	실전 테스트를 통해 성능테스트를 진행하고 발생 및 가능성이 있는 코드 디버깅을 진행한다.

TEAM: BACK-END

- User

서비스(메뉴)	기능	상세기능
User	로그인	이메일 존재 여부를 확인한 후 입력된 비밀번호로 로그인을 시도한다. 로그인이 성공하면 JWT token을 반환하고, 에러가 발생하면 Error Message를 반환한다.
	이메일 존재 여부	입력된 이메일이 존재하는지 확인하고, 이메일 형식이 아니거나 이메일이 존재하지 않으면 Error Message를 반환한다.
	닉네임 존재 여부	입력된 닉넨임이 존재하는지 확인하고, 닉네임이 존재하지 않으면 Error Message를 반환한다.
	이메일 찾기	입력된 닉네임이 존재하는지 확인하고, 존재하지 않으면 Error Message를 반환한다. 존재한다면 해당하는 회원 이메일을 반환한다.
	비밀번호 재설정	입력된 이메일이 존재하는지 확인하고, 이메일 형식이 아니거나 이메일이 존재하지 않으면 Error Message를 반환한다. 이메일이 존재한다면 비밀번호를 랜덤으로 재설정하고 안내 메일을 발송한다.
	카카오 Oauth 지원	카카오 회원가입, 로그인을 사용하여 유저가 Oauth를 통해 간편하게 로그인 할 수 있도록 지원한다. Front로부터 카카오 토큰을 전달받고, 카카오 토큰으로부터 유저 정보를 읽어들여 회원가입 및 로그인을 진행한다.
	프로필 사진	프로필 사진을 추가하고 수정할 수 있다.
	회원가입	이메일이 존재하는지, 비밀번호가 조건에 맞는지, 닉네임이 존재하는지 확인한다. 문제가 없다면 회원가입을 통해 유저를 데이터베이스에 추가하고, JWT token을 반환한다.

- User Info

서비스(메뉴)	기능	상세기능
UserInfo	내 정보	이름, 생일, 닉네임으로 구성된 내 정보를 반환한다.
	닉네임 수정	사용자의 입력으로 닉네임을 받고, 존재하지 않는 닉네임이라면 수정한다. 존재하는 닉네임이라면 에러를 반환한다.

- OCR

서비스(메뉴)	기능	상세기능
OCR	이미지 업로드	사용자로부터 입력받은 이미지를 OCR모델에 송신한다
	Tabular Data 수신	OCR결과로 반환되는 Tabular Data를 수신한다.

- Receipt

서비스(메뉴)	기능	상세기능
영수증	영수증 목록	전체 영수증 목록을 반환한다.
	영수증 상세기록	영수증 세부 기록을 csv 형태로 전송한다.
	영수증 상세기록 수정	세부기록 수정 요청에 따라 csv 파일의 내용을 수정한다.
	영수증 추가	입력된 이미지를 Object Storage에 저장후 OCR에 전송하고, OCR로부터 전달받은 CSV파일을 Object Storage에 저장한다. 이미지와 CSV 파일을 User Table에 저장 후, 상세 기록을 반환한다.
	전체 기록 엑셀 저장	입력받은 사용자가 소유한 전체 기록을 조회한 후, 전체 기록의 CSV 파일을 반환한다.
	영수증 기간별 분석	쿼리 조회 및 Java Streams를 통해 연도별, 월별 총액을 구해 반환한다.
	영수증 품목별 분석	JPA 조회를 통해 품목별 소비 금액을 반환한다.
	전체 분석 사진 저장	입력받은 사용자가 소유한 전체 기록의 분석 사진을 조회한 후, 전체 사진을 반환한다.

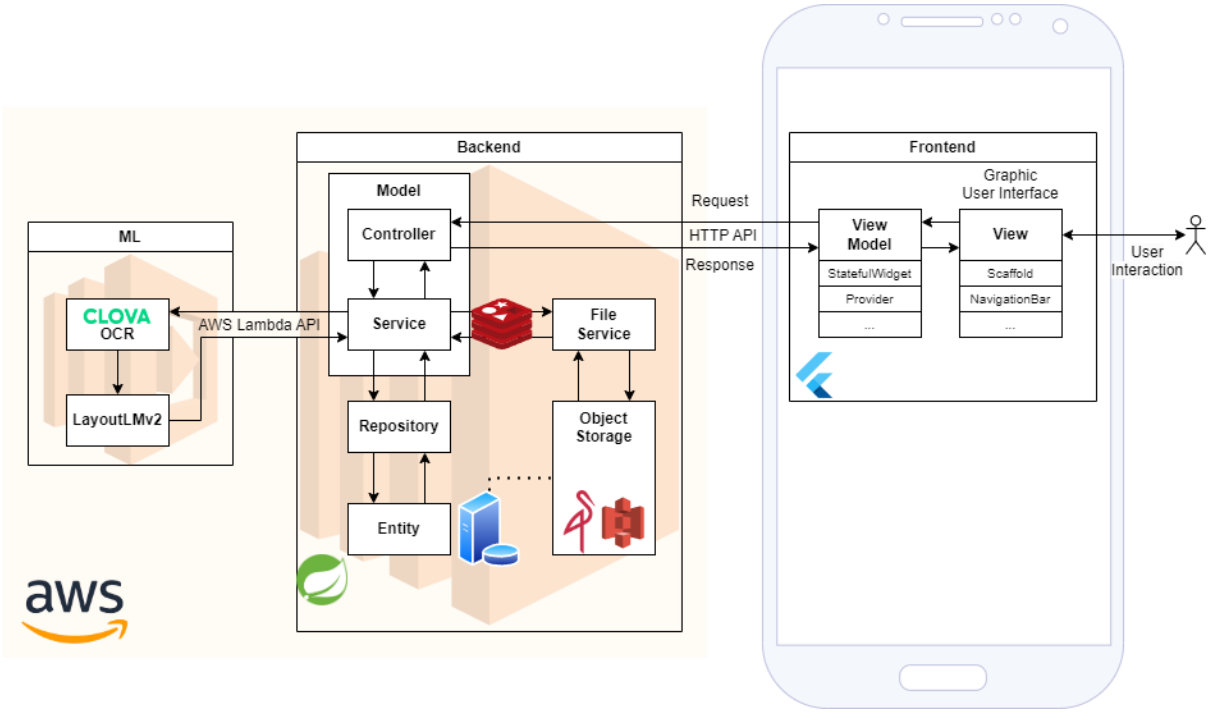
TEAM: ML

- OCR & parsing text

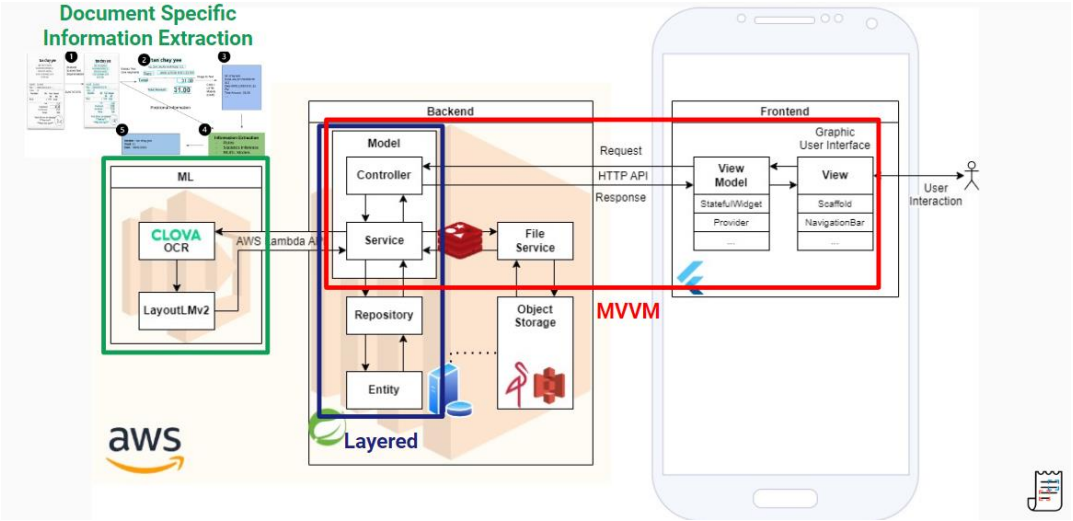
서비스(메뉴)	기능	상세기능
OCR	영수증 OCR	영수증 사진이 입력되면 해당 Text가 있는 위치와 어떤 Text인지 분석할 수 있어야 한다.
OCR Data to Tabular	Layoutlmv 모델을 활용하여 OCR Data	OCR을 통해 얻은 Bounding Box 데이터를 사용하여 해당 Box는 어떤 Contents인지 분류가 가능해야 한다.
	Parsing을 통한 데이터 테블라화	Tagging 작업을 통해 얻은 데이터를 테블라화가 가능해야 한다.
Output	소비 추이 그래프 데이터	그래프로 출력 가능한 데이터의 형식으로 데이터 저장

Architecture

The overall architecture is as follows.



In detail, each endpoint uses architecture as follows.



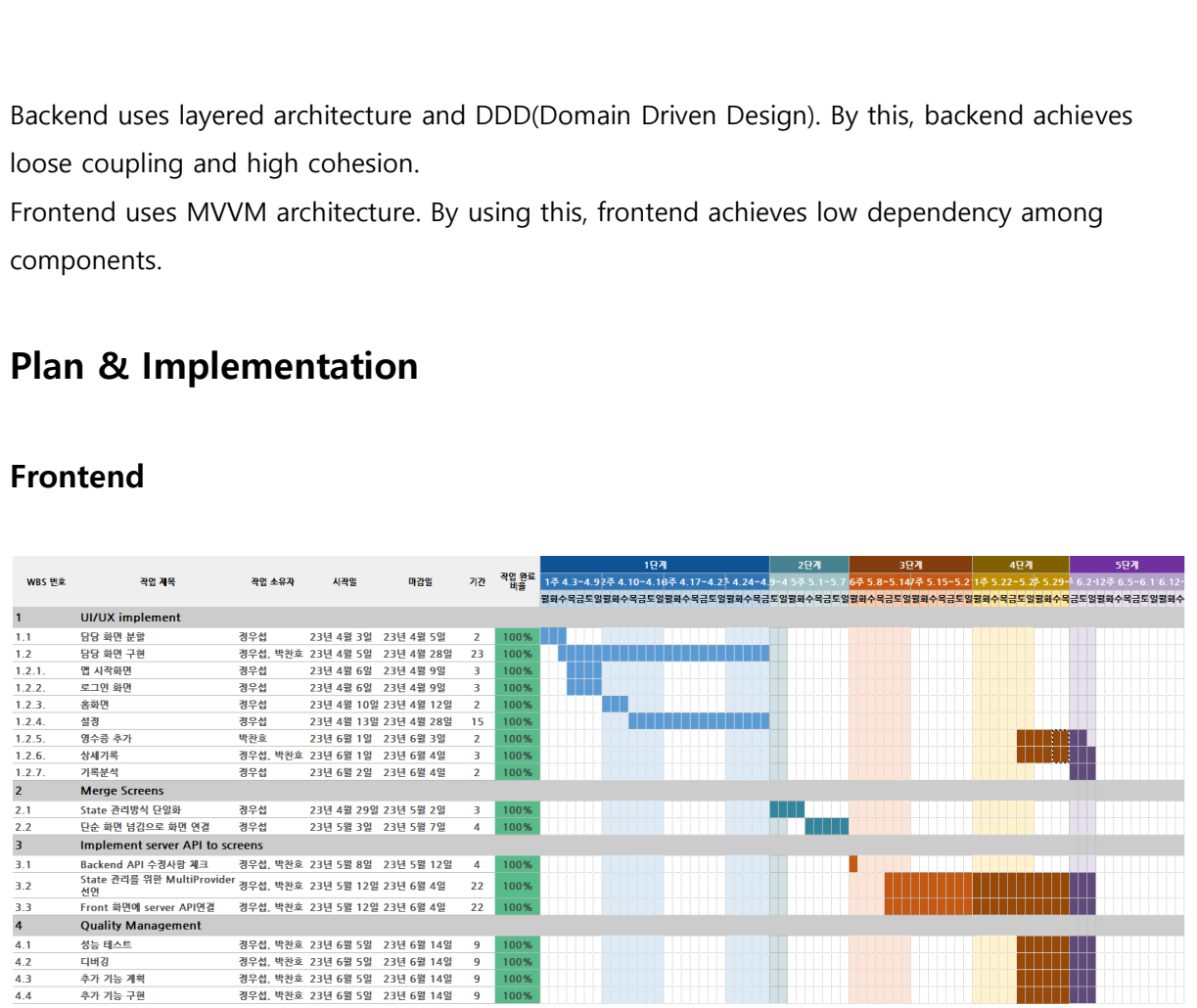
OCM model uses document specific information extraction. First, by using Naver CLOVA OCR API, we can get text embedding and positional embedding from receipt image. After preprocessing those embeddings, use LayoutLMv3 to get layout and then postprocess the output for JSON.

Backend uses layered architecture and DDD(Domain Driven Design). By this, backend achieves loose coupling and high cohesion.
Frontend uses MVVM architecture. By using this, frontend achieves low dependency among components.

Plan & Implementation

Frontend

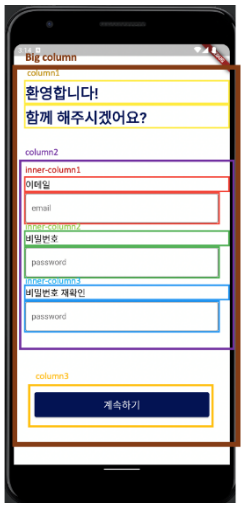
WBS 번호	작업 계목	작업 소유자	시작일	마감일	기간	작업 완료 비율
1	UI/UX implement					
1.1	담당 화면 분할	경우섭	23년 4월 3일	23년 4월 5일	2	100%
1.2	담당 화면 구현	경우섭, 박찬호	23년 4월 5일	23년 4월 28일	23	100%
1.2.1.	앱 시작화면	경우섭	23년 4월 6일	23년 4월 9일	3	100%
1.2.2.	로그인 화면	경우섭	23년 4월 6일	23년 4월 9일	3	100%
1.2.3.	홈화면	경우섭	23년 4월 10일	23년 4월 12일	2	100%
1.2.4.	설정	경우섭	23년 4월 13일	23년 4월 28일	15	100%
1.2.5.	영수증 추가	박찬호	23년 6월 1일	23년 6월 3일	2	100%
1.2.6.	상세기록	경우섭, 박찬호	23년 6월 1일	23년 6월 4일	3	100%
1.2.7.	기록분석	경우섭	23년 6월 2일	23년 6월 4일	2	100%
2	Merge Screens					
2.1	State 관리방식 단일화	경우섭	23년 4월 29일	23년 5월 2일	3	100%
2.2	단순 화면 넘김으로 화면 연결	경우섭	23년 5월 3일	23년 5월 7일	4	100%
3	Implement server API to screens					
3.1	Backend API 수정사항 체크	경우섭, 박찬호	23년 5월 8일	23년 5월 12일	4	100%
3.2	State 관리를 위한 MultiProvider 선언	경우섭, 박찬호	23년 5월 12일	23년 6월 4일	22	100%
3.3	Front 화면에 server API연결	경우섭, 박찬호	23년 5월 12일	23년 6월 4일	22	100%
4	Quality Management					
4.1	성능 테스트	경우섭, 박찬호	23년 6월 5일	23년 6월 14일	9	100%
4.2	디버깅	경우섭, 박찬호	23년 6월 5일	23년 6월 14일	9	100%
4.3	추가 기능 계획	경우섭, 박찬호	23년 6월 5일	23년 6월 14일	9	100%
4.4	추가 기능 구현	경우섭, 박찬호	23년 6월 5일	23년 6월 14일	9	100%



Frontend finished all implementations as planned.

Robust Widget Hierarchy

We robustly structured hierarchy of widgets to avoid overflow and provide same design for different devices.



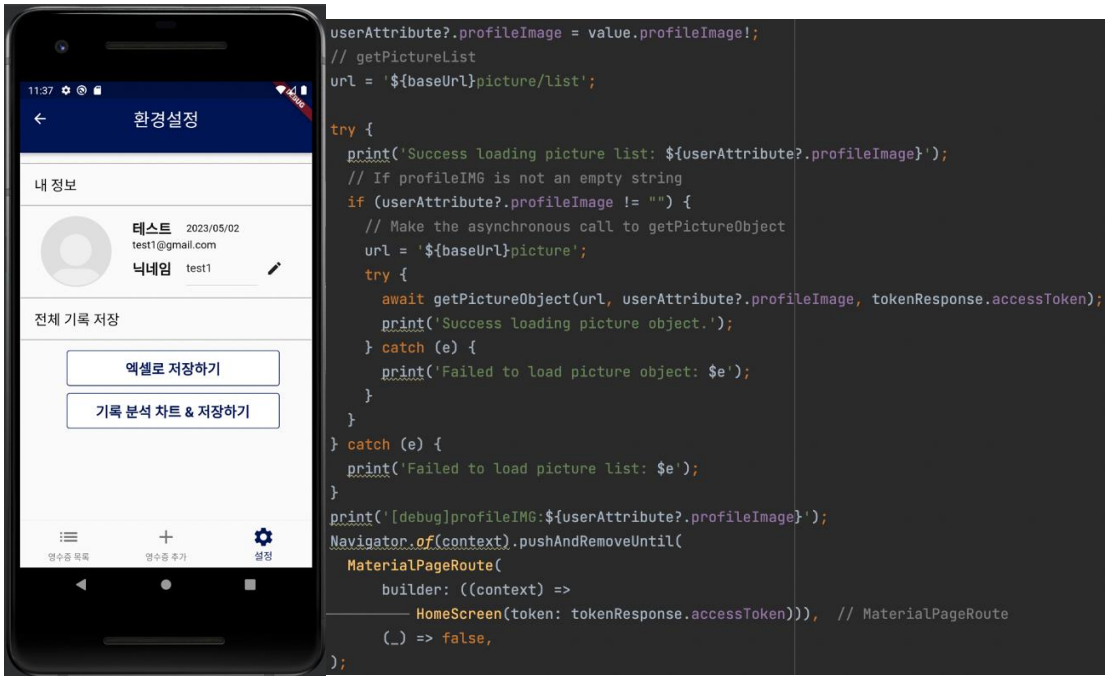
Profile Picture

Asynchronous Profile Picture Retrieval

When retrieving a profile picture from the server in the "My Information" window, there was a problem of slow loading due to the time it took to send and receive the picture file. To solve this, we chose to pre-fetch and save the picture file asynchronously as soon as the user logs in.

The image above shows the login screen. When the login button enclosed in a red box is pressed, the login request function is executed, and if the login information is valid, the user information is returned and the value of Provider UserAttribute is initialized.

If the value of UserAttribute.profileImage is an empty string (""), the picture is not separately retrieved, and the home screen is displayed. In the "My Information" screen, a default profile image is displayed as shown below.



Profile Picture Caching

In the same manner as initializing the profile picture during login, we implemented caching by using the unique value of the image_key, which is the unique identifier for all pictures on the backend, as the file name to avoid duplication.

Once various values related to user information are updated, the logged-in user becomes identifiable immediately. Using the Bearer Token and profileImage from the user information, we execute the getPictureObject function in the picture below to retrieve the profile picture of the corresponding user.

The getExternalStorageDirectory() function in getPictureObject allows us to obtain the relative path of the device where the app is installed, making it possible to manage pictures in the same directory.

```dart
Future getPictureObject(String url, String? object, String? token) async {
  var httpClient = HttpClient();
  var request = await httpClient.getUrl(Uri.parse('$url/$object'));

  // Bearer token
  request.headers.set('authorization', 'Bearer $token');

  print('[debug]url:$url/$object');
  var response = await request.close();
  var bytes = await consolidateHttpClientResponseBytes(response);
  final dir = await getExternalStorageDirectory();
  File file = File('${dir!.path}/$object');
  await file.writeAsBytes(bytes);
  return file;
}
```

Request-Response Structure

We implemented it using the http package in Dart as the basic package. The picture below shows the basic format for GET and POST requests.

```dart
Future<List<String>> getNames(String url, String? token) async {
  final response = await http.get(
    Uri.parse(url),
    headers: <String, String>{
      'Content-Type': 'application/json; charset=UTF-8',
      'Authorization': 'Bearer $token',
    },
  );
  print('[debug] response status code : ${response.statusCode}');
  if (response.statusCode == 200) {
    // If the server returns a 200 OK response,
    // then parse the JSON.
    List<dynamic> body = jsonDecode(utf8.decode(response.bodyBytes))["names"];
    List<String> names = body.cast<String>();

    return names;
  } else {
    // If the server did not return a 200 OK response,
    // then throw an exception.
    throw Exception('Failed to load names');
  }
}
```

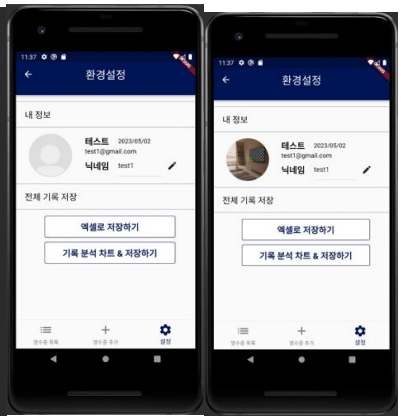
```dart
Future<MessageResponse> resetPassword(String url, EmailRequest emailRequest) async {
  final response = await http.post(
    Uri.parse(url),
    headers: <String, String>{
      'Content-Type': 'application/json; charset=UTF-8',
    },
    body: jsonEncode(emailRequest.toJson()),
  );

  if (response.statusCode == 200 ||  response.statusCode == 400) {
    return MessageResponse.fromJson(jsonDecode(utf8.decode(response.bodyBytes)));
  } else {
    throw Exception('Failed to reset password.');
  }
}
```

Based on the response format of the request, we declared the following response data model:

```dart
Future<MessageResponse> resetPassword(String url, EmailRequest emailRequest) async {
  final response = await http.post(
    Uri.parse(url),
    headers: <String, String>{
      'Content-Type': 'application/json; charset=UTF-8',
    },
    body: jsonEncode(emailRequest.toJson()),
  );

  if (response.statusCode == 200 ||  response.statusCode == 400) {
    return MessageResponse.fromJson(jsonDecode(utf8.decode(response.bodyBytes)));
  } else {
    throw Exception('Failed to reset password.');
  }
}
```

The challenging part was the savePicture request, where we send the profile picture file along with the request. For this, we used the MultipartRequest in http. By using request.files.add(http.MultipartFile()) as shown in the code below, we can add the desired picture file to the request.

```dart
Future<KeyResponse> savePicture(String url, File file, String? token) async {
  var request = http.MultipartRequest('POST', Uri.parse(url));

  // Bearer token 추가
  request.headers.addAll({'Authorization': 'Bearer $token'});

  request.files.add(http.MultipartFile('file', file.readAsBytes().asStream(), file.lengthSync(),
                                                          filename: file.path.split("/").last));

  var response = await request.send();

  if (response.statusCode == 200) {
    var responseData = await response.stream.toBytes();
    var responseString = String.fromCharCodes(responseData);
    return KeyResponse.fromJson(json.decode(responseString));
  } else {
    throw Exception('Failed to save picture');
  }
}
```



When the response is returned as a list, we declared the request and data model as follows:

```dart
Future<List<ByPeriod>> getByMonth(String url ,String? token) async {
  final response = await http.get(
    Uri.parse(url),
    headers: <String, String>{
      'Content-Type': 'application/json; charset=UTF-8',
      'Authorization': 'Bearer $token',
    },
  );
  print('[debug] response status code : ${response.statusCode}');
  if (response.statusCode == 200) {
    // If the server returns a 200 OK response,
    // then parse the JSON.
    List<dynamic> body = jsonDecode(utf8.decode(response.bodyBytes))["byPeriods"];
    List<ByPeriod> byPeriods = body
        .map(
          (dynamic item) => ByPeriod.fromJson(item),
    )
        .toList();

    return byPeriods;
  } else {
    // If the server did not return a 200 OK response,
    // then throw an exception.
    throw Exception('Failed to load ByPeriod');
  }
}
```

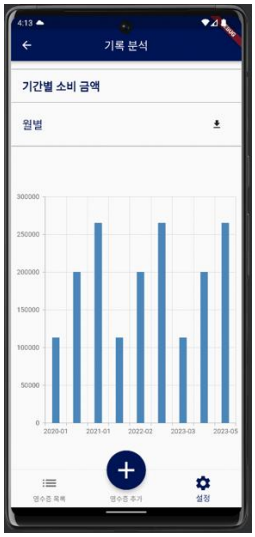
```dart
class ByPeriod {

  final DateTime date;
  final int amount;
  final int analysisId;

  ByPeriod({
    required this.date,
    required this.amount,
    required this.analysisId,
  });

  factory ByPeriod.fromJson(Map<String, dynamic> json) {
    return ByPeriod(
      date: DateTime.parse(json['date']),
      amount: json['amount'],
      analysisId: json['analysisId'],
    );
  }
}
```

By utilizing this, we save the response in a List with the declared type and perform the necessary operations.

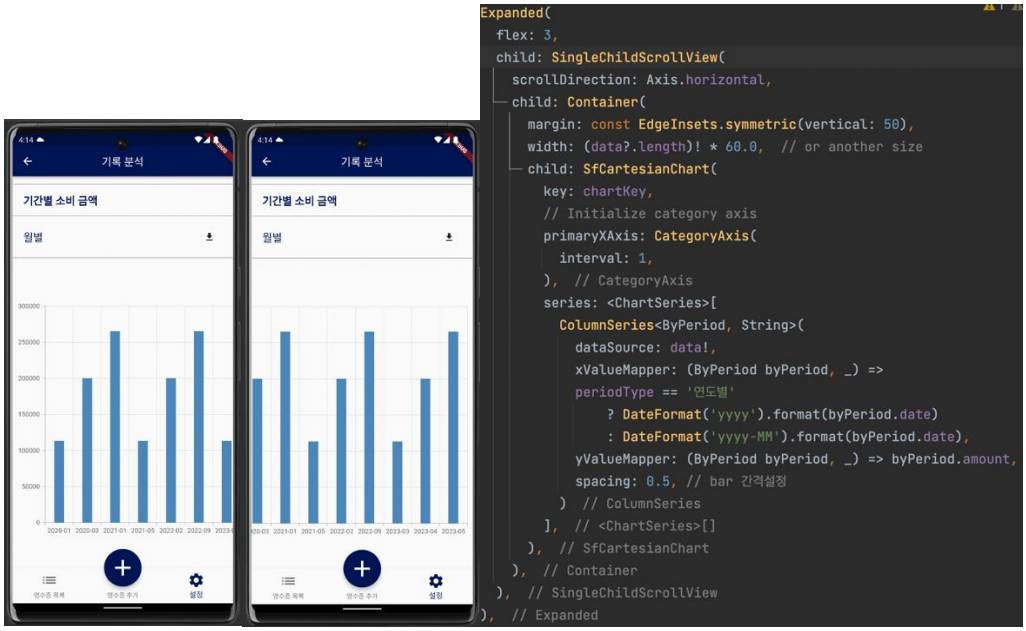
```dart
else { // 화면 넘김용일때
  List<ByPeriod>? ApiResponse;
  if (item == '연도별') {
    String url = '${baseUrl}analysis/year';
    print('[debug]연도별 clicked:$url');
    ApiResponse = await getByYear(url,tokenResponse.accessToken);
  }
  if (item == '월별') {
    String url = '${baseUrl}analysis/month';
    print('[debug]월별 clicked:$url');
    ApiResponse = await getByMonth(url,tokenResponse.accessToken);
  }
  // TODO: Navigator 결과와 같이 넘기기(response 형태에 따라 다르게 구현)
  Navigator.of(context).pushReplacement(MaterialPageRoute(
      builder: ((context) => AnalysisChart2Screen(apiResponse:ApiResponse,periodType:item))
  ));  // MaterialPageRoute
}
```

Overflow in Analysis Chart

When displaying the chart on the receipt data analysis screen, the chart appeared too small.
Before the modification:

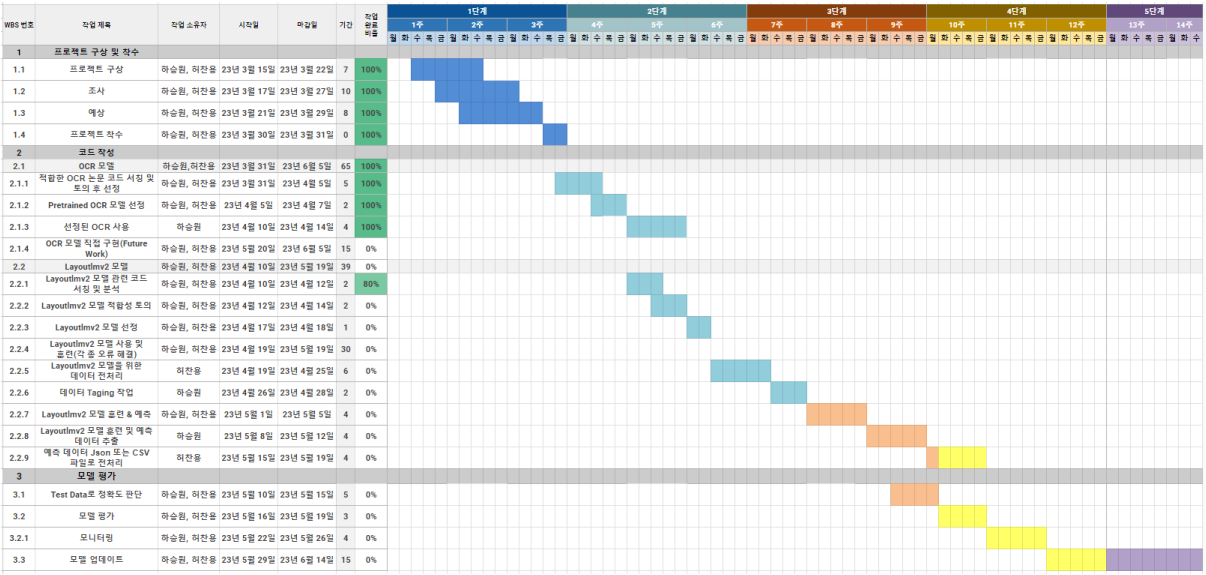


To solve this, we used the SingleChildScrollView widget to display the analysis chart with horizontal scrolling when the data becomes too large. After the modification, the code and image look like this:



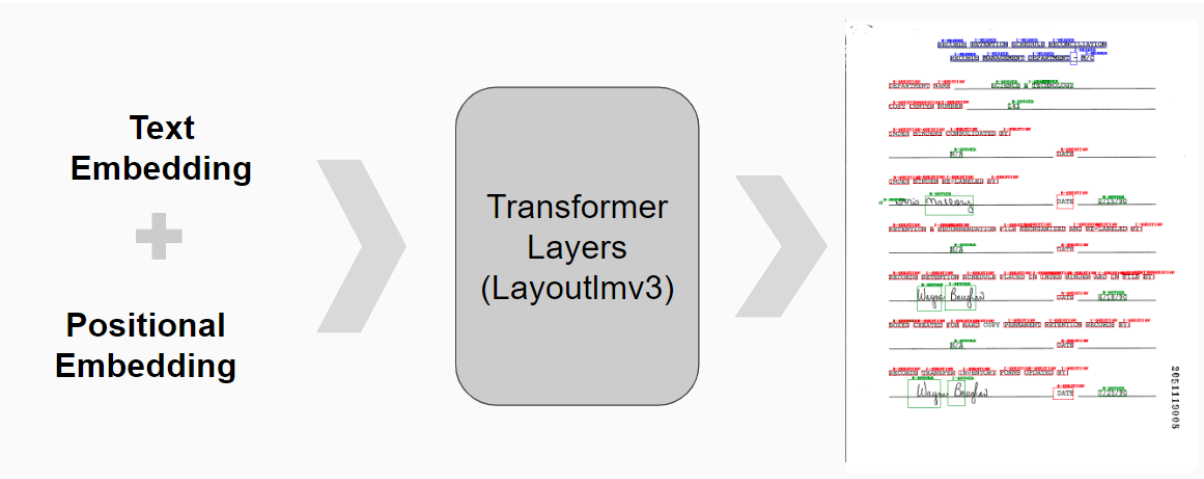
OCM Model

WBS 번호	작업 제목	작업 소유자	시작일	마감일	기간	작업 완료 퍼센트
1	프로젝트 구상 및 착수					
1.1	프로젝트 구상	하승원, 허찬웅	23년 3월 15일	23년 3월 22일	7	100%
1.2	조사	하승원, 허찬웅	23년 3월 17일	23년 3월 27일	10	100%
1.3	예상	하승원, 허찬웅	23년 3월 21일	23년 3월 29일	8	100%
1.4	프로젝트 착수	하승원, 허찬웅	23년 3월 30일	23년 3월 31일	0	100%
2	코드 작성					
2.1	OCR 모델	하승원,허찬웅	23년 3월 31일	23년 6월 5일	65	100%
2.1.1	적합한 OCR 논문 코드 서칭 및 토의 후 선정	하승원, 허찬웅	23년 3월 31일	23년 4월 5일	5	100%
2.1.2	Pretrained OCR 모델 선정	하승원, 허찬웅	23년 4월 5일	23년 4월 7일	2	100%
2.1.3	선정된 OCR 사용	하승원	23년 4월 10일	23년 4월 14일	4	100%
2.1.4	OCR 모델 직접 구현(Future Work)	하승원, 허찬웅	23년 5월 20일	23년 6월 5일	15	0%
2.2	Layoutlmv2 모델	하승원, 허찬웅	23년 4월 10일	23년 5월 19일	39	0%
2.2.1	Layoutlmv2 모델 관련 코드 서칭 및 분석	하승원, 허찬웅	23년 4월 10일	23년 4월 12일	2	80%
2.2.2	Layoutlmv2 모델 적합성 토의	하승원, 허찬웅	23년 4월 12일	23년 4월 14일	2	0%
2.2.3	Layoutlmv2 모델 선정	하승원, 허찬웅	23년 4월 17일	23년 4월 18일	1	0%
2.2.4	Layoutlmv2 모델 사용 및 훈련(각 종 오류 해결)	하승원, 허찬웅	23년 4월 19일	23년 5월 19일	30	0%
2.2.5	Layoutlmv2 모델을 위한 데이터 전처리	허찬웅	23년 4월 19일	23년 4월 25일	6	0%
2.2.6	데이터 Taging 작업	하승원	23년 4월 26일	23년 4월 28일	2	0%
2.2.7	Layoutlmv2 모델 훈련 & 예측	하승원, 허찬웅	23년 5월 1일	23년 5월 5일	4	0%
2.2.8	Layoutlmv2 모델 훈련 및 예측 데이터 추출	하승원	23년 5월 8일	23년 5월 12일	4	0%
2.2.9	예측 데이터 Json 또는 CSV 파일로 전처리	허찬웅	23년 5월 15일	23년 5월 19일	4	0%
3	모델 평가					
3.1	Test Data로 정확도 판단	하승원, 허찬웅	23년 5월 10일	23년 5월 15일	5	0%
3.2	모델 평가	하승원, 허찬웅	23년 5월 16일	23년 5월 19일	3	0%
3.2.1	모니터링	하승원, 허찬웅	23년 5월 22일	23년 5월 26일	4	0%
3.3	모델 업데이트	하승원, 허찬웅	23년 5월 29일	23년 6월 14일	15	0%



All implementations are finished as planned.

By using CLOVA API, get text embedding and positional embedding from receipt and send it to transformer layers (layout lmv3)

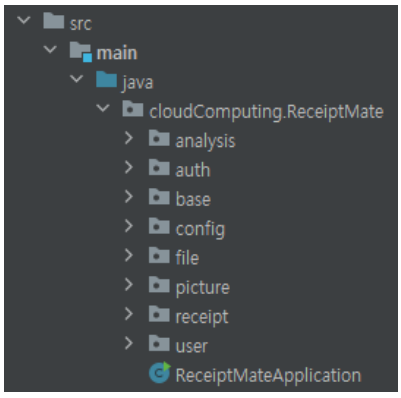


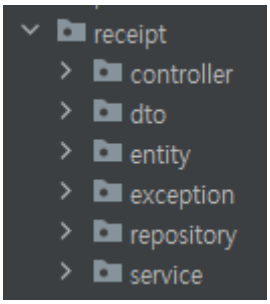
Details are written in Deployment part

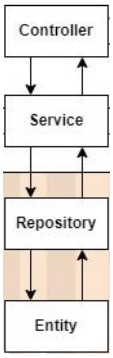
Backend

WBS 번호	작업 제목	작업 소유자	시작일	마감일	기간	작업 완료 비율
1	**Architecture Design**					
1.1	Figma 작업	박찬호	23년 4월 3일	23년 4월 3일	0	100%
1.2	Architecture Design	박찬호	23년 4월 3일	23년 4월 10일	7	100%
1.3	ER Diagram 작성	박찬호	23년 4월 11일	23년 4월 13일	2	100%
1.4	API 목록 작성	박찬호	23년 4월 13일	23년 4월 15일	2	100%
2	**Implementation**					100%
2.1	Entity 작성	박찬호	23년 4월 17일	23년 4월 19일	2	100%
2.2	Domain 단위 layer 구현	박찬호	23년 4월 29일	23년 5월 20일	21	100%
2.2.1	User Repository, Controller	박찬호	23년 4월 29일	23년 5월 1일	2	100%
2.2.2	User Service	박찬호	23년 4월 29일	23년 5월 1일	2	100%
2.2.3	Receipt Repository, Service	박찬호	23년 5월 1일	23년 5월 10일	9	100%
2.2.4	Receipt Service, Controller	박찬호	23년 5월 10일	23년 5월 20일	10	100%
2.2.5	Min.io Object Storage	박찬호	23년 5월 18일	23년 5월 20일	2	100%
2.3	Integration Test	박찬호	23년 5월 3일	23년 5월 22일	19	100%
3	**Quality Management**					100%
4.1	성능 테스트	박찬호	23년 5월 22일	23년 5월 29일	7	100%
4.2	디버깅	박찬호	23년 5월 24일	23년 5월 28일	4	100%

All implementations are finished as planned.







File Tree is arranged by Domain and each domain holds layered architecture

```
private final FileService fileService;

private final AuthService authService;

private final UserRepository userRepository;

private final ReceiptRepository receiptRepository;

private final ByPeriodRepository byPeriodRepository;

private final ByProductRepository byProductRepository;

private final AnalysisRepository analysisRepository;
```

The service of a domain refers other domain by using repository. Main logic (service) not depending on other domain's logic (service)

Layers of Layered Architecture

Controller

Controller only handles handling request, validation, and response at endpoint. It doesn't contain any main logic.

```
no usages  CharliePark
@PostMapping("/existsEmail")
public ResponseEntity<StringResponse> checkEmailAvailability(@Valid @RequestBody EmailRequest emailRequest) {
    return ResponseEntity.ok().body(userService.checkEmailAvailability(emailRequest));
}

no usages  Charlieppark
@PostMapping("/existsNickname")
public ResponseEntity<StringResponse> checkNicknameAvailability(@RequestBody NicknameRequest nicknameRequest) {
    return ResponseEntity.ok().body(userService.checkNicknameAvailability(nicknameRequest));
}

no usages  CharliePark
@PostMapping("/signup")
public ResponseEntity<UserResponse> signUp(@RequestBody SignUpRequest signUpRequest) {
    return ResponseEntity.ok().body(userService.signUp(signUpRequest));
}

no usages  CharliePark
@PostMapping("/login")
public ResponseEntity<UserResponse> logIn(@RequestBody LogInRequest logInRequest) {
    return ResponseEntity.ok().body(userService.logIn(logInRequest));
}

no usages  Charlieppark
@PostMapping("/kakao/signup")
public ResponseEntity<UserResponse> kakaoSignUp(@RequestBody KakaoSignUpRequest kakaoSignUpRequest) {
    return ResponseEntity.ok().body(userService.signUpByKakao(kakaoSignUpRequest));
}

no usages  Charlieppark
@PostMapping("/kakao/login")
public ResponseEntity<UserResponse> kakaoLogIn(@RequestBody KakaoLogInRequest kakaoLogInRequest) {
    return ResponseEntity.ok().body(userService.logInByKakao(kakaoLogInRequest));
}

no usages  Charlieppark
@GetMapping("/kakao/exist/{id}")
public ResponseEntity<BooleanResponse> kakaoExist(@PathVariable("id") String id) {
    return ResponseEntity.ok().body(userService.existByKakaoId(id));
}
```

Service

Service handles main logic of the domain

```
1 usage  CharliePark +1 *
@Transactional
public UserResponse signUp(SignUpRequest signUpRequest) {

    if (userRepository.existsByEmail(signUpRequest.getEmail())) throw new DuplicateEmailException();
    if (userRepository.existsByNickname(signUpRequest.getNickname())) throw new DuplicateNicknameException();

    signUpRequest.setPassword(authService.encodePassword(signUpRequest.getPassword()));

    User user = UserMapper.INSTANCE.requestToUser(signUpRequest);
    user.setIsKakao(false);

    final User savedUser = userRepository.save(user);

    UserResponse userResponse = UserMapper.INSTANCE.userToResponse(savedUser);
    userResponse.setTokenResponse(jwtUtil.generateToken(getTokenInfo(savedUser)));

    return userResponse;
}
```

Repository

Repository handles communication with DataBase and query of the domain.

```
4 usages  Charlieppark +1
@Repository
public interface ByProductRepository extends JpaRepository<ByProduct, Long> {
    no usages  Charlieppark
    List<ByProduct> findAllByYearAndName(Integer year, String name);

    no usages  Charlieppark
    List<ByProduct> findAllByYearAndMonthAndName(Integer year, Integer month, String name);

    no usages  Charlieppark
    List<ByProduct> findAllByYearAndMonthAndDayAndName(Integer year, Integer month, Integer day, String name);

    1 usage  Charlieppark
    List<ByProduct> findAllByNameAndAnalysis(String name, Analysis analysis);

    1 usage  Charlieppark
    @Query("select p.name from ByProduct p where p.analysis = ?1")
    List<String> getAllNamesByAnalysis(Analysis analysis);

    1 usage  Charlieppark
    void deleteAllByOriginalReceiptId(Long originalReceiptId);

    1 usage  Charlieppark
    Boolean existsByOriginalReceiptId(Long originalReceiptId);
}
```

Entity

Entity is an object that represents table in DB

```
21 usages  Charlieppark +1
@Entity
@Builder
@Getter
@NoArgsConstructor
@AllArgsConstructor
@DynamicUpdate
@Setter

public class Receipt {

    @Id
    @GeneratedValue
    private Long id;

    private Date createdDate;

    private String detailKey;

    @ElementCollection
    @CollectionTable(
            name = "info",
            joinColumns = {@JoinColumn(name = "receipt_id", referencedColumnName = "id")})
    @MapKeyColumn
    @Column(name = "info_name")
    private Map<String, String> info;

    @ManyToOne(fetch = FetchType.LAZY)
    @JoinColumn(name = "user_id")
    private User owner;
}
```

API List

analysis-controller Analysis Controller

- GET /api/analysis/month getByMonth
- GET /api/analysis/names getNames
- POST /api/analysis/product getByName
- GET /api/analysis/year getByYear

file-controller File Controller

- GET /api/file/list list

picture-controller Picture Controller

- GET /api/picture/{object} getObject
- POST /api/picture/save save

receipt-controller Receipt Controller

- GET /api/receipt/{object} getObject
- POST /api/receipt/add addReceipt
- DELETE /api/receipt/delete/{id} deleteReceipt
- GET /api/receipt/list listReceipt
- PUT /api/receipt/update updateReceipt

user-controller User Controller

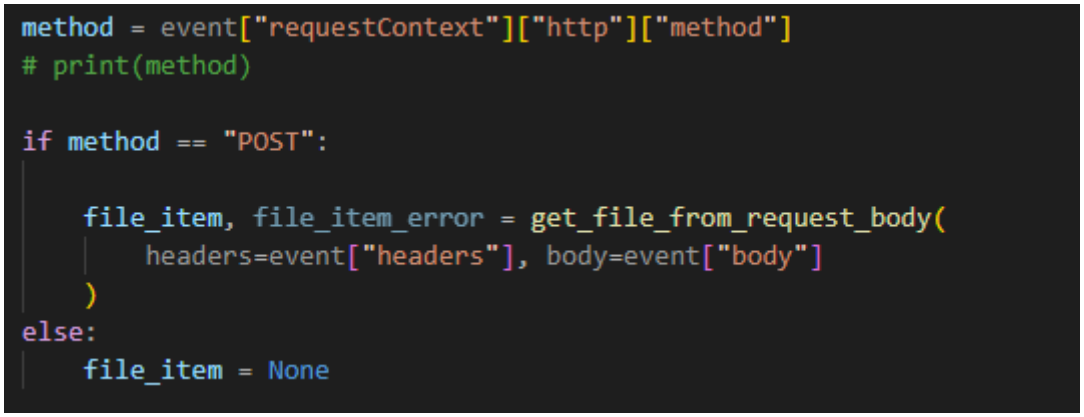
- POST /api/user/existsEmail checkEmailAvailability
- POST /api/user/existsNickname checkNicknameAvailability
- POST /api/user/getNickname getEmail
- GET /api/user/kakao/exist/{id} kakaoExist
- POST /api/user/kakao/login kakaoLogIn
- POST /api/user/kakao/signup kakaoSignUp
- POST /api/user/login logIn
- POST /api/user/modifyNickname modifyNickname
- POST /api/user/reset resetPassword
- POST /api/user/signup signUp

Deployment (cloud)

OCR Model source code for AWS Lambda

1. Lambda Function

```
def lambda_handler(event, context):
    print(event)
    method = event["requestContext"]["http"]["method"]
    # print(method)

    if method == "POST":

        file_item, file_item_error = get_file_from_request_body(
            headers=event["headers"], body=event["body"]
        )
    else:
        file_item = None

    body = json.dumps(file_item, ensure_ascii=False).encode('utf-8')  # Encode the response as

    body_base64 = base64.b64encode(body).decode('utf-8')  # Convert the UTF-8 encoded response

    return {
        'isBase64Encoded': True,
        'statusCode': 200,
        'headers': {
            'Content-Type': 'application/json; charset=utf-8'  # Specify UTF-8 encoding in the
        },
        'body': body_base64
    }
```



```
method = event["requestContext"]["http"]["method"]
# print(method)

if method == "POST":

    file_item, file_item_error = get_file_from_request_body(
        headers=event["headers"], body=event["body"]
    )
else:
    file_item = None
```

2. If the request method of the Lambda REST API is POST, execute the entire process code.

get_file_from_request_body()

```
import cgi
def get_file_from_request_body(headers, body):
    fp = io.BytesIO(base64.b64decode(body)) # decode
    environ = {"REQUEST_METHOD": "POST"}
    headers = {
        "content-type": headers["content-type"],
        "content-length": headers["content-length"],
    }

    fs = cgi.FieldStorage(fp=fp, environ=environ, headers=headers)

    print("FS: ", fs)


    input_file = fs["file"].file.read() # input으로 사용
    model_name = 'seungwon12/cloud_computing_project'

    img = Image.open(BytesIO(input_file)).convert('RGB')
    predicter = Predict(model_name, img)

    preprocess_data = predicter.predict_preprocess()
    box, predict, text = predicter.predict(preprocess_data)

    output = predicter.predict2output(box, predict, text)


    return [output, None]
```

```
img = Image.open(BytesIO(input_file)).convert('RGB')
```

Open the input_file as a byte stream and open the image using Image.open. Then, convert the image to RGB format using convert('RGB') and save it in the img variable.

```
predicter = Predict(model_name, img)
```

```
class Predict:
    def __init__(self,Model_path,img):

        self.label_list= ['others','key','value','total','column_name','item','count','money']
        # 전처리를 위한 모델 호출
        self.processor =  AutoProcessor.from_pretrained("/opt/ml/process", apply_ocr=False)
        self.model=AutoModelForTokenClassification.from_pretrained("/opt/ml/model", num_labels=len(self.label_list))
        self.image=img
        self.api_url = 'https://yartf43wok.apigw.ntruss.com/custom/v1/18186/0c8ba49a3121aa041c2fa2d2ef06b11b5f61d50a1
        self.secret_key = '
```

Specify the class of the code for prediction and initialize the received image data and the model's name.

3. Extract text from the image through the Naver OCR API using the received image.

predict_preprocess()

```
preprocess_data = predicter.predict_preprocess()
```

Call the function to preprocess the image data into the necessary format for model prediction.

```
def predict_preprocess(self):
    bytes_stream = io.BytesIO()

    # Convert the PIL Image to bytes and store it in the BytesIO stream
    self.image.save(bytes_stream, format='JPEG')  # Specify the desired format (JPEG, PNG,

    # Read the contents of the BytesIO object
    img = bytes_stream.getvalue()

    file_data=base64.encodebytes(img).decode('utf-8')
    # naver api 를 호출하기 위해서 파일 형식을 json형식으로 api를 호출하며 데이터를 넘긴다.
    request_json = {
        'images': [
            {
                'format': 'jpg',
                'name': 'demo',
                'data': file_data
            }
        ],
        'requestId': str(uuid.uuid4()),
        'version': 'V2',
        'timestamp': int(round(time.time() * 1000)),
        'enableTableDetection': True
    }

    payload = json.dumps(request_json).encode('UTF-8')
    headers = {
    'X-OCR-SECRET': self.secret_key,
    'Content-Type': 'application/json'
    }


#requests함수를 이용하여 api를 호출 결과값을 result에 저장
    response=requests.post(self.api_url, headers=headers, data = payload)
    result=response.json()
```

Send the request_json to the Naver API using the post command to obtain the result. Convert the OCR result obtained through post into json format and save it.

4. Preprocess the extracted data for applying to the model.

This process uses the table format obtained from the Naver OCR API to perform preprocessing. Extract the text and related information about the text position from the image data separately, as only the text and its position-related data are used.

```
# naver ocr api 반환값으로 이미지에서 모든 텍스트를 'text' 배열에 저장
# 그리고 그 텍스트(바운딩 박스)의 위치 정보는 'location' 배열에 저장
text=[]
location=[]
for i in result['images'][0]['fields']:
    text.append(i['inferText'])
    location.append([int(i['boundingPoly']['vertices'][0]['x']),int(i['boundingPoly']['vertices'][0]


    table_lable=[0 for _ in range(len(text))]

embedding_loaction=[]

image=self.image
```

Extract the bounding box positions of the text from the data obtained through the Naver OCR API and save them.

```
for _ in range(3):
    for index , value in enumerate(embedding_loaction):
        #print(text[index],'\t',table_lable[index],'\t',embedding_loaction[index])
        if len(embedding_loaction) <= index +1:
            break
        else:
            if abs(embedding_loaction[index][2]-embedding_loaction[index+1][0]) < 4:
                embedding_loaction[index+1]=[embedding_loaction[index][0],embedding_loaction[index][1],
                                             embedding_loaction[index+1][2],embedding_loaction[index+1][3]]
                del embedding_loaction[index]

                text[index+1]=text[index]+' '+text[index+1]
                del text[index]

                del table_lable[index+1]
```

Preprocess the bounding box positions into positions suitable for model usage.

```
temp={}
temp['id'] =0
temp['words'] = text#text
temp['bboxes'] = embedding_loaction # make sure to normalize your bounding boxes
temp['ner_tags']=table_lable
```

Generate a new training data from the preprocessed data.

5. Make predictions using the preprocessed data as input.

```
import cgi
def get_file_from_request_body(headers, body):
    fp = io.BytesIO(base64.b64decode(body)) # decode
    environ = {"REQUEST_METHOD": "POST"}
    headers = {
        "content-type": headers["content-type"],
        "content-length": headers["content-length"],
    }

    fs = cgi.FieldStorage(fp=fp, environ=environ, headers=headers)

    print("FS: ", fs)


    input_file = fs["file"].file.read() # input으로 사용
    model_name = 'seungwon12/cloud_computing_project'

    img = Image.open(BytesIO(input_file)).convert('RGB')
    predicter = Predict(model_name, img)

    preprocess_data = predicter.predict_preprocess()
    box, predict, text = predicter.predict(preprocess_data)

    output = predicter.predict2output(box, predict, text)


    return [output, None]
```

```
box, predict, text = predicter.predict(preprocess_data)
```

After all the prediction steps are completed, return to the 'get_file_from_request_body()' function, which is the starting point of the prediction process. Execute the 'predict2output()' function to convert the predicted results into the data format expected by the API requester. Pass the parameters received, which include the box containing the position data of the text, the layout values predicted by the Layoutlmv3 model for each text, and the text data extracted from the image.

```
  output = predicter.predict2output(box, predict, text)


  return [output, None]
```

predict2output()

The process of generating the result using the predicted text classification data and the data obtained during image preprocessing.

Classify the text layout using the Layoutlmv3 model. When the loop is executed, the text data corresponding to each layout is classified and stored in 'output_dict'.

```
output_dict={}
for i in self.label_list:
    output_dict[i]=[]

for i in range(len(layout)):
    if str(layout[i])=='others':
        output_dict['others'].append(text[i])

    elif str(layout[i])=='key':
        output_dict['key'].append(text[i])
    elif str(layout[i])=='value':
        output_dict['value'].append(text[i])
    elif str(layout[i])=='total':
        output_dict['total'].append(text[i])
    elif str(layout[i])=='column_name':
        output_dict['column_name'].append(text[i])
    elif str(layout[i])=='item':
        output_dict['item'].append(text[i])
    elif str(layout[i])=='count':
        output_dict['count'].append(text[i])
    elif str(layout[i])=='money':
        output_dict['money'].append(text[i])
```

Store the items predicted as item, quantity, and amount in the receipt image in a new column 'new_dict'.

```
output_dict['csv']=[]

try:
    new_dict = {
        #'column_name': output_dict['column_name'],
        'item': output_dict['item'],
        'count': output_dict['count'],
        'money': output_dict['money']
    }
    #품목 수량 단위 금액
    max_length = max( len(new_dict['item']), len(new_dict['count']), len(new_dict['money']))

    print('max_length',max_length)
    for key in ['item', 'count', 'money']:
        new_dict[key] += [''] * (max_length - len(new_dict[key]))
    print('new_dict',new_dict)

    for indx in range(len(new_dict['money'])):
        new_dict['money'][indx] = new_dict['money'][indx].replace(',', '')  # Remove commas from the string
        new_dict['money'][indx] = new_dict['money'][indx].replace('.', '')
        new_dict['money'][indx] = new_dict['money'][indx].replace('원', '')  # Remove the '원' symbol from the string
        new_dict['money'][indx]=int(new_dict['money'][indx])


    col_name=['품목','수량','금액']

    df=pd.DataFrame.from_dict(new_dict)
    count_list = []
    df.columns=col_name
```

Then convert 'new_dict' dictionary into a dataframe.

```
col_name=['품목','수량','금액']

df=pd.DataFrame.from_dict(new_dict)
count_list = []
df.columns=col_name
```

Add a unit column to the API request value. If the strings 'g', 'ml', 'kg', 'l' exist in the 'item' column of the data, update the unit column with the data and update the quantity according to the unit.

```
import re
for index, row in df.iterrows():
    match = re.search(r"(\d+)(g|ml|kg|l)", row['품목'])
    if match:
        quantity = int(match.group(1))
        unit = match.group(2)
        print("Item:", row['품목'])
        print("Quantity:", quantity)
        print("Unit:", unit)
        print()
        count_list.append(quantity)
        if unit == 'g':
            df.at[index, '수량'] *= quantity
        elif unit == 'ml':
            df.at[index, '수량'] *= quantity
        elif unit == 'kg':
            df.at[index, '수량'] *= quantity * 1000
        elif unit == 'l':
            df.at[index, '수량'] *= quantity * 1000
    else:
        count_list.append('개')

df['단위'] = count_list

# Change the column order
df = df[['품목', '수량', '단위', '금액']]


csv_string = df.to_csv(index=False)
print('output_dict',output_dict)
```

The code that updates the quantity of each item by iterating through each row. This code adjusts the quantity to the appropriate unit for products like meat or vegetables that use g, kg, and liquid products that use l, ml.

```python
def lambda_handler(event, context):
    print(event)
    method = event["requestContext"]["http"]["method"]
    # print(method)

    if method == "POST":

        file_item, file_item_error = get_file_from_request_body(
            headers=event["headers"], body=event["body"]
        )
    else:
        file_item = None

    body = json.dumps(file_item, ensure_ascii=False).encode('utf-8')  # Encode the response as

    body_base64 = base64.b64encode(body).decode('utf-8')  # Convert the UTF-8 encoded response

    return {
        'isBase64Encoded': True,
        'statusCode': 200,
        'headers': {
            'Content-Type': 'application/json; charset=utf-8'  # Specify UTF-8 encoding in the
        },
        'body': body_base64
    }
```

Check the HTTP method of the input event and perform the corresponding operation. The response is returned in JSON format, and the response body is encoded in UTF-8 and encoded in base64.

```python
body = json.dumps(file_item, ensure_ascii=False).encode('utf-8')  # Encode the response as UTF-8

body_base64 = base64.b64encode(body).decode('utf-8')  # Convert the UTF-8 encoded response to base64
```

'body' is the response encoded in UTF-8 format.
'body_base64' is a variable that reconverts 'body' back to base64 format.

```python
return {
    'isBase64Encoded': True,
    'statusCode': 200,
    'headers': {
        'Content-Type': 'application/json; charset=utf-8'  # Specify UTF-8 encoding in the Content-Type header
    },
    'body': body_base64
}
```

The code returns the results obtained through the above process.

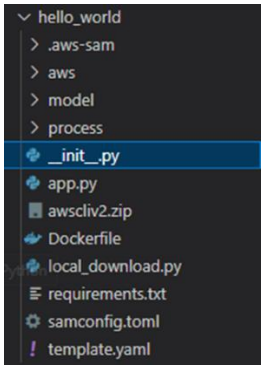
AWS Lambda Deployment and Creation

Reference (https://www.youtube.com/watch?v=tL3-PqQvdF4&t=756s)

1. Specify the files for creating a Docker image.

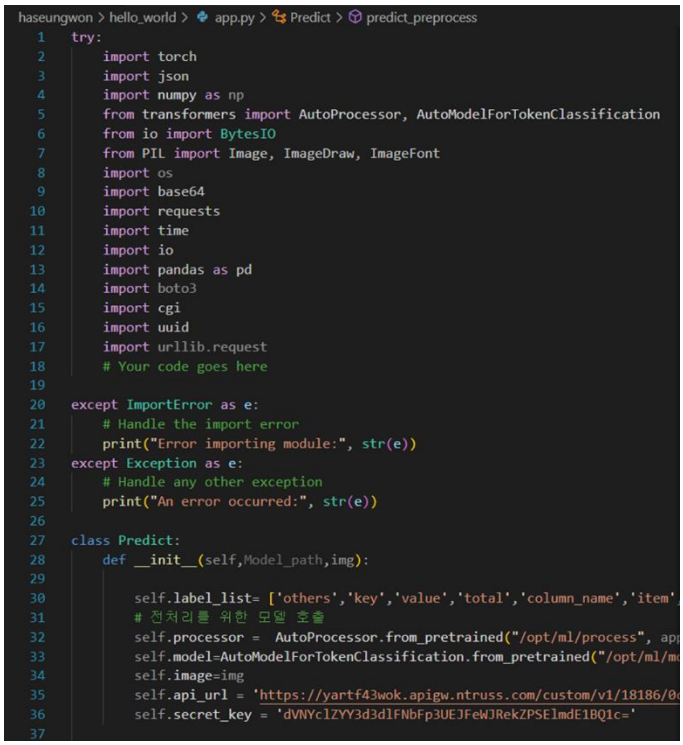
(Code: https://github.com/ramsrigouthamg/deployment-question-answering-aws-lambda)

Code files



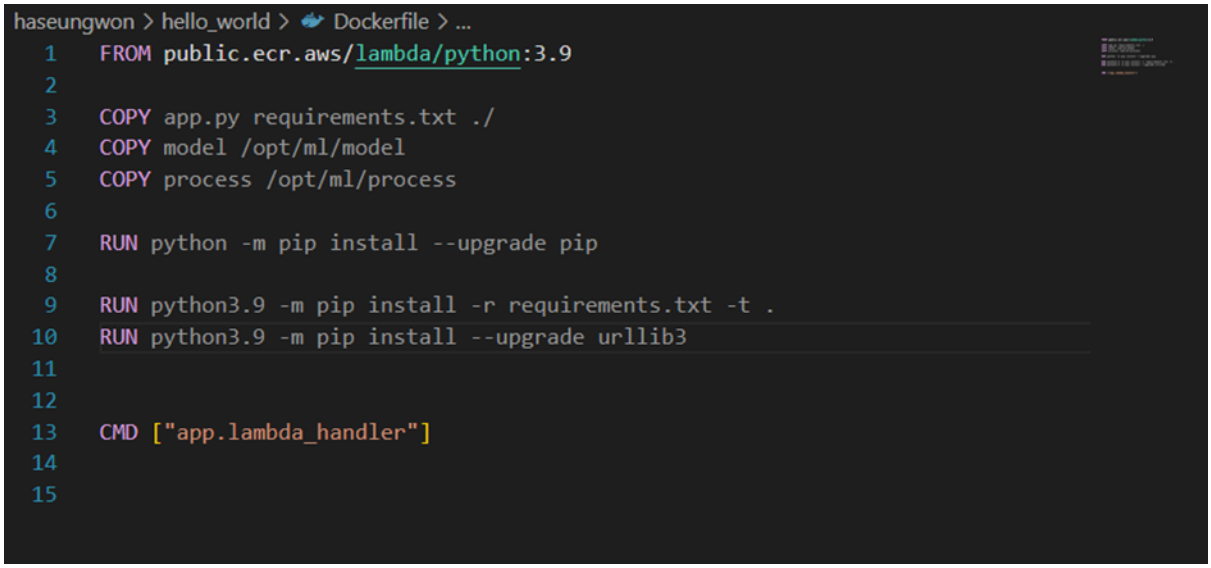
app.py:

This file is the AWS Lambda function that is executed after creating the Docker image. It is usually written in Python and contains the main logic of the Lambda function. This file defines the tasks that the Lambda function will perform when it is executed.



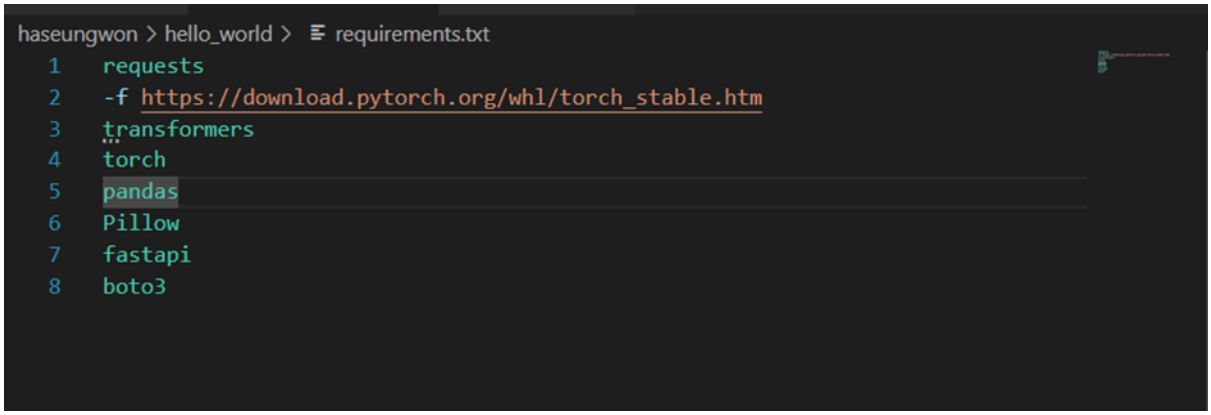
Dockerfile:

It is a file that configures the environment of a Docker image and installs the necessary libraries and dependencies. The Dockerfile provides instructions for building a Docker image. This file is read by the Docker client when building a Docker image. The Dockerfile defines the execution environment of the application and specifies the required packages, libraries, configurations, etc., for the application to run.



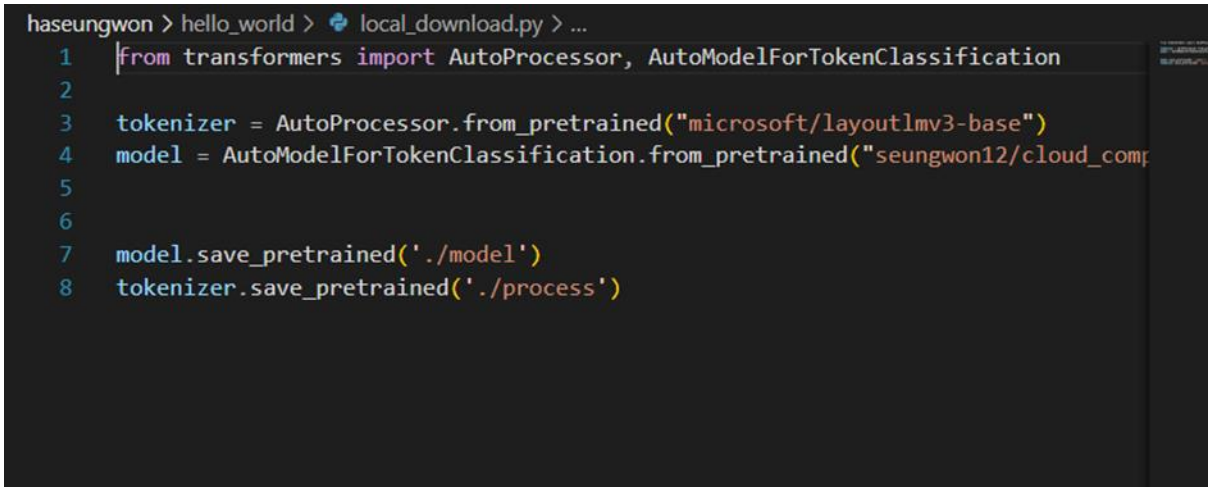
requirements.txt:

This file contains a list of dependencies for a Python application. It is typically used along with the pip install command for installing packages. It is read by the Dockerfile to install dependencies within the Docker image. requirements.txt records the Python packages and version information required for the application to run.



local_download.py:

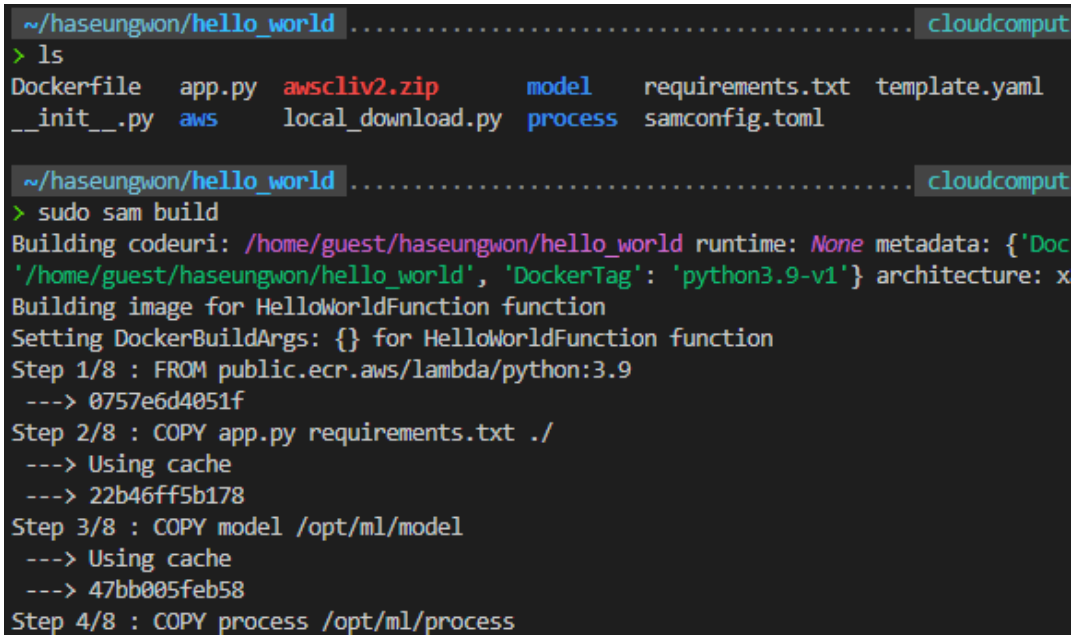
During the process of creating a Docker image for use in Lambda, it loads and saves the model to the Lambda image path.



2. Creating Docker Image and Lambda Using AWS SAM

Building the Docker image:

Generate a Docker image using the command 'sam build' from the Docker image file path. This creates an image based on the Dockerfile, which includes the environment in which the image will run, the required libraries for the control function, and the model to be loaded and saved.



Once the Docker image is created, executing the sam --guided command prompts for interactive input of the following configuration options:

```
> sudo sam deploy --guided

Configuring SAM deploy
======================

        Looking for config file [samconfig.toml] :  Found
        Reading default arguments  :  Success

        Setting default arguments for 'sam deploy'
        =========================================
        Stack Name [project]: test
        AWS Region [us-east-1]:
        #Shows you resources changes to be deployed and require a 'Y' to initiate deploy
        Confirm changes before deploy [Y/n]: n
        #SAM needs permission to be able to create roles to connect to the resources in your template
        Allow SAM CLI IAM role creation [Y/n]: n
        Capabilities [['CAPABILITY_IAM']]: n
        #Preserves the state of previously provisioned resources when an operation fails
        Disable rollback [Y/n]: n
        HelloWorldFunction may not have authorization defined, Is this okay? [y/N]: n
```

Stack Name [project]:: Prompt for the name of the CloudFormation stack to create.

AWS Region [us-east-1]:: Prompt for the AWS region to use. Users can input their desired AWS region here.

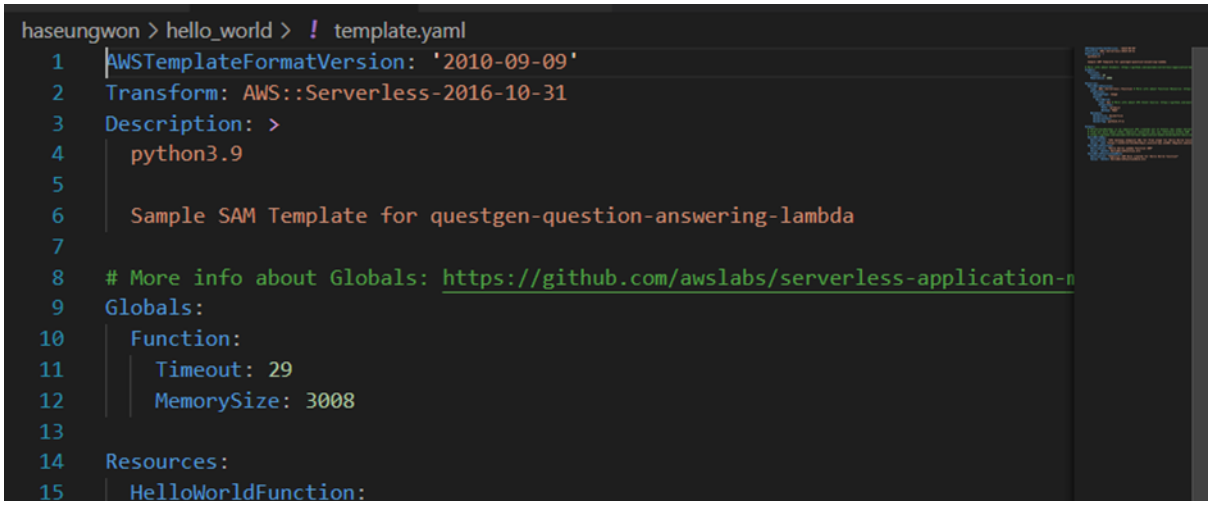
Confirm changes before deploy [Y/n]:: Prompt to confirm whether to review changes before deployment.

Allow SAM CLI IAM role creation [Y/n]:: Prompt to grant SAM CLI permission to create IAM roles.

Capabilities [['CAPABILITY_IAM']]:: Prompt for permissions related to IAM functionality.
Disable rollback [Y/n]:: Prompt to disable rollback functionality.

HelloWorldFunction may not have authorization defined, Is this okay? [y/N]: n: Warning message that authorization is not defined for 'HelloWorldFunction'.

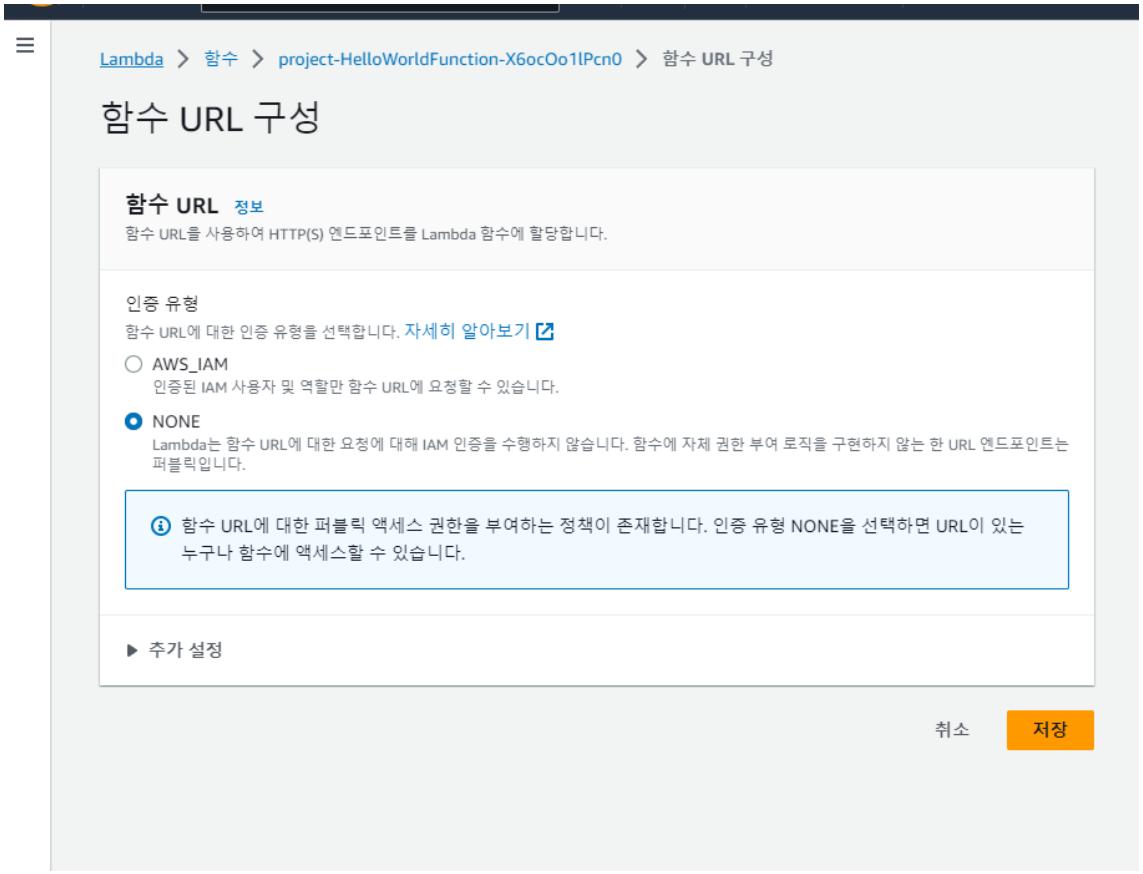
After completing the sam --guided command, a template.yaml file is generated based on the configured values. This file is a SAM template used by AWS CloudFormation and includes the Lambda function to be deployed and its corresponding settings. Subsequently, the sam deploy command can be used to deploy the Lambda function based on the template.yaml file. The sam deploy command internally creates and updates the CloudFormation stack, provisions Lambda functions and related resources in AWS.

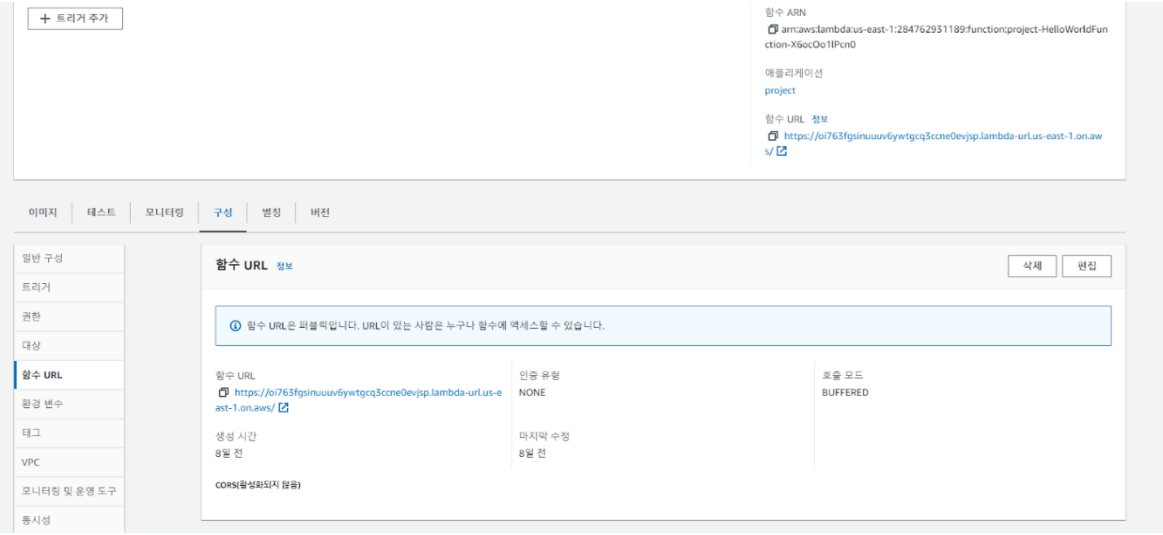


Therefore, after reviewing the template.yaml file generated by the sam --guided command, you can deploy the Lambda function based on that file by executing the sam deploy command.

3. Deploying Lambda with Docker Image After AWS SAM Deployment

Creating Lambda function URL: The URL of a Lambda function is generated based on the ARN of the Lambda function. Typically, the URL of a Lambda function has the following format:
https://lambda.{region}.amazonaws.com/2015-03-31/functions/{function-arn}/invocations
This URL is provided to external users to enable them to use the Lambda function. However, this method does not provide additional features such as authentication, authorization, logging, throttling, etc., as it does not use API Gateway. Using API Gateway is generally the recommended approach, but you can also create and use the Lambda function URL directly by following the above procedure.


Lambda > 함수 > project-HelloWorldFunction-X6ocOo1lPcn0 > 함수 URL 구성
함수 URL 구성
함수 URL 정보
함수 URL을 사용하여 HTTP(S) 엔드포인트를 Lambda 함수에 할당합니다.
인증 유형
함수 URL에 대한 인증 유형을 선택합니다. 자세히 알아보기
AWS_IAM
인증된 IAM 사용자 및 역할만 함수 URL에 요청할 수 있습니다.
NONE
Lambda는 함수 URL에 대한 요청에 대해 IAM 인증을 수행하지 않습니다. 함수에 자체 권한 부여 로직을 구현하지 않는 한 URL 엔드포인트는 퍼블릭입니다.
함수 URL에 대한 퍼블릭 액세스 권한을 부여하는 정책이 존재합니다. 인증 유형 NONE을 선택하면 URL이 있는 누구나 함수에 액세스할 수 있습니다.
추가 설정
취소
저장


+ 트리거 추가
함수 ARN
arn:aws:lambda:us-east-1:284762931189:function:project-HelloWorldFunction-X6ocOo1lPcn0
애플리케이션
project
함수 URL 정보
https://oi763fgsinuuuv6ywtgcq3ccne0evjsp.lambda-url.us-east-1.on.aws/
이미지
테스트
모니터링
구성
별칭
버전
일반 구성
트리거
권한
대상
함수 URL
환경 변수
태그
VPC
모니터링 및 운영 도구
동시성
함수 URL 정보
삭제
편집
함수 URL은 퍼블릭입니다. URL이 있는 사람은 누구나 함수에 액세스할 수 있습니다.
함수 URL
https://oi763fgsinuuuv6ywtgcq3ccne0evjsp.lambda-url.us-east-1.on.aws/
인증 유형
NONE
호출 모드
BUFFERED
생성 시간
8일 전
마지막 수정
8일 전
CORS(활성화되지 않음)

Result:

Input image data

model output

lambda output

Reasons for Model Selection

After trying various options such as Naver OCR, Tesseract, and Cloud Vision API, we found that the accuracy for Korean language was significantly lower. Among them, we chose Naver OCR, which had the highest accuracy for Korean language.

Comparative analysis between Layoutlmv2 and Layoutlmv3 models showed that Layoutlmv3 was lighter and demonstrated relatively higher performance in terms of accuracy. Therefore, based on the expected results, we selected the Layoutlmv3 model for the project.

Challenges and Solutions

During the preprocessing stage, we identified an issue where the positional embedding caused the text's position after OCR to have a ratio less than 0. To address this, we modified the code and treated cases where the ratio became less than 0 as error cases, handling them with exception handling.

During the OCR process, noise occurred in the image data when extracting data from receipt images. For example, even for the same receipt, the image data could vary slightly due to angles or light reflections. Additionally, running OCR caused changes in the position of the text. Due to this issue, when extracting text, the problem of classifying unrelated words into the same category occurred, even if they were not in the same category. For example, in receipt images, the word

"구매자" ("Buyer") was recognized as a single text and the coordinates were returned. However, cases occurred where a word like ":" was also included in the same range. To address this noise issue, we removed text containing punctuation marks and applied additional processes to convert the values of bounding boxes that indicate the position of the text accordingly.

The AWS Lambda service had the limitation of not supporting various functions we used, such as transformers and pandas. To address this, we pre-installed the necessary libraries and utilized a Docker container image to configure the environment and generate the Lambda image. During the Docker image building process, the required dependency libraries were installed to resolve this limitation.

The functions for downloading the models, which are invoked while running the functions, were also pre-specified, so when the image is built, they are stored in the environment where the image is executed. As a result, it reduced the processing time required for downloading the model every time the function is executed.

Using AWS SAM, the process of building a Docker image and deploying it to AWS services took approximately 30 minutes. This posed a problem because it consumed too much time to make code modifications or format changes to the results. To address this, we only performed the relatively faster Docker image building process and used the 'sam local start-api' function provided by SAM service to test the API in a local host manner, thereby resolving the issue.

Backend Deployment settings

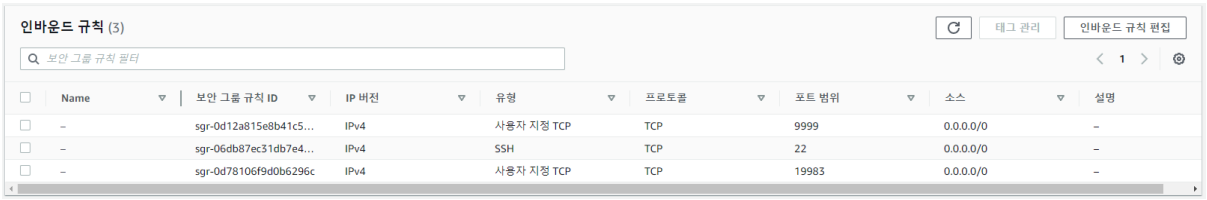
Backend is deployed to AWS EC2 t2.micro which has 1 vCPU (1 physical core), 1 GiB ram.

이름	vCPU(가상 CPU)	RAM(GiB)	CPU 크레딧/시간	온디맨드 요금/시간*	1년 약정 예약 인스턴스 실질 시간당*	3년 약정 예약 인스턴스 실질 시간당*
t2.nano	1	0.5	3	0.0058 USD	0.003 USD	0.002 USD
t2.micro	1	1.0	6	0.0116 USD	0.007 USD	0.005 USD
t2.small	1	2.0	12	0.023 USD	0.014 USD	0.009 USD
t2.medium	2	4.0	24	0.0464 USD	0.031 USD	0.021 USD
t2.large	2	8.0	36	0.0928 USD	0.055 USD	0.037 USD
t2.xlarge	4	16.0	54	0.1856 USD	0.110 USD	0.074 USD
t2.2xlarge	8	32.0	81	0.3712 USD	0.219 USD	0.148 USD

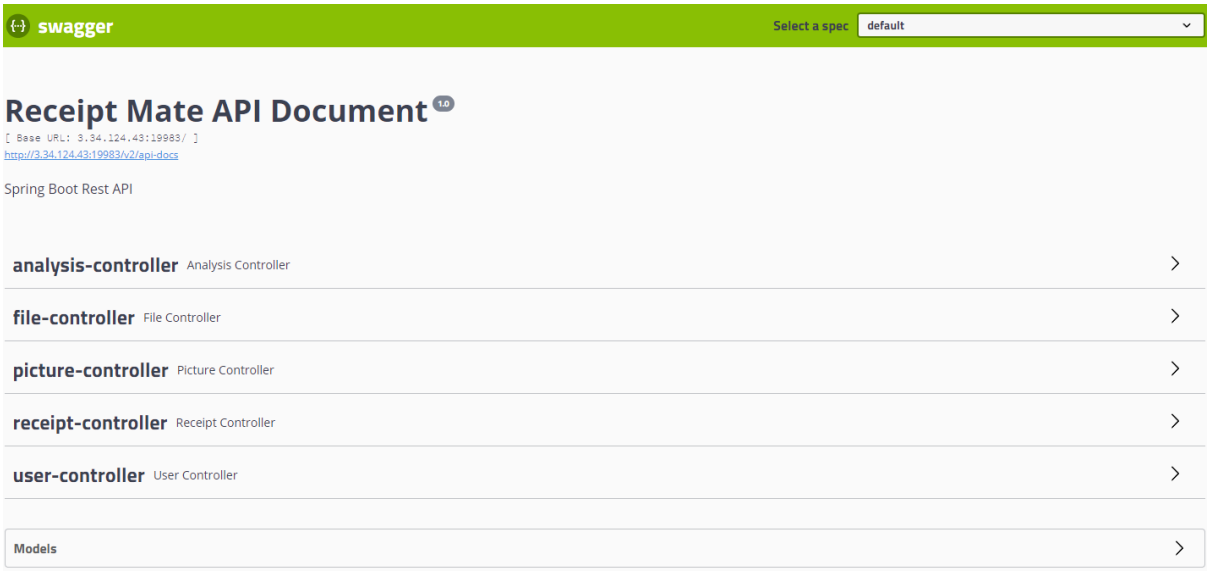
It is stably running on t2.micro instance for 2 weeks without shutting down because of lack of memory (27.7 ~ 80% of memory).

```
45814 ubuntu    20   0 2311M  267M  9856 S  0.0 27.7 12:39.65 java -jar ReceiptMate-0.0.1-SNAPSHOT.jar
```

Inbound rules are set to open port 9999 (minio), 19983 (HTTP API)



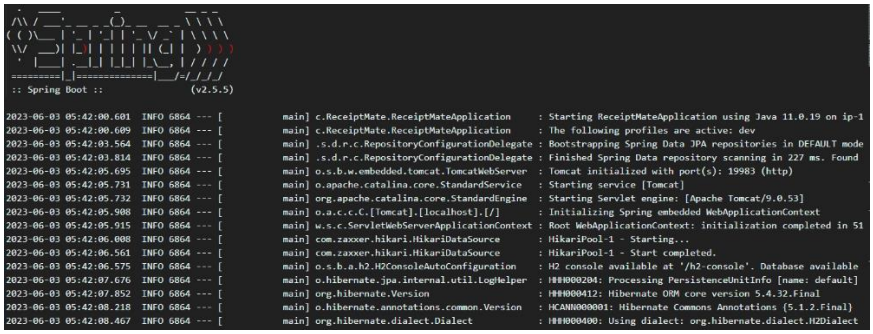
API Server running on http://3.34.124.43:19983/



Backend used running on nohup and background

```
ubuntu@ip-172-31-41-115:~$ nohup java -jar ReceiptMate-0.0.1-SNAPSHOT.jar &
```

Log is managed by nohup.out



Ⅲ Test

Demo

Demos can be found in the links of youtube

Signup : https://www.youtube.com/watch?v=i7k4ixNJqy0
Login : https://www.youtube.com/shorts/eISYiSmGbU0
Kakao Authentication : https://www.youtube.com/shorts/cxV3A97wpW8
Add Receipt: https://www.youtube.com/shorts/qvUaKynuBPY
Detail : https://www.youtube.com/shorts/a-SidSMV3A0
Settings : https://www.youtube.com/shorts/dlTwpxRiRIY
Analysis : https://www.youtube.com/shorts/-fnkmiuP26M

Test results

Endpoint	Domain	Type	Result	Notes
Backend	User	Integration	Pass	Logic of finding login email should be revised for security
Backend	Receipt	Integration	Pass	Updating needs delete and recreate, so it should be more robust
Backend	Picture	Integration	Pass	
Backend	File	Integration	Pass	Save functions should be refactored
Backend	Analysis	Integration	Pass	
Frontend	User	E2E	Pass	
Frontend	Receipt	E2E	Pass	
Frontend	Analysis	E2E	Pass	
Frontend	Settings	Integration	Pass	Logic of profile images should be revised for multi device
OCR Model	Model	Integration	Pass	Postprocessing should be enhanced for receipts with complexity.
OCR Model	Lambda	E2E	Pass	

IV Conclusion

Team Organization & Contribution

Name	Role	Tech Stack	Contribution
박찬호	Team Leader, Frontend, Backend	Spring Boot, Flutter	Design (Features, Structure, Figma, Userflow diagram, Logo), Assisting Lambda deployment, Implementation and deployment of HTTP API Server, Implementation of Frontend screens
정우섭	Frontend	Flutter	Code style convention for Flutter, Configuration of Flutter baseline codes, Implementation of Frontend screens,
하승원	OCR Model	PyTorch, Layoutlmv3	Labeling work using the collected data, Collected OCR result from receipts, Embedding and preprocessing of OCR data, Training and evaluating Layoutlmv3 model, Lambda deployment
허찬용	OCR Model	PyTorch, Layoutlmv2	Labeling work using the collected data, Collected OCR result from receipts, Embedding and preprocessing of OCR data, Training and evaluating Layoutlmv2 model, Parsing output result for backend

Detailed Contribution

TEAM: Front-End

정우섭

- Scheduling, creating a Gantt Chart (100%)
- Creating a Requirement Definition (100%)
- Preparation of interim presentation data (100% - implementation method, responsible for architecture)
- Make code style convention (100%)
- Completed Implementation of all the screens in charge (100%)

박현찬

- Creating a Requirement Definition (40%)

박찬호

- Draw Figma for UI (100%)
- Draw Userflow diagram (100%)
- Configure color and style for frontend design (100%)

Team: BACK-END

The Backend team completed Architecture Design and completed about 90% of the planned implementation. For more information, User Domain, Min.io connection and implementation have been completed, and Receipt Domain is currently under development. Receipt Domain is about 80% implemented. In addition, additional implementations (OAuth, detailed household accounts by period and item) added after the interim announcement are also ongoing, with a 50% of implementation.

Team: ML

하승원

- Labeling work using the collected data. (50%)
- Using the Naver OCR API, collected Json-type OCR data from the receipt image. (100%)
- Embedding and preprocessing of OCR data obtained through the Naver OCR API for Layoutlmv3 model learning. (100%)
- Training and evaluating Layoutlmv3 model using the preprocessed data. (100%)
- Creating a Requirement Definition (100%)
- Lambda deployment (100%)

허찬용

- Labeling work using the collected data. (50%)
- Embedding and preprocessing of OCR data obtained through the Naver OCR API for Layoutlmv2 model learning. (100%)
- Training and evaluating Layoutlmv2 model using the preprocessed data. (100%)
- Scheduling, creating a Gantt Chart (100%)
- parsing output result for backend (100%)

Challenges and solutions

TEAM: Front-End

There were two objectives of making code style conventions. First is to prevent output overflows at different screen sizes and ratios on different devices. Second is to easily maintain code having independence between modules. Therefore, the code was structured using the following method.

When implementing the front end by setting a px value in the Padding widget, there was a problem that the widget frequently overflowed the screen due to different device environments. To solve this, we wrapped the screen components (ex. Text, TextFormField, and ElevatedButton) in a container widget. Then we adjusted the alignment by aligning container widget by using Column and Row to position them properly. After alignment, the Expanded widget is used to distribute the amount of space each element occupies to ensure that the layout is consistent with other devices.

TEAM: Back-End

TEAM: ML

As a result of using not only Naver OCR, but also tesseract and cloud vision api, it was found that the accuracy of receipts in Korean was significantly lower than that of receipts in English. Among those, we chose NAVER OCR, which has the highest accuracy in Korean.

Comparing LayoutLM v2 and v3, it was more convenient to use v2 for preprocessing. However, v3 is lighter and performs with better accuracy.

So, when comparing OUTPUT like the “expected result” pictures, we chose the Layoutlmv3 model that showed higher performance and efficiency as the project model.

During the preprocessing process, there were cases that the position of the text after OCR became less than 0. Therefore, we added exception handling for the case.

When data is extracted through a receipt image during the OCR process, noise is generated between images. For example, even with the same receipt, the image data varies slightly depending on the angle or light reflection. In addition, if we proceed with OCR, the position of the text will be changed. Due to this problem, when extracting text, there was a problem of classifying it into the same category even if it was not the same category of words. For example, in a receipt image, the word "buyer" should be recognized as a single text and the coordinate position should be returned, but the word ":", such as "buyer:", was included in the same range. So, we removed this noise from the text that contained the

travel letters, and we applied an additional process to convert the values for the bounding box that represent the location of the text accordingly.

Future Plans

TEAM: Front-End

Home screen, addition of receipts, detailed records, and record analysis screen implementation will be implemented by 정우섭, 박찬호 (was in charge of 박현찬)
Connection to Implemented Screen API

TEAM: Back-End

After completing the implementation of the Receipt domain, domain that is managing detailed analyses will be added. Then Oauth will be added at the same time as backend team proceeds with the connection with the Front-end. Except for the connection with the front-end and Oauth, the work will be completed by 5/28, and the Oauth of the backend will be carried out at the same time as the front-end work.

TEAM: ML

Implementation of prediction code: Currently, model training and data preprocessing parts have been implemented, but we will work on code to show the results using the trained model.

Code restructuring and refactoring: We are currently working with Python code, and we are going to restructure the code so that we can provide APIs for training, prediction, and preprocessing tasks according to API requests.

OCR API: NAVER OCR API is currently running 100 cases per month for free. So in small operations, it's not yet relevant, but if you need a lot of data, you'll be charged. Therefore, we are planning to implement a model that can directly work on Korean OCR.

V References

Gallup Korea. (30 Jun, 2022). *Gallup Korea*. 8 Apr, 2023, https://www.gallup.co.kr/gallupdb/reportContent.asp?seqNo=1309

박주석. (2023). 마이데이터 개념과 추진 현황. *ICT Standard Weekly 제 1117 호*, 4-6.

'22 년 연간, '22 년 12 월 주요유통업체 매출동향. (2 Feb, 2023). *산업통상자원부, 대한민국 정책브리핑*. 9 Apr, 2023, https://www.korea.kr/news/pressReleaseView.do?newsId=156550526

[디지털 시대, 오프라인의 귀환]MZ 세대 '3 감' 잡아라...구찌 버거, IWC 커피, 시몬스 수세미 손에 잡히는 경험 마케팅 열풍. (23 Apr, 2022). *서정민, 중앙선데이*. 9 Apr, 2023, https://www.joongang.co.kr/article/25065674#home

2014 가계부와 물가 관련 조사. (Aug, 2022). *트렌드모니터*. 9 Apr, 2023, https://www.trendmonitor.co.kr/tmweb/trend/allTrend/detail.do?bIdx=1239&code=0602&trendType=CKOREA

'아직도 종이영수증을 쓰십니까?' 기업에 부는 친환경 바람. (30 May, 2022). *이나영, 사례뉴스*. 9 Apr, 2023, http://www.casenews.co.kr/news/articleView.html?idxno=11329

[오맞! 이 정책] 종이 대신 '전자영수증' 받고 탄소중립실천포인트 쌓으세요!. (18 Apr, 2022). *이정운, 대한민국 정책브리핑*. 9 Apr, 2023, https://www.korea.kr/news/visualNewsView.do?newsId=148900784

Rombach, R., Blattmann, A., Lorenz, D., Esser, P., & Ommer, B. (2022). High-resolution image synthesis with latent diffusion models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition* (pp. 10684-10695).

Google Trends. (n.d.). *Google Trends*. 4 월 2 일, 2023, https://trends.google.com/trends/explore?date=today%205-y&geo=KR&q=Diffusion,Stable%20Diffusion&hl=ko

Automatic1111/stable-diffusion-webui. (n.d.). *Github*. 4 월 2 일, 2023, https://github.com/AUTOMATIC1111/stable-diffusion-webui

All You Need Is One GPU: Inference Benchmark For Stable Diffusion. (10 월 5 일, 2022). *Lambda*. 4 월 2 일, 2023, https://lambdalabs.com/blog/inference-benchmark-stable-diffusion

RTX 3080 Power Consumption Guide. (5 월 5 일, 2022). *EcoEnergyGeek*. 4 월 2 일, 2023, https://www.ecoenergygeek.com/rtx-3080-power-consumption/