1. INTRODUCTION

1.1 PROJECT SUMMARY

- Project is based on filling attendance by face recognition.
- Each and every student can fill their attendance.
- Built for making attendance process easier.

1.2 PURPOSE

- This web app detects the face of student, recognizes who is the student and then fills the attendance of respective student in the records.
- The main purpose of this web app is to fill the attendance easier just by showing our face.

1.3 SCOPE

- This web app is somewhat similar compared to other face recognize apps available but adding few more ideas. Adding more ideas, we can record the attendance and also we can generate report based on the attendance.
- There are two type of user 1. Faculty 2. Students
- Faculties people can add remove students and fill their attendance and generate report via daily, weekly, monthly and semester vise.
- Student can only see attendance on daily basis. They able to see report if particular faculty gives them permission. Otherwise they can only see before one week of internals.

1.4 OBJECTIVE

- The main objective of making this application is to remove the hectic process to fill the attendance one by one.
- Hence saving the time and workload.
- The system is automatic hence it reduces the paperwork and workload of taking attendance of respective students.
- Hence now onwards students will not be able to fill proxy of others.

1.5 TECHNOLOGY AND LITERATURE REVIEW

- This project is made in OpenCV, Netbeans and the language is Python, Java, MySQL. [1]
- This application will be run in Chrome compactible at any version.
- OpenCV is released under a BSD license and hence it's free for both academic and commercial use. It has C++, C, Python and Java interfaces and supports Windows, Linux, Mac OS, iOS and Android. OpenCV was designed for computational efficiency and with a strong focus on real-time applications. Written in optimized C/C++, the library can take advantage of multi-core processing. Enabled with OpenCL, it can take advantage of the hardware acceleration of the underlying heterogeneous compute platform.[1]
- A Java servlet is a Java program that extends the capabilities of a server. Although servlets can respond to any types of requests, they most commonly implement applications hosted on Web servers. Such Web servlets are the Java counterpart to other dynamic Web content technologies such as PHP and ASP.NET.[3]

2. OVERALL DESCRIPTION

2.1 PRODUCT PERSPECTIVE

- Following diagram shows perspective of product.

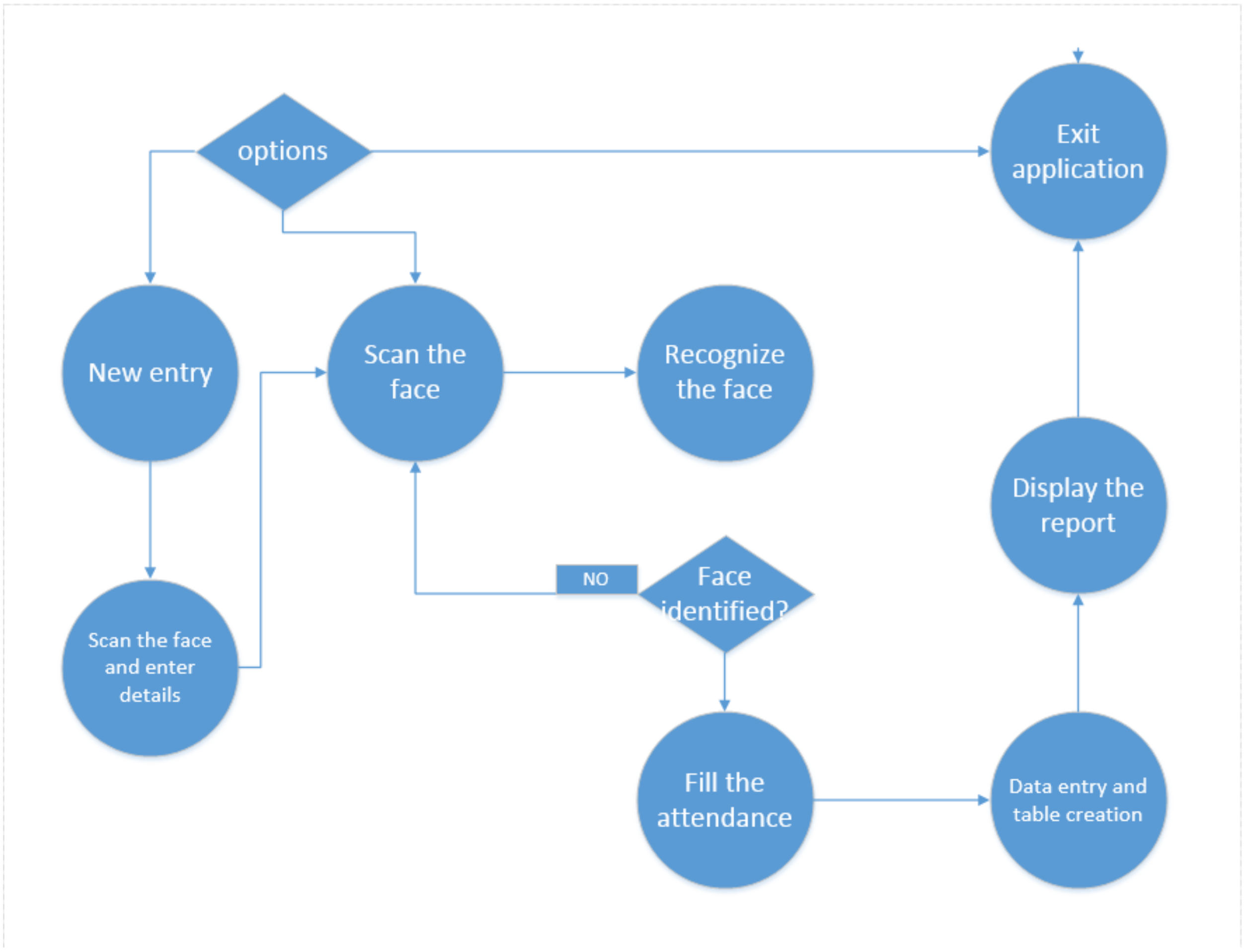


Fig 2.1 perspective of product

2.2 PRODUCT FUNCTIONS

- The web app will firstly detect our face by enabling the camera by itself when that site will be opened. Then after detecting the face, the app will automatically recognizes the face by comparing the data with the database available. Then app will fill the attendance of that student in the attendance record and then teachers can generate attendance report based on that database.

2.3 USER CHARCTERISTICS

- Generally this type of web app can be accessed by both teachers as well as student.
- This web application can detect our face, recognize it and fill the attendance in database.

2.4 CONSATRAINTS

- This web app is not the only web app which detects and recognize the face. And this app may not be that efficient as the other apps available today.
- The web app is implemented in several languages. Java, CSS, HTML, MySQL, Python. Also, to make this app work; XAMPP, OpenCV, Notepad is used. Hence increasing complexity to make this project work and understand.

2.5. ASSUMPTIONS AND DEPENDENCIES

- The web app will be fully functional and error free, running without any crash.
- The web app remains stable and compatible with Windows 7 and greater.

3. SYSTEM STUDY

3.1 REQUIREMENT

3.1.1 Functional Requirement

- **SCAN THE FACE:**

This application should scan the face when we place our face towards camera.

Input: Our face.

Output: The application scans the face provided as an input.

- **IDENTIFY THE FACE:**

When the face is scanned, the application should recognize who's face it is.

Input: Our face.

Output: Identify who's face it is.

3.1.2 Non Functional Requirement

LOADING:

Face should be identified fast.

DETAIL:

Application should provide more details of the student.

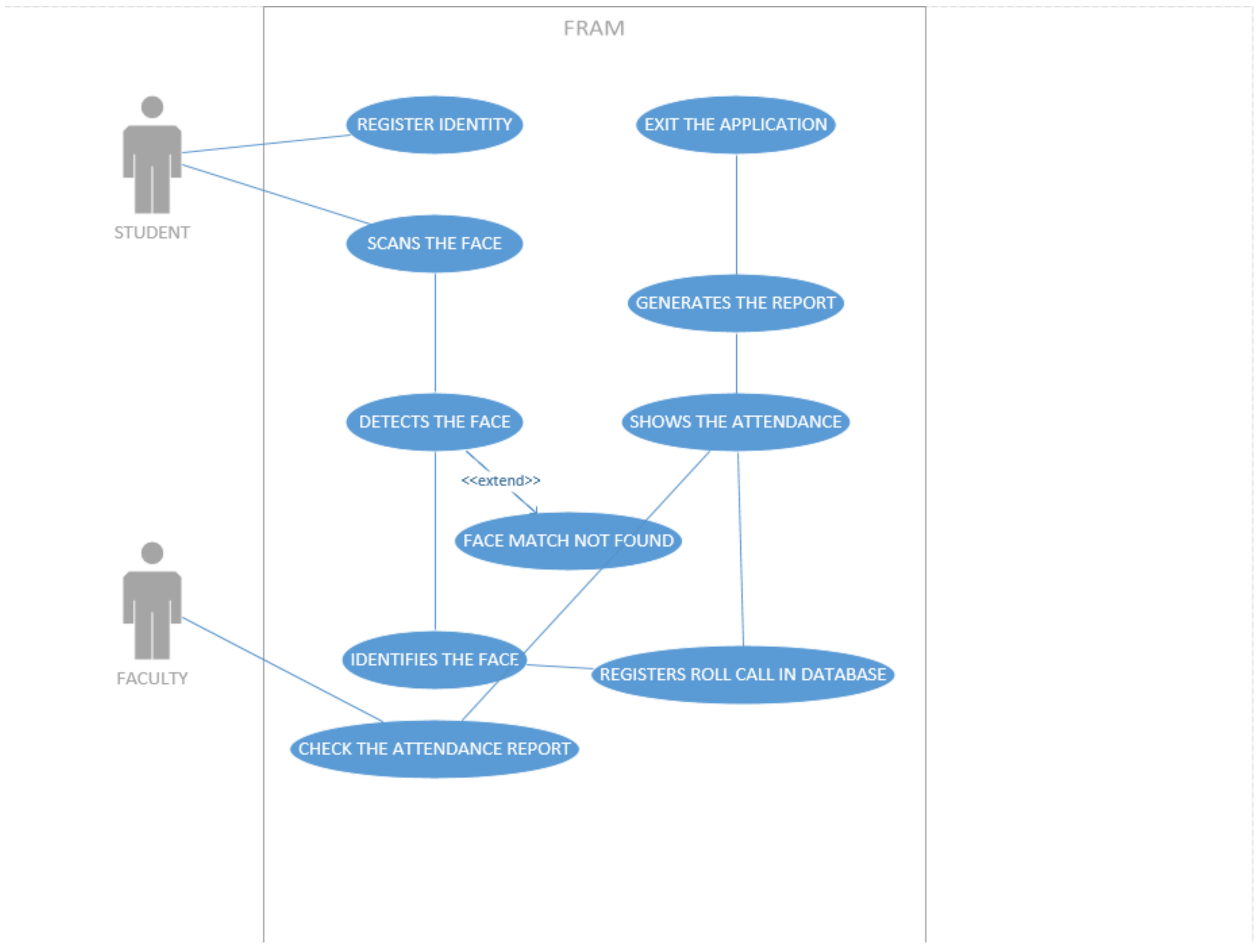
UI:

It should be easily understandable, in other words, it should be handy.

3.2 FEASIBLITY STUDY

- **Does the system contribute to the overall objectives of the organization?**
  - YES
- **Can the system be implemented using the current technology and within the given cost and schedule constraints?**
  - YES
- **Can the system be integrated with other system which are already in place?**
  - NO

3 SYSTEM ACTIVITY(Use Case)



3.2 CLASS DIAGRAM

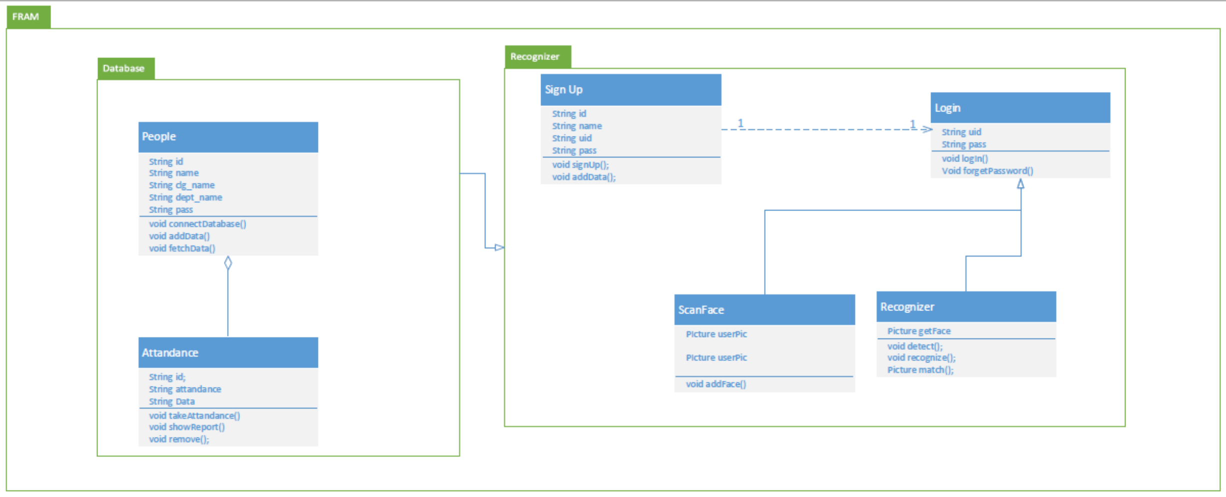
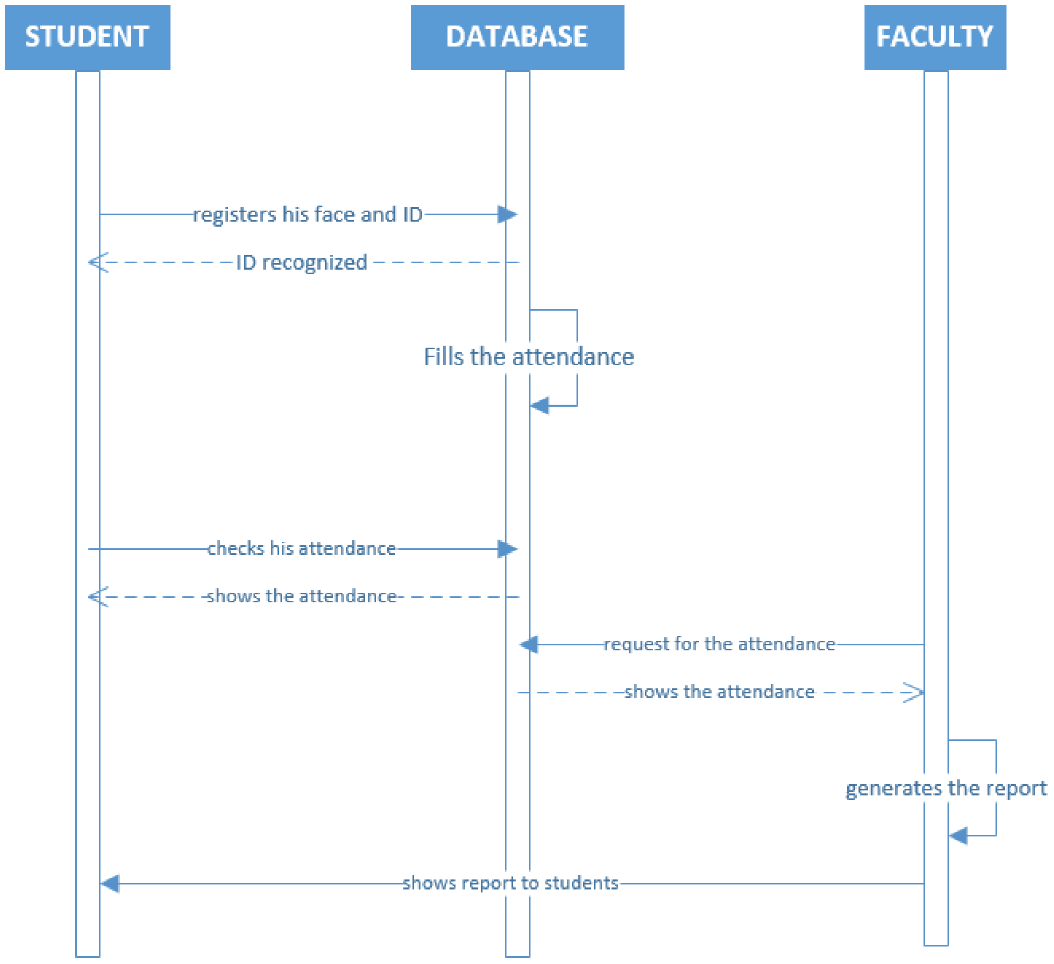


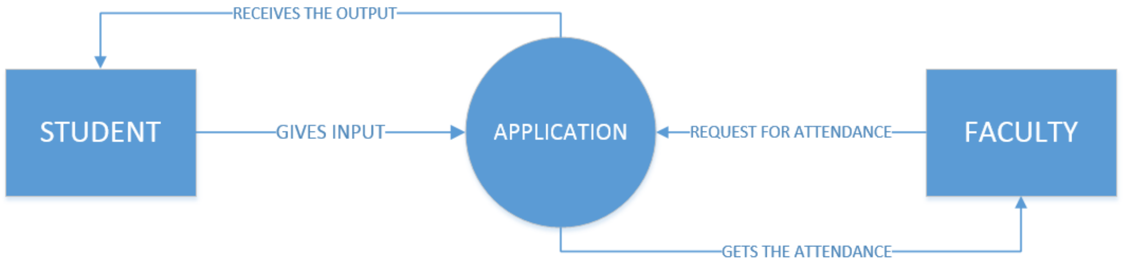
FIG 3.2 CLASS DIAGRAM

3.3 SEQUENCE DIAGRAM



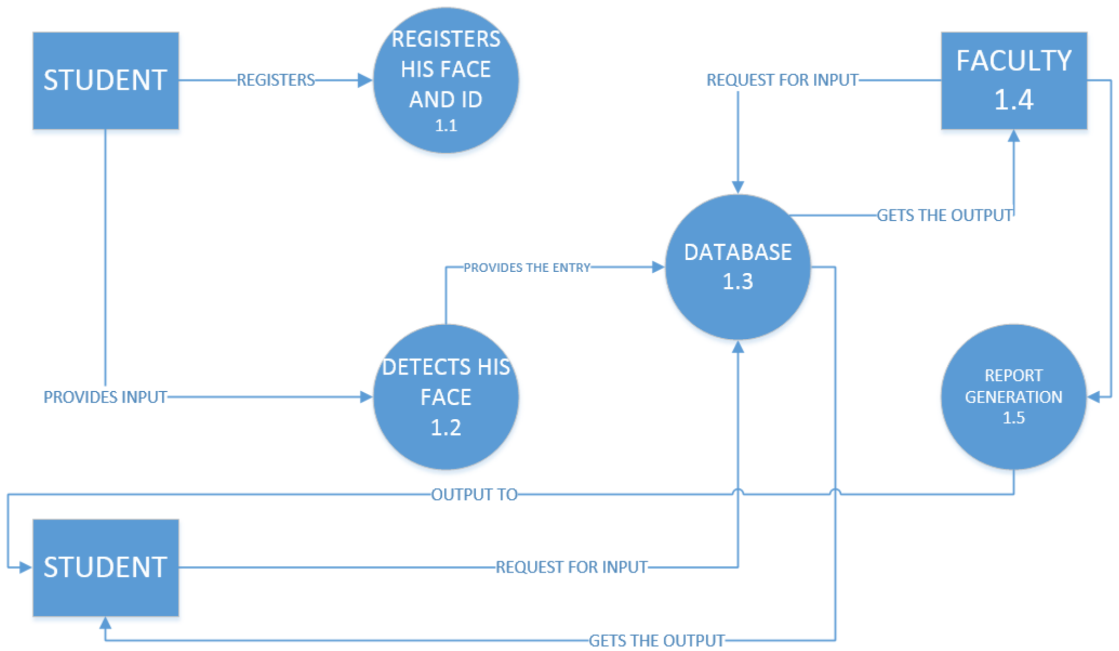
3.3 sequence diagram

3.4 CONTEXT DIAGRAM



3.4 Context diagram

3.5 DFD DIAGRAM



3.5 DFD diagram

4 PROJECT ESTIMATION

4.3 ESTIMATION METHOD USED[2]

- We used cocomo method for our project.
- Formula :FP=count total*[0.65+ (0.01* ∑Fi)]
- Function point count for FRAM.

Table 4.1 General System characteristics

Adjustment factor	value
1. Does the system require reliable backup and recovery?	4
2. Are data communications required?	2
3. Are there distributed processing functions?	3
4. Is performance critical?	3
5. Will the system run in an existing, heavily utilized operational environment?	2
6. Does the system require on-line data entry?	0
7. Does the on-line data entry require the input transaction to be built over multiple screens or operations?	0
8. Are the master file updated on-line?	0
9. Are the inputs, outputs, files, or inquiries complex?	0
10. Is the internal processing complex?	3
11. In the code designed to be reusable?	1
12. Are conversion and installation included in the design?	4
13. Is the system designed for multiple installations in different organizations?	3
14. Is the application designed to facilitate change and ease of use by the user?	5
Total	30

Table 4.2

Type of components	Complexity of components	Value
External inputs	2*4	12
External outputs	1*5	20
External inquiries	1*4	4
External logical files	1*10	10
External interface files	1*7	7
	Total	34

Unadjusted function point

- Formula
- FP = count total*[0.65+ (0.01*ΣFi)]
- = 34*[0.65+ (0.01*30)]
- = 32.3
- Function point count for FPS Game= 58.28
- Lines of code=32.3*30=969=0.969 KLOC
- Estimation of effort using Cocomo Model:
  a=2.4; b=1.05

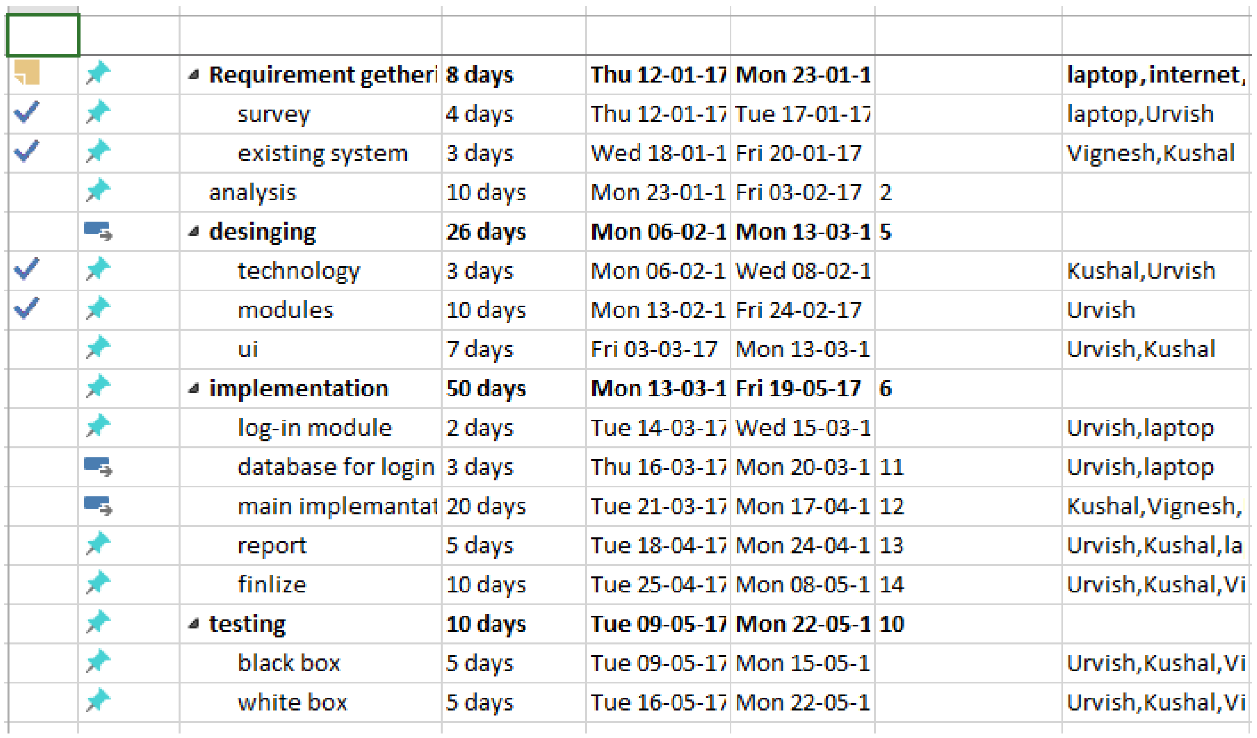
  Effort =a*(KLOC) b
  =$2.4*(0.969)^{1.05}$
  =2.24 person-month

- Estimation of time using Cocomo Model:
  Type of Project = Organic. Then, c=2.5; d=0.38

  Time = (Effort) d * c
  = (2.24) 0.38 * 2.5
  = 2.13 months
- Estimation of Cost using Cocomo Model:
  Cost = 2.13 * 5000
  = Rs. 10,611

5 SCHEDULE IN WHICH

5.1 BREAKDOWN STRUCTRE



		◢ Requirement getheri	**8 days**	**Thu 12-01-17**	**Mon 23-01-1**		**laptop,internet,**
✓		survey	4 days	Thu 12-01-17	Tue 17-01-17		laptop,Urvish
✓		existing system	3 days	Wed 18-01-1	Fri 20-01-17		Vignesh,Kushal
		analysis	10 days	Mon 23-01-1	Fri 03-02-17	2	
		◢ desinging	**26 days**	**Mon 06-02-1**	**Mon 13-03-1**	**5**	
✓		technology	3 days	Mon 06-02-1	Wed 08-02-1		Kushal,Urvish
✓		modules	10 days	Mon 13-02-1	Fri 24-02-17		Urvish
		ui	7 days	Fri 03-03-17	Mon 13-03-1		Urvish,Kushal
		◢ implementation	**50 days**	**Mon 13-03-1**	**Fri 19-05-17**	**6**	
		log-in module	2 days	Tue 14-03-17	Wed 15-03-1		Urvish,laptop
		database for login	3 days	Thu 16-03-17	Mon 20-03-1	11	Urvish,laptop
		main implemantat	20 days	Tue 21-03-17	Mon 17-04-1	12	Kushal,Vignesh,
		report	5 days	Tue 18-04-17	Mon 24-04-1	13	Urvish,Kushal,la
		finlize	10 days	Tue 25-04-17	Mon 08-05-1	14	Urvish,Kushal,Vi
		◢ testing	**10 days**	**Tue 09-05-17**	**Mon 22-05-1**	**10**	
		black box	5 days	Tue 09-05-17	Mon 15-05-1		Urvish,Kushal,Vi
		white box	5 days	Tue 16-05-17	Mon 22-05-1		Urvish,Kushal,Vi

FIG 5.1 Breakdown structure

5.2 TASK NETWORK

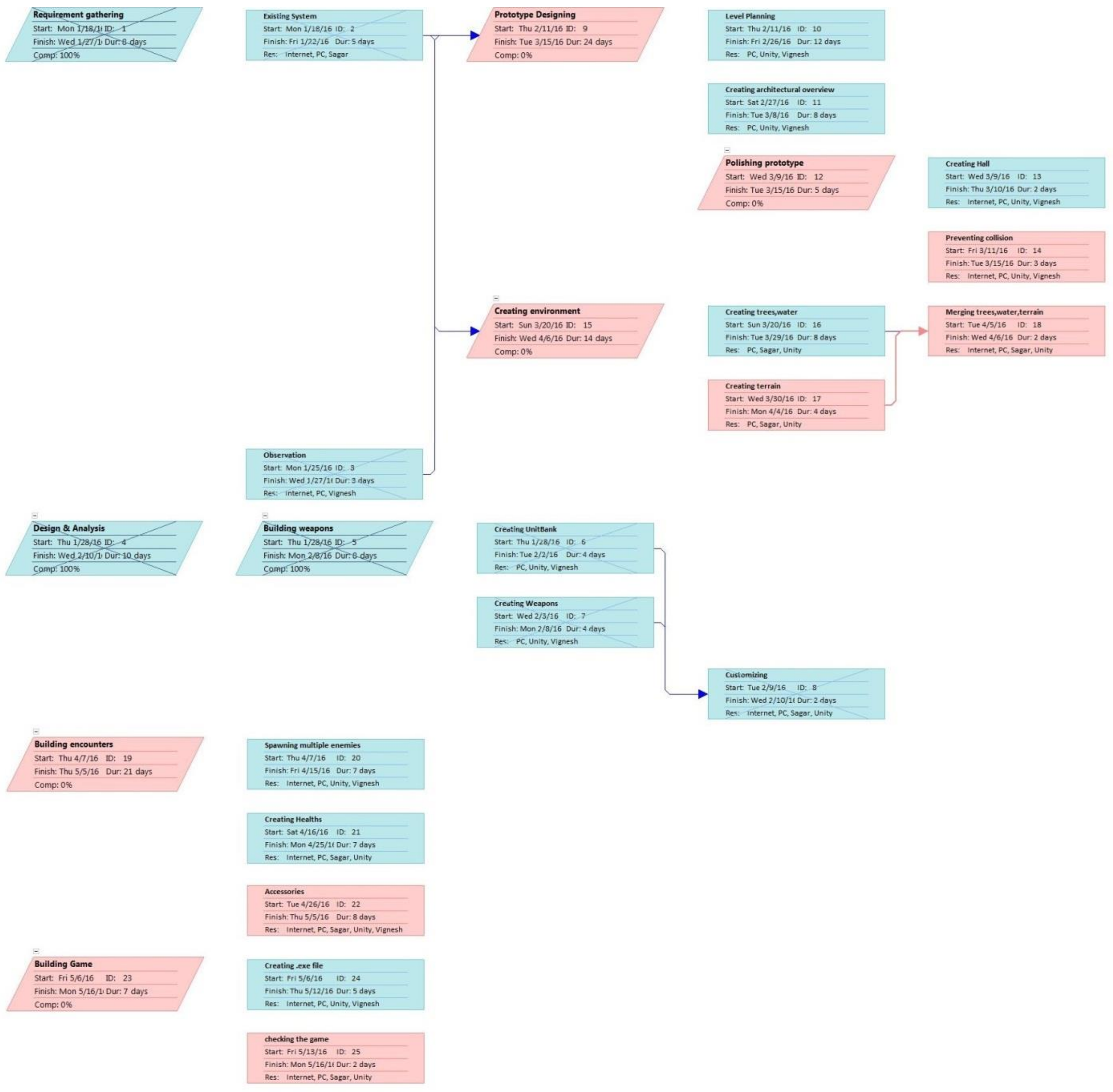
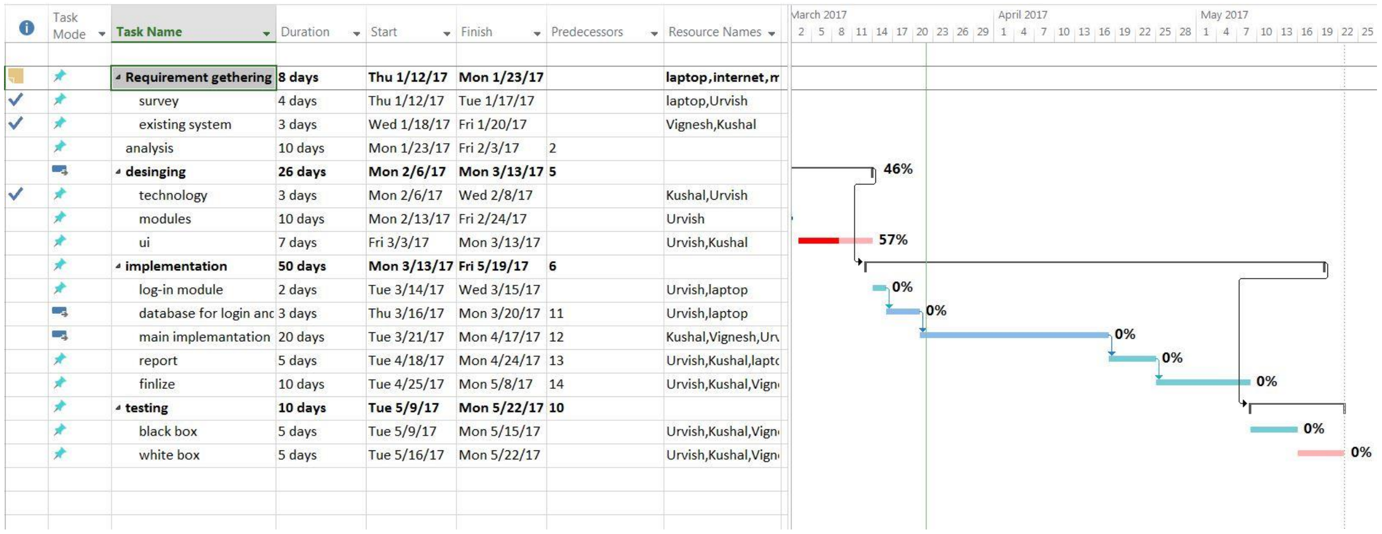

Requirement gathering
Start: Mon 1/18/16 ID: 1
Finish: Wed 1/27/16 Dur: 8 days
Comp: 100%
Existing System
Start: Mon 1/18/16 ID: 2
Finish: Fri 1/22/16 Dur: 5 days
Res: Internet, PC, Sagar
Prototype Designing
Start: Thu 2/11/16 ID: 9
Finish: Tue 3/15/16 Dur: 24 days
Comp: 0%
Level Planning
Start: Thu 2/11/16 ID: 10
Finish: Fri 2/26/16 Dur: 12 days
Res: PC, Unity, Vignesh
Creating architectural overview
Start: Sat 2/27/16 ID: 11
Finish: Tue 3/8/16 Dur: 8 days
Res: PC, Unity, Vignesh
Polishing prototype
Start: Wed 3/9/16 ID: 12
Finish: Tue 3/15/16 Dur: 5 days
Comp: 0%
Creating Hall
Start: Wed 3/9/16 ID: 13
Finish: Thu 3/10/16 Dur: 2 days
Res: Internet, PC, Unity, Vignesh
Preventing collision
Start: Fri 3/11/16 ID: 14
Finish: Tue 3/15/16 Dur: 3 days
Res: Internet, PC, Unity, Vignesh
Creating environment
Start: Sun 3/20/16 ID: 15
Finish: Wed 4/6/16 Dur: 14 days
Comp: 0%
Creating trees,water
Start: Sun 3/20/16 ID: 16
Finish: Tue 3/29/16 Dur: 8 days
Res: PC, Sagar, Unity
Merging trees,water,terrain
Start: Tue 4/5/16 ID: 18
Finish: Wed 4/6/16 Dur: 2 days
Res: Internet, PC, Sagar, Unity
Creating terrain
Start: Wed 3/30/16 ID: 17
Finish: Mon 4/4/16 Dur: 4 days
Res: PC, Sagar, Unity
Observation
Start: Mon 1/25/16 ID: 3
Finish: Wed 1/27/16 Dur: 3 days
Res: Internet, PC, Vignesh
Design & Analysis
Start: Thu 1/28/16 ID: 4
Finish: Wed 2/10/16 Dur: 10 days
Comp: 100%
Building weapons
Start: Thu 1/28/16 ID: 5
Finish: Mon 2/8/16 Dur: 8 days
Comp: 100%
Creating UnitBank
Start: Thu 1/28/16 ID: 6
Finish: Tue 2/2/16 Dur: 4 days
Res: PC, Unity, Vignesh
Creating Weapons
Start: Wed 2/3/16 ID: 7
Finish: Mon 2/8/16 Dur: 4 days
Res: PC, Unity, Vignesh
Customizing
Start: Tue 2/9/16 ID: 8
Finish: Wed 2/10/16 Dur: 2 days
Res: Internet, PC, Sagar, Unity
Building encounters
Start: Thu 4/7/16 ID: 19
Finish: Thu 5/5/16 Dur: 21 days
Comp: 0%
Spawning multiple enemies
Start: Thu 4/7/16 ID: 20
Finish: Fri 4/15/16 Dur: 7 days
Res: Internet, PC, Unity, Vignesh
Creating Healths
Start: Sat 4/16/16 ID: 21
Finish: Mon 4/25/16 Dur: 7 days
Res: Internet, PC, Sagar, Unity
Accessories
Start: Tue 4/26/16 ID: 22
Finish: Thu 5/5/16 Dur: 8 days
Res: Internet, PC, Sagar, Unity, Vignesh
Building Game
Start: Fri 5/6/16 ID: 23
Finish: Mon 5/16/16 Dur: 7 days
Comp: 0%
Creating .exe file
Start: Fri 5/6/16 ID: 24
Finish: Thu 5/12/16 Dur: 5 days
Res: Internet, PC, Unity, Vignesh
checking the game
Start: Fri 5/13/16 ID: 25
Finish: Mon 5/16/16 Dur: 2 days
Res: Internet, PC, Sagar, Unity

FIG 5.2 Task Network

5.3 GANTT CHART



	Task Mode	Task Name	Duration	Start	Finish	Predecessors	Resource Names
		Requirement gethering	8 days	Thu 1/12/17	Mon 1/23/17		laptop,internet,m
✓		survey	4 days	Thu 1/12/17	Tue 1/17/17		laptop,Urvish
✓		existing system	3 days	Wed 1/18/17	Fri 1/20/17		Vignesh,Kushal
		analysis	10 days	Mon 1/23/17	Fri 2/3/17	2	
		desinging	26 days	Mon 2/6/17	Mon 3/13/17	5	
✓		technology	3 days	Mon 2/6/17	Wed 2/8/17		Kushal,Urvish
		modules	10 days	Mon 2/13/17	Fri 2/24/17		Urvish
		ui	7 days	Fri 3/3/17	Mon 3/13/17		Urvish,Kushal
		implementation	50 days	Mon 3/13/17	Fri 5/19/17	6	
		log-in module	2 days	Tue 3/14/17	Wed 3/15/17		Urvish,laptop
		database for login anc	3 days	Thu 3/16/17	Mon 3/20/17	11	Urvish,laptop
		main implemantation	20 days	Tue 3/21/17	Mon 4/17/17	12	Kushal,Vignesh,Urv
		report	5 days	Tue 4/18/17	Mon 4/24/17	13	Urvish,Kushal,laptc
		finlize	10 days	Tue 4/25/17	Mon 5/8/17	14	Urvish,Kushal,Vign
		testing	10 days	Tue 5/9/17	Mon 5/22/17	10	
		black box	5 days	Tue 5/9/17	Mon 5/15/17		Urvish,Kushal,Vign
		white box	5 days	Tue 5/16/17	Mon 5/22/17		Urvish,Kushal,Vign

FIG 5.3 GANTT CHART

6 PROJECT RESOURCE

6.1 PEOPLE

PEOPLE	TASK	ROLE
Mr. Vignesh Patel	designing, backend, database connectivity, documentation, frontend.	Developer
Mr. Urvish Rana	designing, backend, database connectivity, testing, frontend.	Developer
Mr. Kushal Reshamdalal	database connectivity, testing, frontend.	Developer

6.2HARDWARE AND SOFTWARE

- Minimum requirements

Os	Windows 7.0 and above
Ram	512mb
Storage	1gb

7. SYSTEM DESIGN

7.1. SYSTEM APPLICATION DESIGN

7.1.1. Method Pseudo code

```
import cv2
import numpy as np
import sqlite3

faceDetect = cv2.CascadeClassifier('haarcascade_frontalface_default.xml');
cam = cv2.VideoCapture(0);
rec = cv2.createLBPHFaceRecognizer();
rec.load("recognizer\\training.yml")
atm1=0
def getProfile(id):
    conn = sqlite3.connect("FaceBase.db")
    cmd = "SELECT * FROM People WHERE ID="+str(id)

    cursor = conn.execute(cmd)

    profile = None
    for row in cursor:
        profile=row
    conn.close()
    return profile

procedure= """
asdfgh
```

```
fdsgh
adgfshnj
dgfsdjf
"""

def setAt(ids):
    conn = sqlite3.connect("FaceBase.db")
    cursor = conn.cursor()
    for id in ids:
        cmd1= "SELECT att FROM People WHERE id="+str(id)
        cursor1 = conn.execute(cmd1)
        print cmd1

    for row in cursor1:
        atm1=row[0]
        print atm1

    for id in ids:
        atm1=atm1+1
        cmd="UPDATE People SET att="+str(atm1)+" WHERE id="+str(id)
        cursor.execute(cmd)
        print("hello got ",id,cmd);
    conn.commit()

    conn.close()

id=0
flag=set()
```

```
font =
cv2.cv.InitFont(cv2.cv.CV_FONT_HERSHEY_COMPLEX_SMALL,2,1,0,2)
while(True):
    ret,img = cam.read();
    gray = cv2.cvtColor(img,cv2.COLOR_BGR2GRAY)
    faces = faceDetect.detectMultiScale(gray,1.3,5);
    for(x,y,w,h) in faces:
        cv2.rectangle(img,(x,y),(x+w,y+h),(255,0,0),2)
        id,conf = rec.predict(gray[y:y+h,x:x+w])
        profile = getProfile(id)
        if(profile!=None):

            cv2.cv.PutText(cv2.cv.fromarray(img),str(profile[1]),(x,y+h+30),font,255)
            cv2.cv.PutText(cv2.cv.fromarray(img),str(profile[2]),(x,y+h+60),font,255)
            cv2.cv.PutText(cv2.cv.fromarray(img),str(profile[3]),(x,y+h+90),font,255)

cv2.cv.PutText(cv2.cv.fromarray(img),str(profile[4]),(x,y+h+120),font,255)
            print("id",id)
            flag.add(id)
    cv2.imshow("Face",img);
    if(cv2.waitKey(1) == ord('q')):
        break;

setAt(flag)



cam.release()
cv2.destroyAllWindows()
```

7.1.2. Interface

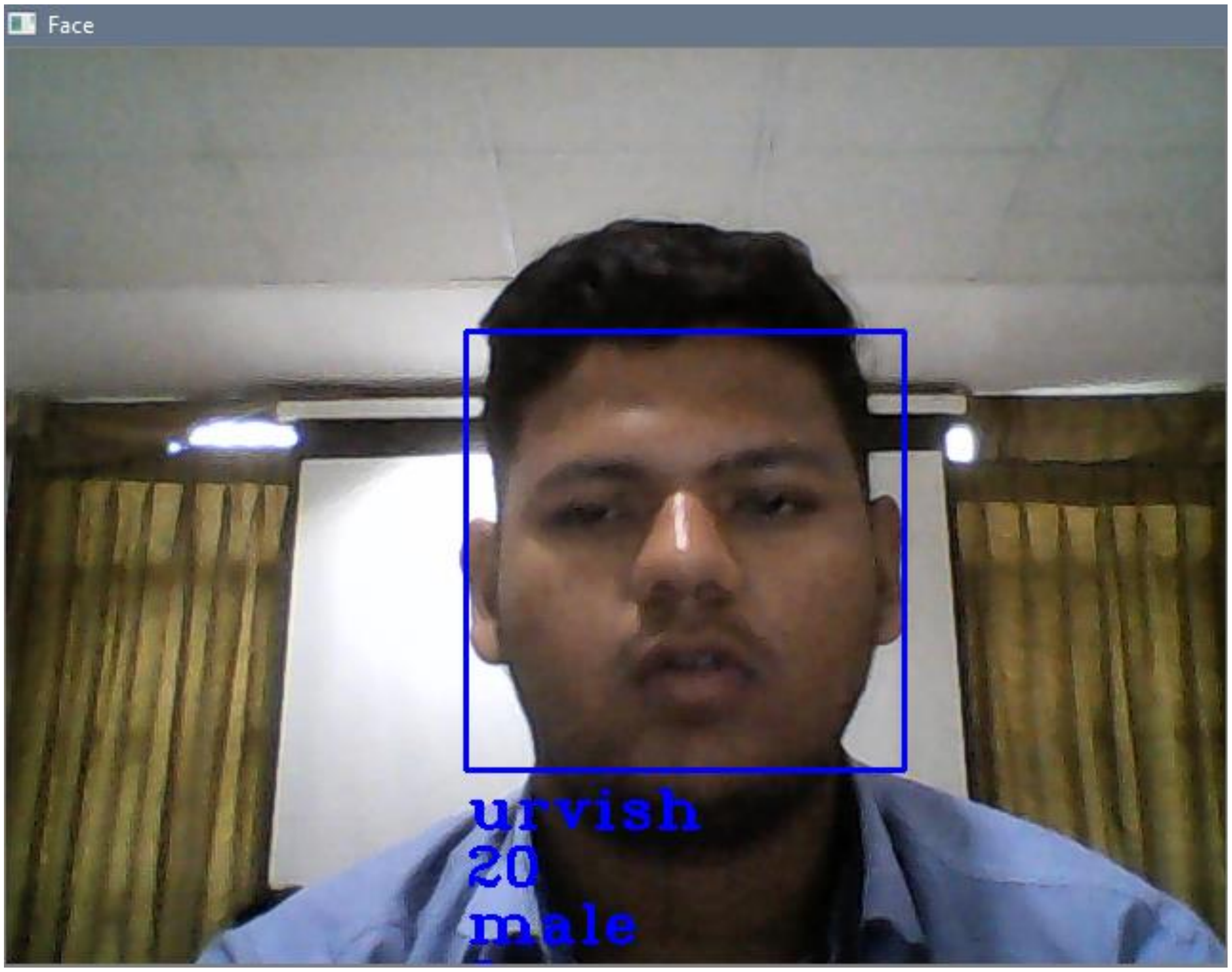
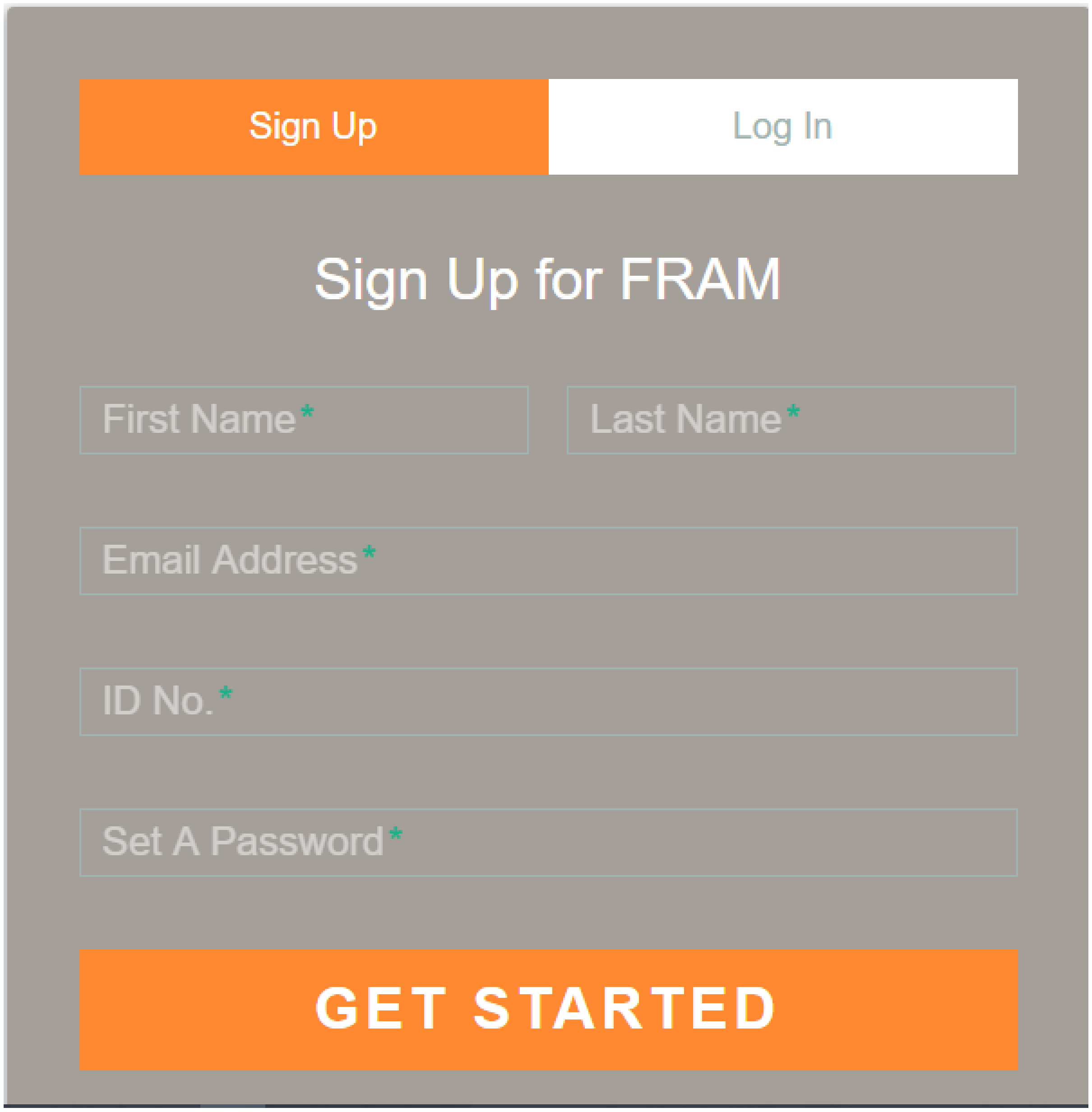


FIG 7.1 START UP



Sign Up for FRAM

First Name*

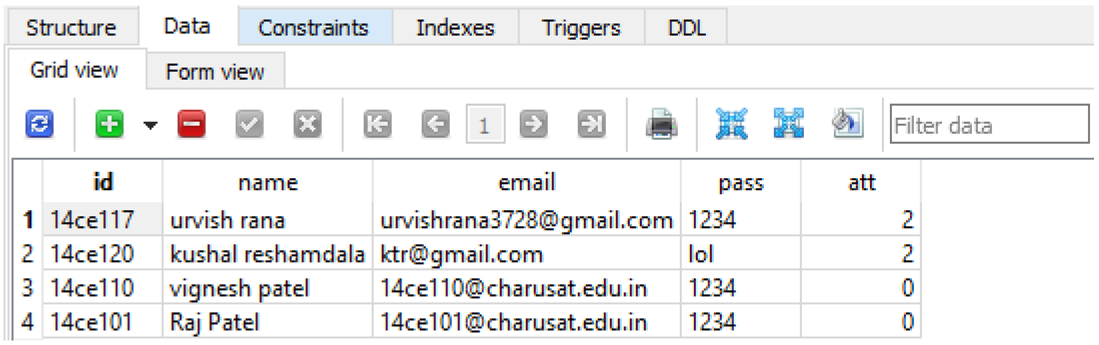
Last Name*

Email Address*

ID No.*

Set A Password*

FIG 7.2 LOGIN PAGE



	id	name	email	pass	att
1	14ce117	urvish rana	urvishrana3728@gmail.com	1234	2
2	14ce120	kushal reshamdala	ktr@gmail.com	lol	2
3	14ce110	vignesh patel	14ce110@charusat.edu.in	1234	0
4	14ce101	Raj Patel	14ce101@charusat.edu.in	1234	0

FIG 7.3 DATATBASE

8. TESTING

8.1 TESTING PLAN

- We are doing two types of testing here black box and white box.

8.2 TEST CASE DESIGN

- **Register function test suit:**

Test Suites No: 1
Test Suite Detail: User wants to register his/her face and ID.

Test Case ID	Function Name	Test Case (condition)	Expected Results	Actual Result	Pass/Fail
1	registernew	User clicks the 'register' button	New entry should be done	New entry actually gets done.	Pass

- **Student scans his/her face.**

Test Suites No: 1

Test Suite Detail: Student scans his/her face to get his/her attendance filled.

Test Case ID	Function Name	Test Case (condition)	Expected Results	Actual Result	Pass/Fail
1	entry	Student presents his/her face	Attendance should be filled	Attendance gets filled	Pass

Test Case ID	Function Name	Test Case (condition)	Expected Results	Actual Result	Pass/Fail
2	entry	Student presents his/her face without registering	Attendance should not be filled	Attendance doesn't get filled	Pass

- Student wants to see the attendance.

Test Suites No: 1

Test Suite Detail: Student wants to see his/her attendance of so far.

Test Case ID	Function Name	Test Case (condition)	Expected Results	Actual Result	Pass/Fail
1	checkattendance	Student tries to see his/her attendance	Attendance upto this far should be shown	Attendance upto this far is shown	Pass

- Faculty wants to see the attendance.

Test Suites No: 1
Test Suite Detail: Faculty wants to get the attendance of all students along with ID and their other data this far.

Test Case ID	Function Name	Test Case (condition)	Expected Results	Actual Result	Pass/Fail
1	attendanceid	Faculty wants to get the attendance of all students along with their other data	Attendance along with other data should be shown.	Attendance along with other data is shown.	Pass

Test Suites No: 2
Test Suite Detail: Faculty generates the report of each and every student.

Test Case ID	**Function Name**	**Test Case (condition)**	**Expected Results**	**Actual Result**	**Pass/Fail**
1	generatereport	Faculty wants to generate the report	Report should be generated	Report gets generated	Pass

9. LIMITATION AND FUTURE ENHANCEMENTS

9.1 LIMITATIONS

- There might be a possibility that sometimes the face isn't get scanned.
- Also, sometimes some unusual change in face like specs, beard shave might not detect the face.

9.2 FUTURE ENHANCEMENTS

- We will try to improvise the face detection better.
- Application will catch and recognize more number of faces once.

10. CONCLUSION AND DISCUSSION

10.1 Self-Analysis of Project Viabilities

- From this project we studied about face detection and recognition

10.2 Problem Encountered and Possible Solutions

- OpenCV was not able to recognize the face of student earlier.
    - **Solution:** specific changes to recognizer set has been done.
- Database connectivity was not done profoundly.
    - **Solution:** specific modules are now added and appropriate modules are replaced.

10.3 Summary of Project work

- This project is about web app development. This app is developed for student purpose. This app will help to fill the attendance easier just by showing our face.

BIBLIOGRAPHY

WEBSITE	**TOPIC**
[1].www.opencv.org	**image processing library**
[2].www.getcocomoonline.com	**cocomo model**
[3]. www.en.wikipedia.com	**java servlet**