하루내컷(My Daily Life Cut)

The Application integrates emotion recognition, returning the confidence across a set of emotions for each face in the imagesuch as anger, contempt, disgust, fear, happiness, neutral, sadness, and surprise.

Young Hoon, Lee
Dept. Earth Resources and Environmental Engineering
Hanyang University
Seoul, Korea
nobel6018@gmail.com

David, Jeon
Dept. Computer Science
Hanyang University
Seoul, Korea
iuwios@naver.com

Ji Won, Hur
Dept. Information System
Hanyang University
Seoul, Korea
hanante0203@gmail.com

Ji Yun, Park
Dept. Business School
Hanyang University
Seoul, Korea
yuunnii315@gmail.com

***Abstract*— This application detects one human face in an image and gets back face rectangles for where in the image the face is, along with some attributes which contain machine learning-based predictions of facial features. Algorithm in application utilizes some of sentiment data and returns meaningful analysis results.**

Keywords*— *face detection, machine learning, application diary, sentiment, appearance analysis, record

I. INTRODUCTION

Sentiment Analysis Camera application provides users 4 basic functions: (1) analyze human emotion when uploading face photo, (2) record results on the account's database with date, (3) input user's personal comments with the photo, (4) AI chat service which offers words of sympathy depending on user's mood.

II. SERVICE CONCEPT & PROPOSAL

This application focuses on an image and analyze human's emotion on it. The image, which is a photo of one's face, sometimes could be more honest and raw data offered from human. Texts can be fabricated as much as he/she wants, on the other hands, face is hardly concealed. Most people write diary to record their mood of the day. However, we suggest NEW idea to record feelings in much simple and fabrication-free way. We aim for everyone's facial expression at every moment.

In the application, main source is machine learning technology. We highly use Microsoft Azure Face API to detect facial features. 27 types of data are extracted from one-time detection. They are processed and expanded to new service.

III. DEVELOPMENT ENVIRONMENT

A. Software development platform

a) Android/IOS Platform

Ideally, the team will aim to develop an application compatible for both Android and IOS. This is to ensure the best daily usability as most people spend majority of their time with smartphones. The concept of taking a picture and recording any time the user wants is most efficient in a small device which would be a phone. This allows any user to take a picture any place or time he or she is carrying a phone which is also in the same direction that this application aims.

b) Programming Language

- Java

Java is a language that is needed to code in an IDE called Android Studio. The main execution codes are written in java such as calling a widget for use from an xml or connecting to different servers.

- Python 3.7

To use Django version 2 or higher, the recommended python version is 3.

- JavaScript

React Native needs JavaScript library for building user interfaces.

- XML

XML is needed to design the user interface in Android Studio applications. Font, text, widgets, application format is all designed in the XML.

c) Cost Estimation

The team has estimated the monthly cost for a target user of 1,000 and an average of 3 images taken per user daily. The total cost is 164.2USD per month. There is no extra cost for software or hardware.

- Microsoft Azure Cost

The team is using a 12 month-free version of Microsoft Azure with 20 transactions per minute and 30,000 free transactions per month. However, for long term maintenance the Standard version is required which taking into consideration that one person takes about 3 images a day and 1,000 users as a target. The price for 1,000 transaction is 0.8$ which would make monthly cost (3images*1000users*30days)*(0.8USD/1000transactions) = 72USD.

- Cloudinary Cost

Cloudinary offers one of the best prices for saving media data. For 1,000 users taking an average of 3 images a day takes 3images * 1000 users* 30days = 90,000 images in one month. Depending on the image quality, the data size may differ but the average for 90,000 images is set to 60 gigabytes which costs 89USD monthly.

d) Framework

- Django 2.1.2
- Django Rest Framework 3.10.2
- React Native 0.59.8
- React ^16.8.3
- Expo ^35.0.0
- Android Studio 5.1

e) API

- Microsoft Azure Face 1.24.0
- Cloudinary 1.20.0

f) Repository

- Github

B. Software in use

a) Daylio

It helps users to establish more productive lifestyles and to better understanding their own habits by keeping track of their moods while writing a diary. Its major features are as follows:

i. The most prominent feature of this is that if users record several emotions in a day, their average emotion sets as a representative value.

ii. Although it sets 'Very good', 'Good', 'Normal', 'Bad' and 'Very bad' to default values, users can change or add emotions they want.

iii. In [Statistics] tab, users can check 'Change of moods over the month', 'Consecutively recorded days', 'The number of moods per day' and 'Statistics of moods by day'.

iv. It sets 'Very good' to 'Very bad' from 5 to 1 and by colors, it is easy to see the emotion changes by date on the line graph.

v. In [Calendar] tab, if the user has recorded n emotions, each emotion may be displayed by one-nth at that date on the calendar and can also be viewed as an average of the emotions.

Daylio

b) Mind i/o (마인드 i/o)

It is an integrated platform for managing the health of the mind. Its major features are as follows:

i. It has a data-based machine learning chatbot that learns the words users use, which is called 'io chatbot' learning minds.

ii. It provides various meditation methods and healing music from the basic course.

iii. It allows users to record their feelings in writing with photos using 'a mind diary'. If the diary is public, it can be used as SNS. When the diary is private, it can be used as a personal diary.

iv. It allows users to communicate with other users who have similar interests or pains.

Mind i/o

c) Emotional Diary (감정일기)

It is an application that tells you how you felt at that time by recording users' emotions at the end of each day. Its major features are as follows:

i. It allows users to write 'an emotional diary'. This includes an emotion, the intensity of the emotion and contents. Users can add locations and photos to their diary. Based on the number of diaries stored in a database, it outputs a user-only emotional charts.

ii. It has 'emotional folders' where users can check their diaries stored in a database by emotions. Regardless of emotions, users can view the entire diaries they wrote previously.

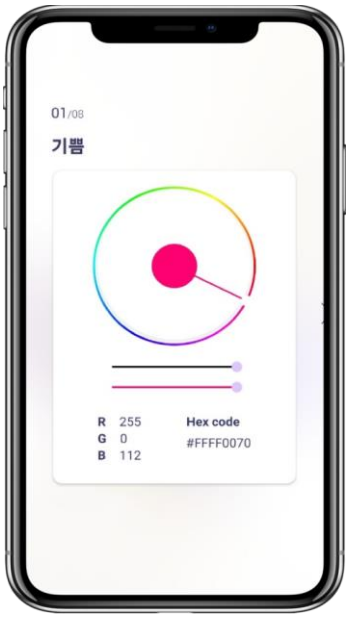
iii. It provides a recording service where users can record their own feelings by talking to their cellphones.

iv. It marks users' current locations and ones where users were writing a diary at that moment. This service is based on Google-Map.

Emotional Diary

d) Bora - Look at your feelings (당신의 감정을 보라)

It is the simplest one among emotional analyzing applications. It only shows the monthly representative emotion by colors on the calendar. The major feature is that user can set the desired color out of eight emotions: joy, happiness, excitement, peace, sadness, anxiety, anger, and (custom value). After this setting being completed, the colors that a user set are displayed in a palette format.



Bora

C. *Development task distribution*

Idea was from Software Engineering Project group in Hanyang University. The group is composed of Y.H. Lee, David Jeon, J.W. Hur, and J.Y. Park. Every team member participates in planning, designing, and developing. Also, we manage overall development process in Github public repository. Details in *XI. ROLE ALLOCATION*.

IV. SPECIFICATIONS

This part is a collection of requirements for 하루내컷, a sentiment analysis camera application. The 하루내컷 requirements document is based on the VOLERE Template. Copyright © 1995-2004 The Atlantic Systems Guild Limited.

The term ‘requirements’ is used according to the following definition.

Requirement:
something a system must do or a property a system must have. A requirement shall always be testable by just testing the system.

1. Login

1.1 The purpose of the Login page

This is the screen that user face for the first time when running the application. An introduction to the app appears on the screen, allowing the user to briefly know what the app is about. Also, to sign up, a user can subscribe by linking his or her Google account or creating a new account within the application. This allows users to store recorded emotions and photos in their own database, so that data can be preserved if the mobile phone is changed or reinstalled after deleting the application.

1.2 Function details

1.2.1 Link to user’s Google account
ID: 4101
Type: Requirement

This application integrates Google Login with itself. When linked to a Google Account, the user can create a nickname, password, and password checkboxes for use in the application and select the sex corresponding to him or her.

Process

- Click the 'Login with Google Account' button.
- Select the account that he or she wants to use in the application when having multiple Google accounts.
- Complete the nickname, password, and password checkboxes for the application to use and selects the appropriate sex.

1.2.2 Creating a new account
ID: 4102
Type: Requirement

This application can authenticate users with an email address and password. Unlike 1)'s *'Link to user’s Google account'*, users need to set up separate e-mails to be used in the application before being authenticated, and then create an account after filling in the nickname, password and password check box, just like *‘Link to user’s Google account’.*

Process

- Click the *'Creating a new account'* button.
- Complete the email, password, and password check boxes for the application to use.
- Type in the nickname, date of birth, and select the sex.
- Click the OK or Cancel button to complete the membership.

Issues

- Database administration: sign-in, sign-up

2. Main(home)

2.1 The purpose of the Main page

This section is the center of the application which is the first screen to be seen after a user creates an account or log in to the application. Robot with the speech bubble shows up on the screen. Text prints out words that suit a user's mood and shows various emotional variation graphs daily, weekly, or monthly. Since this application is to record each user's daily, weekly, and monthly emotional changes, it is important to show a user-centered screen rather than a community-centered one on the first screen page.

2.2 Function details

2.2.1 AI robot with speech bubble

ID: 4201

Type: Requirement

The robot with speech bubble is a service that allows users to share emotional interactions by showing messages that suit the user's chosen mood. Therefore, it sympathizes with and encourages the user's feelings.

Process

- A user clicks on the Select box at the top of the speech bubble.
- A list of eight emotions pops up on the screen.
- A user chooses the feelings that he feels on that day.
- A sentence or a word suitable for the emotion appears in the speech balloon. At this time, the writing can be one of the contents of a diary previously written by the user, or it can use a famous phrase brought from Naver famous phrase book API.
- When the Refresh button at the bottom of the speech balloon is being pressed, different words will appear on the screen each time.

Issues

- Message database management: default message data gets from external api and user-created message data will be additionally stored on the top of database.
- User-created message input: text type, category list, maximum 100 characteristics.

2.2.2 Mind Control Graph

This section shows a total of three graphs by obtaining statistics based on the user's photographic and written feelings. The first graph is radial which shows how often a user feels the emotions over a week or a month. The second is an emotional trend graph that identifies the number of emotions or the intensity change of emotions depending on how they are developed. The third is the total emotional trend line, which allows a user to see the eight emotions changing at once.

2.2.2.1 Radial graph

ID: 4202

Type: Requirement

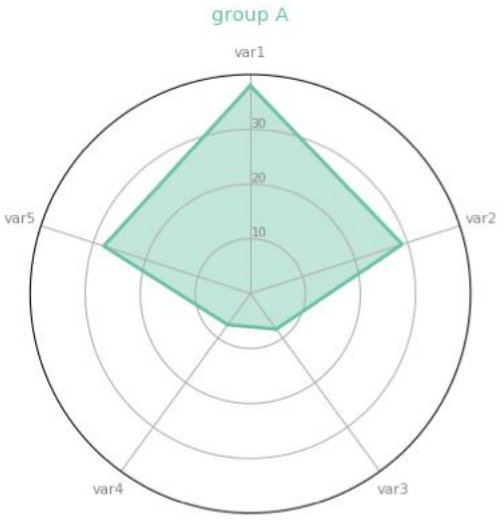
Eight emotions are grouped into five sections, allowing users to see how often the group of emotions appeared during the week or month. The eight emotions are 'anger', 'contempt', 'disgust', 'fear', 'happiness', 'normal', 'sadness' and 'surprise'. In this application, 'anger' is classified as the first group, 'contempt' and 'disgust' as the second group, 'fear' as the third group, 'sadness' as the fourth group, 'happiness' as the fifth group, and 'surprise' as the sixth group.

Process

- On the first screen, users can check the frequency of their emotions over a week in a radial graph format.
- If the user clicks the right arrow at the top of the graph, it shows the feelings that a user has felt over a month.

Issues

- Imbalanced data quantity: utilization rate of rare emotions such as ‘disgust’, ‘surprise’ is much less.
- Radial graph comparison standard(month average)



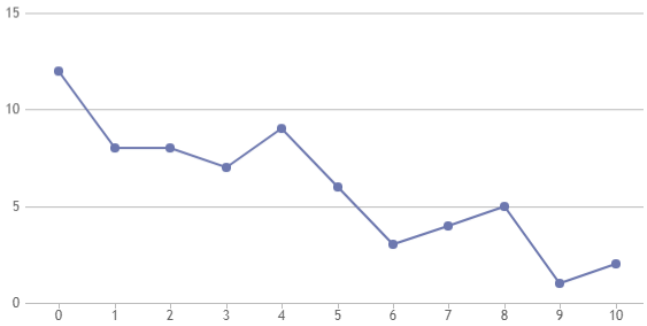
Radial graph type

2.2.2.2 Trend line graph by emotion
ID: 4203
Type: Requirement

Of the total eight emotions in the Select box, a graph is displayed that corresponds only to the feeling selected by the user. The frequency of each emotion is stored in the application's database user by user, and the stored value is shown in the graph. When developed by date, the x-axis of the graph displays the date and day, and the y-axis displays the intensity of the emotion from 0 to 100%. The default value of the intensity of the emotion is zero, and it is displayed on the graph if no emotion has been felt on a day.
When developed by week, the x-axis of the graph displays the corresponding month and week, and the y-axis is the frequency of the emotion, with a value of 0 to 7.

Process

- A user selects one emotion from the Select box.
- A trend line graph of the selected emotion is displayed at the bottom of the Select box.



Trendline graph type

2.2.2.3 Total emotional trend line graph
ID: 4204
Type: Requirement

All eight emotions are displayed on the graph at once, so a user can see the changes in all emotions. Like the trend line graph of (2), the values of the x- and y-axes vary depending on whether to develop by the date or by the week. In the former case, the x-axis displays the date and day of the week, and the y-axis displays the intensity of the emotion. However, in the latter case, the x-axis displays the corresponding month and the week, and the y-axis displays the frequency of the emotions. Each emotional graph line is differentiated by colors.

Process

- All eight emotions are displayed on the graph at once.
- Users can swipe graphs left and right with their fingers.
- If a user clicks on the emotions shown on the left side of the y-axis, the matching emotion graph line will appear highlighted.

3. Camera

3.1 The purpose of the Camera page

This section is to provide camera activation and data analysis support. The following steps are rough user experience. Taking a photo, saving photo, and user can get final analysis results in the user interface. In the backend system, face detection, face attribute recognition, parsing data are happened.

Moreover, user can input text which is about the photo or daily feeling. Each data is independently saved and managed in the user account's database. User can read any records (photo and user-creating text) of exact date anytime.

3.2 Function details

3.2.1 Photo taking and saving
ID: 4301
Type: Requirement

The application shall activate mobile built-in camera application when accessed by user. Both front-side camera and back-side camera are possible. Photo size is 3:4 or 9:16 or 1:1. Any case to take a photo is possible when it returns a photo of user face. Also, user can import another photo from mobile album. Video can't be used.

- taking a photo service
- saving a photo in a designated album service
- importing a photo from album service
- an album linking service

3.2.2 Image detection
ID: 4302
Type: Requirement

The application returns face expression analysis results when a photo is entered which is an image data. Before analyzing, the system detects one of more human faces in the image and gets back face rectangles for where in the image the face is. The red-colored rectangle

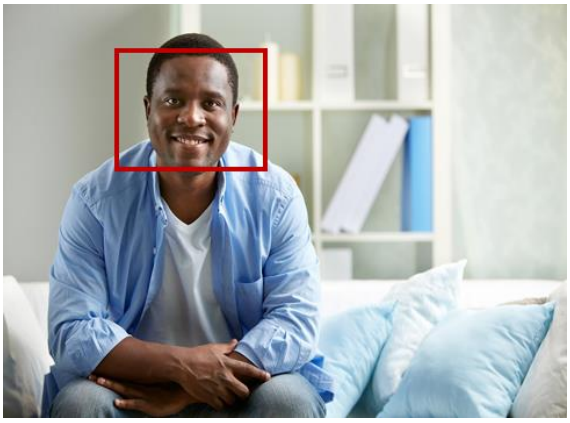
represents that it has success in finding one human face.

- an image loading service
- a face detection service

3.2.3 Image analysis
ID: 43021
Type: Requirement

The application detects and recognizes several attributes in an image. The face attribute features available are: age, emotion, gender, pose, smile, and facial hair. It doesn't show all to user. The data processing occurs in parsing and partial extraction. And here are more details about meaningful data list.

- hair: bald / color (brown, black, red, blond, gray, other)
- face hair: moustache, beard, sideburns
- smile
- head pose: pitch, roll, yaw
- gender
- age
- make-up: eye make-up, lip make-up
- emotion: anger, contempt, disgust, fear, happiness, neutral, sadness, surprise
- occlusion: forehead, eye, mouth
- accessories: glasses



3.2.4 Sentiment prediction
ID: 4303
Type: Requirement

After detection, the application returns face expression analysis results. It is along with some attributes of facial features as previously stated. They are machine learning-based predictions. The algorithm extracts 27 types of data for each face in the image. And it makes a meaningful analysis results based on application system needs.

- a sentiment prediction service
- a 27 types of facial features extraction service
- a data grouping service: emotions, appearance, etc.

3.2.5 Comment suggestion
ID: 4304
Type: Requirement

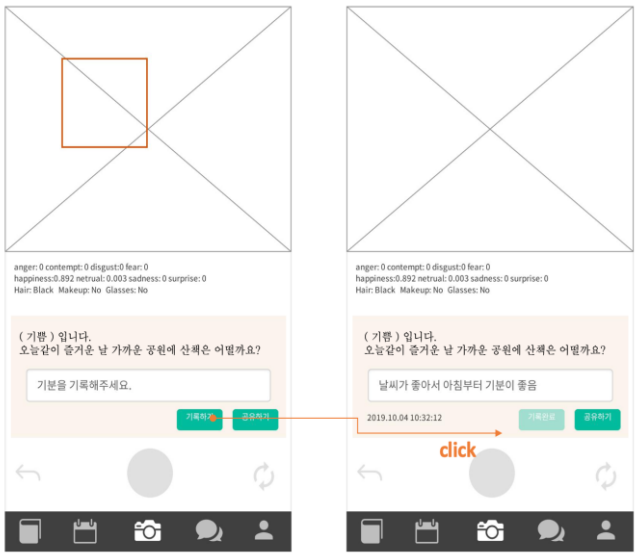
The application automatically suggests some comments according to user's mood. They are represented with analysis results at the same time. For example, when the application detects user's face and concludes 'happiness', the comment like 'why don't you go to park and have a coffee to fully enjoy your happiness?!' appears. Database has several text data and they are set by developer in advance.

- a touching comment providing service

3.2.6 Data record
ID: 4305
Type: Requirement

The application provides not only emotion analysis service but also other additional services utilizing the photo. Most of all, the application aims for user's free expression at every moment. In this reason, the application supports data record service as well. Data includes photo, sentiment analysis result, and text(comment).

- a user comment record service
- an account-utilization service: every data store in user's own database
- a data saving service: photo, text, date, sentiment analysis result



pre-wireframe

4. Archive

4.1 The purpose of the Archive page

This section is to provide user's history. Users can review her or his emotion analyzed by AI backend system. The history contains not only the picture but also emotion analysis, her/his comments.

Users can see their history after taking a photo. User can see the history since every photo, emotion analysis and comment are saved in database. Cloudinary saves photos. It also saves comments and analyzed emotions in another database.

4.2 Function details

4.2.1 API call with authentication

ID: 4401

Type: Requirement

The archive needs user authentication. Photo and emotion is important private information. The application takes user's unique id through authentication process. Whenever user call data, retrieve photos, comments and emotion list in this section, authentication is required.

Process

- User should call API with header which has "Authorization".
- Backend service requires the header's authorization parameter.
- If authoritative value is correct and valid, backend responses with status 200 with contents
- If authorization value is expired or invalid, backend responses with status 403 error (Authorization Forbidden)

4.2.2 Call Cloudinary API

ID: 4402

Type: Requirement

User photos is necessary component to archive. Photos are saved in cloudinary. Cloudinary provides many APIs, from CRUD (create, retrieve, update, delete) to resizing, rotating and so on(for more, refer 4.2.3). In this section, retrieve images API is used. The application calls retrieve API with the header which has authorization key-value pair.

Process

- The application makes retrieve image API call. Authorization is essential.
- Backend service check authorization value and then bring user images back if authorization is verified.
- The application takes and displays the images.

4.2.2 Display archive history list

ID: 4403

Type: Requirement

Each history of archive consists of picture, representative emotion and shortened comments. The application asks all the images from cloudinary API, all emotions and comments from the database. And then reformat those things and make archive history list.

4.2.3 Display history detail view

ID: 4404

Type: Requirement

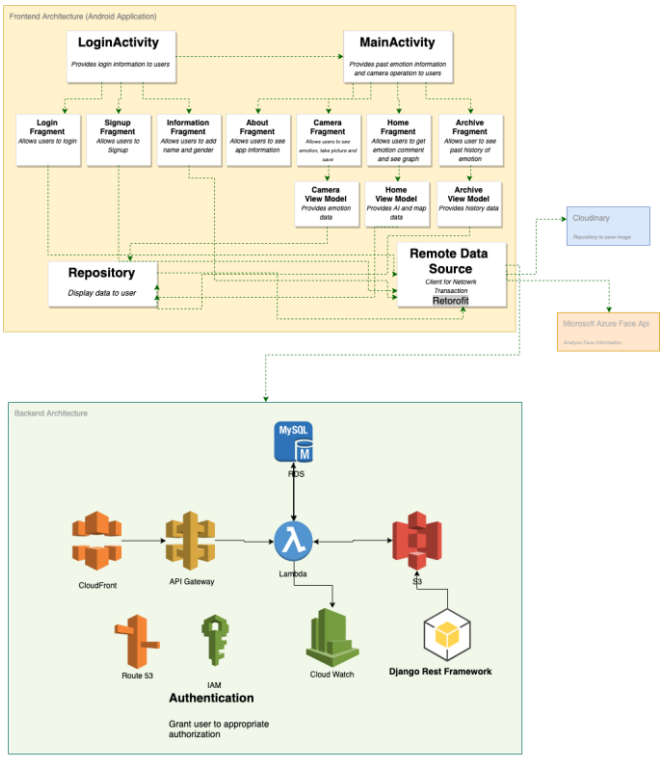
Each history detail has user photo, emotion and comments. Detail archive history can present more information. Detail emotion statistics and full comments.

The application requests a specific photo from cloudinary API and equivalent emotion and comment from database.

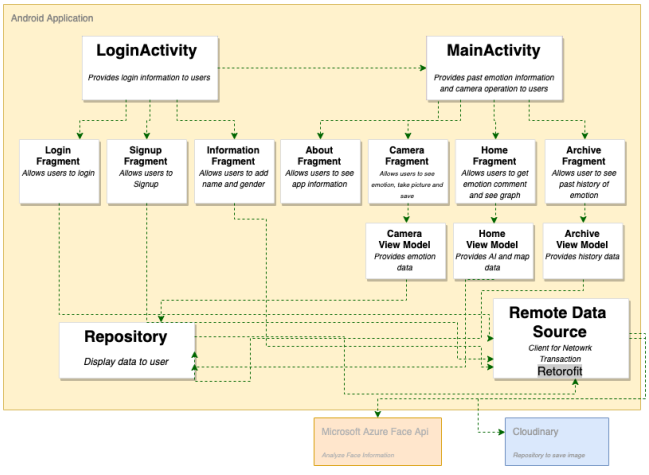
V. ARCHITECTURE DESIGN & IMPLEMENTATION

A. Overall Architecture

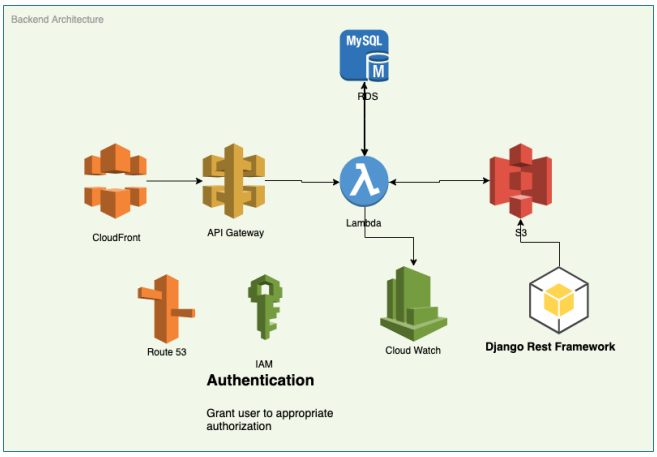
- Overall Architecture



- Frontend Architecture Component diagram for Android Application



- Backend Architecture



B. Directory Organization

5.1 Frontend

The front-end development of 하루내컷 is based on Android Studio. Directory organization is composed of directory, file name and package. Top directory is LifeCut and several directories are separated according to page or function. There are app, objclass, json, drawable, layout, font, menu, mipmap, values, manifests as subordinate directories. We also attach some image icons which are being used at application front side.

Directory	File Names	Package
/LifeCut/app/src/main/java/com/example/lifecut/app	LoginActivity.java MainActivity.java LoginFragment.java SignUpFragment.java InfoFragment.java AboutFragment.java HomeFragment.java CameraFragment.java ArchiveFragment.java	Package 1
/LifeCut/app/src/main/java/com/example/lifecut/objclass	FaceInfo.java ImageFace.java User.java UserPass.java InfoCard.java PostEmotion.java	Package 2
/LifeCut/app/src/main/java/c	JsonPlaceHolder.java	Package 3

om/example/lifecut/json		
/LifeCut/app/src/main/res/drawable/	ic_about.xml ic_camera.xml ic_archive.xml ic_home.xml robots.png	Package 4
/LifeCut/app/src/main/res/layout/	activity_login.xml activity_main.xml fragment_about.xml fragment_home.xml fragment_camer.xml fragment_archive.xml fragment_login.xml fragment_signup.xml fragment_information.xml	Package 5
/LifeCut/app/src/main/res/font/	montserrat.xml montserrat_bold.xml montserrat_regular.xml montserrat_medium.xml	Package 6
/LifeCut/app/src/main/res/menu/	mnu_main.xml mnu_navigation.xml	Package 7
/LifeCut/app/src/main/res/mipmap/	ic_launcher.png	Package 8
/LifeCut/app/src/main/res/values/	attr.xml colors.xml config.xml dimens.xml strings.xml styles.xml swatches.xml ui_confil.xml	Package 9
/LifeCut/app/src/main/manifests/	AndroidManifest.xml	Package 10
/LifeCut/app/	build.gradle	Package 11

5.1.1 Package1
The Package is in charge of putting all the information from created object classes, xml design files, permission files together – then displays it for usability to the users.

a) LoginFragment.java
To get user's username and password, then sends the information to Login Activity through an interface. User can also move to SignupFragment.

b) SignupFragment.java
To get user's username, password and email address, then sends the information to Login Activity through FragmentLogListener. When LoginActivity sends response, the fragment is moved to InformationFragment.

c) InformationFragment.java
To get user's id and gender, then sends the information to LoginActivity through InfromationListener.

d) LoginActivity.java
Login information from SignupFragment is sent to Django through url-address. The caller then gets a response from Django in token form if the username and password exist. Then through intent the activity switches to MainActivity. From SignupFragment, the information is sent to Django the same way and a new user is then created. The MainActivity then sends a request back to go on to InformationFragment. When information comes from Information Fragment, then the data are sent to Django to be saved and through intent the activity switches to MainActivity.

e) HomeFragment.java
Float information about sentiments such as anger, contempt, happiness, surprise, neutral, sadness is received from MainActivity and implemented in linear graph. Comment information from the user is also received and added to the word cloud in the robot. When the plus sign is pressed, the string comment information inside the robot is sent to the MainActivity.

f) AboutFragment.java
To display information about application. Image file is displayed through an ImageView.

g) Camera Fragment.java
When picture is taken, the bitmap of the image is sent to MainActivity through a Listener. The response from MainActivity is then analyzed through getEmotion and drawRectangleOnBitmap method and displayed in ImageView. The text response data is displayed in the Emotion TextView. The data from the previous two methods are then sent back to MainActivity.

h) Archive Fragment
Date, emotion, and comment information are received from MainActivity. Each set of the user's information are displayed through an adapter in ImageView.

i) MainActivity.java
Bitmap data received from FragmentCamListener is sent to MS Azure Server and the response data is sent back to the CameraFragment. The data received from FragmentCamListenerSave is sent to Django with the current user token in POST. MainActivity also sends emotion and date data received from Django in Json format to HomeFragment and ArchiveFragment through a Listener.

5.1.2 Package 2
Object classes for constructors, getters, setters are saved here. They are used to save emotion, date, bitmap, url informatins.

a) FaceInfo.java
Object class with bitmap image, String comment, String date, float anger, float contempt, float, disgust, float fear, float happiness, float surprise, float sadness, float surprise constructors and getter.

b) ImageFace.java
Object class with MS Azure Face[] face, Bitmap image constructor and getter.

c) User.java
Object class with String token constructor and getter.

d) UserPass.java
Object class with String username, String password constructor, getter and setter.

5.1.3 Package 3
To hold interface to access JSON files.

a) JsonPlaceHolder.java
To hold all relative url values to access JSON file.

5.1.4 Package 4
Drawable folder holds icon information in xml, png and jpg format.

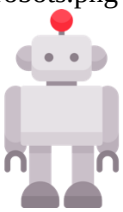
a) ic_about.xml

b) ic_home.xml

c) ic_camera.xml

d) ic_archive.xml

e) robots.png



5.1.5 Package 5
To contain all the designs for the Fragment and Activity pages.

a) activity_login.xml
To hold a FrameLayout for the interface to be switched through fragments.

b) activity_main.xml
There is a header, constraint layout with FrameLayout and menu-bar below nested in Linear Layout.

c) fragment_camera.xml
To holds an ImageView, constraint-layout with EditText, two TextView and two Buttons nested inside a LinearLayout.

d) fragment_home.xml
To hold robots.png and GraphView inside LinearLayout.

e) archive_fragment.xml
To hold an ImageView inside ScrolViewLayout.

f) about_fragment.xml
To hold an ImageView inside Linear Layout.

g) login_fragment.xml
To hold three Buttons, three TextView and tow EditText inside LinearLayout..

h) signup_fragment.xml
To hold three one button, two TextView, and two EditText inside LinearLayout..

i) information_fragment.xml
To hold two TextView and two EditText inside LinearLayout.

5.1.6 Package 6
To hold font information not given in Android Studio. There is only one font type named "montserrat" in different styles such as bold, italic, normal.

5.1.7 Package 7
To hold navigation menu which has information for the default layout in MainActivity.

a) mnu_main.xml
To hold information for the header of the MainApplication.

b) menu_navigation.xml
To holds 4 sections for navigation bar divided by archive, home, about and camera.

5.1.8 Package 8
To hold a launcher image of application

5.1.9 Package 9
To hold color, attribute and configure information for use in design.

5.1.10 Package 10
To contain internet permission request and Activity permission requests.

5.1.11 Package 11
To hold gradle information. The version of API's such as cloudinary and MS Azure Face, dependencies, java version, implementations are stored.

5.2 Backend Server
We used "Django and Django Rest Framework" to develop backend. Backend is restful as well. It consists of 'app' and 'core' app. The 'app' considers global setting. The 'core' app covers the main functions – models, views, urls and serializers.

Directory	File Names	Package
/	.gitignore manage.py requirements.txt	root
/app	__init__.py settings.py urls.py wsgi.py	app
/core	__init__.py admin.py apps.py models.py tests.py	core
/core/api	__init__.py serializers.py urls.py views.py	core.api

5.2.1 Root Package
The root has default files. It covers django default wrapper, versioning and git settings.

a) manage.py
The manage.py is a thin wrapper around django-madin.py that takes care of two things for you before delegating to django-admin.py. It puts your project's package on sys.path. And it sets the DJANGO_SETTINGS_MODULE environment variable so that it points to your project's settings.py file

b) .gitignore
.gitignore tells git which files (or patterns) it should ignore. It's usually used to avoid committing transient files from your working directory that aren't useful to other collaborators, such as compilation products, temporary files IDEs(pycharm, visual studio code) create, etc. My team used gitignore.io which provides a predefined set of gitignore.

c) requirements.txt
Requirements files are files containing a list of items to be installed using pip install like so: pip install -r

requirements. txt. Details on the format of the files are here: Requirements File Format. Logically, a Requirements file is just a list of pip install arguments placed in a file. This file could easily be made: pip freeze > requirements.txt

5.2.2 App Package
App package deals with overall current django proejct setting.

a) settings.py
settings.py has project global settings. It specify project base directory, and that it is debug mode or not, allowed_hosts, installed_apps, database and third party default setting, etc. It cointains all global setting. It could be called and used by other files as well.

b) urls.py
urls.py is url blueprint. It combines urls and app's urls

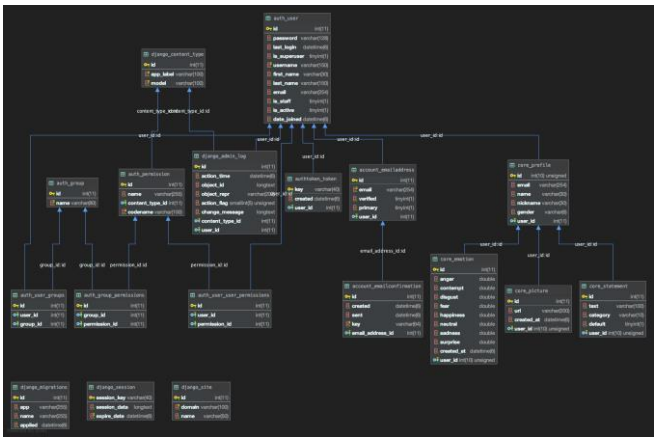
c) wsgi.py
WSGI is Web Server Gateway Interface. It is a specification that describes how a web server communicates with web applications, and how web applications can be chained together to process one request. It is used to forward requests from a web server (such as Apache or NGINX) to a backend Python web application or framework. From there, responses are then passed back to the webserver to reply to the requestor. wsgi.py exposes the WSGI callable as a module-level variable.

5.2.3 Core Package
This is general setting for core.api. Admin utility and app name and defining models can be done in this package.

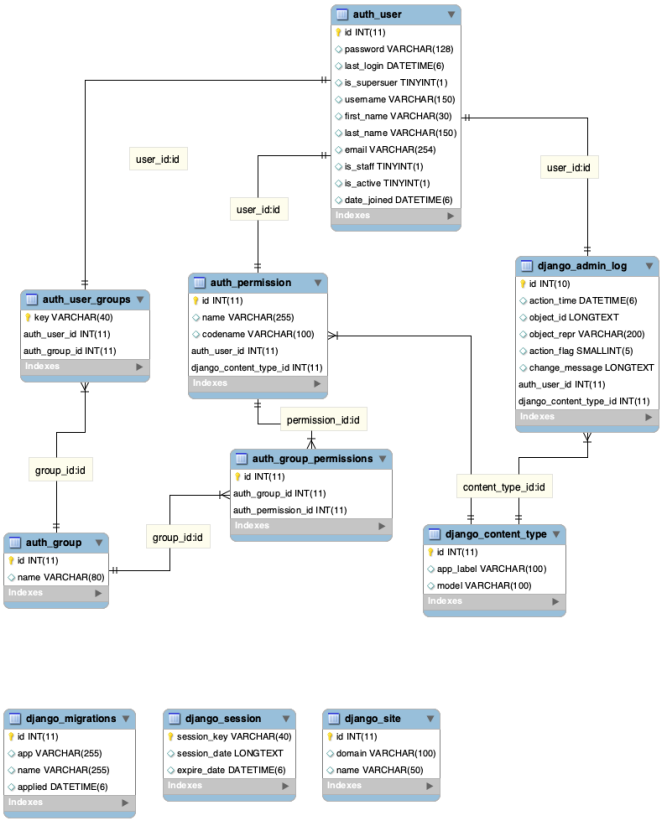
a) admin.py
This is where models can be registered. The registered model can be managed in GUI interface. The row(s) could be read, created, updated and deleted. In this way, the developer can easily build admin page.

b) apps.py
This apps.py specify app name in singular and verbose name. The speicfied name is called at app.settings.py. This allows you to make use of AppConfig features without requiring your users to update their INSTALLED_APPS setting. Besides this use case, it's best to avoid using default_app_config and instead specify the app config class in INSTALLED_APPS as described next.

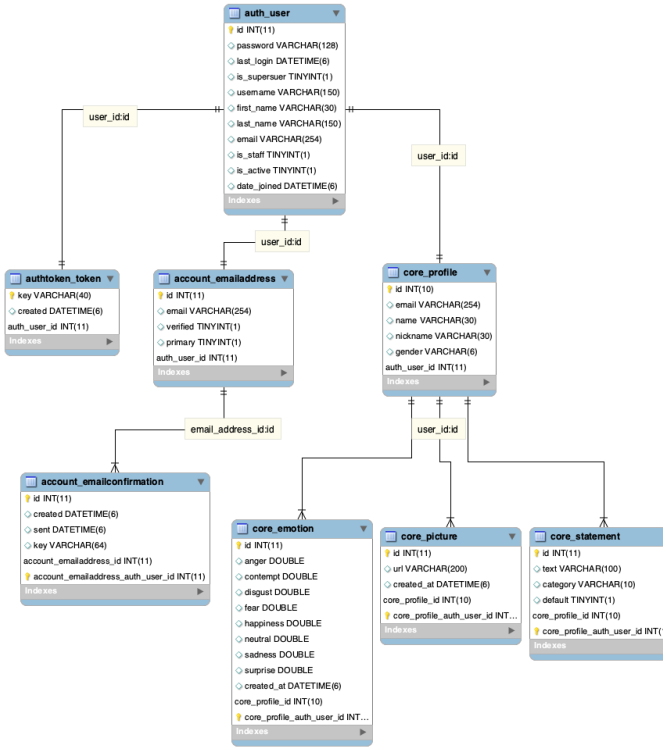
c) models.py
A model is the single, definitive source of information about your data. It contains the essential fields and behaviors of the data you're storing. Generally, each model maps to a single database table. Each model is a Python class that subclasses django.db.models.Model. Each attribute of the model represents a database field. With all of this, Django gives you an automatically-generated database-access API; see Making queries. We have user, profile, emotion, picture, statement and diary models. The ERD (Entity Relationship Diagram) is attached below.



Backend Overall ERD



Left part of overall ERD



Right part of overall ERD

d) tests.py
The tests.py is where django model test and api test codes are written. Test codes are highly recommended because of the project stability and extensibility and expandability. The bigger project size the harder to find out bugs.

5.2.3 core.api Package
a) serializers.py
Serializers allow complex data such as querysets and model instances to be converted to native Python datatypes that can then be easily rendered into JSON, XML or other content types. Serializers also provide deserialization, allowing parsed data to be converted back into complex types, after first validating the incoming data. Django Rest Framework provides many serializer class as well as, ModelSerializer class which provides a useful shortcut for creating serializers that deal with model instances and querysets.

b) urls.py
This urls.py define the app's urls. The global blueprint url is defined in app.urls.py. The actual and detail api urls are attached.

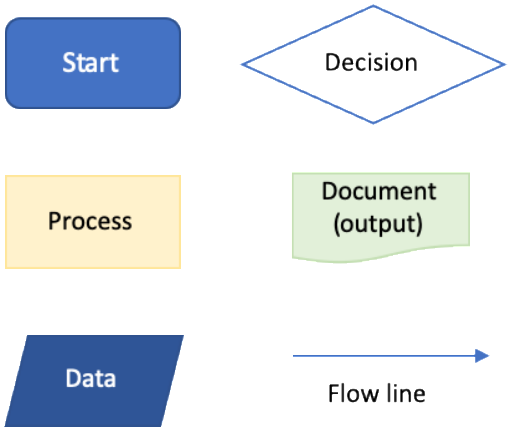
c) views.py

When client sends request to specific url. Django seek out the matching view. The view is actual working logic. It calls serailizer and response in correct form for example json, xml, etc. Django Rest Framework allows you to combine the logic for a set of related views in a single class, called a ViewSet. A ViewSet class is simply a type of class-based View. The method handlers for a ViewSet are only bound to the corresponding actions at the point of finalizing the view, using the .as_view() method.

C. Module 1 or Software Component 1 or Class Name 1
...blank

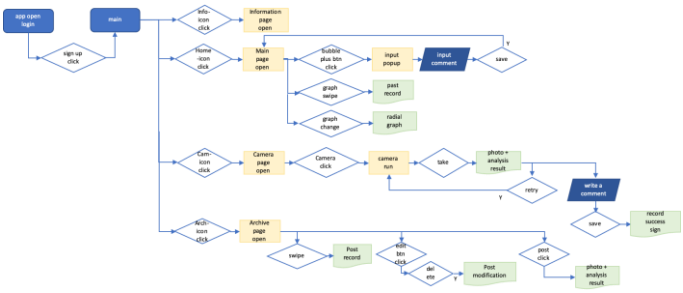
VI. USE CASES

According to the application in real, we made user experience. In visualization, service-flow-chart shows all process that user go through the application. There are 4 parts in UX and they are menu in UI at the same time: info, home, camera, archive.
The service flow chart is written by the underneath flow chart tutorial.



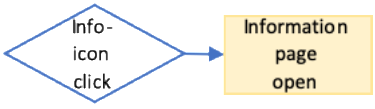
Flow chart tutorial

A. Service Flow Chart

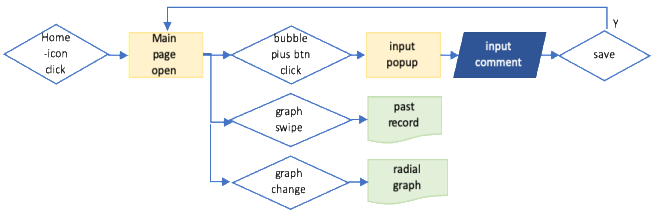


total service flow

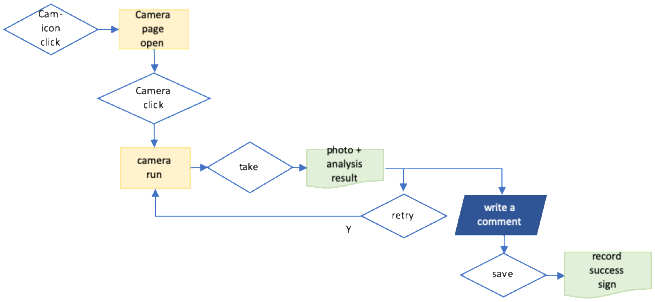
a) information page



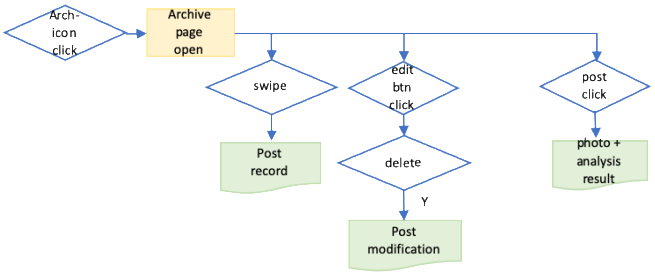
b) home page



c) camera page



d) archive page



B. Service Scenario

Name	Description
Use case	The name of use case
Brief description	A brief description of the role and purpose of the use case
Pre-condition	A textual description that defines a constraint on the application when the use case may start
Post-condition	A textual description that defines a constraint on the application when the use case has terminated
Basic path	A textual description of what the application does in regard to the use case. The description is understandable by the customer rather than the description about how specific problems are solved by the application
Exceptional path	A list of location within the basic path of the use case at which additional behavior can be inserted using the extended relationship

a) information page

Use case	Draw process
Brief description	A basic information about the application which includes application title, service usage explanation, service purpose, and short description.
Pre-condition	- Login
Post-condition	-
Basic path	-
Exceptional path	-

b) home page

Use case	Draw process
Brief description	2 types of service are on the home page which are robot with speech bubble and emotional record graph. It implicates emotion data and visualize data records.
Pre-condition	- Emotion data input - Text type phrase data input - Emotion analysis record database connection - User phrase record database connection

Post-condition	- Text phrase output to speech bubble - Trend line graph output - Radial graph output
Basic path	1. This use case starts after login access 2. In the speech bubble, a phrase is printed out that corresponds a user's mood from data 3. In the graph, various emotional variation shows by daily, weekly, monthly. 4. 2 types of graph convert 5. Input text phrase
Exceptional path	At step 2 of the basic path, daily data doesn't exist. At step 3 of the basic path, swiping action doesn't work

c) camera page

Use case	Draw process
Brief description	Camera activation and data analysis support. Input text box for user comment about result such as daily feeling.
Pre-condition	- Image data input - Text data input - Camera activation - Album connection - Analysis record database connection - MS Azure face api response
Post-condition	- Image output - Analysis with comment save in database
Basic path	1. This use case starts with device built-in camera activation 2. Image output 3. Emotion analysis result output 4. Input text comment 5. Save all data
Exceptional path	At step 1 of the basis path, camera activation fails At step 2 of the basis path, image citation fails At step 3 of the basis path, analysis result citation fails At step 5 of the basis path, saving fails

d) archive page

Use case	Draw process
Brief description	Provide user's history. Review emotion result which is analyzed by AI backend system. The history contains images, emotion analysis, text data.
Pre-condition	- Analysis record database connection
Post-condition	- Image output - Emotion analysis output - Text output
Basic path	1. A list of data record is in grid 2. Box type grid 3. Image, analysis result, comment on the box
Exceptional path	At step 1 of the basis path, data citation fails At step 2 of the basis path, disordering

e) sing-in, sign-up page

Use case	Draw process
Brief description	Link user's Google account or create a new account for the application
Pre-condition	- User data input: name, email, password
Post-condition	- User extra data input: nickname, gender

Basic path	1. This use case starts with application access. 2. Application display input box 3. If authentication passes, main page appears
Exceptional path	At step 3 of the basic path, authentication trial fails.

C. Wireframe

...blank

VII. EXPECTANCY EFFECTS

A. Application Distribution

- Amazon RDS for MySQL

B. Customer Values

a) Qualitative Values

…

b) Quantitative Values

…

C. Market Research

…

VIII. ROLE ALLOCATION

1	Roles	Name	Task description
	User	J.W.	QA, market searching
	Customer	-	-
	SW developer 1	David	programming
	SW developer 2	Y.H.	programming
	Development manager	J.Y.	planning, designing
2	Roles	Name	Task description
	User	David J.W. J.Y. Y.H.	user-side experience
	Customer	-	-
	SW developer	David	programming
	Development manager	J.W. J.Y. Y.H.	planning, designing
3	Roles	Name	Task description
	User	J.Y. J.W.	QA user interface user experience
	Customer	-	-
	SW developer	Y.H.	programming(BE)
	Development manager	David	programming(FE)
4	Roles	Name	Task description
	User		
	Customer		
	SW developer		
	Development manager		

**Every role is variable according to situation*