Look IT

Face Classification System

160538L

Sabesan S.

Contents

Definitions, Acronyms, and Abbreviations

Emotion API – The API which gives results by getting real time pictures from users as input and giving probability of an emotion to the given input as output

User – Person who interacts with the system

JIT – Just in Time

SOA – Service Oriented Architecture

RUP – Rational Unified Process

MVC – Model View Controller

IDE - Integrated Development Environment

GUI - Graphical User interface

Abstract

Look it is an Android based application which is to be used to classify and find the emotion of a user. Now-a-days, emotional management is a growing skill in industry. This is a big task and there are few persons can identify the other person's emotions. The main vision of the system is to detect the face and classify the emotions. Since the emotional detector find manually, it may various from the original emotion, it will not give correct emotion. If we implement this system, mostly the results will be correct and accurate.

This system will have a database and hosted in the application. So, the data will be stored in many places and can be managed in every system. The system is developed by using Android Studio and MySQL as a database system to manage and store the data. This system has 3 levels of users with different priorities. In this application, admin is responsible for update all database. This system allows to send notifications to the user. Admin is the main authority who can do addition, deletion, and modification if required. Each can separately view the results generated by the system.

1 Introduction

1.1 Background of the application problem

Look It is Android based application with a database system which will automate the face detection and emotional analysis. Since it is an Android based application, it will allow user to take image from the mobile and update their image. The database will keep the record of all the negative emotions and YouTube videos link according to every negative emotion. The validity of the results will be checked by the Microsoft Azure and the accuracy of the result will be depend on the Microsoft Azure Face API.

1.2 Motivation for the selected system development

During the proposal phase of the system, there was a chance of getting to know about people's emotions problem and heard about several emotional survey websites implemented in many countries. Inspired by those applications, this system was planned and being developed. This system will automate the face detection and emotional classification. The results collected through manual process will not be accurate. So, to overcome these issues, this system was planned and being developed. This application will have the facilities such as notifying user about updates, results provide by the Microsoft Azure Face API.

1.3 Importance and main purpose of the system

The Look It result collection and storage can be accessed and managed in each de-centralized system and data will be up to date. The need for manual process of emotional detector, manual advices for a person will be reduced by a considerable amount. It will be very useful since it reduces the time consumption for providing results by a huge amount. The old data can't be saved and compared with new updates and can check for the changes because the persons should forget their past negative experiences. User access privileges are given at different levels of priorities. This enables the system to keep control of the system security at a secured level. This system is implemented in such a way, it is much user friendly with users, since it has to deal with all types of users

1.4 Overview of the application

Look It is Android based application with a database system which will automate the face detection and emotional analysis. It will be very useful to keep the database up to date. In this application, admin is responsible for all the data management and adding videos to the system. A common user can use without register to the system as a user. The user will be notified about the result. System would be capable of generating results for the existing approved emotions.

2 Literature Review

2.1 The key idea

Facial recognition is being used in many businesses such as Payments, Access and security, Criminal identification, Advertising, Healthcare [1] Over the last ten years or so, face recognition has become a popular area of research in computer vision and one of the most successful applications of image analysis and understanding. Because of the nature of the problem, computer science researchers are not only interested in it, but also neuroscientists and psychologists' interest in it. It is the general opinion that advances in computer vision research will provide useful insights to neuroscientists and psychologists into how human brain works, and vice versa.[2]

These problems can be overcome by this Look It which automate the above manual detection. This is an Android based application which can be accessed by all the people. There is no registration process because each user access independently. The system is an Android application which handles the results data of the Emotion API and hands them over to the users when their systems are online. When a classification is done by an Emotion API using a certain user images, the user will be notified through text reply informing that the classification processing has been finished. Thereafter the user has to decide whether he/she want to see a video or not. But when the user fails to collect the results within a certain period, the input image as well as the results text will be erased from the application. Thus, a REST API has to be developed so as to receive the results data from Emotion API, send them to the user when the user's system is online. It will make sure that the user will receive the results data without loss.

This system is to be developed in order to handle a very large amount of data in future and send notifications to the users, members via SMS about the updates and etc.

2.2 Related developments and Product available

There are thousands of Applications available in the Google's play store and Apple's app store. These are some famous applications.

Trueface.AI – Fraud detection has a deep learning algorithm which was reportedly trained on thousands of examples of "attacks" that the team collected over the years. It has been reported that True face [3]. Yale center has been developed a Mood meter for all people and especially for children. It helps to concentrate the kids' education [4]. App store has an app. This app focuses on helping kids read body languages and understand emotions by looking at gorgeous pictures and figuring out which person is expressing a given emotion [5].

But these systems haven't the database for negative emotions and YouTube links to reduce the negative emotions

3 System Model

3.1 System Requirements

Functional Requirements

The main requirement of the system is detecting the face and find the emotion in the given image. So here, many use cases have been created such as capture an image, view received results, share results, receive results from Emotion API, Suggest Videos, sent image to the Emotion API given below is the use case diagram for the system

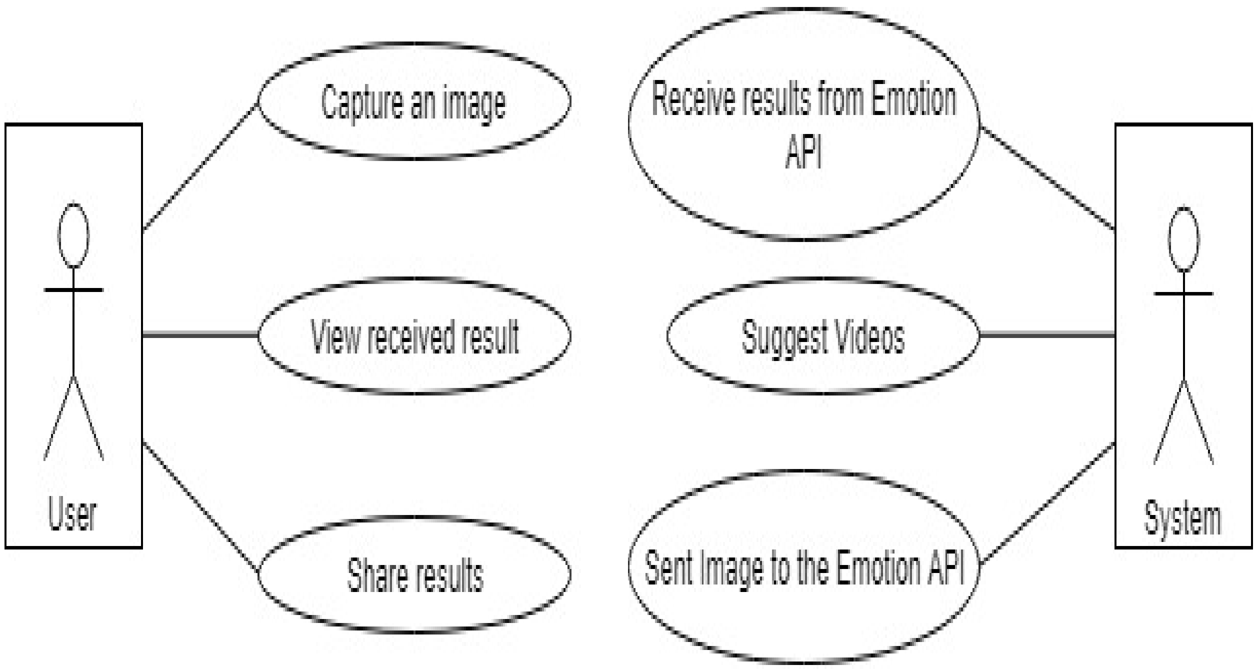


Figure 1- Use Case Diagram

When the user requests to take an image of his/her face in his/her Android application, the system should get the image from the user and when the user requests to view the details of the received results text, the system should display the text of the received result of him/her image in text format. If the user requests to share the details of the received results text, the system should share the text of the received result of him/her image in his / her Social media page. When the classification is finished at the Emotion API, the results will be received to the Look It application by using the REST API. If the classification is finished at the Emotion API, the results will be received to the Look It application by using the REST API. If result shows that user have negative emotions, then system suggest videos. When User take his face image and request to get results, System Sent the image through REST API to Emotion API to get the results.

Non- Functional Requirements

Usability

Users of the system should be able to interact with the system easily and the interaction should be easily understandable without any ambiguities. The users will not need any extra skills or high technical knowledge to use the system. Basic knowledge about using the Android Mobile will be enough to use the system.

Reliability

The Look It Android Application will be download through online and available anytime for any users. Unless there are any server-side problems, the system will have no limitations for the users. The system will not lose any results from the Emotion API and will make sure that the results are sent to the respective users.

Performance and Security

System should be able to provide secured for user's images as input of the system. The results data may be containing personal or confidential information for some user. Thus, it is necessary to have secure for user images. There are possibilities to be attacked by injections. System should be able to protect its users from the injections. Thus, it should provide a secured way to input and output data to and from the database sub-system.

Supportability

As it is an Android Application system, any users can access this system using their Android Mobile easily. System is developed so as to respond to all popular Android Mobiles. So, system may be adapted to the new environment. Moreover, the system should be able to all sizes of mobile screens such as Huawei, Samsung, OPPO, etc. mobile screens. There is a one configuration to access camera needed to reach the system in the user's perspective. Anyways, user need to have Internet connection to reach the Online System.

3.2 System Design

Main Architecture

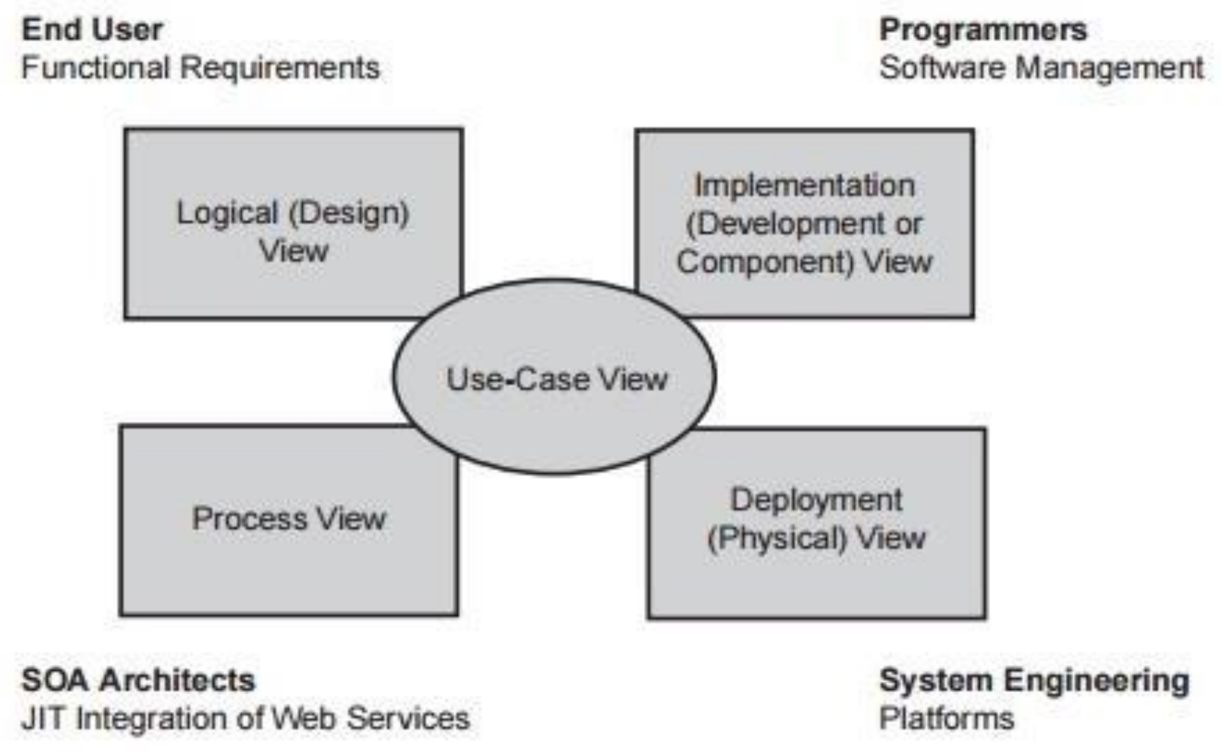


Figure 2 - "4+1" view model

The structure of this system is based on 4+1 architectural model which is used to describe the architecture of software-intensive systems based on the concurrent views. The above figure 3 describes about the "4+1" architectural model view of the system. According to this, this system is divided into 3 separate components called model, view and controller. They are interconnected with each other. MVC supports flexibility with the separation of responsibilities. Following figure illustrates the MVC structure of the system.

Deployment View

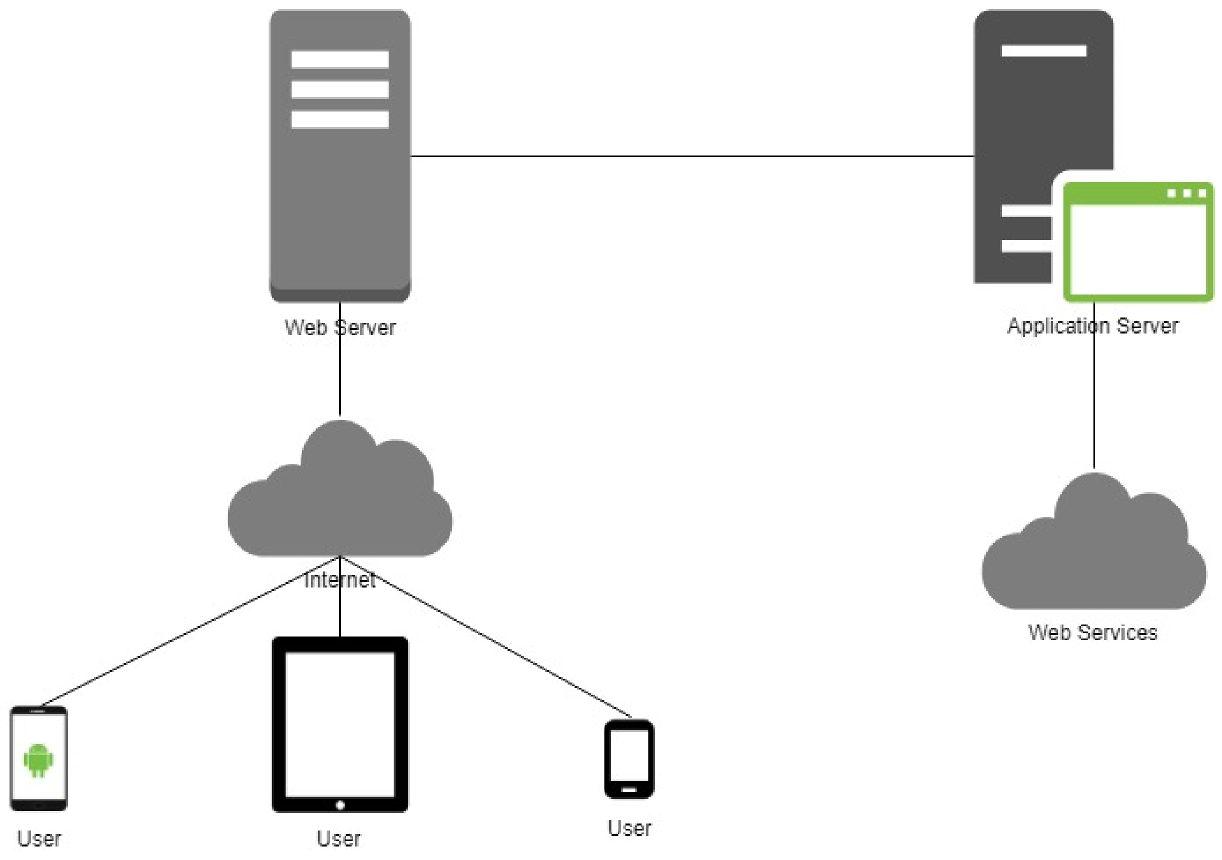


Figure 3- High level Architecture

The above Figure - 3 is the high level picture of the architecture of Look It. User of the system access it through the internet. Web server does the tasks in order to response the requests. All the user activities of the system will be saved in the system.

Logical View

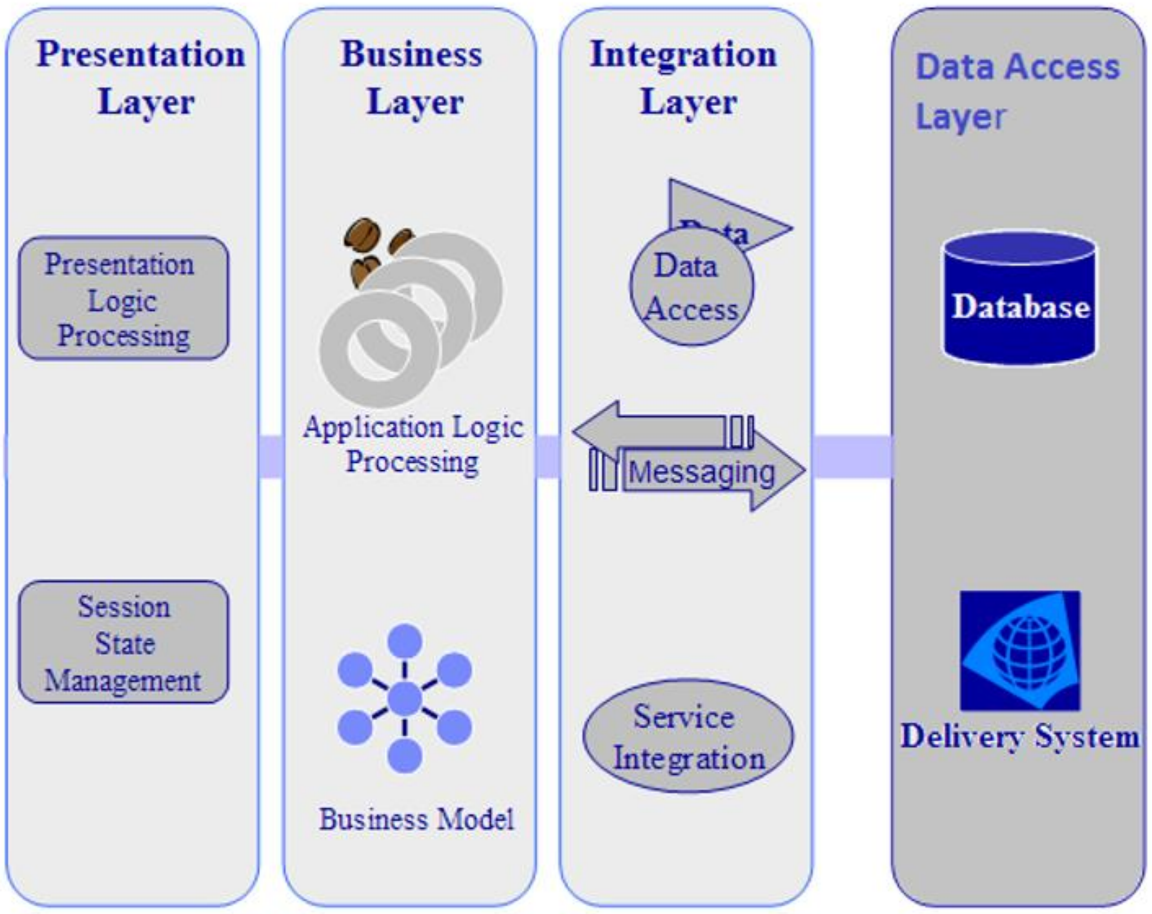


Figure 4- N-tier architecture

Figure 4: The Look It is divided into layers based on the N-tier architecture. The above figure is based on the responsibility layering strategy that associates with a particular responsibility. This strategy has been selected because it isolates the system responsibilities from one another. So it will improves both system development and maintenance.

Class Diagram

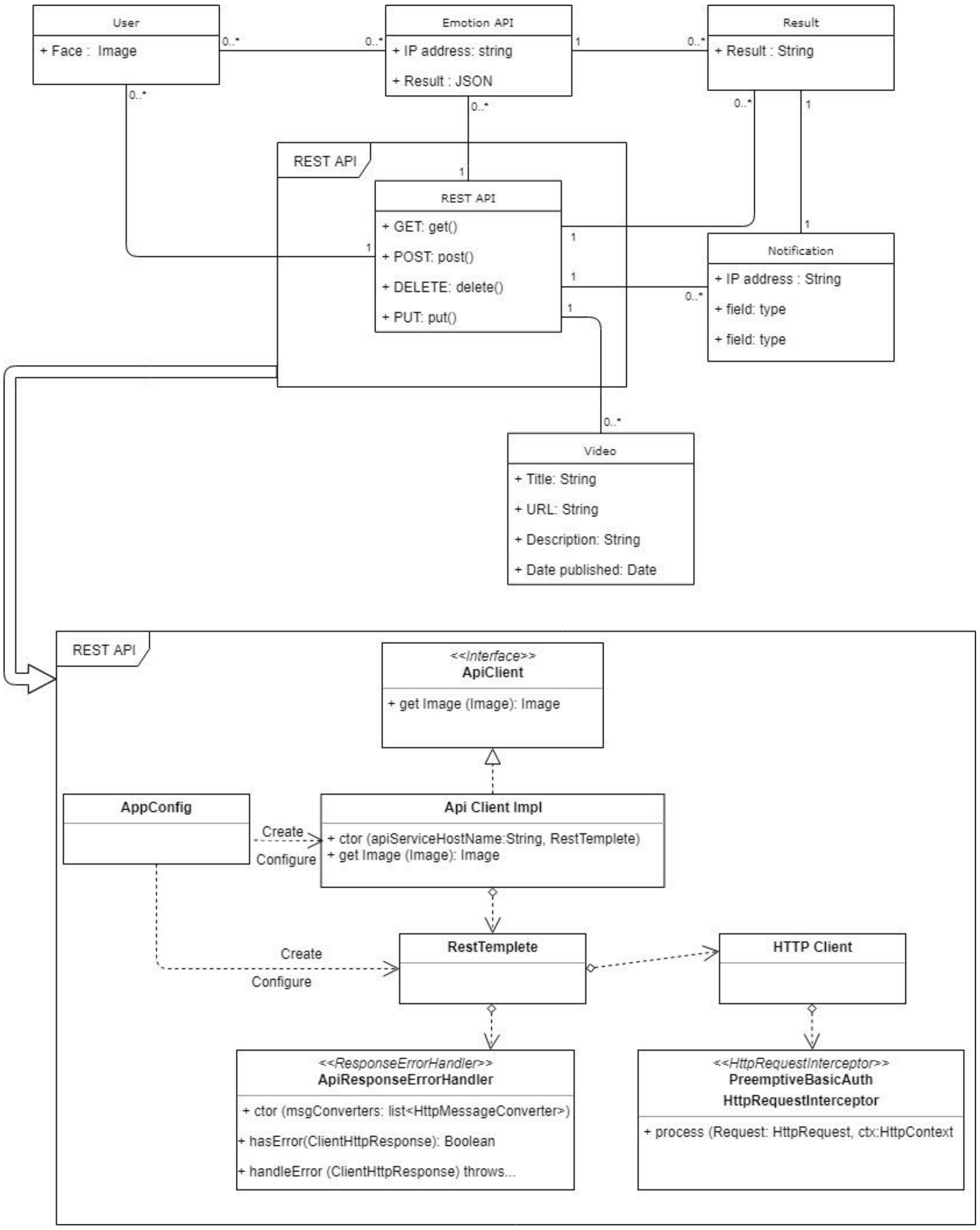

User
+ Face : Image
Emotion API
+ IP address: string
+ Result : JSON
Result
+ Result : String
REST API
+ GET: get()
+ POST: post()
+ DELETE: delete()
+ PUT: put()
Notification
+ IP address : String
+ field: type
+ field: type
Video
+ Title: String
+ URL: String
+ Description: String
+ Date published: Date
<<Interface>> ApiClient
+ get Image (Image): Image
AppConfig
Create
Configure
Api Client Impl
+ ctor (apiServiceHostName:String, RestTemplete)
+ get Image (Image): Image
RestTemplete
HTTP Client
<<ResponseErrorHandler>> ApiResponseErrorHandler
+ ctor (msgConverters: list<HttpMessageConverter>)
+ hasError(ClientHttpResponse): Boolean
+ handleError (ClientHttpResponse) throws...
<<HttpRequestInterceptor>> PreemptiveBasicAuth HttpRequestInterceptor
+ process (Request: HttpRequest, ctx:HttpContext
0..*
1

Figure 5- Class Diagram

The above figure describes the class diagram of the Look It. Each class has associate relationship with each other. Here clearly describe the Rest API infrastructure.

Process View

Capture Image

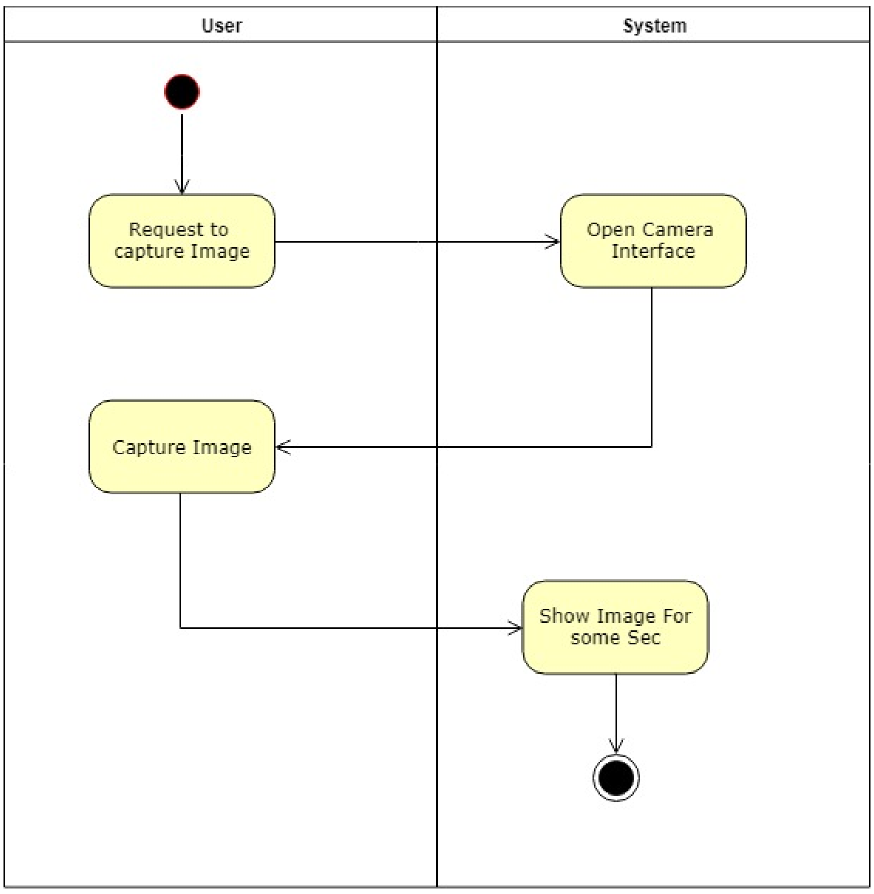


Figure 6- Activity Diagram- Capture Image

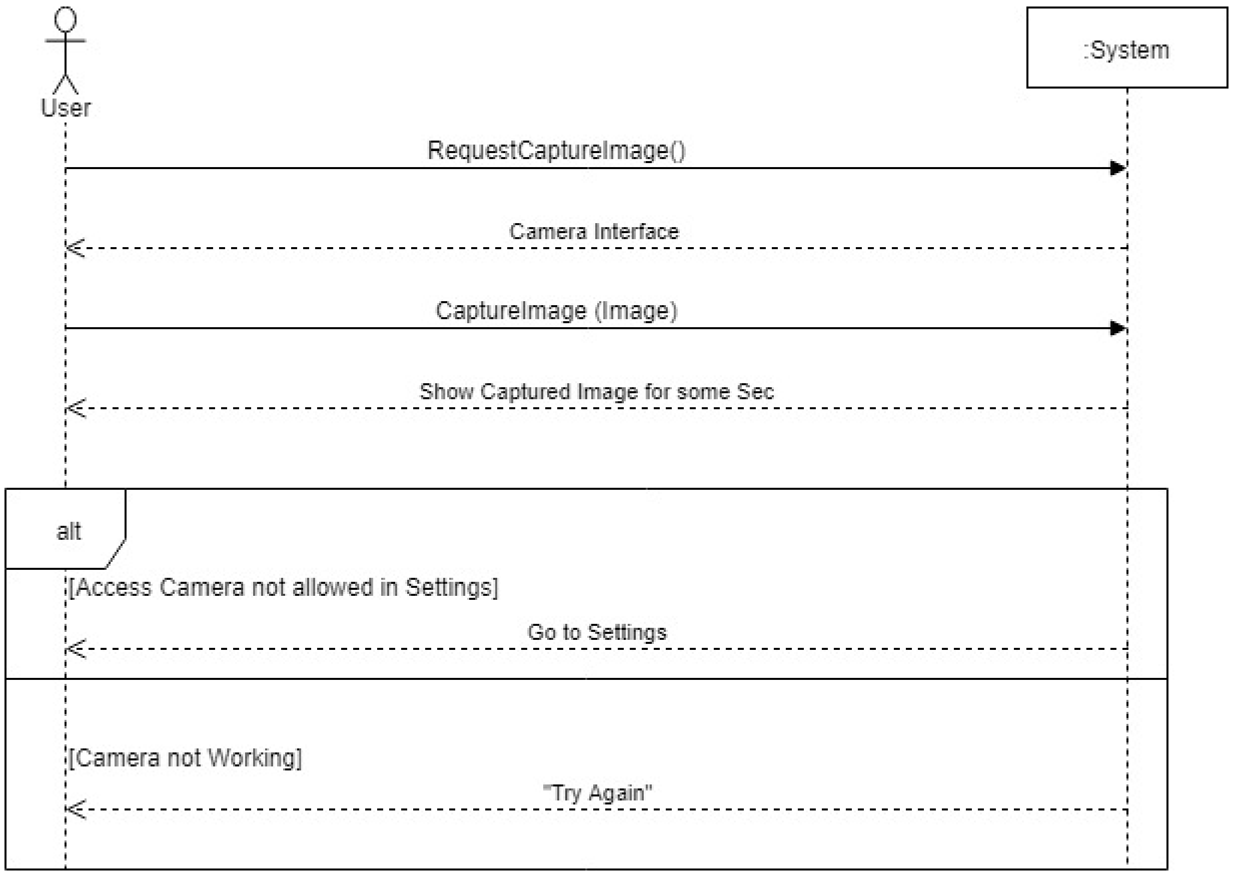


Figure 7- Sequence Diagram- Capture Image

View received results

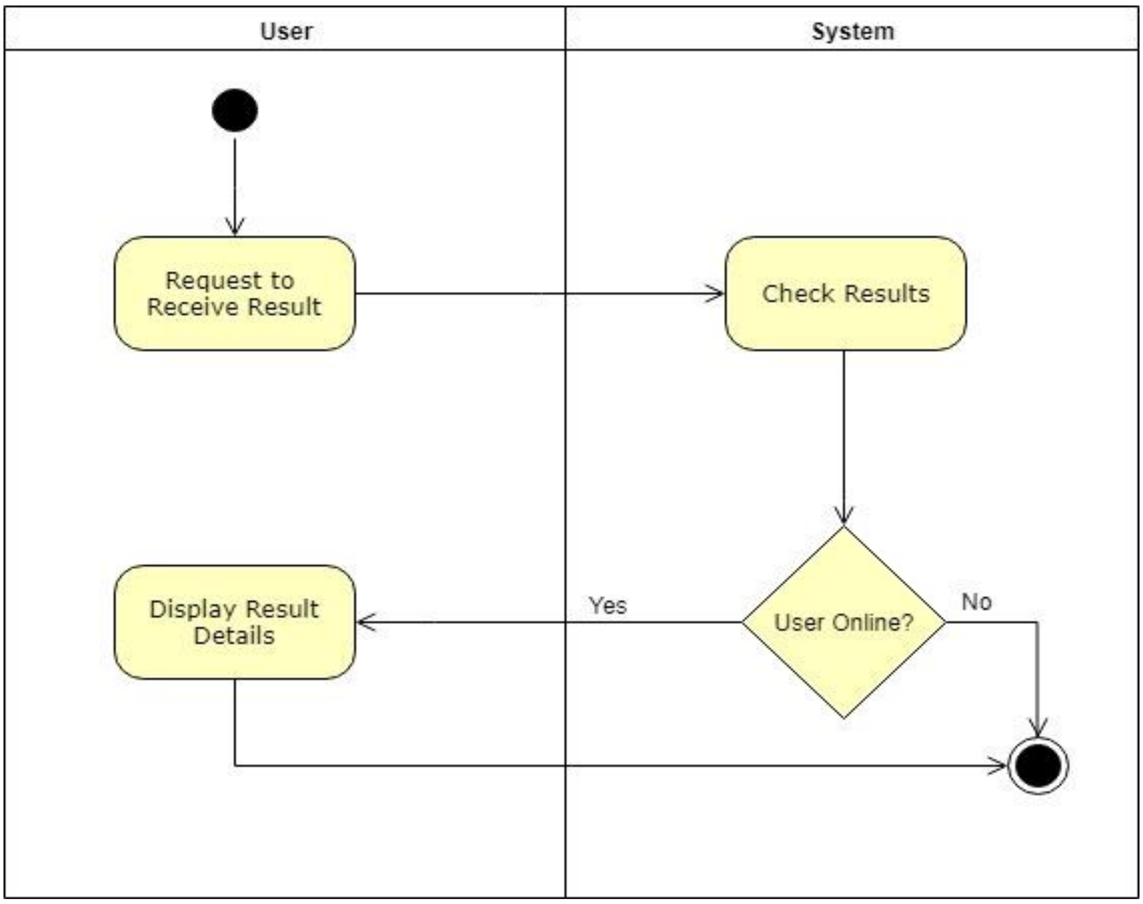


Figure 8- Activity Diagram- View received results

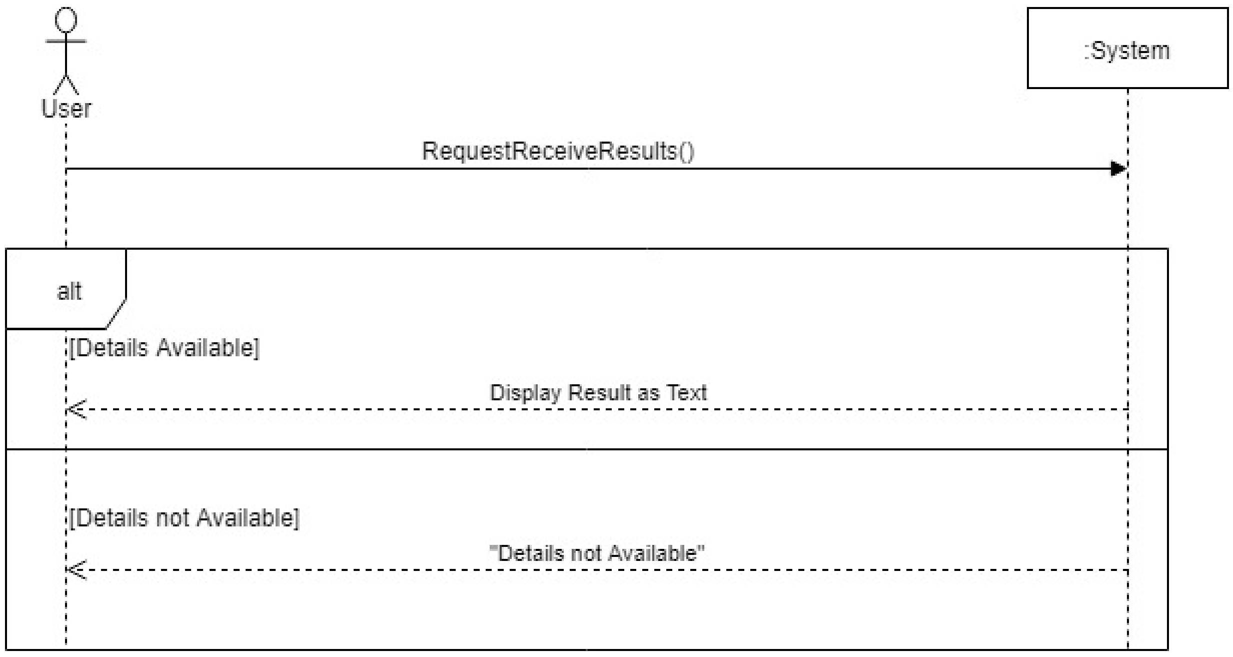


Figure 9- Sequence Diagram- View received results

Share results

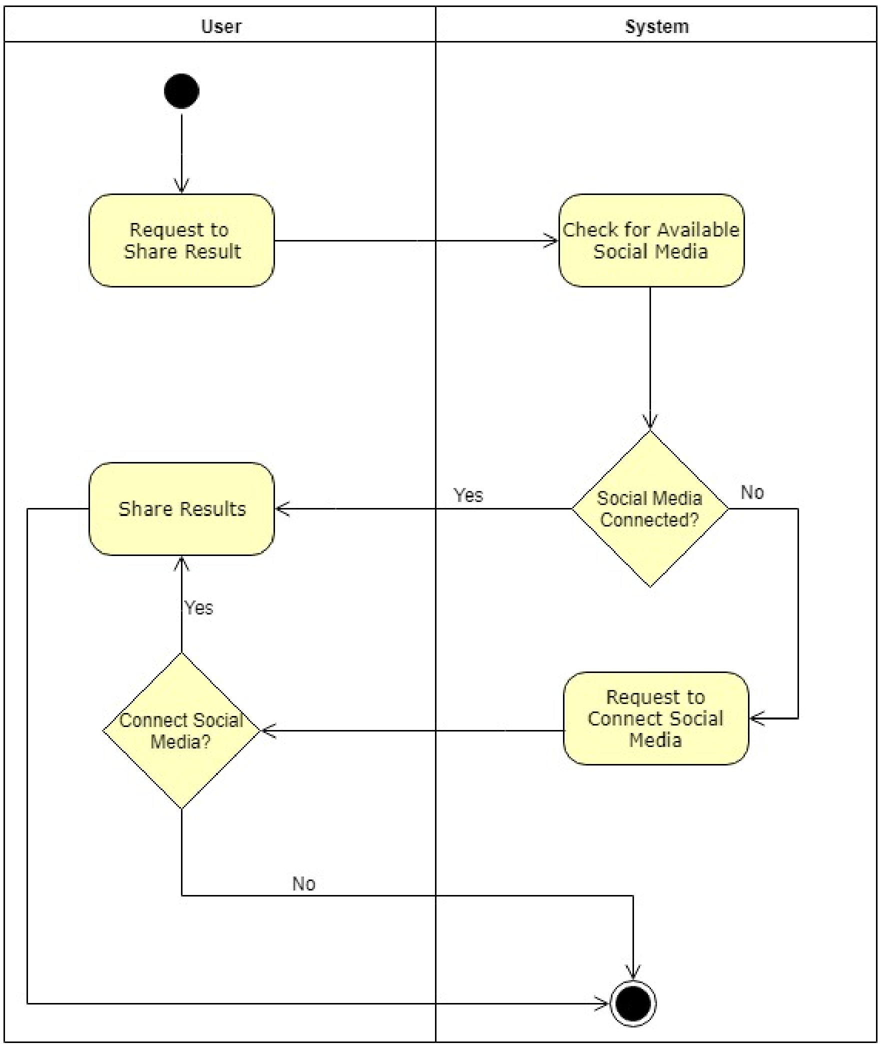


Figure 10- Activity Diagram- Share results

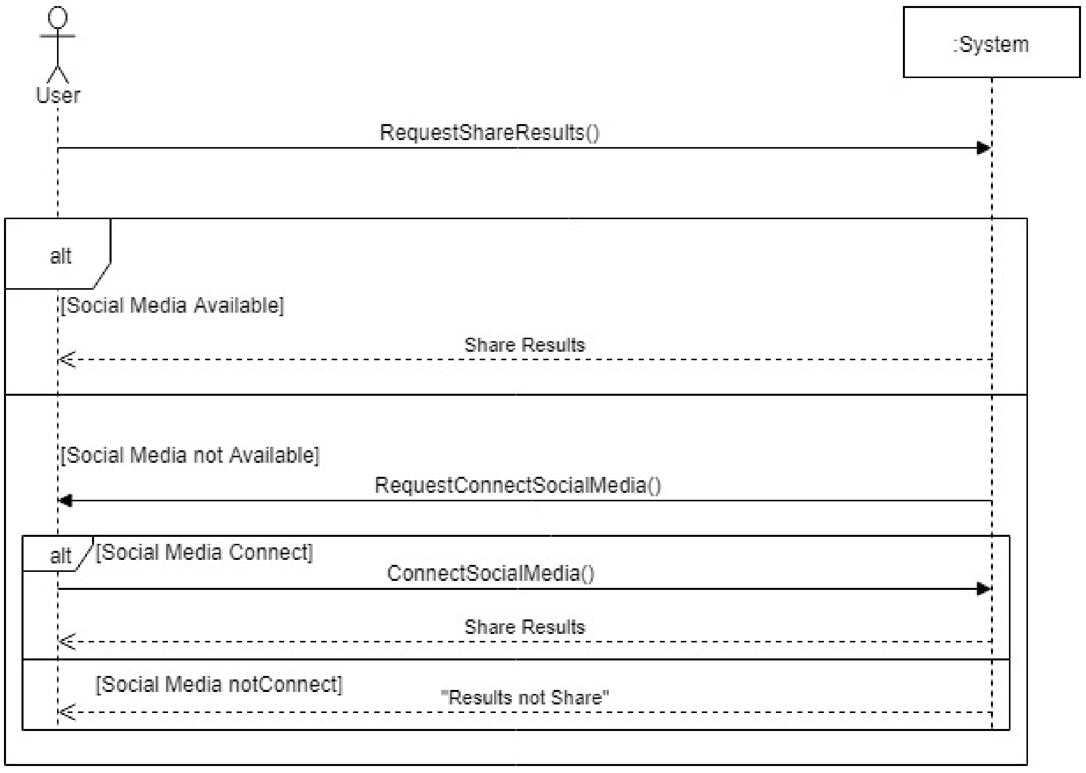


Figure 11- Sequence Diagram- Share results

Suggest Videos

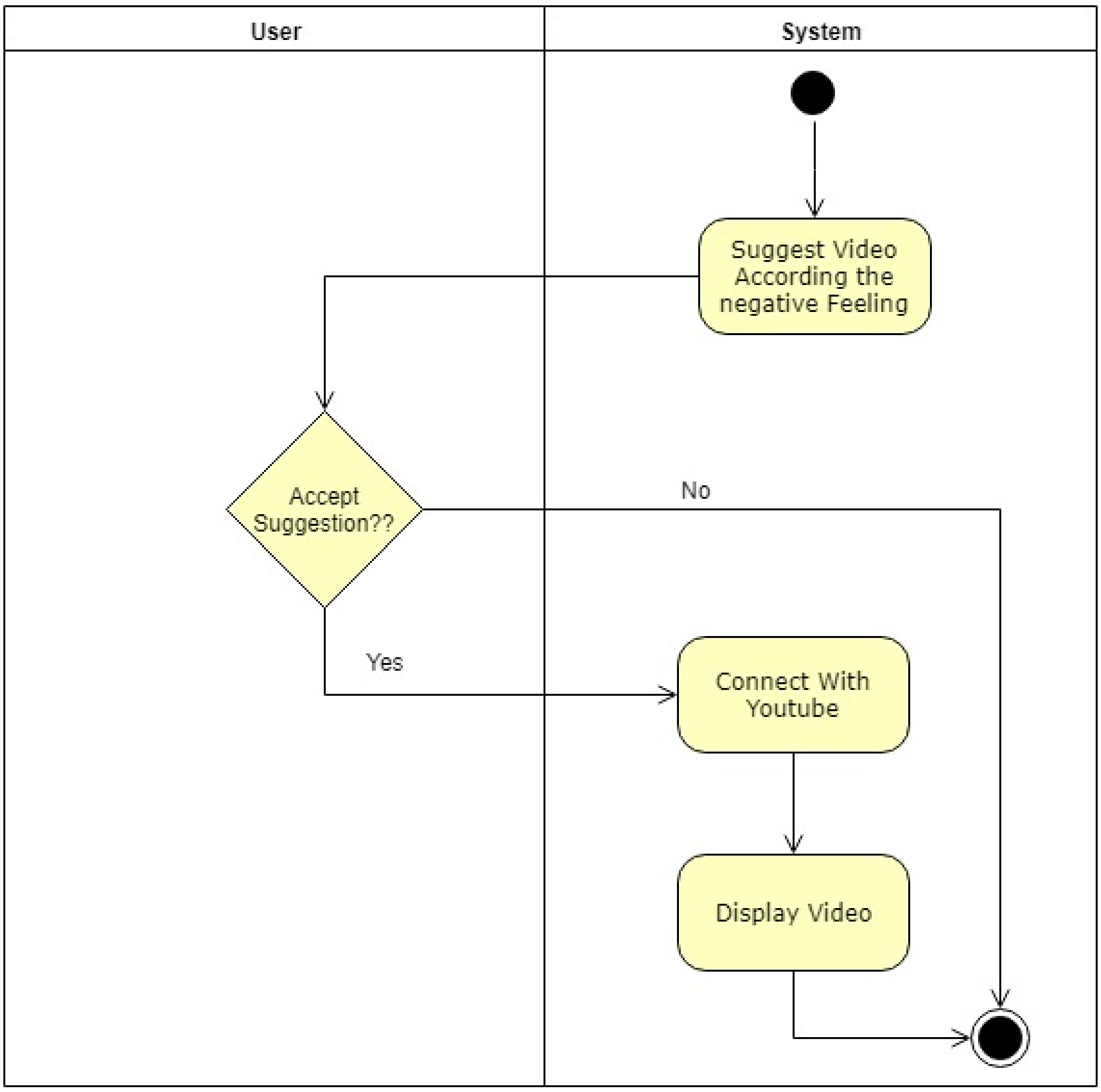


Figure 12- Activity Diagram- Suggest Videos

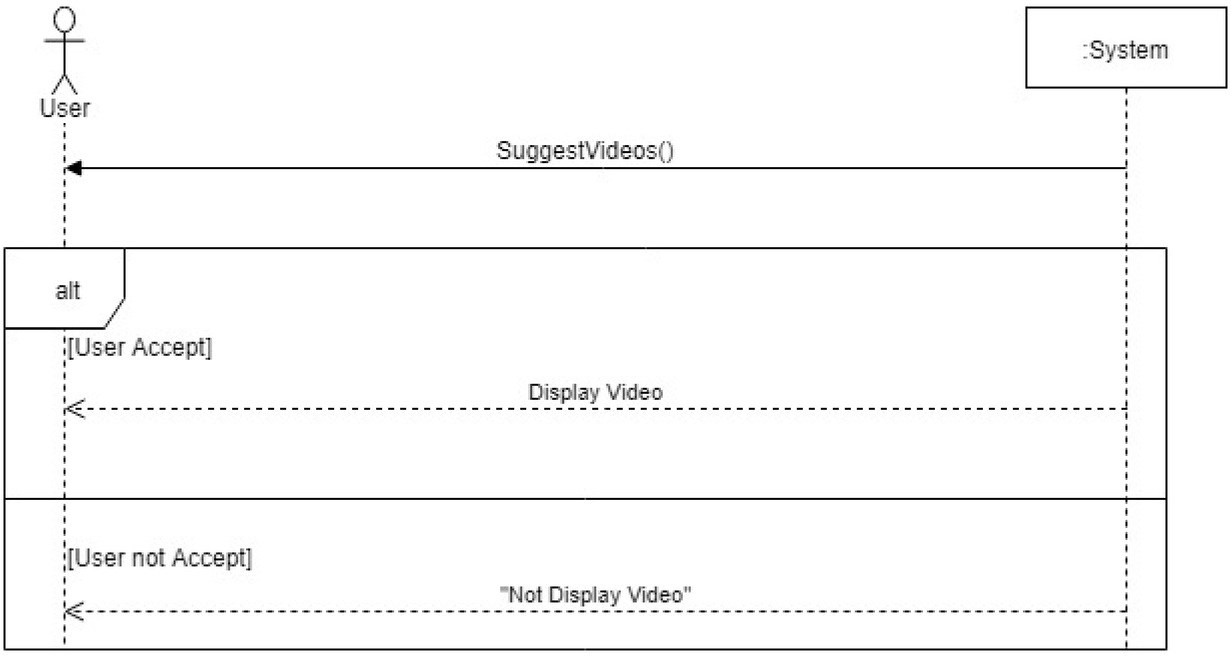
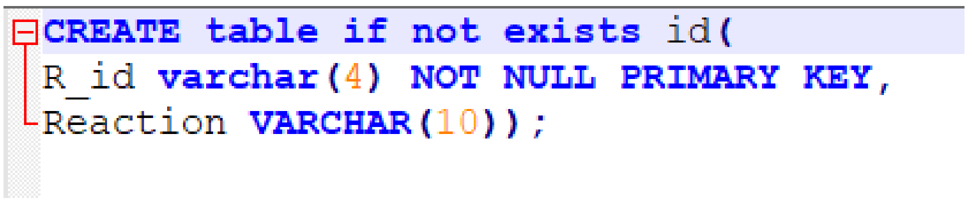


Figure 13- Sequence Diagram- Suggest Videos

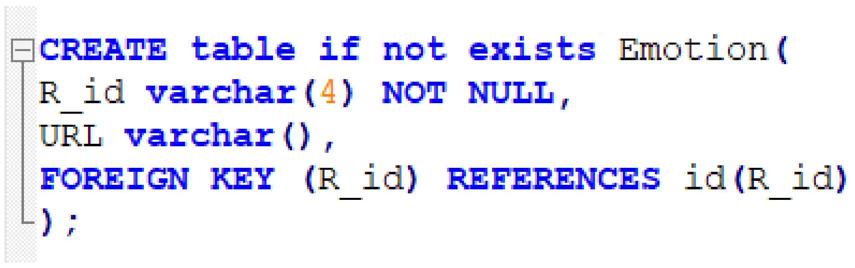
3.3 Database deign

The following is the database structure of this Look It



```
CREATE table if not exists id(
R_id varchar(4) NOT NULL PRIMARY KEY,
Reaction VARCHAR(10));
```

Figure 14- Table- id



```
CREATE table if not exists Emotion(
R_id varchar(4) NOT NULL,
URL varchar(),
FOREIGN KEY (R_id) REFERENCES id(R_id)
);
```

Figure 15- Table- Emotion

4 System Implementation

4.1 Implementation Procedure

This system was implemented by using Android Studio. Android Studio IDE was used as the tool for implementation.

Unified Process methodology is used for this Look It. This method is a popular iterative and incremental software development process framework. This UP methodology has clear objectives and goals for the four phases of the lifecycle of the system development. The feasibility study of the system is conducted in the Inception Phase of the lifecycle. In this feasibility study, the analysis of existing systems and business case, understanding the project scope, requirements gathering and defining all the required use cases were done. The risks were assessed and a project timeline was estimated and produced.

In Elaboration Phase, the majority of the requirements were defined and analyzed. The main functional and non-functional requirements of the system were identified and analyzed. The creation of use case diagrams with use case descriptions, class diagrams and architectural diagrams. For this Look It, the MVC architecture pattern was used to implement the system.

In construction phase, the largest part of the system development process is done. The output from the previous phase were used in this construction phase for the implementation of the system. The Activity diagrams, sequence diagrams, and interaction overview diagrams were used in this phase.

In the final phase, transition phase, the documentation and system training were given to the client. The maintenance of the system will be continued in this phase.

4.2 Materials

There is an existing API implement in it. Microsoft Azure Face API is that existing API.

4.3 The algorithm

If user give not an input, It will show a message that "Please take a Picture."

```
process.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View v) {
        if (ready) {
            detectandFrame(mBitmap);
        } else {
            makeToast("Please take a picture.");
        }
    }
});
```

Otherwise no any special algorithms used in this project. This is a Face classification System. No any special algorithms needed for this project implementation.

4.4 Main Interfaces

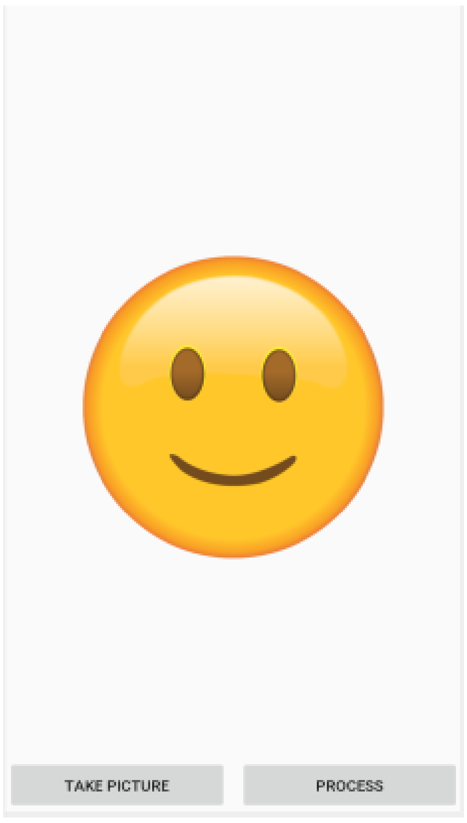


Figure 16- Front Interface

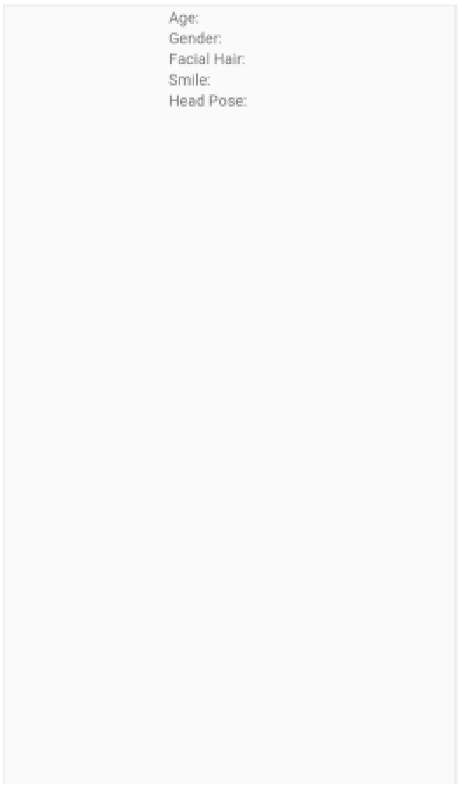

Age:
Gender:
Facial Hair:
Smile:
Head Pose:

Figure 17- Result Interface

5 System Testing and Analysis

A test plan has been planned and developed to ensure that the system meets its requirements without any failure or compromise. The objectives of this test plan was partially met. There is no any guarantee that the system will work without any single failure. But there are ways to reduce the risks related to the system and ensures the successful construction of the system.

- ❖ Unit Testing

The unit testing is used to verify the smallest unit of the software. The success and failure criteria of each unit will depend on the functionality of the particular unit or the subsystem. This unit testing will ensure that each sub-system works correctly as a separate module. The unit testing is done for result system.

- ❖ Integration Testing

Integration testing verifies whether the integrated sub parts are working properly. This testing is done by giving input programmatically and checking the activities. List of Items which have to be tested in Look It.

- View Results
- Share Results

- ❖ User Interface Testing

User interface testing is the process of testing the system's GUI to ensure that it meets its specifications. Not only the functionalities, the quality of design elements, quality of user elements such as layout, color, fonts, buttons, icons, links, and the content of the UI will be evaluated and verified whether they meets the user friendliness. This testing will include user screen validation checking, verify the object states and field formats such as numeric fields and etc.

UI items of Look It that has to be tested:

- Front Interface
- Result Interface

6 Conclusion and Future Work

6.1 Conclusion

Look It has been developed in less than 4 months and it works well. The main functionality of the system is detecting the Real time face and classify that face. Initial aim of the system was detecting the face. There were many use cases developed for main functionality such as Capture an image from User, View received result details, receive results from Emotion API, Suggest Videos, Sent Image to the Emotion API, Share results. more than that, System saves the YouTube links in a Database.

6.2 Future Work

In future this system will be developed in different platforms such as iOS, Windows, Linux etc. This system can be implemented in video conference software such as Skype, Hangouts. If a user got negative emotions, then there can be some desktop animations will show that user to relief from that negative emotions. Or If It will implement in Music players then Songs can be automatically changed in the music player. There are lot of opportunities to develop this system in future.

7 Reference

[1] disruptionhub [Online]. Available: https://disruptionhub.com/5-applications-facial-recognition-technology/

[2] hackernoon [Online]. Available: https://hackernoon.com/face-recognition-using-opencv-a-step-by-step-guide-to-build-a-facial-recognition-system-8da97cd89847

[3] emerj [Online]. Available: https://emerj.com/ai-sector-overviews/facial-recognition-applications/

[4] ei.yale.edu, [Online]. Available: http://ei.yale.edu/mood-meter-app/

[5] itune.apple.com, [Online]. Available: https://itunes.apple.com/us/app/touch-and-learn-emotions/id451685022?mt=8

Tool used to draw the diagrams:

draw.io. [Online]. Available: www.draw.io

lucid chart. [Online]. Available: www.lucidchart.com

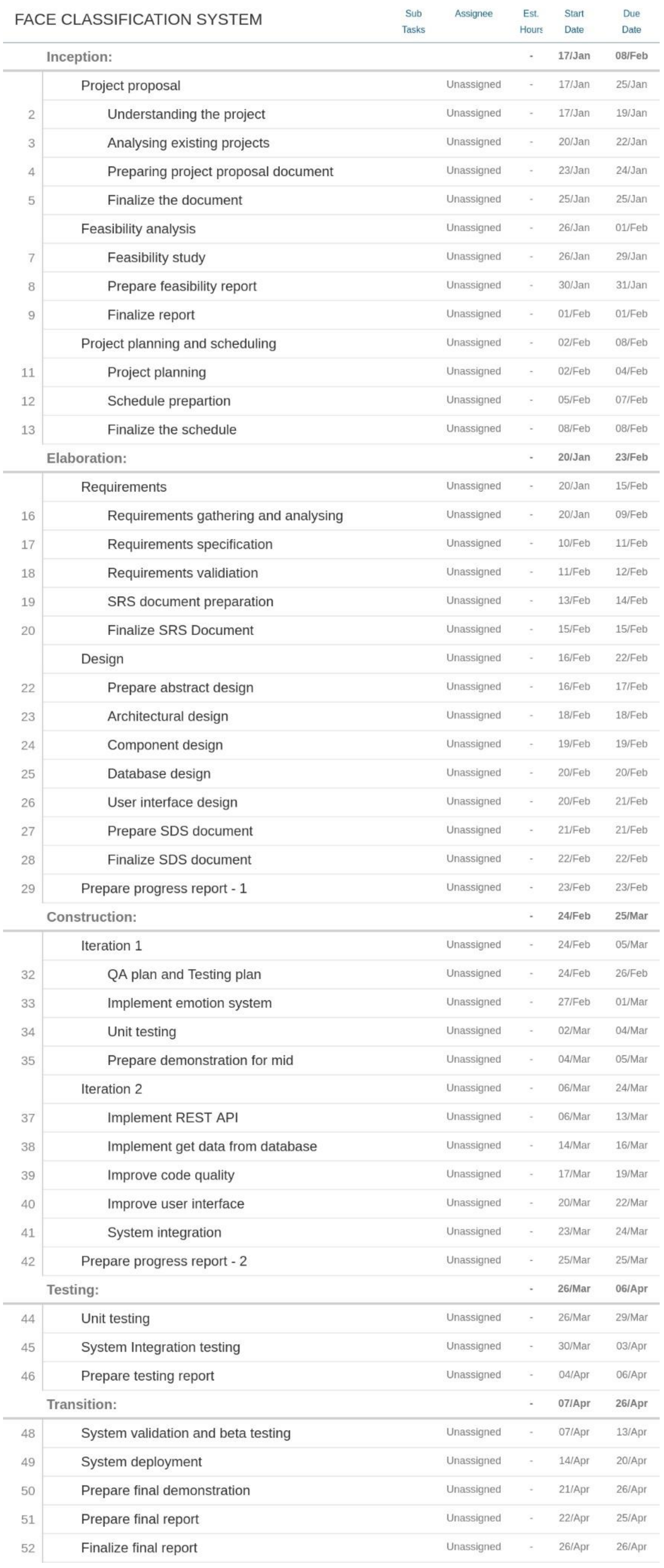
8 Appendix

Appendix 1:

Table of Figures:

Appendix 2:

Project Schedule



	FACE CLASSIFICATION SYSTEM	Sub Tasks	Assignee	Est. Hours	Start Date	Due Date
	Inception:			-	**17/Jan**	**08/Feb**
	Project proposal		Unassigned	-	17/Jan	25/Jan
2	Understanding the project		Unassigned	-	17/Jan	19/Jan
3	Analysing existing projects		Unassigned	-	20/Jan	22/Jan
4	Preparing project proposal document		Unassigned	-	23/Jan	24/Jan
5	Finalize the document		Unassigned	-	25/Jan	25/Jan
	Feasibility analysis		Unassigned	-	26/Jan	01/Feb
7	Feasibility study		Unassigned	-	26/Jan	29/Jan
8	Prepare feasibility report		Unassigned	-	30/Jan	31/Jan
9	Finalize report		Unassigned	-	01/Feb	01/Feb
	Project planning and scheduling		Unassigned	-	02/Feb	08/Feb
11	Project planning		Unassigned	-	02/Feb	04/Feb
12	Schedule prepartion		Unassigned	-	05/Feb	07/Feb
13	Finalize the schedule		Unassigned	-	08/Feb	08/Feb
	Elaboration:			-	**20/Jan**	**23/Feb**
	Requirements		Unassigned	-	20/Jan	15/Feb
16	Requirements gathering and analysing		Unassigned	-	20/Jan	09/Feb
17	Requirements specification		Unassigned	-	10/Feb	11/Feb
18	Requirements validiation		Unassigned	-	11/Feb	12/Feb
19	SRS document preparation		Unassigned	-	13/Feb	14/Feb
20	Finalize SRS Document		Unassigned	-	15/Feb	15/Feb
	Design		Unassigned	-	16/Feb	22/Feb
22	Prepare abstract design		Unassigned	-	16/Feb	17/Feb
23	Architectural design		Unassigned	-	18/Feb	18/Feb
24	Component design		Unassigned	-	19/Feb	19/Feb
25	Database design		Unassigned	-	20/Feb	20/Feb
26	User interface design		Unassigned	-	20/Feb	21/Feb
27	Prepare SDS document		Unassigned	-	21/Feb	21/Feb
28	Finalize SDS document		Unassigned	-	22/Feb	22/Feb
29	Prepare progress report - 1		Unassigned	-	23/Feb	23/Feb
	Construction:			-	**24/Feb**	**25/Mar**
	Iteration 1		Unassigned	-	24/Feb	05/Mar
32	QA plan and Testing plan		Unassigned	-	24/Feb	26/Feb
33	Implement emotion system		Unassigned	-	27/Feb	01/Mar
34	Unit testing		Unassigned	-	02/Mar	04/Mar
35	Prepare demonstration for mid		Unassigned	-	04/Mar	05/Mar
	Iteration 2		Unassigned	-	06/Mar	24/Mar
37	Implement REST API		Unassigned	-	06/Mar	13/Mar
38	Implement get data from database		Unassigned	-	14/Mar	16/Mar
39	Improve code quality		Unassigned	-	17/Mar	19/Mar
40	Improve user interface		Unassigned	-	20/Mar	22/Mar
41	System integration		Unassigned	-	23/Mar	24/Mar
42	Prepare progress report - 2		Unassigned	-	25/Mar	25/Mar
	Testing:			-	**26/Mar**	**06/Apr**
44	Unit testing		Unassigned	-	26/Mar	29/Mar
45	System Integration testing		Unassigned	-	30/Mar	03/Apr
46	Prepare testing report		Unassigned	-	04/Apr	06/Apr
	Transition:			-	**07/Apr**	**26/Apr**
48	System validation and beta testing		Unassigned	-	07/Apr	13/Apr
49	System deployment		Unassigned	-	14/Apr	20/Apr
50	Prepare final demonstration		Unassigned	-	21/Apr	26/Apr
51	Prepare final report		Unassigned	-	22/Apr	25/Apr
52	Finalize final report		Unassigned	-	26/Apr	26/Apr