

1.INTRODUCTION

PROJECT OVERVIEW

Internet is gaining more importance day after day in all life aspects, especially in business and marketing due to the amazing increase in internet users around the world with an estimate of 2.4 billion users in 2012, when comparing this number to the number of internet users in 2000, a growth of 566% can be noticed. This is making internet the fastest media of all time in both growth rate and number of users (internet world stats, 2012). Based on a comprehensive study in 2011, the number of advertisements circulated over the net was more than 3.5 million daily. Internet became one of the most efficient ways to conduct business. In developed and well-developed countries, internet proved to be of much help for local enterprises where it provides great potential for such enterprises to compete worldwide. The main idea is to implement an online system for managing the internet customers and complaint system for customers for raising complaints on the issues related to ISP provider and provide best customer care service for users using this application. There are many Internet security providers in a country that will provide internet services for users on different packages. Basically ISP works on three connections, Dial Up using telephone service, Broad band and wireless connections

PURPOSE

The main idea is to implement an online system for managing the internet customers and complaint system for customers for raising complaints on the issues related to ISP provider and provide best customer care service for users using this application. By adding more entries to the data base store, the application can respond to more number of queries from the customers. The importance is given on giving correct reply to the input queries.

2.LITERATURE SURVEY

a. EXISTING PROBLEM

TITLE: Design of Internet Provider E-CRM System on CV. Ahyein Pratama Mandiri

Air Joman

AUTHOR: Irfan Hakim Nasution¹

Currently conducts its business activities by waiting for customers to come to the office or customers providing information about places, privileges and products to other potential customers. Inadequate reporting of customer complaints makes it difficult to know the level of customer satisfaction, causing the opportunity to get potential customers to the maximum not to be fulfilled. Dissemination of information that is not neat makes it difficult for customers to know the latest information, thus influencing customers to switch to other providers. Therefore CV. Ahyein Pratama Mandiri requires CRM (Customer Relationship Management) which is applied to the information system, where this system can facilitate CV. Ahyein Pratama Mandiri in managing services for customer satisfaction. In carrying out this research, the researcher uses a qualitative research method which is a method that discusses the problem by describing, interpreting and writing down a situation or event that will be analyzed and then draw a general conclusion from the problems discussed. As a result, the application of CRM on CV. Ahyein Pratama Mandiri can make it easier for companies to provide information to consumers, provide convenience in ordering products that can be done anytime and anywhere, and increase customer loyalty. Customer relationship management (CRM) is a special type of management that addresses theories for managing the relationship between a business and its customers and for improving customer relationships to achieve growth. Healthy business development . CRMs are designed to improve profits, sales, and customer satisfaction by helping businesses of all kinds accurately identify their customers, attract more customers, and maintain customer loyalty. A business strategy that includes customized software and services. CRM is a sales, marketing and service strategy that uses information technology through a customer-centric corporate

philosophy and culture to make business processes more efficient . Information is data processed in a form that is significant to the recipient and has real value that can be perceived in making decisions now or in the future

i. **TITLE:** The impact of the magnitude of service failure and complaint handling

on satisfaction and brand credibility in the banking industry

AUTHOR: Ghazal Shams¹ · Mohsin Abdur Rehman

The present research aims to investigate the effects of service failure and complaint handling on customer satisfaction with complaint handling which consequently impacts overall satisfaction and brand credibility. To examine the objectives of the present research, the authors deployed a sample of 384 respondents in Persian banks within Iran. Structural equation modeling has been used to analyze the data. The findings suggest that the magnitude of service failure negatively affects customer satisfaction with complaint handling. Complaint handling positively affects customer satisfaction with complaint handling. In addition, the results suggest that customer satisfaction with complaint handling positively influences brand credibility and overall satisfaction. Finally, overall satisfaction positively impacts brand credibility. The results revealed that if the complaint handling occurs instantly at the right time, it would have been a positive influence on customer satisfaction and ultimately develop brand credibility. Therefore, banks can adopt customer relationship management systems and processes which enable quick responses to customer complaints. Bank managers could find the results of the present study useful and beneficial in developing complaint handling efforts and expanding appropriate service recovery and brand credibility strategies. stakeholders involved in the service process. Service recovery follows service failure and helps business to recover the damaged service experience. An effective service recovery strategy transforms the existing service processes to avoid the recurrence of a service failure. Service sector adopts more technological advancements to tailor customers dynamic expectations. However, business service processes are facing numerous challenges to sustaining brand value.

- ii. **TITLE:** Exploring the influence of the human factor on customer satisfaction in
call centres

AUTHOR: Dorina Chicua,*, Maria del Mar Pàmies

The aim of this study is to explore the human or employee-related factors that shape customer satisfaction in the context of call centres. The literature review draws from a range of disperse disciplines including Service Quality, Human Resource Management and Marketing. The empirical study explores the different variables identified to obtain a nuanced analysis of the employee-related paths that lead to customer satisfaction in call centres. The study employs data from 109 call centres and utilises PLS for our exploratory purposes. Call centre managers should note that investing in HR practices will pay off in terms of improving the elusive phenomenon of customer satisfaction within call centres. The call centre industry is a peculiar service industry, in as much as it is almost entirely based on a voice-to-voice encounter between the employee and the customer, on opposite ends of the telephone line. In general, customers are less satisfied with the service they receive from call centres than from the more traditional brick n' mortar, or face to face service encounters. In call centres, employees (call centre operators) are the main connection between the organization and the customer. Employees are often required to undertake many different tasks at the same time. They are expected to display ambidextrous behaviour, being able to accomplish managerial requirements such as: maintaining service quality, including attentiveness, perceptiveness, responsiveness and assurance, satisfy customers, solve problems attend a large number of calls in a short time while ensuring first call resolution.

iii. **TITLE:**Analyzing and Implementing a System For Reporting, Follow Up
and

Resolving of Complaints

AUTHOR: Angham AL Abbas, Khadeeja Alzayer

In every aspect of life either it is personal or professional we use internet. It makes life easier, and overcomes unsatisfactory and unacceptable services or issues on various fields. We can use online complaint management system which is considered as an essential part of quality services. Complaints and compliments are valuable sources of information that organizations can use it to improve program delivery and service. “A web system for reporting, follow-up and resolving of complaints” is a web application analyzed and developed for managing various complaints in any place such as universities, hospitals shopping centers, damaged roads, unwanted load Shedding or sewerage proble.... etc. This work aims to make complaints easier to be reported, coordinated, monitored, tracked and resolved, and to provide governments with effective tool to keep records of complaint data, to use them for identifying problem areas and to improve services. Today's development cycles for web-applications such as Portals and Marketplaces are short, and getting shorter with continuous improvements and enhancements as new requirements and features become apparent. Therefore, developing “Web Services” using the “Service-Oriented Architecture” paradigm is a widely accepted concept. On the other side, most of user's complaints are apparent when a system has inappropriate communication between the organizations, their employees and customers (Citizens). Poor communication can result in poor services or products being provided by the organization or Government. Whilst concentrating on

the topic of complaint handling, organizations can achieve an efficient success factor by increasing their user satisfaction and their loyalty. Therefore each organization needs to develop its internal and external communication towards its staff and customers to achieve success. Although appropriate communication can reduce user dissatisfaction; it cannot eliminate complaint

iv. **TITLE:** Leveraging unstructured call log data for customer churn prediction

AUTHOR: Nhi N.Y. Vo ^a , Shaowu Liu ^b , Xitong L

Customer retention is important in the financial services industry. Machine learning has been incorporated into customer data analytics to predict client churn risks. Despite its success, existing approaches primarily use only structured data, e.g., demographics and account history. Data mining with unstructured data, e.g., customer interaction, can reveal more insights, which has not been adequately leveraged. In this research, we propose a customer churn prediction model utilizing the unstructured data, which is the spoken contents in phone communication. We collected a large-scale call center dataset with two million calls from more than two hundred thousand customers and conducted extensive experiments. The results show that our model can accurately predict the client churn risks and generate meaningful insights using interpretable machine learning with personality traits and customer segments. We discuss how these insights can help managers develop retention strategies customized for different customer segments. Customer relationship management (CRM) has always been a core business function for any company. Among the components of CRM, increasing customer engagement and loyalty is one of the most challenging tasks. Although customer acquisition and retention are both important, prior research has showed that acquiring a new customer is typically five times more expensive than retaining an existing customer. Because of the high cost of customer acquisition, established businesses focus more on customer retention instead of acquisition. In customer retention, predicting customer churn risk is an important task. Each single percentage increase in customer churn prediction accuracy could potentially lead to a substantial revenue saving.

REFERENCES

1. Irfan Hakim Nasution¹, Design of Internet Provider E-CRM System on CV. Ahyein Pratama Mandiri Air Joman, 2022.
2. Ghazal Shams¹ · Mohsin Abdur Rehman, The impact of the magnitude of service failure and complaint handling on satisfaction and brand credibility in the banking industry, 2020.
3. Dorina Chicua^{*}, Maria del Mar Pàmies, Exploring the influence of the human factor on customer satisfaction in call centres, 2018.
4. Angham AL Abbas, Khadeeja Alzayer, Analyzing and Implementing a System For Reporting, Follow Up and Resolving of Complaints, 2019.
5. Nhi N.Y. Vo^a , Shaowu Liu^b , Xitong L, Leveraging unstructured call log data for customer churn prediction, 2020.

PROBLEM STATEMENT DEFINITION

Call center management is the way in which organizations manage the daily operations of call center, including forecasting, scheduling, employee training, reporting and all customer interactions. Call center management can be modernized with workforce optimization (WFO) solutions. A call center management system refers to a software solution that helps improve customer interactions, service levels, and user experience. Simply put, it's a modern way of managing the day-to-day call center operations-training, forecasting, reporting, scheduling, and many others. Callers should be able to leave messages in cases where all agents are preoccupied and no one can take the call. A call center function through operators, known as agents or sometimes customer representatives, and computerized telephony (CTI).


3.IDEATION AND PROPOSED SOLUTION

3.1 EMPATHY MAP CANVAS



3.2 IDEATION & BRAINSTORMING

Template



Brainstorm & idea prioritization

Use this template in your own brainstorming sessions so your team can unleash their imagination and start shaping concepts even if you're not sitting in the same room.

- 10 minutes to prepare
- 1 hour to collaborate
- 2-8 people recommended

[Share template feedback](#)

➔

Before you collaborate

A little bit of preparation goes a long way with this session. Here's what you need to do to get going.

10 minutes

A

Team onboarding

Define who should participate in the session and send an invite. Share relevant information to pre-work ahead.

B

Get the goal

Think about the problem you'll be focusing on solving in the brainstorming session.

C

Learn how to use the facilitator's tools

Use the Facilitator's Superpowers to run a happy and productive session.

[Open article](#)

1

Define your problem statement

What problem are you trying to solve? Frame your problem as a How Might We statement. This will be the focus of your brainstorm.

5 minutes

problem

How might we can solve the issue *caused* by the customer?

Key rules of brainstorming

To run an smooth and productive session

- Stay in topic
- Encourage wild ideas
- Defer judgment
- Listen to others
- Go for volume
- If possible, be visual



Need some inspiration?

Get a full week's worth of the template to kickstart your work.

[Open example](#)

2

Brainstorm

Write down any ideas that come to mind that address your problem statement.

10 minutes

TP

You can refer to sticky notes and 16 the panel (which is working) to start drawing!

FLAVOR & S

Light Biscuits	Protein Bars in Cakes	Protein Bars in Cakes
Custom Bars	Protein Bars in Cakes	Protein Bars in Cakes
Protein Bars in Cakes	Protein Bars in Cakes	Protein Bars in Cakes

DRISCHVEN

Custom Biscuits	Protein Bars in Cakes	Protein Bars in Cakes
Protein Bars in Cakes	Protein Bars in Cakes	Protein Bars in Cakes
Protein Bars in Cakes	Protein Bars in Cakes	Protein Bars in Cakes

RAMBERGSHEND

Protein Bars in Cakes	Protein Bars in Cakes	Protein Bars in Cakes
Protein Bars in Cakes	Protein Bars in Cakes	Protein Bars in Cakes
Protein Bars in Cakes	Protein Bars in Cakes	Protein Bars in Cakes

GURUKAUN

Protein Bars in Cakes	Protein Bars in Cakes	Protein Bars in Cakes
Protein Bars in Cakes	Protein Bars in Cakes	Protein Bars in Cakes
Protein Bars in Cakes	Protein Bars in Cakes	Protein Bars in Cakes

3

Group ideas

Take turns sharing your ideas while clustering similar or related notes as you go. In the last 10 minutes, give each cluster a sentence-like label. If a cluster is bigger than six sticky notes, try and see if you can break it up into smaller sub-groups.

20 minutes

**CUSTOMER**

Selection for customer needs	Marketing Customer	Providing Customer	Providing service on time
Selection for customer	Over with product quality	Customer Service	

TP

When that problem happens to sticky notes to start to make the system, improve, and improve the system. It's a process within your team.

**CHATBOX**

Chat box	Protein Bars in Cakes
----------	-----------------------

**FEEDBACKS**

Customer Feedback	Self Feedback	Rating for rating
-------------------	---------------	-------------------

**INFORMATION**

Light Biscuits	Light Biscuits in Cakes
----------------	-------------------------

**SECURITY**

Security	Customer Service
----------	------------------

**SERVICES**

Providing service on time	Protein Bars in Cakes	Marketing Customer	Providing Customer	Providing service on time	Customer Service
---------------------------	-----------------------	--------------------	--------------------	---------------------------	------------------



3.3 PROPOSED SOLUTION

This proposed system provides an online way of solving the problems faced by the public by saving time and eradicate corruption, and The ability of providing many of the reports on the system, and add to Facilitate the process of submitting a complaint. In this project we can design web application to analyze the complaints and to provide automatic forwarding system of user's complaints. User is easily known about status of complaints. If the action can't be taken properly means, send to higher authorities. The proposed system is supposed to handle as more number of customers as possible in any particular time. The mail service is also provided to have a communication between the admin and the users. The user queries should be periodically referred and the solution should be provided quickly.

3.4 PROBLEM SOLUTION FIT

The existing system is handled manually. The system has a formatted call centre management for customers in paper work like files and document format. The customers are waiting a call to taken by the call centre employee pick their calls. So any urgent work we didn't get any important response from the call centre. So the proposed system the manager will look after it and then he will take care about the customer's problems. After that the manager will enquire and allocate the problem to the specified person in that department. The person will enquire the problem and then rectifies it.

4.REQUIREMENT ANALYSIS

4.1 FUNCTIONAL REQUIREMENT

Admin

- Login
 - Add Employees
 - Add Customers
 - View Details
1. Employees
 2. Customers
 3. Feed Back

Employee

- Login
- View Complaints
- Send Notification (Through Email)
- View Feed Back

Customer

- Login

- Post Complaint
- View Notification
- Feed Back

MODULE DESCRIPTION

Admin

- Login

In this module, the admin can login in the system using his/her username and password.

- Add Employees

In this module, the admin can add the employee information like employee name, id, phone number, mail id, location etc.

- Add Customers

In this module, the admin can add the customer information like customer name, id, phone number, mail id, location etc.

- View Details

In this module, the admin can view the employee details, customer details and feedback details.

Employee

- Login

In this module, the employee can login in the system using his/her username and password.

- View Complaints

In this module, the employee can view the customer complaint using this application.

- Send Notification (Through Email)

The employee can sent the notification to the user through the email for update status of the complaint using this system.

- View Feed Back

In this module, the employee can view the user feedback.

Customer

- Login

In this module, the customer can login in the system using his/her username and password.

- Post Complaint

In this module, the customer can post the internet service related complaint to this system.

- View Notification

If the employee can update the status of the complaint, the user can get the automatic notification.

- Feed Back

In this module, the user can post the feedback of products or service.

a. NON FUNCTIONAL REQUIREMENTS

4.2 Non - Functional Requirements

Usability

The system shall allow the users to access the system with pc using web application. The system uses a web application as an interface. The system is user friendly which makes the system easy

Availability

The system is available 100% for the user and is used 24 hrs a day and 365 days a year. The system shall be operational 24 hours a day and 7 days a week.

Scalability

Scalability is the measure of a system's ability to increase or decrease in performance and cost in response to changes in application and system processing demands.

Security

A security requirement is a statement of needed security functionality that ensures one of many different security properties of software is being satisfied.

Performance

The information is refreshed depending upon whether some updates have occurred or not in the application. The system shall respond to the member in not less than two seconds from the time of the requestsubmittal. The system shall be allowed to take more time when doing large processing jobs. Responses to view information shall take no longer than 5 secondsto appear on the screen.

Reliability

The system has to be 100% reliable due to the importance of data and the damages that can be caused by incorrect or incomplete data. The system will run 7 days a week, 24 hours a day.

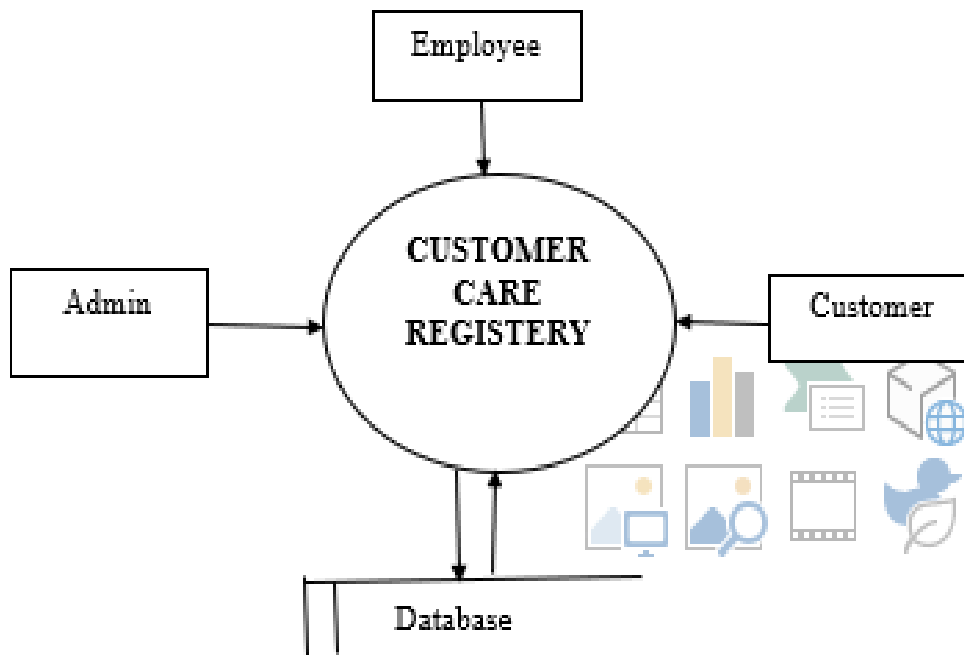
5. PROJECT DESIGN

5.1 DATA FLOW DIAGRAMS

A data-flow diagram is a visual representation of how data moves through a system or a process (usually an information system). The DFD additionally gives details about each entity's inputs and outputs as well as the process itself. A data-flow diagram lacks control flow, loops, and decision-making processes. Using a flowchart, certain operations depending on the data may be depicted.

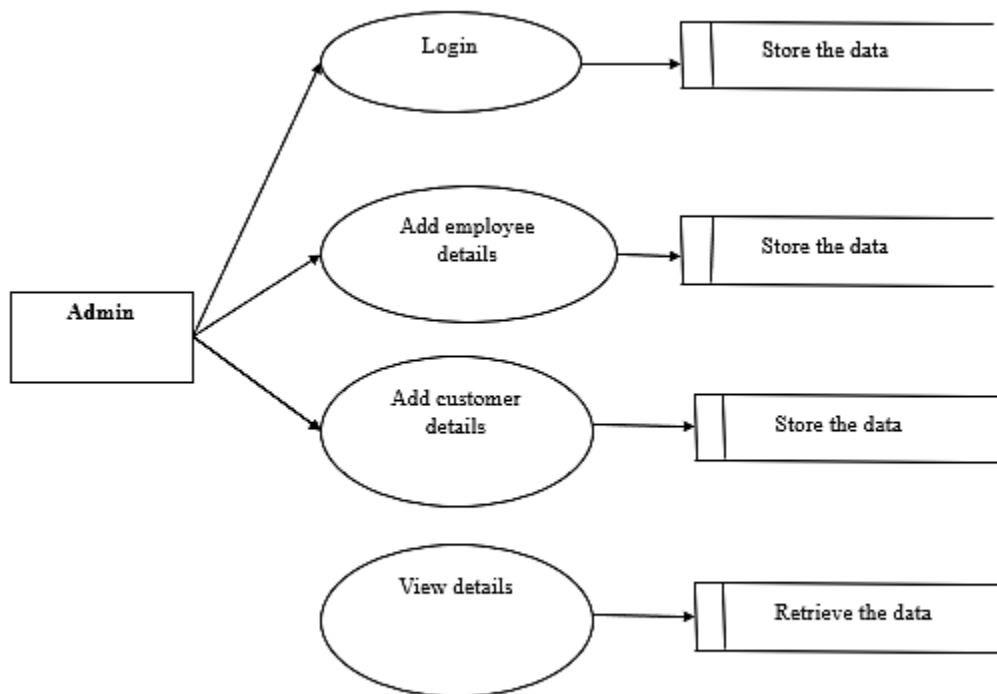
LEVEL 0

It is also known as a context diagram. It's designed to be an abstraction view, showing the system as a single process with its relationship to external entities. It represents the entire system as a single bubble with input and output data indicated by incoming/outgoing arrows.



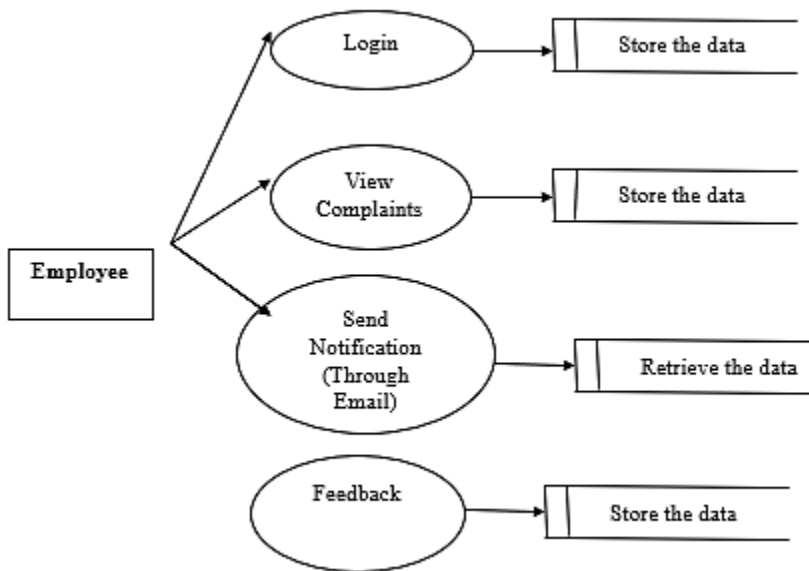
LEVEL 1

In 1-level DFD, the context diagram is decomposed into multiple bubbles/processes. In this level, we highlight the main functions of the system and breakdown the high-level process of 0-level DFD into sub processes.



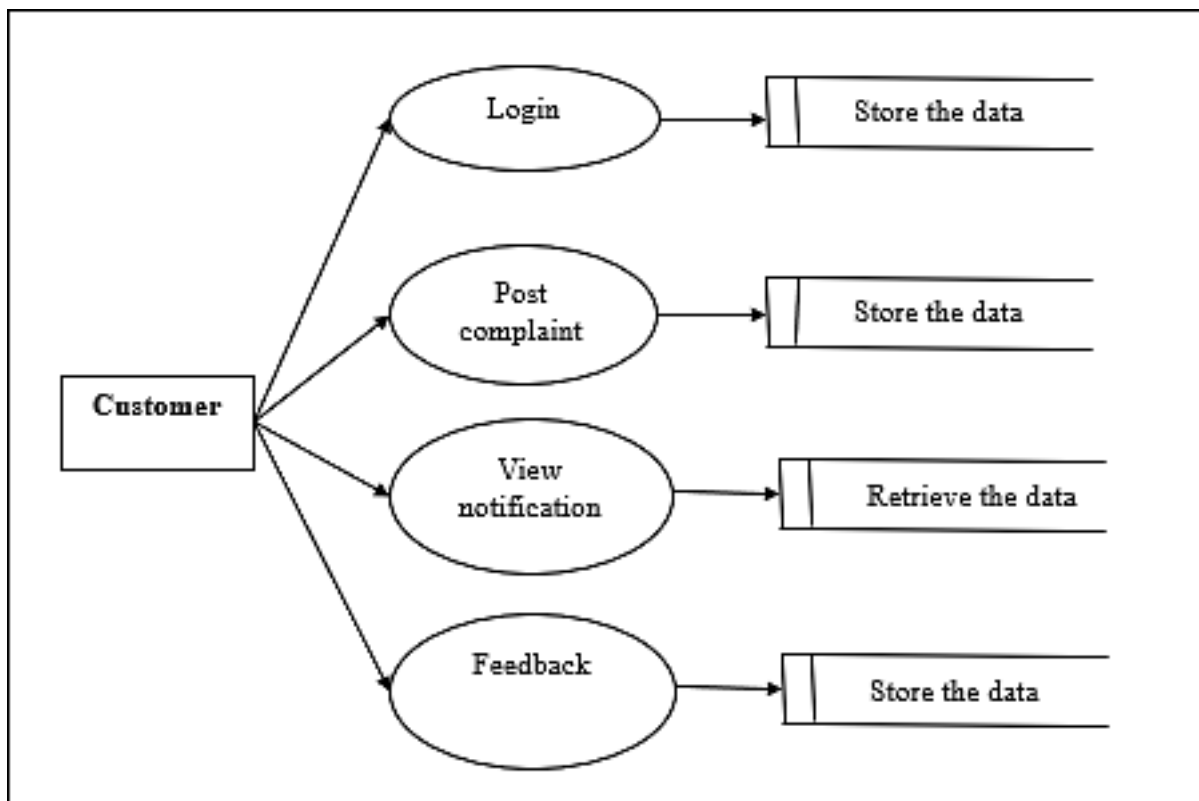
LEVEL 2

2-level DFD goes one step deeper into parts of 1-level DFD. It can be used to plan or record the specific/necessary detail about the system's functioning.

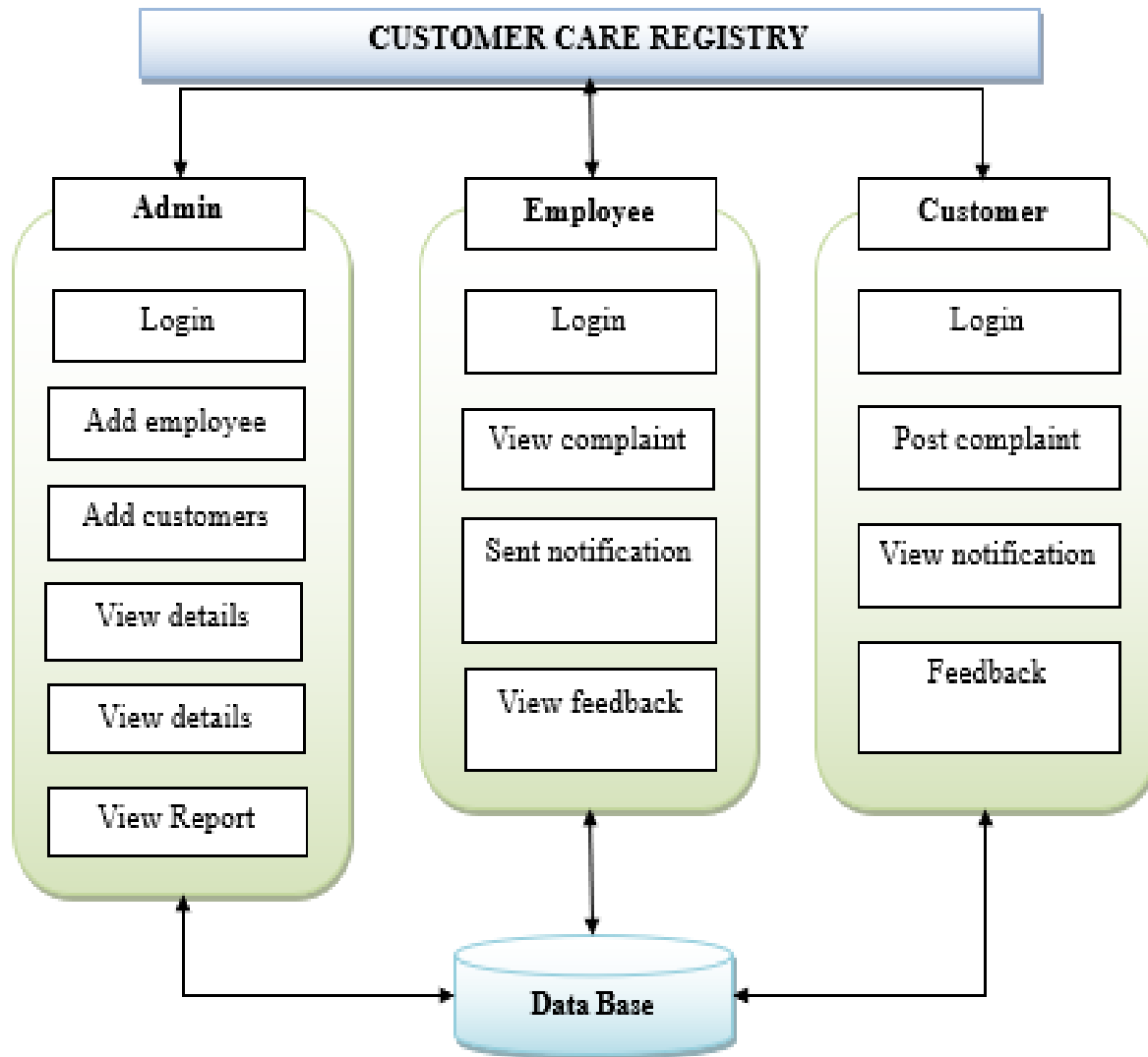


LEVEL 3

A data flow diagram (DFD) is a graphical representation of the flow of data through an information system. A DFD shows the flow of data from data sources and data stores to processes, and from processes to data stores and data sinks. DFDs are used for modelling and analyzing the flow of data in data processing systems, and are usually accompanied by a data dictionary, an entity-relationship model, and a number of process descriptions.



5.2 SOLUTION & TECHNICAL ARCHITECTURE



5.3 User Stories

User Type	Functional Requirement (Epic)	User Story Number	User Story / Task	Acceptance criteria	Priority	Release
Customer (Mobile user)	Registration	USN-1	As a customer, I can register for the application by entering my email, password, and confirming my password.	I can access my account / dashboard	High	Sprint-1
	login	USN-2	As a customer, I can login to the application by entering correct email and password.	I can access my account/dashboard.	High	Sprint-1
	Dashboard	USN-3	As a customer, I can see all the orders raised by me.	I get all the info needed in my dashboard.	Low	Sprint-2
	Order creation	USN-4	As a customer, I can place my order with the detailed description of my query	I can ask my query	Medium	Sprint-2
	Address Column	USN-5	As a customer, I can have conversations with the assigned agent and get my queries clarified	My queries are clarified.	High	Sprint-3
	Forgot password	USN-6	As a customer, I can reset my password by this option in case I forgot my old password.	I get access to my account again	Medium	Sprint-4
Agent (web user)	Order details	USN-7	As a Customer, I can see the current stats of order.	I get a better understanding	Medium	Sprint-4
	Login	USN-1	As an agent I can login to the application by entering Correct email and password.	I can access my account / dashboard.	High	Sprint-3
	Dashboard	USN-2	As an agent, I can see the order details assigned to me by admin.	I can see the tickets to which I could answer.	High	Sprint-3
	Address column	USN-3	As an agent, I get to have conversations with the customer and clear his/her doubts	I can clarify the issues.	High	Sprint-3
	Forgot password	USN-4	As an agent I can reset my password by this option in case I forgot my old password.	I get access to my account again.	Medium	Sprint-4

6.PROJECT PLANNING & SCHEDULING

6.1 SPRINT PLANNING & ESTIMATION

Project Tracker, Velocity & Burndown Chart: (4 Marks)

Sprint	Total Story Points	Duration	Sprint Start Date	Sprint End Date (Planned)	Story Points Completed (as on Planned End Date)	Sprint Release Date(Actual)
Sprint-1	20	6 Days	24 Oct 2022	29 Oct 2022		29 Oct 2022
Sprint-2	20	6 Days	31 Oct 2022	05 Nov 2022		05 Nov 2022
Sprint-3	20	6 Days	07 Nov 2022	12 Nov 2022		12 Nov 2022
Sprint-4	20	6 Days	14 Nov 2022	19 Nov 2022		19 Nov 2022

Velocity:

Imagine we have a 10-day sprint duration, and the velocity of the team is 20 (points per sprint). Let's calculate the team's average velocity (AV) per iteration unit (story points per day)

$$AV = \frac{\text{sprint duration}}{\text{velocity}} = \frac{20}{10} = 2$$

6.2 SPRINT DELIVERY SCHEDULE

Sprint	Functional Requirement (Epic)	User Story Number	User Story / Task	Story Points	Priority	Team Members
Sprint-1	User Panel	USN-1	The user will login into the website and go through the services available on the webpage	20	High	AJITHKUMAR K
Sprint-2	IBM DB2	USN-2	The role of the admin is to check out the database about the availability and have a track of all the things that the users are going to service	20	High	DHIVAKAR N
Sprint-3	E-mail Delivery	USN-3	The user can directly talk to the customer and receive e-mail services. Get the recommendations based on information provided by the user.	20	High	DHINESHKUMAR K
Sprint-4	final delivery	USN-4	Container of applications using docker kubernetes and deployment the application. Create the documentation and final submit the application	20	High	VETRIVEL T

6.3 REPORT FROM JIRA

▼ CCR Sprint 1 18 Nov – 2 Dec (2 issues)

0 0 0

Complete sprint

...

CCR-6 As a customer, I can login to the application by entering corre...

DONE▼

CCR-5 As a customer, I can register for the application by entering m...

DONE▼

+ Create issue

▼ CCR Sprint 2 18 Nov – 2 Dec (2 issues)

0 0 0

Complete sprint

...

CCR-3 As a customer, I can see all the orders raised by me.

DONE▼

CCR-4 As a customer, I can place my order with the detailed descripti...

DONE▼

▼ CCR Sprint 3 18 Nov – 2 Dec (4 issues)

0 0 0

Complete sprint

...

CCR-9 As an agent, I can see the order details assigned to me by admin

DONE▼

CCR-7 As a customer, I can have conversations with the assigned agen...

DONE▼

CCR-10 As an agent, I get to have conversations with the customer an...

DONE▼

CCR-8 As an agent I can login to the application by entering Correct e...

DONE▼

▼ CCR Sprint 4 18 Nov – 2 Dec (3 issues)

0 0 0

Complete sprint

...

CCR-11 As a customer, I can reset my password by this option incase I ...

DONE▼

CCR-12 As a Customer ,I can see the current stats of order.

DONE▼

CCR-13 As an agent I can reset my password by this option in case I fo...

DONE▼

7.CODING AND SOLUTIONING

7.1 Feature 1

```
def sendmsg(Mailid,message):  
    import smtplib  
    from email.mime.multipart import MIMEMultipart  
    from email.mime.text import MIMEText  
    from email.mime.base import MIMEBase  
    from email import encoders  
  
    fromaddr = "sampletest685@gmail.com"  
    toaddr = Mailid  
  
    # instance of MIMEMultipart  
    msg = MIMEMultipart()  
  
    # storing the senders email address  
    msg['From'] = fromaddr  
  
    # storing the receivers email address  
    msg['To'] = toaddr  
  
    # storing the subject  
    msg['Subject'] = "Alert"
```

```
# string to store the body of the mail
body = message

# attach the body with the msg instance
msg.attach(MIMEText(body, 'plain'))

# creates SMTP session
s = smtplib.SMTP('smtp.gmail.com', 587)

# start TLS for security
s.starttls()

# Authentication
s.login(fromaddr, "hneucvnontsuwgpj")

# Converts the Multipart msg into a string
text = msg.as_string()

# sending the mail
s.sendmail(fromaddr, toaddr, text)

# terminating the session
s.quit()

if __name__ == '__main__':
    app.run(debug=True, use_reloader=True)
```

7.2 Feature 2

```
1. def newcom() :  
  
    if request.method == 'POST':  
  
        name = request.form['name']  
  
        com = request.form['com']  
  
        uname = session['uname']  
  
  
  
        conn = ibm_db.connect(dsn, "", "")  
  
        pd_conn = ibm_db_dbi.Connection(conn)  
  
  
        selectQuery = "SELECT * FROM booktb"  
  
        dataframe = pandas.read_sql(selectQuery, pd_conn)  
  
  
        dataframe.to_sql('booktb', con=engine, if_exists='append')  
  
        data2 = engine.execute("SELECT * FROM booktb").fetchall()  
  
        count = 0  
  
  
        for item in data2:  
  
            count += 1
```

```
Bookingid = "COMID00" + str(count)
```

```
insertQuery = "INSERT INTO booktb VALUES ('" + Bookingid + "','" + uname + "','" + com +  
"',';')"
```

```
insert_table = ibm_db.exec_immediate(conn, insertQuery)
```

```
print(insert_table)
```

```
selectQuery = "SELECT * FROM booktb where UserName= '" + uname + "' "
```

```
dataframe = pandas.read_sql(selectQuery, pd_conn)
```

```
dataframe.to_sql('booktb1', con=engine, if_exists='append')
```

```
data = engine.execute("SELECT * FROM booktb1").fetchall()
```

```
return render_template('UserComplaint.html', data=data)
```

```
@app.route("/AgentAssign", methods=['GET'])
```

```
def AgentAssign():
```

```

cid = request.args.get('id')

    session['cid'] = cid


    conn = ibm_db.connect(dsn, "", "")

pd_conn = ibm_db_dbi.Connection(conn)


selectQuery = "SELECT * FROM agenttb "

dataframe = pandas.read_sql(selectQuery, pd_conn)


    dataframe.to_sql('booktb1', con=engine, if_exists='append')

data = engine.execute("SELECT * FROM booktb1").fetchall()


    return render_template('AgentAssign.html', data=data)

```

```

@app.route("/Action", methods=['GET'])

```

```

def Action():

```

```

    cid = request.args.get('id')

    session['cid'] = cid

```

```

    return render_template('Action.html')

```

```
@app.route("/ass", methods=[ 'GET', 'POST'])

def ass():

    agid = request.form['agid']

    cid = session['cid']

    uname = session['uname']

    conn = ibm_db.connect(dsn, "", "")

    pd_conn = ibm_db_dbi.Connection(conn)

    selectQuery1 = "SELECT * FROM regtb where UserName='" + uname + "'"

    dataframe = pandas.read_sql(selectQuery1, pd_conn)

    dataframe.to_sql('regtb', con=engine, if_exists='append')
```



```
data1 = engine.execute("SELECT * FROM regtb").fetchall()
```

```
for item1 in data1:
```

```
    Mobile = item1[5]
```

```
    Email = item1[4]
```

```
    sendmsg(Email,"Assign Agent id"+agid)
```

```
insertQuery = "update booktb set AgentName='"+ agid +"' where ComplaintId='"+ cid +"' "
```

```
insert_table = ibm_db.exec_immediate(conn, insertQuery)
```

```
alert = 'Agent Assign Send Notication'
```

```
return render_template('goback.html', data=alert)
```

```
@app.route("/acc", methods=[ 'GET' , 'POST'])
```

```
def acc():
```

```
    com = request.form['com']
```

```
    cid = session['cid']
```

```
    uname = session['uname']
```

```

conn = ibm_db.connect(dsn, "", "")

pd_conn = ibm_db_dbi.Connection(conn)

selectQuery1 = "SELECT * FROM regtb where UserName='" + uname + "'"

dataframe = pandas.read_sql(selectQuery1, pd_conn)

dataframe.to_sql('regtb', con=engine, if_exists='append')

data1 = engine.execute("SELECT * FROM regtb").fetchall()

for item1 in data1:

    Mobile = item1[5]

    Email = item1[4]

    sendmsg(Email, "Action Information " + com)

```

7.3 Database Schema

```

conn = ibm_db.connect(dsn, "", "")
pd_conn = ibm_db_dbi.Connection(conn)
selectQuery = "SELECT * FROM agenttb where "
dataframe = pandas.read_sql(selectQuery, pd_conn)
dataframe.to_sql('booktb1', con=engine, if_exists='append')
data = engine.execute("SELECT * FROM booktb1").fetchall()

return render_template('NewAgent.html', data=data)

```

```

@app.route("/AdminHome")
def AdminHome():
    conn = ibm_db.connect(dsn, "", "")

```

```

pd_conn = ibm_db_dbi.Connection(conn)

selectQuery = "SELECT * from regtb "
dataframe = pandas.read_sql(selectQuery, pd_conn)

dataframe.to_sql('Employee_Data',
                 con=engine,
                 if_exists='append')

# run a sql query
data = engine.execute("SELECT * FROM Employee_Data").fetchall()
return render_template('AdminHome.html',data=data)

```

```

@app.route("/UserHome")
def UserHome():
    user = session['uname']

    conn = ibm_db.connect(dsn, "", "")
    pd_conn = ibm_db_dbi.Connection(conn)
    selectQuery = "SELECT * FROM regtb where UserName= '" + user + "'"
    dataframe = pandas.read_sql(selectQuery, pd_conn)
    dataframe.to_sql('booktb1', con=engine, if_exists='append')
    data = engine.execute("SELECT * FROM booktb1").fetchall()
    return render_template('UserHome.html',data=data)

```

```

@app.route("/UserComplaint")
def UserComplaint():
    user = session['uname']

    conn = ibm_db.connect(dsn, "", "")
    pd_conn = ibm_db_dbi.Connection(conn)
    selectQuery = "SELECT * FROM booktb where UserName= '" + user + "'"

```

```
dataframe = pandas.read_sql(selectQuery, pd_conn)
dataframe.to_sql('booktb1', con=engine, if_exists='append')
data = engine.execute("SELECT * FROM booktb1").fetchall()
```

```
return render_template('UserComplaint.html', data=data)
```

```
@app.route("/AdminComplaintInfo")
```

```
def AdminComplaintInfo():
```

```
conn = ibm_db.connect(dsn, "", "")
pd_conn = ibm_db_dbi.Connection(conn)
selectQuery = "SELECT * FROM booktb "
dataframe = pandas.read_sql(selectQuery, pd_conn)
dataframe.to_sql('booktb1', con=engine, if_exists='append')
data = engine.execute("SELECT * FROM booktb1").fetchall()
```

```
return render_template('AdminComplaintInfo.html', data=data)
```

```
@app.route("/adminlogin", methods=['GET', 'POST'])
```

```
def adminlogin():
```

```
error = None
```

```
if request.method == 'POST':
```

```
    if request.form['uname'] == 'admin' or request.form['password'] == 'admin':
```

```
        conn = ibm_db.connect(dsn, "", "")
        pd_conn = ibm_db_dbi.Connection(conn)
```

```
        selectQuery = "SELECT * from regtb "
        dataframe = pandas.read_sql(selectQuery, pd_conn)
```

```
        dataframe.to_sql('Employee_Data',
                        con=engine,
                        if_exists='append')
```

```
        # run a sql query
```

```
        data = engine.execute("SELECT * FROM Employee_Data").fetchall()
        return render_template('AdminHome.html', data=data)
```

```
else:
```

```
    return render_template('index.html', erro
```

8.TESTING

8.1 TEST CASES

A test case has components that describe input, action and an expected response, in order to determine if a feature of an application is working correctly. A test case is a set of instructions on “HOW” to validate a particular test objective/target, which when followed will tell us if the expected behavior of the system is satisfied or not.

Characteristics of a good test case:

- Accurate: Exacts the purpose.
- Economical: No unnecessary steps or words.
- Traceable: Capable of being traced to requirements.
- Repeatable: Can be used to perform the test over and over.
- Reusable: Can be reused if necessary

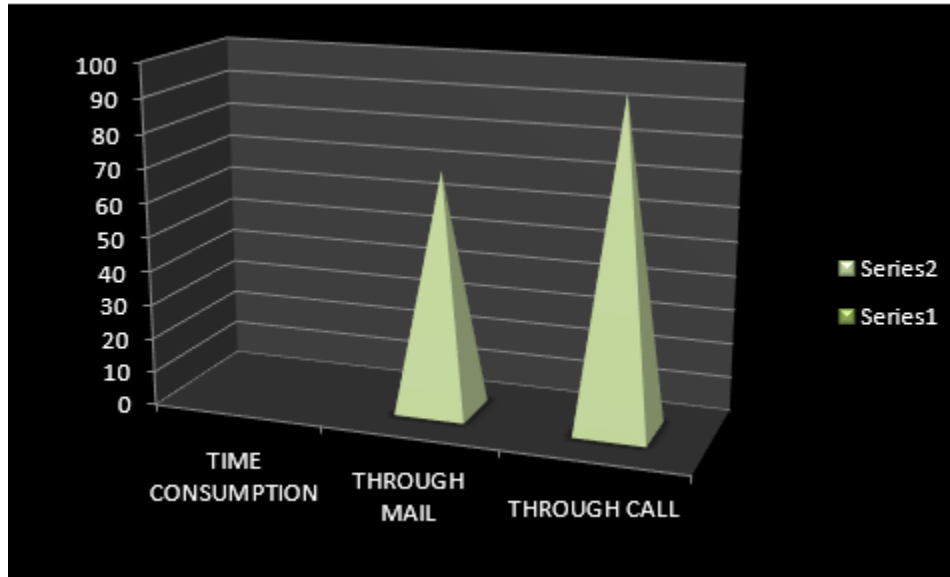
S.NO	Scenario	Input	Excepted output	Actual output
1	Admin Login Form	User name and password	Login	Login success.
2	Employee Login Form	User name and password	Login	Login success.
3	User Registration Form	User basic details	Registered successfully	User basic details are stored in the database.
4	User Login Form	User name and password	Login	Login success.

8.2 USER ACCEPTANCE TESTING

This is a type of testing done by users, customers, or other authorised entities to determine application/software needs and business processes. Acceptance testing is the most important phase of testing as this decides whether the client approves the application/software or not. It may involve functionality, usability, performance, and U.I of the application. It is also known as user acceptance testing (UAT), operational acceptance testing (OAT), and end-user testing.

9.RESULTS

9.1 PERFORMANCE METRICS



10.ADVANTAGES AND DISADVANTAGES

DISADVANTAGES

- Requires an active internet connection.
- System may provide inaccurate results if the data entered incorrectly.
- Difficult to provide proper intimation system
- Current system is manual process
- Cannot always taking a call
- Tower problem during call conversation

ADVANTAGES

- System is easy to understand and user friendly.
- The system is purely based on prediction which predicts an internet plan for the customer.
- Admin can easily view employee report based on the resolution provided on the complaint.
- Handle large number of contextual information.
- User friendly and time consuming process.
- Using this project, the user can know about status of complaint through website.
- Keep track of daily information exchange at the server by the administrator.
- Increase in processing and transfer speeds of information over the network.

11.CONCLUSION

Application software has been computed successfully and was also tested successfully by taking “test cases”. It is user friendly, and has required option, which can be utilized by the user to perform the desired operations. Application meets the information requirements specified to a great extent. The system has been designed keeping in view the present and future requirements in mind and made very flexible. The goals that are achieved by the software are Instant access, improved productivity, Optimum utilization of resources, Efficient management of records, Simplifications of the operations, Less processing time and getting required information, User friendly, Portable and flexible for further enhancement. The system has the benefits of easy access because it is developed as a platform independent web application, so the admin can maintain a proper contact with their users, which may be accessed anywhere. All communications between the police and administrator have been done through the online, so this communication cost is also reduced.

12.FUTURE SCOPE

In future we can develop this project in android application with extra features like customer complaint system and collect the feedback form from the customer about the system.

1. APPENDIX

SOURCE CODE

```
from flask import Flask, render_template, flash, request, session, send_file
from flask import render_template, redirect, url_for, request

import ibm_db
import pandas
import ibm_db_dbi
from sqlalchemy import create_engine

engine = create_engine('sqlite://',
                       echo = False)

dsn_hostname = "b70af05b-76e4-4bca-a1f5-
23dbb4c6a74e.c1ogj3sd0tgtu0lqde00.databases.appdomain.cloud"
dsn_uid = "crc83247"
dsn_pwd = "eHGBftxhodLDnNpM"

dsn_driver = "{IBM DB2 ODBC DRIVER}"
dsn_database = "BLUDB"
dsn_port = "32716"
dsn_protocol = "TCPIP"
dsn_security = "SSL"

dsn = (
    "DRIVER={0};"
    "DATABASE={1};"
    "HOSTNAME={2};"
```

```
"PORT={3};"
"PROTOCOL={4};"
"UID={5};"
"PWD={6};"
"SECURITY={7};").format(dsn_driver, dsn_database, dsn_hostname, dsn_port, dsn_protocol,
dsn_uid, dsn_pwd,dsn_security)
```

```
try:
```

```
    conn = ibm_db.connect(dsn, "", "")
    print ("Connected to database: ", dsn_database, "as user: ", dsn_uid, "on host: ",
dsn_hostname)
```

```
except:
```

```
    print ("Unable to connect: ", ibm_db.conn_errormsg() )
```

```
app = Flask(__name__)
app.config['DEBUG']
app.config['SECRET_KEY'] = '7d441f27d441f27567d441f2b6176a'
```

```
@app.route("/")
```

```
def homepage():
```

```
    return render_template('index.html')
```

```
@app.route("/AdminLogin")
```

```
def AdminLogin():
```

```
    return render_template('AdminLogin.html')
```

```
@app.route("/UserLogin")
def UserLogin():
    return render_template('UserLogin.html')
```

```
@app.route("/NewUser")
def NewUser():
    return render_template('NewUser.html')
```

```
@app.route("/NewComplaint")
def NewComplaint():
    user = session['uname']
    return render_template('NewComplaint.html',uname=user)
```

```
@app.route("/NewAgent")
def NewAgent():
```

```
    conn = ibm_db.connect(dsn, "", "")
    pd_conn = ibm_db_dbi.Connection(conn)
    selectQuery = "SELECT * FROM agenttb where "
    dataframe = pandas.read_sql(selectQuery, pd_conn)
    dataframe.to_sql('booktb1', con=engine, if_exists='append')
    data = engine.execute("SELECT * FROM booktb1").fetchall()
```

```
return render_template('NewAgent.html',data=data)
```

```
@app.route("/AdminHome")
```

```
def AdminHome():
```

```
    conn = ibm_db.connect(dsn, "", "")
```

```
    pd_conn = ibm_db_dbi.Connection(conn)
```

```
    selectQuery = "SELECT * from regtb "
```

```
    dataframe = pandas.read_sql(selectQuery, pd_conn)
```

```
    dataframe.to_sql('Employee_Data',
```

```
                    con=engine,
```

```
                    if_exists='append')
```

```
    # run a sql query
```

```
    data = engine.execute("SELECT * FROM Employee_Data").fetchall()
```

```
    return render_template('AdminHome.html',data=data)
```

```
@app.route("/UserHome")
```

```
def UserHome():
```

```
    user = session['uname']
```

```
    conn = ibm_db.connect(dsn, "", "")
```

```
    pd_conn = ibm_db_dbi.Connection(conn)
```

```
    selectQuery = "SELECT * FROM regtb where  UserName= '" + user + "' "
```

```
    dataframe = pandas.read_sql(selectQuery, pd_conn)
```

```
    dataframe.to_sql('booktb1', con=engine, if_exists='append')
```

```
data = engine.execute("SELECT * FROM booktb1").fetchall()
return render_template('UserHome.html',data=data)
```

```
@app.route("/UserComplaint")
```

```
def UserComplaint():
```

```
    user = session['uname']
```

```
    conn = ibm_db.connect(dsn, "", "")
```

```
    pd_conn = ibm_db_dbi.Connection(conn)
```

```
    selectQuery = "SELECT * FROM booktb where UserName= ' " + user + " "
```

```
    dataframe = pandas.read_sql(selectQuery, pd_conn)
```

```
    dataframe.to_sql('booktb1', con=engine, if_exists='append')
```

```
    data = engine.execute("SELECT * FROM booktb1").fetchall()
```

```
    return render_template('UserComplaint.html',data=data)
```

```
@app.route("/AdminComplaintInfo")
```

```
def AdminComplaintInfo():
```

```
    conn = ibm_db.connect(dsn, "", "")
```

```
    pd_conn = ibm_db_dbi.Connection(conn)
```

```
    selectQuery = "SELECT * FROM booktb  "
```

```
    dataframe = pandas.read_sql(selectQuery, pd_conn)
```

```
    dataframe.to_sql('booktb1', con=engine, if_exists='append')
```

```
    data = engine.execute("SELECT * FROM booktb1").fetchall()
```

```
    return render_template('AdminComplaintInfo.html',data=data)
```

```

@app.route("/adminlogin", methods=['GET', 'POST'])
def adminlogin():
    error = None
    if request.method == 'POST':
        if request.form['uname'] == 'admin' or request.form['password'] == 'admin':

            conn = ibm_db.connect(dsn, "", "")
            pd_conn = ibm_db_dbi.Connection(conn)

            selectQuery = "SELECT * from regtb "
            dataframe = pandas.read_sql(selectQuery, pd_conn)

            dataframe.to_sql('Employee_Data',
                             con=engine,
                             if_exists='append')

            # run a sql query
            data = engine.execute("SELECT * FROM Employee_Data").fetchall()
            return render_template('AdminHome.html' , data=data)

    else:
        return render_template('index.html', error=error)

@app.route("/userlogin", methods=['GET', 'POST'])
def userlogin():

    if request.method == 'POST':
        username = request.form['uname']

```



```

password = request.form['password']
session['uname'] = request.form['uname']

conn = ibm_db.connect(dsn, "", "")
pd_conn = ibm_db_dbi.Connection(conn)

selectQuery = "SELECT * from regtb where UserName='" + username + "' and password='"
+ password + "'"

dataframe = pandas.read_sql(selectQuery, pd_conn)

if dataframe.empty:
    data1 = 'Username or Password is wrong'
    return render_template('goback.html', data=data1)
else:
    print("Login")
    selectQuery = "SELECT * from regtb where UserName='" + username + "' and
password='" + password + "'"
    dataframe = pandas.read_sql(selectQuery, pd_conn)

    dataframe.to_sql('Employee_Data',
                    con=engine,
                    if_exists='append')

    # run a sql query
    data = engine.execute("SELECT * FROM Employee_Data").fetchall()

    return render_template('UserHome.html', data=data )

@app.route("/newuser", methods=['GET', 'POST'])

```

```

def newuser():
    if request.method == 'POST':

        name1 = request.form['name']
        gender1 = request.form['gender']
        Age = request.form['age']
        email = request.form['email']
        pnumber = request.form['phone']
        address = request.form['address']

        uname = request.form['uname']
        password = request.form['psw']

        conn = ibm_db.connect(dsn, "", "")

        insertQuery = "INSERT INTO regtb VALUES ('" + name1 + "','" + gender1 + "','" + Age +
        "','" + email + "','" + pnumber + "','" + address + "','" + uname + "','" + password + "')"

        insert_table = ibm_db.exec_immediate(conn, insertQuery)
        print(insert_table)

    return render_template('UserLogin.html')

@app.route("/newage", methods=['GET', 'POST'])
def newage():
    if request.method == 'POST':

```

```
name1 = request.form['name']
gender1 = request.form['gender']
Age = request.form['age']
email = request.form['email']
pnumber = request.form['phone']
address = request.form['address']
```

```
uname = request.form['uname']
```

```
conn = ibm_db.connect(dsn, "", "")
pd_conn = ibm_db_dbi.Connection(conn)
```

```
insertQuery = "INSERT INTO agenttb VALUES ('" + name1 + "'," + gender1 + "'," + Age
+ "'," + email + "'," + pnumber + "'," + address + "'," + uname + "')"
insert_table = ibm_db.exec_immediate(conn, insertQuery)
print(insert_table)
```

```
selectQuery = "SELECT * FROM agenttb "
dataframe = pandas.read_sql(selectQuery, pd_conn)
dataframe.to_sql('booktb1', con=engine, if_exists='append')
data = engine.execute("SELECT * FROM booktb1").fetchall()
```

```
return render_template('NewAgent.html',data=data)
```

```

@app.route("/newcom", methods=['GET', 'POST'])
def newcom():
    if request.method == 'POST':

        name = request.form['name']
        com = request.form['com']
        uname = session['uname']

        conn = ibm_db.connect(dsn, "", "")
        pd_conn = ibm_db_dbi.Connection(conn)

        selectQuery = "SELECT * FROM booktb"
        dataframe = pandas.read_sql(selectQuery, pd_conn)

        dataframe.to_sql('booktb', con=engine, if_exists='append')
        data2 = engine.execute("SELECT * FROM booktb").fetchall()
        count = 0

        for item in data2:
            count += 1

        Bookingid = "COMID00" + str(count)

        insertQuery = "INSERT INTO booktb VALUES ('" + Bookingid + "', '" + uname + "', '" +
com + "', ', ')"

```

```
insert_table = ibm_db.exec_immediate(conn, insertQuery)
print(insert_table)
```

```
selectQuery = "SELECT * FROM booktb where UserName= '' + uname + '' "
dataframe = pandas.read_sql(selectQuery, pd_conn)
```

```
dataframe.to_sql('booktb1', con=engine, if_exists='append')
data = engine.execute("SELECT * FROM booktb1").fetchall()
return render_template('UserComplaint.html', data=data)
```

```
@app.route("/AgentAssign", methods=['GET'])
```

```
def AgentAssign():
```

```
    cid = request.args.get('id')
```

```
    session['cid'] = cid
```

```
    conn = ibm_db.connect(dsn, "", "")
```

```
    pd_conn = ibm_db_dbi.Connection(conn)
```

```
    selectQuery = "SELECT * FROM agenttb "
```

```
    dataframe = pandas.read_sql(selectQuery, pd_conn)
```

```
    dataframe.to_sql('booktb1', con=engine, if_exists='append')
```

```
    data = engine.execute("SELECT * FROM booktb1").fetchall()
```

```
    return render_template('AgentAssign.html', data=data)
```

```
@app.route("/Action", methods=['GET'])
```

```
def Action():
    cid = request.args.get('id')
    session['cid'] = cid

    return render_template('Action.html')

@app.route("/ass", methods=['GET', 'POST'])
def ass():

    agid = request.form['agid']

    cid = session['cid']

    uname = session['uname']

    conn = ibm_db.connect(dsn, "", "")
    pd_conn = ibm_db_dbi.Connection(conn)

    selectQuery1 = "SELECT * FROM regtb where UserName='" + uname + "'"
    dataframe = pandas.read_sql(selectQuery1, pd_conn)

    dataframe.to_sql('regtb', con=engine, if_exists='append')
    data1 = engine.execute("SELECT * FROM regtb").fetchall()

    for item1 in data1:
        Mobile = item1[5]
        Email = item1[4]
        sendmsg(Email,"Assign Agent id"+agid)
```

```
insertQuery = "update booktb set AgentName='"+ agid +" where ComplaintId='"+ cid +"" "
```

```
insert_table = ibm_db.exec_immediate(conn, insertQuery)
```

```
alert = 'Agent Assign Send Notication'
```

```
return render_template('goback.html', data=alert)
```

```
@app.route("/acc", methods=['GET', 'POST'])
```

```
def acc():
```

```
com = request.form['com']
```

```
cid = session['cid']
```

```
uname = session['uname']
```

```
conn = ibm_db.connect(dsn, "", "")
```

```
pd_conn = ibm_db_dbi.Connection(conn)
```

```
selectQuery1 = "SELECT * FROM regtb where UserName='" + uname + """
```

```
dataframe = pandas.read_sql(selectQuery1, pd_conn)
```

```
dataframe.to_sql('regtb', con=engine, if_exists='append')
```

```
data1 = engine.execute("SELECT * FROM regtb").fetchall()
```

```
for item1 in data1:
```

```
    Mobile = item1[5]
```

```
    Email = item1[4]
```

```
sendmsg(Email,"Action Information "+com)
```

```
insertQuery = "update booktb set ACTIONINFO='"+ com +" where ComplaintId='"+ cid +" "
```

```
insert_table = ibm_db.exec_immediate(conn, insertQuery)
```

```
alert = 'Action Info Saved Send Notication'
```

```
return render_template('goback.html', data=alert)
```

```
def sendmsg(Mailid,message):
```

```
    import smtplib
```

```
    from email.mime.multipart import MIMEMultipart
```

```
    from email.mime.text import MIMEText
```

```
    from email.mime.base import MIMEBase
```

```
    from email import encoders
```

```
    fromaddr = "sampletest685@gmail.com"
```

```
    toaddr = Mailid
```

```
    # instance of MIMEMultipart
```

```
    msg = MIMEMultipart()
```

```
    # storing the senders email address
```

```
    msg['From'] = fromaddr
```

```
    # storing the receivers email address
```

```
    msg['To'] = toaddr
```

```
    # storing the subject
```



```
msg['Subject'] = "Alert"

# string to store the body of the mail
body = message

# attach the body with the msg instance
msg.attach(MIMEText(body, 'plain'))

# creates SMTP session
s = smtplib.SMTP('smtp.gmail.com', 587)

# start TLS for security
s.starttls()

# Authentication
s.login(fromaddr, "hneucvnontsuwgpj")

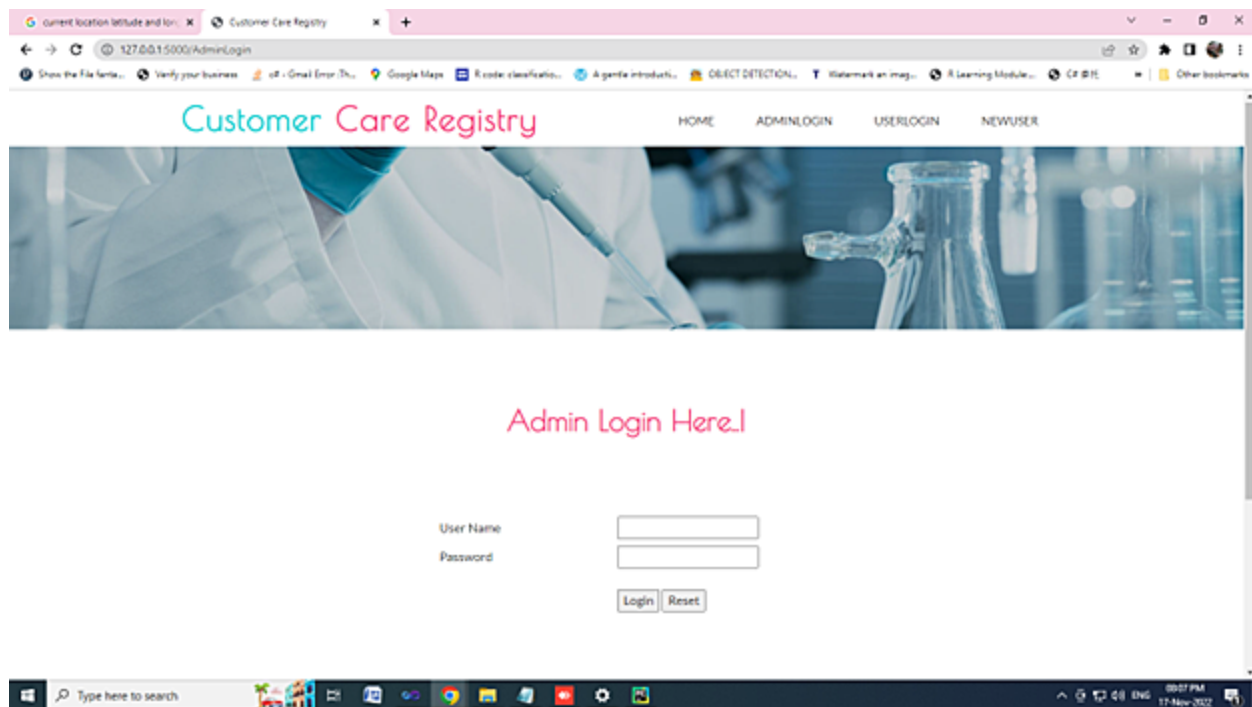
# Converts the Multipart msg into a string
text = msg.as_string()

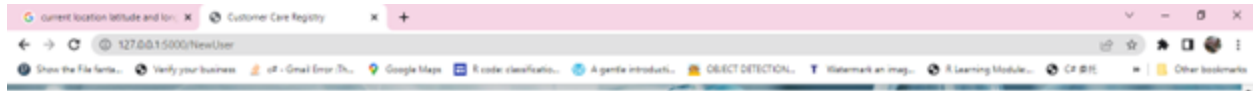
# sending the mail
s.sendmail(fromaddr, toaddr, text)

# terminating the session
s.quit()

if __name__ == '__main__':
    app.run(debug=True, use_reloader=True)
```

SCREENSHOTS





New User Registration

Name

Gender ☐ Male ☐ Female

Age

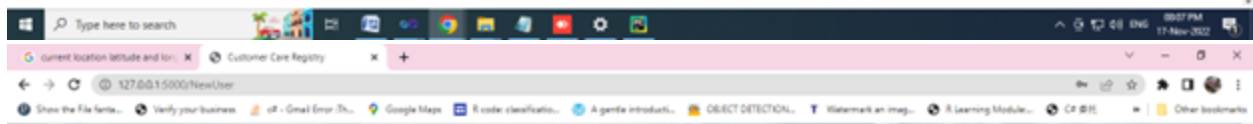
Email Id

Phone Number

Address

User Name

Password



New User Registration

Name

Gender ☒ Male ☐ Female

Age

Email Id

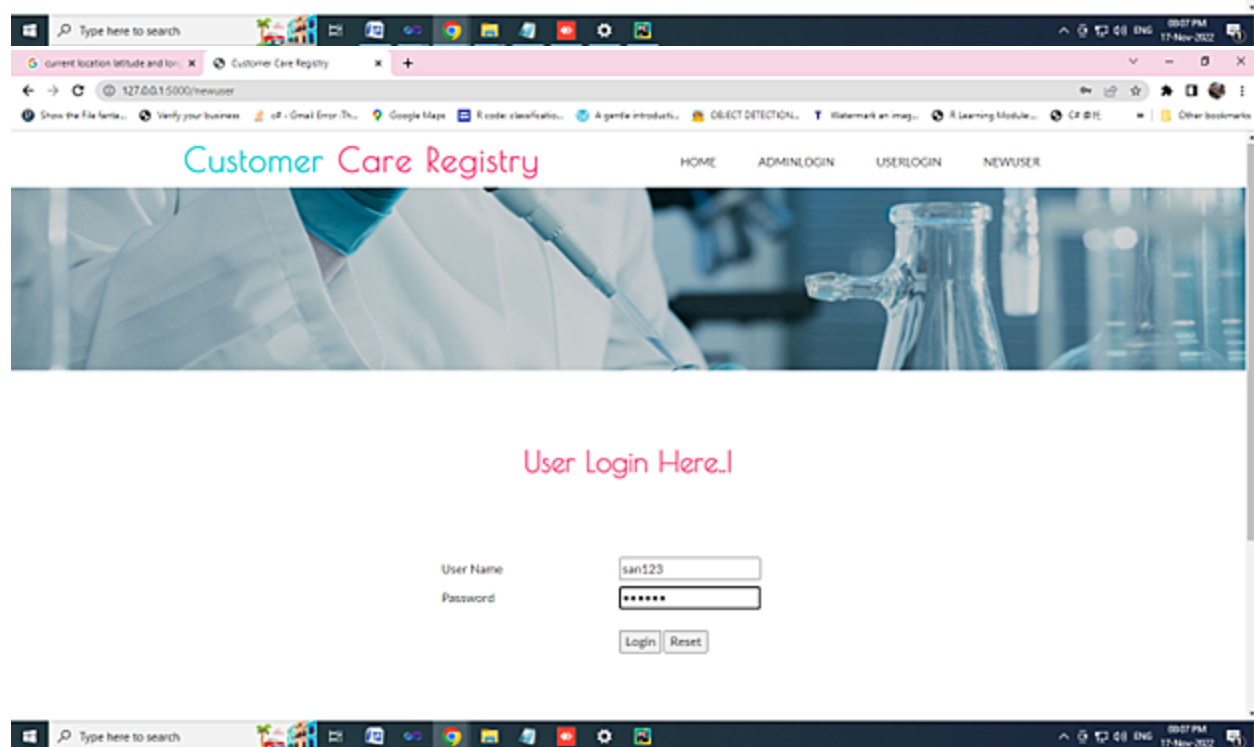
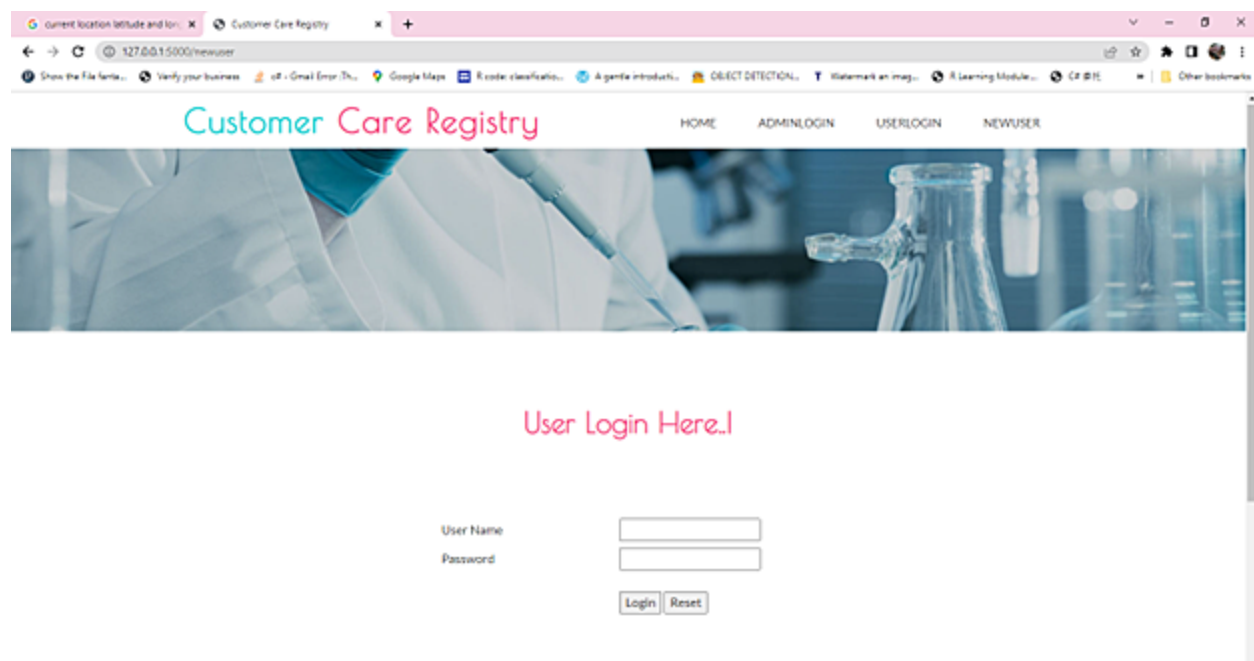
Phone Number

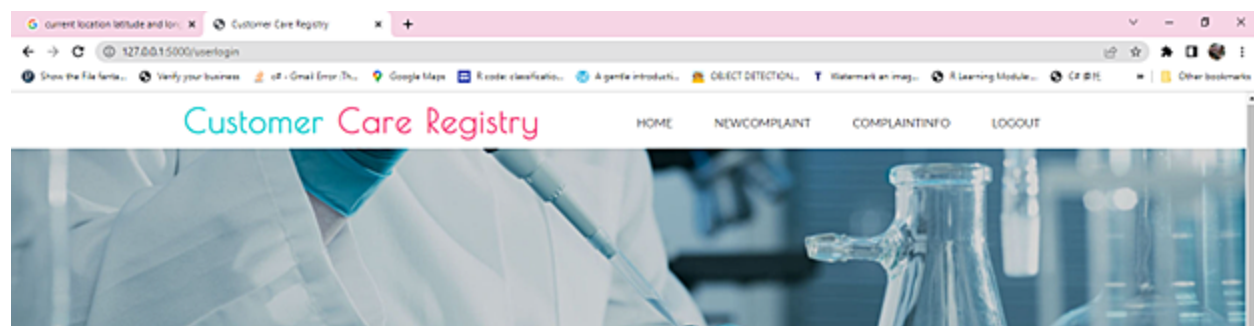
Address

User Name

Password

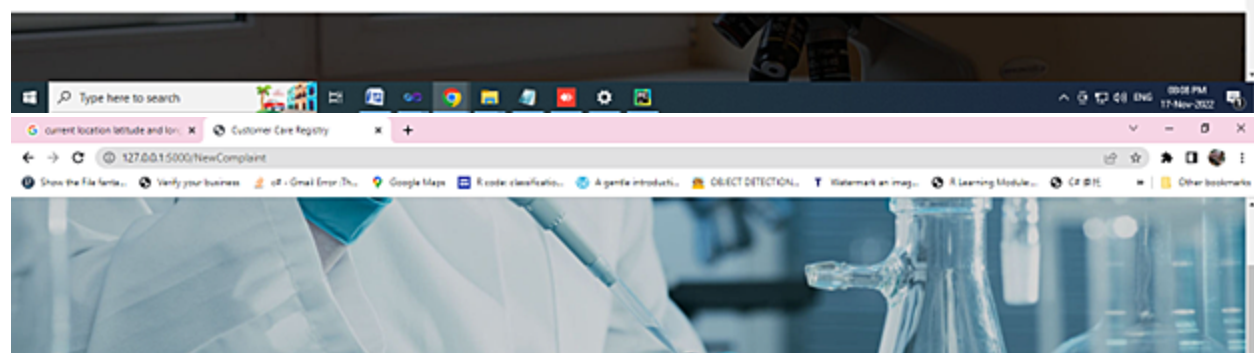






Personal Info

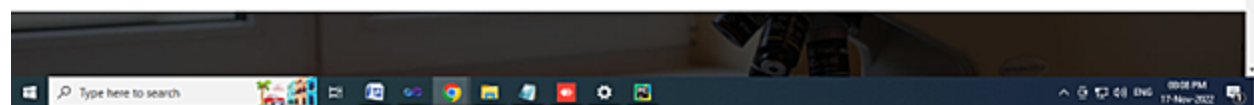
Name	Gender	Age	EmailId	Phone	UserName
san	male	20	sangeeth5535@gmail.com	9486365535	san123
san	male	20	sangeeth5535@gmail.com	9486365535	san123

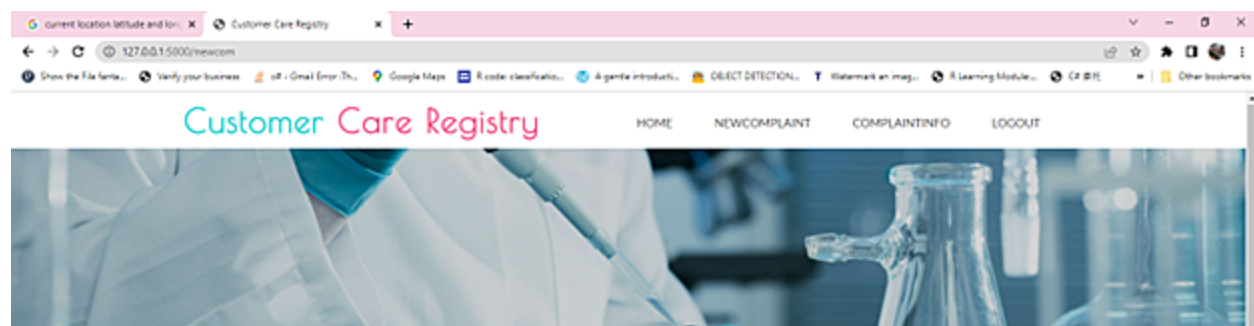


New Complaint Info

UserName

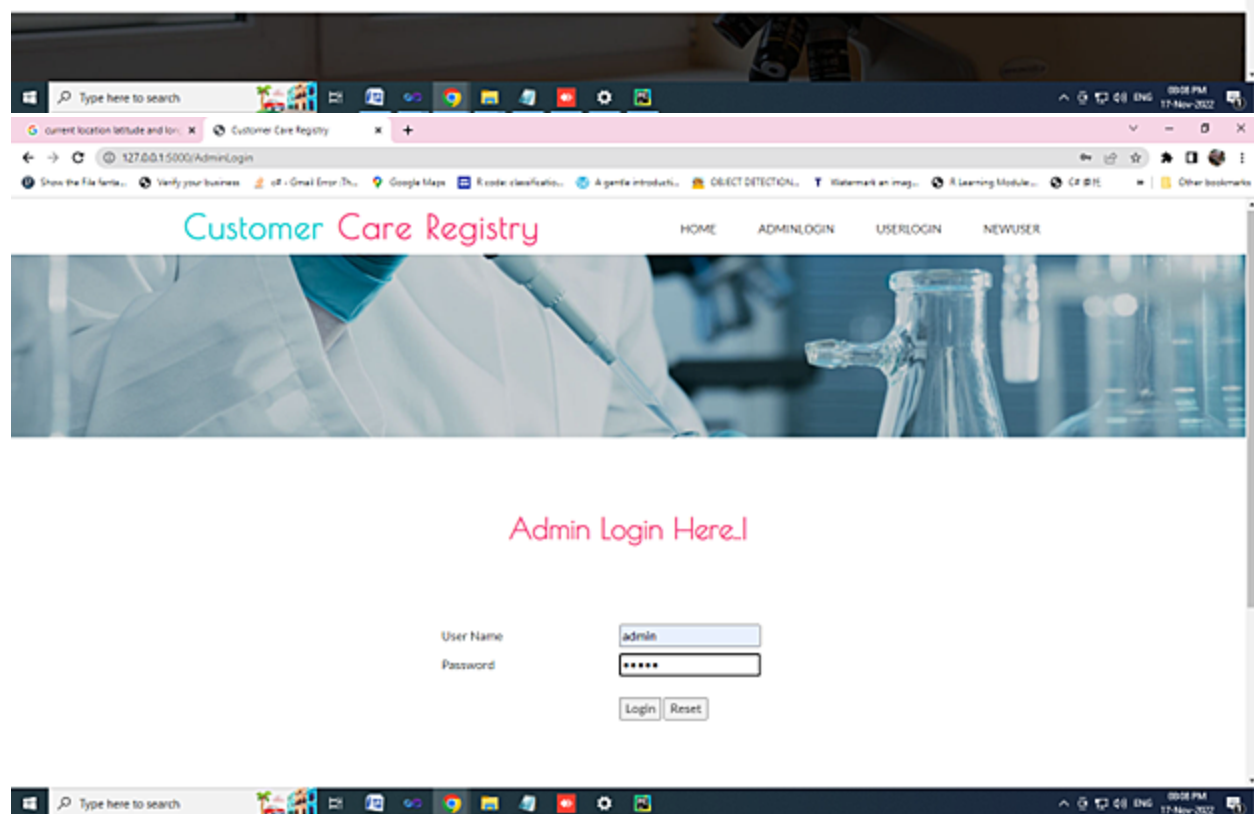
Complaint Info





Complaint Action Information!

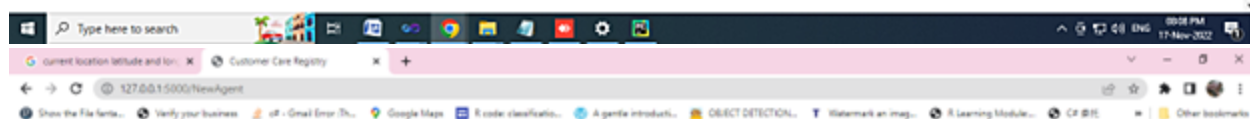
ComplaintId	UserName	Info	AgentName	ActionInfo
COMID002	san123	info	Ag345	solved
COMID005	san123	problem		





User Information

Name	Gender	Age	EmailId	Phone	UserName
san	male	20	sangeeth5535@gmail.com	9087556035	san
san	male	20	sangeeth5535@gmail.com	9486365535	san123
ajithkumar	male	20	velajithkumar@gmail.com	9629008825	ajith
vasanth	male	21	vasanthias52@gmail.com	9150602819	vasa
san	male	20	sangeeth5535@gmail.com	9486365535	san123



New Agent Registration

Name

Gender ☐ Male ☐ Female

Age

Email Id

Phone Number

Address

AgentId

Agent Information

Name	Gender	Age	EmailId	Phone	AgentId
san	male	20	san@gmail.com	9087556035	Ag001
AjithKumar	male	20	sangeeth5535@gmail.com	9486365535	Ag345



Customer Care Registry

HOME NEWAGENT COMPLAINTINFO LOGOUT



Complaint Action Information!

ComplaintId	UserName	Info	AgentName	ActionInfo	Assign	Action
COMID000	san	example problem	Ag001		AgentAssign	Action
COMID001	san	water problem	Ag345	completed	AgentAssign	Action
COMID002	san123	acids	Ag345	solved	AgentAssign	Action
COMID003	ajith	no water supply	Ag001	solved problems	AgentAssign	Action
COMID004	ajith	water supply problem			AgentAssign	Action
COMID005	san123	problem			AgentAssign	Action

Customer Care Registry

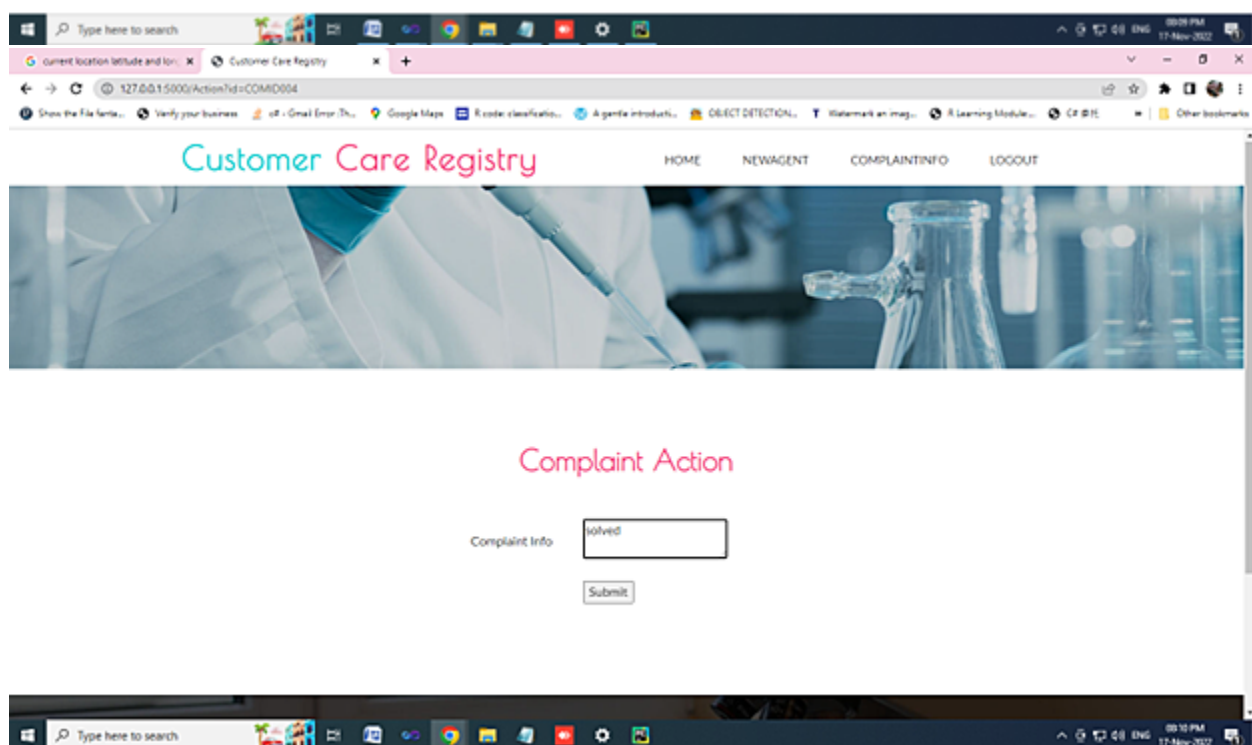
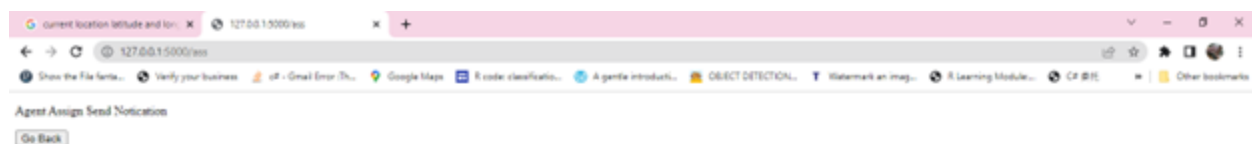
HOME NEWAGENT COMPLAINTINFO LOGOUT

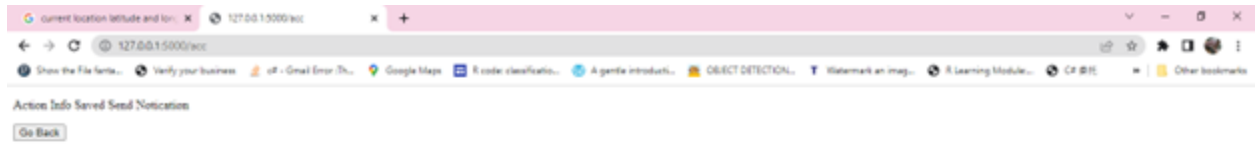


Assign Agent

AgentId Ag001

Submit





GITHUB AND PROJECT DEMO LINK

[Project-Demo-link](#)

<https://github.com/IBM-EPBL/IBM-Project-12740-1659460667>