

Sprint 3

Team ID: PNT2022TMID13422

Team Mates:

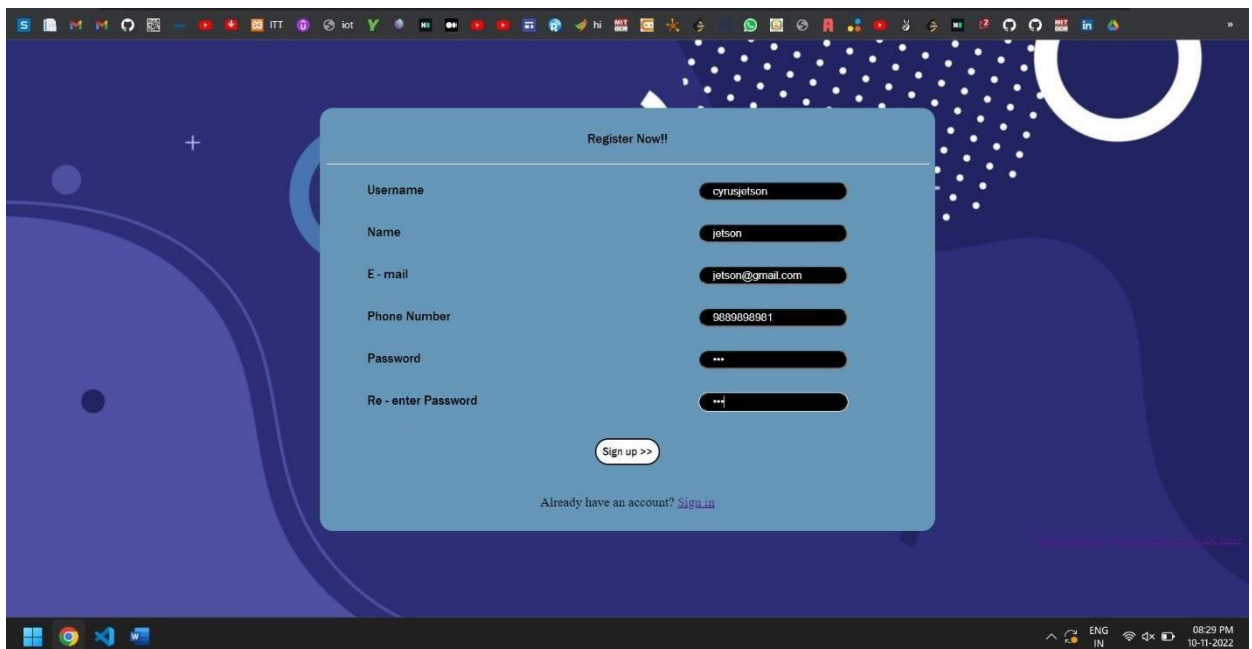
1. Yoga Vignesh MK
2. Vignesh K
3. Vijaya Kumar B
4. Surya V

Sprint 3:

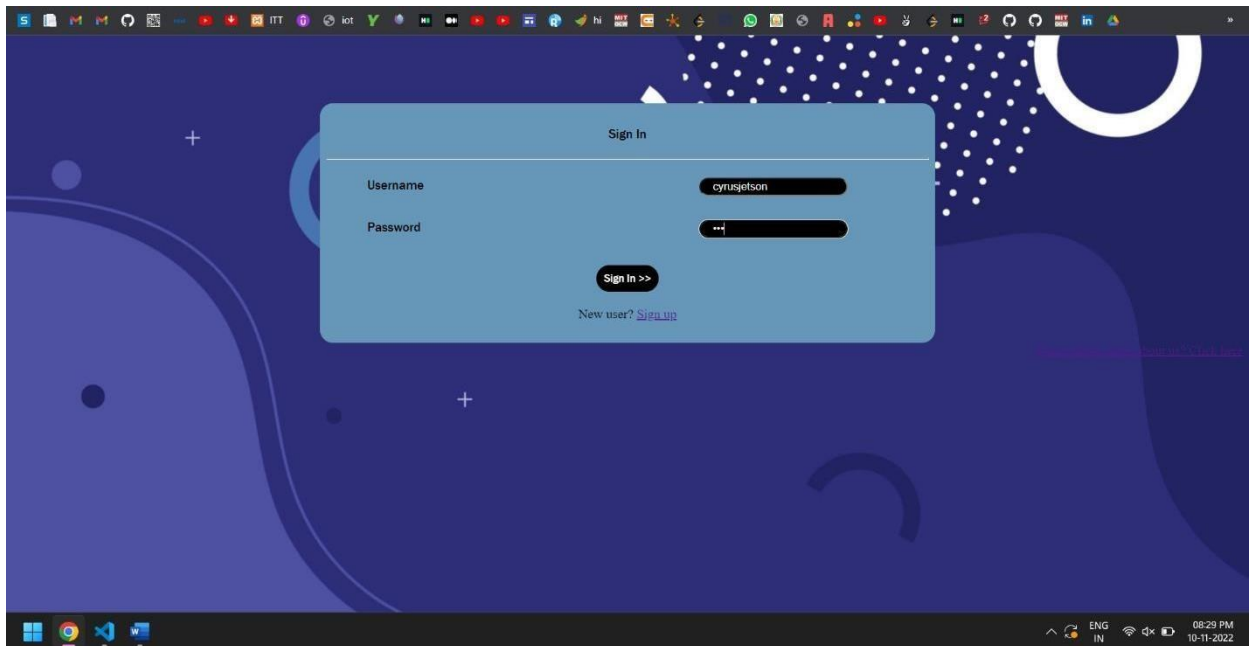
1. Admin Dashboard
2. Agent Dashboard
3. User Dashboard

Output:

1. Signup



2. Sign In



A screenshot of a web application's sign-in page. The page has a dark blue background with abstract white and light blue shapes. A light blue rectangular box is centered on the page, containing the sign-in form. The form has two input fields: 'Username' with the text 'cyrusjetson' and 'Password' with a masked password '****'. Below the fields is a 'Sign In >>' button. At the bottom of the box, it says 'New user? [Sign up](#)'. The browser's taskbar at the bottom shows various icons and the system clock indicating 08:29 PM on 10-11-2022.

Sign In

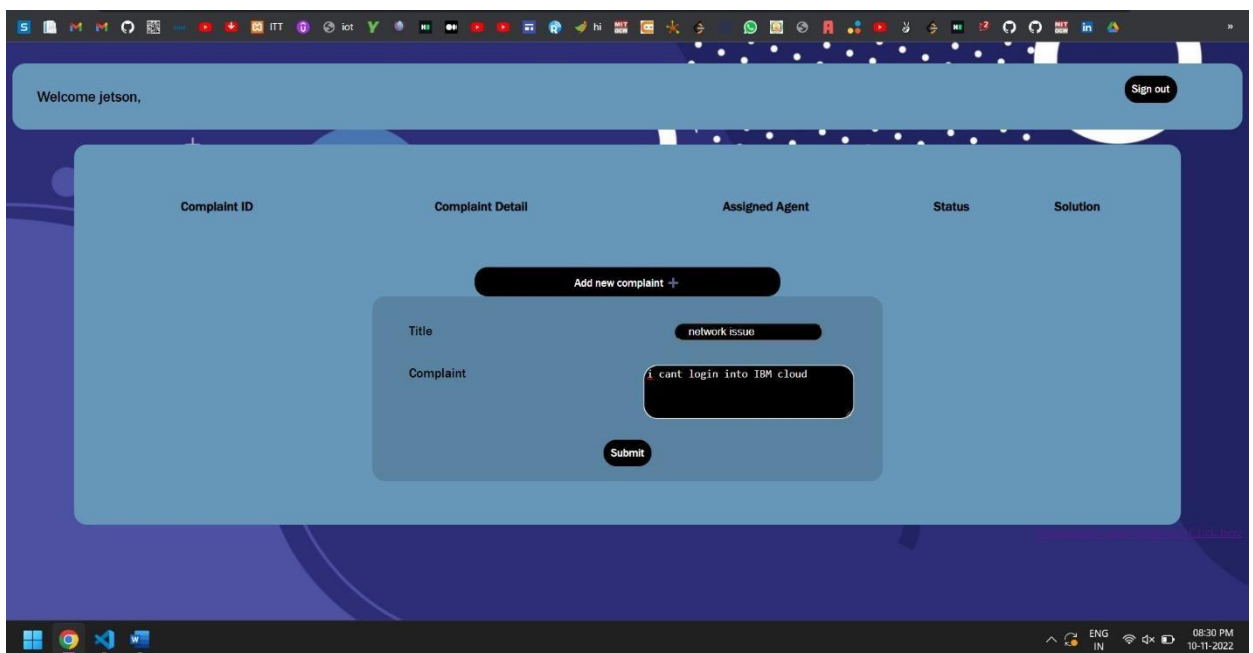
Username cyrusjetson

Password ****

Sign In >>

New user? [Sign up](#)

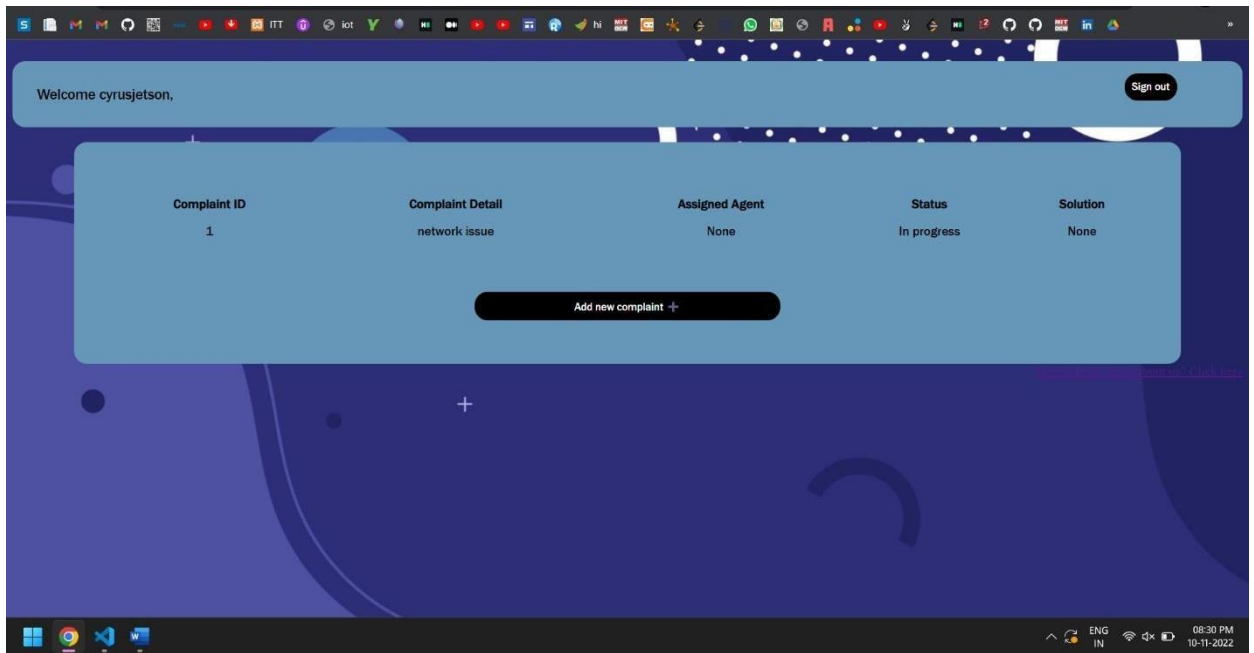
3. User dashboard



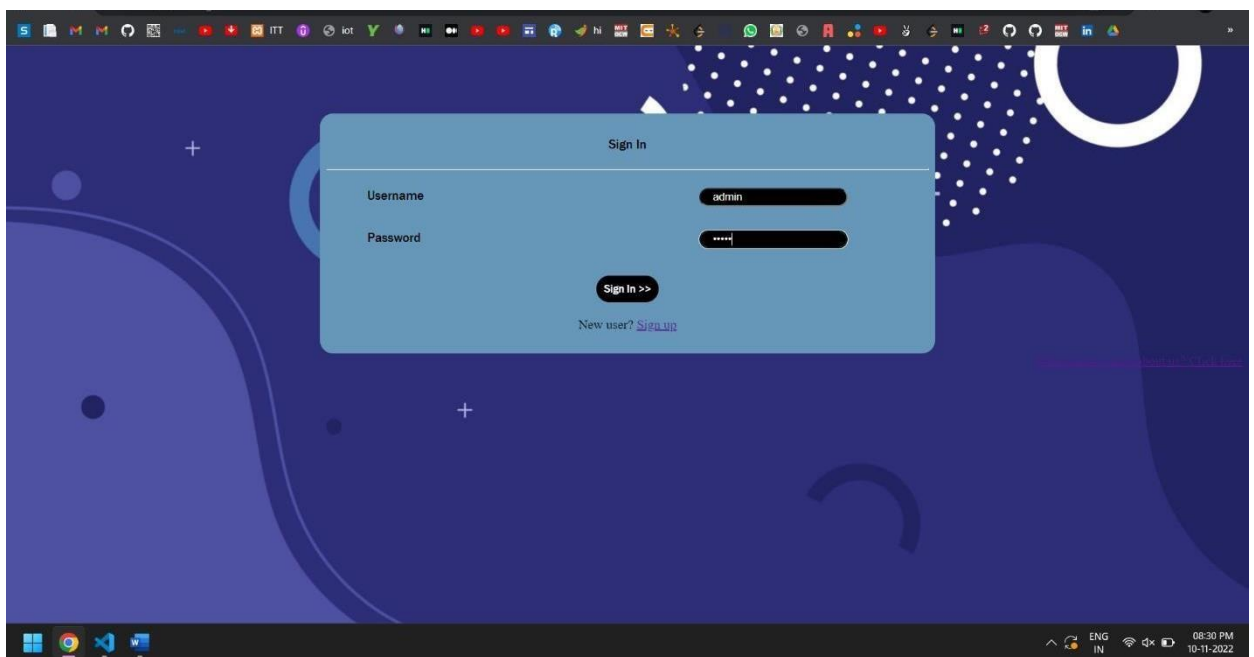
A screenshot of a user dashboard. At the top, a light blue header bar contains the text 'Welcome jetson,' on the left and a 'Sign out' button on the right. Below the header is a large light blue box. Inside this box, at the top, are five column headers: 'Complaint ID', 'Complaint Detail', 'Assigned Agent', 'Status', and 'Solution'. Below these headers is a form to 'Add new complaint'. The form has two input fields: 'Title' with the text 'network issue' and 'Complaint' with the text 'i cant login into IBM cloud'. A 'Submit' button is at the bottom of the form. The browser's taskbar at the bottom shows various icons and the system clock indicating 08:30 PM on 10-11-2022.

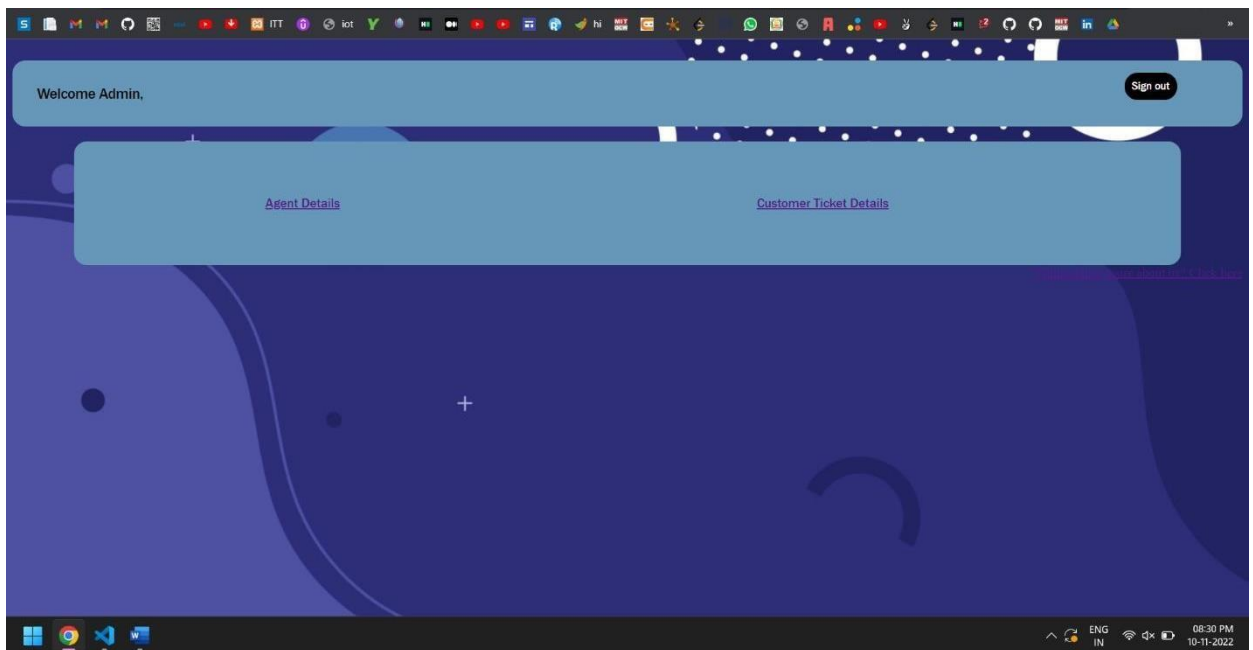
Welcome jetson, Sign out

| Complaint ID | Complaint Detail | Assigned Agent | Status | Solution |
|---|------------------|----------------|--------|----------|
| <div>Add new complaint +</div> <div><div>Titlenetwork issue</div><div>Complainti cant login into IBM cloud</div><div>Submit</div></div> | | | | |

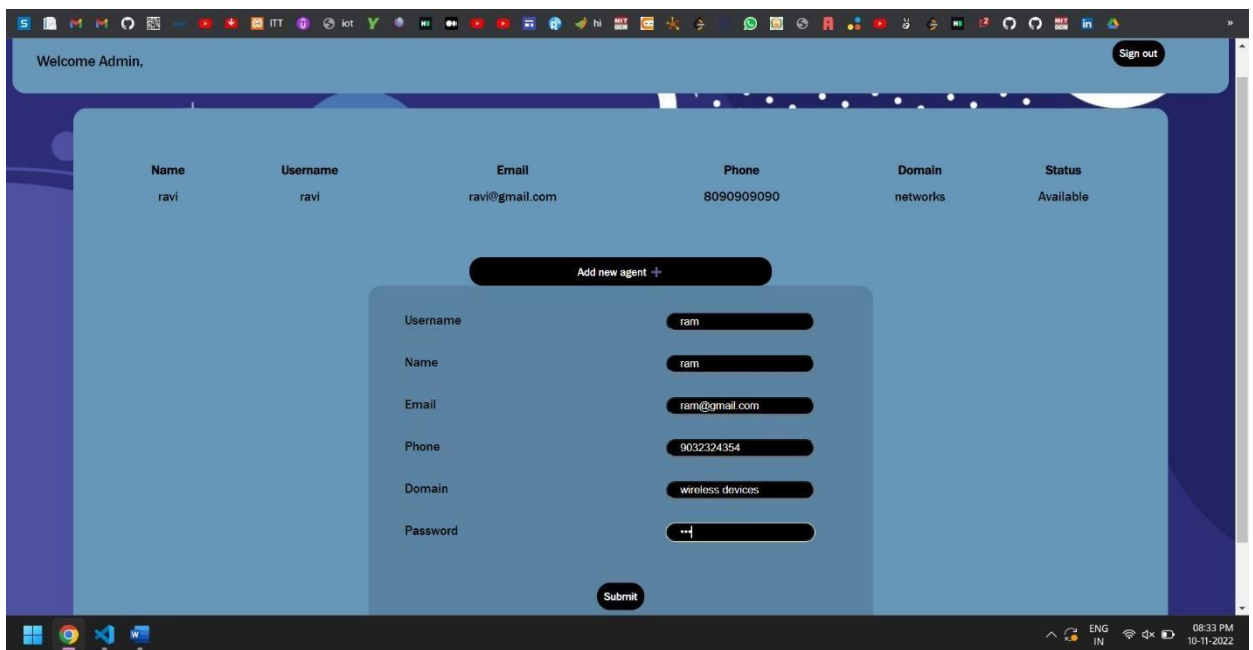


4. Admin dashboard

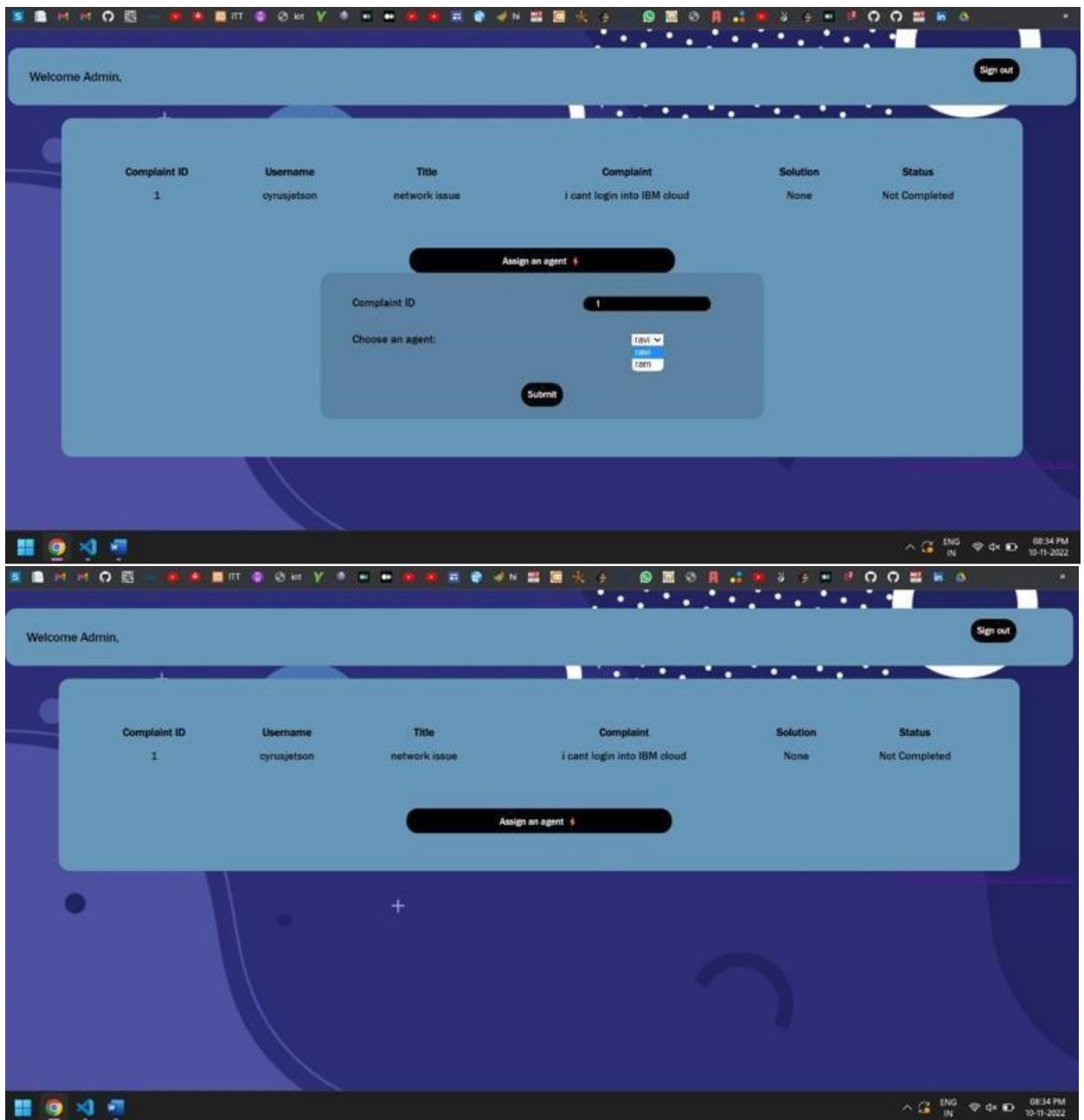




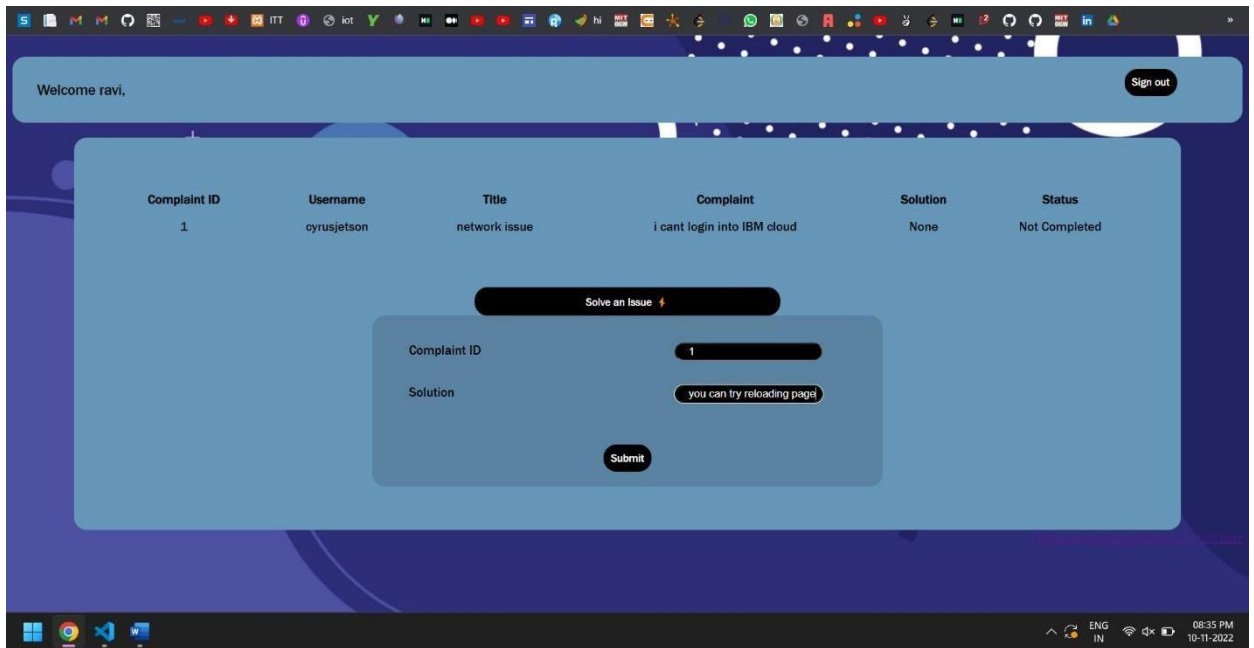
5. Agent details page



6. Assign agent page



7. Agent dashboard



Code:

Admin.html

```
{% extends 'base.html' %}
```

```
{% block head %}
```

```
<title>  
    Admin Dashboard  
</title>
```

```
{% endblock %}
```

```
{% block body %}
```

```
<!-- things  
div 1 welcome  
jetson,    sign out  
div 2 your  
complaints status  
add new complaint -  
>  
<br>
```

{% endblock %}

Agentdas.html

```
{% extends 'base.html' %}
```

```
{% block head %}
```

```
<title>
    Agent Dashboard
</title>
```

```
{% endblock %}
```

```
{% block body %}
```

```
<!-- things
    div 1 welcome
jetson,    sign out
    div 2 your
complaints status
    add new complaint -
->
<br>
<!-- <br>
{% for i in range(11) %}
    {{ i }}
{% endfor %}
```

```
<br>
{% for i in complaints %}
{{ i['USERNAME'] }}
<br>
{% for j in i.values() %}
    {{ j }}
{% endfor %}
<br>
{% endfor %} -->
```

```
<div class="fordashboardtop">
    <div class="fordashboardtopelements1">
        Welcome {{ name }},
    </div>
    <div class="fordashboardtopelements2">
        <a href="/login"><button class="forbutton">Sign out</button></a>
```

</div>

```

</div>
<br>
<div class="outerofdashdetails">

    <div class="fordashboarddetails">
        <br>
        <!-- table of customers complaints -->
        <table class="fortable">
            <thead>
                <th>Complaint ID</th>
                <th class="pad">Username</th>
                <th>Title</th>
                <th>Complaint</th>
                <th>Solution</th>
                <th>Status</th>
            </thead>
            <tbody>
                {% for i in complaints %}
                <tr>
                    <td class="pad">
                        {{ i['C_ID'] }}
                    </td>
                    <td class="pad">
                        {{ i['USERNAME'] }}
                    </td>
                    <td>
                        {{ i['TITLE'] }}
                    </td>
                    <td>
                        {{ i['COMPLAINT'] }}
                    </td>
                    <td>
                        {{ i['SOLUTION'] }}
                    </td>
                    <td>
                        {% if i['STATUS'] == 1 %}
                        Completed
                        {% else %}
                        Not Completed
                        {% endif %}
                    </td>
                </tr>
                {% endfor %}
            </tbody>

        </table>

        <br>
        <center>

            <div class="fordashboarddetails">

```



```

        <button type="button" class="collapsible">Solve an Issue
    ⚡ </button>
    <div class="content">
        <br>
        <form action="/updatecomplaint" method="post">
            <div class="forform">
                <div class="textinformleft">
                    Complaint ID
                </div>
                <div class="textinformright">
                    <input type="name" name="cid">
                </div>
            </div>
            <div class="forform">
                <div class="textinformleft">
                    Solution
                </div>
                <div class="textinformright">
                    <input type="text" name="solution">
                </div>
            </div>

            <br>
            <br>
            <div>
                <button class="forbutton" type="submit"> Submit
    </button>

        </div>
    </form>
    <br>
</div>

    </div>
</center>
</div>

</div>

{% endblock %}

Agents.html
{% extends 'base.html' %}

{% block head %}

```



```
<title>  
  Dashboard  
</title>
```

```

        <th>Username</th>
        <th>Email</th>
        <th>Phone</th>
        <th>Domain</th>
        <th>Status</th>
    </thead>
    <tbody>
        {% for i in agents %}
        <tr>
            <td class="pad">
                {{ i['NAME'] }}
            </td>
            <td class="pad">
                {{ i['USERNAME'] }}
            </td>
            <td>
                {{ i['EMAIL'] }}
            </td>
            <td>
                {{ i['PHN'] }}
            </td>
            <td>
                {{ i['DOMAIN'] }}
            </td>
            <td>
                {% if i['STATUS'] == 1 %}
                Assigned to job
                {% elif i['STATUS'] == 0 %}
                not Available
                {% else %}
                Available
                {% endif %}
            </td>
        </tr>
        {% endfor %}
    </tbody>
</table>

<br>
<center>

    <div class="fordashboarddetails">

        <button type="button" class="collapsible">Add new agent
+ </button>

        <div class="content">
            <br>
            <form action="/addnewagent" method="post">
                <div class="for-form">
                    <div class="textinformleft">
                        Username

```

`</div>`

```
<br>
```

```
        </div>

        </div>
    </center>

    </div>

</div>

{% endblock %}

Dashboard.html
```

```
{% extends 'base.html' %}
```

```
{% block head %}
```

```
<title>
    Dashboard
</title>
```

```
{% endblock %}
```

```
{% block body %}
```

```
<!-- things
    div 1 welcome
jetson,    sign out
    div 2 your
complaints status
```

```
add new complaint -->
```

```
<br>
```

```
<!-- <br>
```

```
{% for i in range(11) %}
```

```
    {{ i }}
```

```
{% endfor %}
```

```
<br>
```

```
{% for i in complaints %}
```

```
{{ i['USERNAME'] }}
```

```
<br>
```

```
{% for j in i.values() %}
```

```
    {{ j }}
```

```
{% endfor %}
```

</tbody>



Login.html

| |
|------|
| |
|------|

```
<div>
    New user? <a href="/signup">Sign up</a>
</div>
<br>
</div>

</div>
</div>

{% endblock %}
```

Signup.html

```
{% extends 'base.html' %}

{% block head %}

<title>
    Sign Up
</title>
{% endblock %}

{% block body %}

<div class="forpadding">

    <!-- for box of the signup form -->
    <div class="sign">
        <div>
            <p class="fortitle">
Register Now!!
            </p>
            <hr>
            <form action="/signup" method="post">
                <div class="forform">
                    <div class="textinformleft">
                        Username
                    </div>
                    <div class="textinformright">
                        <input type="name" name="username">
                    </div>
                </div>
                <div class="forform">
                    <div class="textinformleft">
                        Name
                    </div>
                    <div class="textinformright">
                        <input type="name" name="name">
                    </div>
                </div>
            </form>
        </div>
    </div>
</div>
```



```
</div>
```

{% endblock %}

Tickets.html

```
{% extends 'base.html' %}

{% block head %}

<title>
    Agent Dashboard
</title>

{% endblock %}

{% block body %}

<!-- things
div 1 welcome
jetson,    sign
out       div 2
your complaints
status

add new complaint -->
<br>
<!-- <br>
{% for i in range(11) %}
    {{ i }}
{% endfor %}

<br>
{% for i in complaints %}
    {{ i['USERNAME'] }}
<br>
{% for j in i.values() %}
    {{ j }}
{% endfor %}
<br>
{% endfor %} -->

<div class="fordashboardtop">
    <div class="fordashboardtopelements1">
        Welcome Admin,
    </div>
```




```

<div class="fordashboardtopelements2">
  <a href="/login"><button class="forbutton">Sign out</button></a>
</div>

</div>
<br>
<div class="outerofdashdetails">

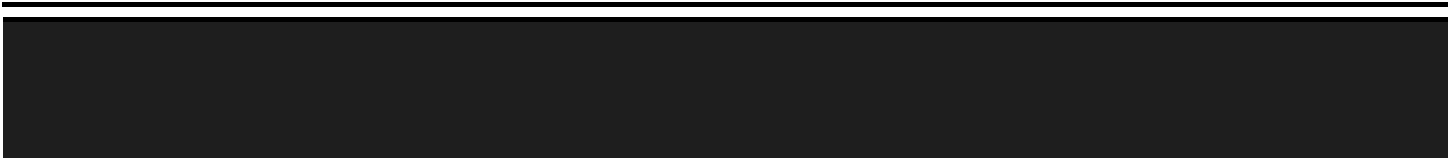
  <div class="fordashboarddetails">
    <br>
    <!-- table of customers complaints -->
    <table class="fortable">
      <thead>
        <th>Complaint ID</th>
        <th class="pad">Username</th>
        <th>Title</th>
        <th>Complaint</th>
        <th>Solution</th>
        <th>Status</th>
      </thead>
      <tbody>
        {% for i in complaints %}
          <tr>
            <td>{{ i['C_ID'] }}</td>
            <td class="pad">
              {{ i['USERNAME'] }}
            </td>
            <td>
              {{ i['TITLE'] }}
            </td>
            <td>
              {{ i['COMPLAINT'] }}
            </td>
            <td>
              {{ i['SOLUTION'] }}
            </td>
            <td>
              {% if i['STATUS'] == 1 %}
                Completed
              {% else %}
                Not Completed
              {% endif %}
            </td>
          </tr>
        {% endfor %}
      </tbody>

    </table>

    <br>
    <center>

```

```
<div class="fordashboarddetails">
```

```

<button type="button" class="collapsible">Assign an agent
⚡ </button>

<div class="content">
  <br>
  <form action="/assignagent" method="post">
    <div class="forform">
      <div class="textinformleft">
        Complaint ID
      </div>
      <div class="textinformright">
        <input type="name" name="ccid">
      </div>
    </div>
    <div class="forform">
      <div class="textinformleft">
        <label for="agent">Choose an agent:</label>
      </div>
      <div class="textinformright">
        <select name="agent" id="agent">
          {% for i in freeagents %}
            <option value={{ i['USERNAME'] }}>{{
i['USERNAME'] }}</option>
          {% endfor %}
        </select>
      </div>
    </div>

    <br>
    <br>
    <div>
      <button class="forbutton" type="submit"> Submit
    </div>
  </form>
  <br>
</div>

</div>

</div>

{% endblock %}

```

App.py:

```

from flask import Flask, render_template, request, redirect, session, url_for
from flask import Flask, render_template, request, redirect, session
import ibm_db
import re

app = Flask(__name__)

import ibm_db
import re

app = Flask(__name__)

# for connection
# conn= ""

app.secret_key = 'a'
print("Trying to connect...")
conn = ibm_db.connect("DATABASE=bludb;HOSTNAME=824dfd4d-99de-440d-9991-629c01b3832d.bs2io90l08kqb1od8lcg.databases.appdomain.cloud;PORT=30119;SECURITY=SSL;SSLServerCertificate=DigiCertGlobalRootCA.crt;UID=qvk70423;PWD=saDlGasU4iQy1yvk;", '', '')
print("connected..")

@app.route('/signup', methods=['GET', 'POST'])
def signup():
    global userid
    msg = ''
    if request.method == 'POST':
        username = request.form['username']
        name = request.form['name']
        email = request.form['email']
        phn = request.form['phn']
        password = request.form['pass']
        repass = request.form['repass']
        print("inside checking")
        print(name)
        if len(username) == 0 or len(name) == 0 or len(email) == 0 or len(phn) == 0 or len(password) == 0 or len(repass) == 0:
            msg = "Form is not filled completely!!"
            print(msg)
            return render_template('signup.html', msg=msg)
        elif password != repass:
            msg = "Password is not matched"
            print(msg)
            return render_template('signup.html', msg=msg)
        elif not re.match(r'[a-z]+', username):

```

```

        print(msg)                return
render_template('signup.html', msg=msg)                elif
not re.match(r'^@+@[^@]+\.[^@]+', email):
    msg = 'Invalid email'
print(msg)                return
render_template('signup.html', msg=msg)                elif
not re.match(r'[A-Za-z]+', name):
    msg = "Enter valid name"
print(msg)                return
render_template('signup.html', msg=msg)                elif
not re.match(r'[0-9]+', phn):                msg = "Enter
valid phone number"                print(msg)
return render_template('signup.html', msg=msg)
    sql = "select * from users where
username = ?"                stmt = ibm_db.prepare(conn,
sql)                ibm_db.bind_param(stmt, 1, username)
ibm_db.execute(stmt)                account =
ibm_db.fetch_assoc(stmt)                print(account)
if account:
    msg = 'Acccount already exists'
else:
    userid = username                insert_sql =
"insert into users values(?,?,?,?,?)"
    prep_stmt = ibm_db.prepare(conn, insert_sql)
    ibm_db.bind_param(prepare_stmt, 1, username)
    ibm_db.bind_param(prepare_stmt, 2, name)
    ibm_db.bind_param(prepare_stmt, 3, email)
    ibm_db.bind_param(prepare_stmt, 4, phn)
    ibm_db.bind_param(prepare_stmt, 5, password)
    ibm_db.execute(prepare_stmt)                print("successs")
    msg = "succesfully signed up"
    return render_template('dashboard.html',                msg=msg,
name=name)                else:
    return
render_template('signup.html')

@app.route('/dashboard')
def dashboard():
    return
render_template('dashboard.html')

@app.route('/login', methods=["GET",
"POST"]) def login():    global userid
msg = ''    if request.method == 'POST':
    username = request.form['username']
    userid = username

```

```
password = request.form['pass']
```

```

        if userid == 'admin' and password == 'admin':
            print("its admin")
return render_template('admin.html')
else:
    sql = "select * from agents where username = ? and password =
?"
    stmt = ibm_db.prepare(conn, sql)
ibm_db.bind_param(stmt, 1, username)          ibm_db.bind_param(stmt, 2,
password)          ibm_db.execute(stmt)          account =
ibm_db.fetch_assoc(stmt)          print(account)          if account:
            session['Loggedin'] = True
session['id'] = account['USERNAME']
userid = account['USERNAME']
session['username'] = account['USERNAME']
msg = 'logged in successfully'

        # for getting complaints details          sql =
"select * from complaints where assigned_agent = ?"
complaints = []          stmt = ibm_db.prepare(conn, sql)
ibm_db.bind_param(stmt, 1, username)
ibm_db.execute(stmt)          dictionary =
ibm_db.fetch_assoc(stmt)          while dictionary != False:
complaints.append(dictionary)          dictionary =
ibm_db.fetch_assoc(stmt)          print(complaints)
return render_template('agentdash.html',
name=account['USERNAME'], complaints=complaints)

    sql = "select * from users where username = ? and password =
?"
    stmt = ibm_db.prepare(conn, sql)
ibm_db.bind_param(stmt, 1, username)          ibm_db.bind_param(stmt,
2, password)          ibm_db.execute(stmt)          account =
ibm_db.fetch_assoc(stmt)          print(account)          if account:
            session['Loggedin'] = True
session['id'] = account['USERNAME']
userid = account['USERNAME']
session['username'] = account['USERNAME']
msg = 'logged in successfully'

        # for getting complaints details          sql =
"select * from complaints where username = ?"
complaints = []          stmt = ibm_db.prepare(conn, sql)
ibm_db.bind_param(stmt, 1, username)
ibm_db.execute(stmt)

```

```
dictionary = ibm_db.fetch_assoc(stmt)
```





```

        while dictionary != False:
            # print "The ID is : ", dictionary["EMPNO"]
# print "The Name is : ", dictionary[1]
complaints.append(dictionary)                dictionary =
ibm_db.fetch_assoc(stmt)                    print(complaints)
return render_template('dashboard.html',
name=account['USERNAME'], complaints=complaints)
else:
    msg = 'Incorrect user credentials'
return render_template('dashboard.html', msg=msg)        else:
return render_template('login.html')

@app.route('/addnew', methods=["GET", "POST"]) def
add():
    if request.method == 'POST':                title =
request.form['title']                des = request.form['des']
try:                sql = "insert into
complaints(username,title,complaint) values(?,?,?)"
stmt = ibm_db.prepare(conn, sql)
ibm_db.bind_param(stmt, 1, userid)
ibm_db.bind_param(stmt, 2, title)
ibm_db.bind_param(stmt, 3, des)                ibm_db.execute(stmt)
except:
    print(userid)                print(title)
print(des)                print("cant insert")                sql
= "select * from complaints where username = ?"
complaints = []                stmt = ibm_db.prepare(conn, sql)
ibm_db.bind_param(stmt, 1, userid)
ibm_db.execute(stmt)                dictionary =
ibm_db.fetch_assoc(stmt)                while dictionary != False:
    # print "The ID is : ", dictionary["EMPNO"]
# print "The Name is : ", dictionary[1]
complaints.append(dictionary)                dictionary =
ibm_db.fetch_assoc(stmt)                print(complaints)
return render_template('dashboard.html', name=userid,
complaints=complaints)

@app.route('/agents') def
agents():

```

```
sql = "select * from agents"
```



```

        agents = []      stmt = ibm_db.prepare(conn, sql)
    ibm_db.execute(stmt)      dictionary =
    ibm_db.fetch_assoc(stmt)  while dictionary != False:
    agents.append(dictionary)  dictionary =
    ibm_db.fetch_assoc(stmt)  return
    render_template('agents.html', agents=agents)

@app.route('/addnewagent', methods=["GET",
"POST"]) def addagent():    if request.method
== 'POST':        username =
request.form['username']    name =
request.form['name']        email =
request.form['email']       phone =
request.form['phone']       domain =
request.form['domain']      password =
request.form['password']    try:
sql = "insert into agents values(?,?,?,?,?,2)"
stmt = ibm_db.prepare(conn, sql)
ibm_db.bind_param(stmt, 1, username)
ibm_db.bind_param(stmt, 2, name)
ibm_db.bind_param(stmt, 3, email)
ibm_db.bind_param(stmt, 4, phone)
ibm_db.bind_param(stmt, 5, password)
ibm_db.bind_param(stmt, 6, domain)
ibm_db.execute(stmt)        except:
print("cant insert")        sql = "select *
from agents"        agents = []      stmt =
ibm_db.prepare(conn, sql)
ibm_db.execute(stmt)        dictionary =
ibm_db.fetch_assoc(stmt)    while
dictionary != False:
agents.append(dictionary)    dictionary
= ibm_db.fetch_assoc(stmt)

        return render_template('agents.html', agents=agents)

@app.route('/updatecomplaint', methods=["GET", "POST"])
def updatecomplaint():
    if request.method == 'POST':
        cid = request.form['cid']        solution
= request.form['solution']        try:
            sql = "update complaints set solution =? where c_id = ?
and assigned_agent=?"        stmt = ibm_db.prepare(conn, sql)

```

```
ibm_db.bind_param(stmt, 1, solution)
```



```

        ibm_db.bind_param(stmt, 2, cid)
    ibm_db.bind_param(stmt, 3, userid)
    ibm_db.execute(stmt)          sql = "update agents set status
=3 where username=?"          stmt = ibm_db.prepare(conn,
sql)          ibm_db.bind_param(stmt, 1, userid)
    ibm_db.execute(stmt)          except:          print("cant
insert")          sql = "select * from complaints where
assigned_agent = ?"          complaints = []          stmt =
    ibm_db.prepare(conn, sql)          ibm_db.bind_param(stmt, 1,
userid)          ibm_db.execute(stmt)          dictionary =
    ibm_db.fetch_assoc(stmt)          while dictionary != False:
        complaints.append(dictionary)
        dictionary = ibm_db.fetch_assoc(stmt)
# print(complaints)          return
render_template('agentdash.html', name=userid,
complaints=complaints)

@app.route('/tickets') def
tickets():
    sql = "select * from complaints"
    complaints = []          stmt =
    ibm_db.prepare(conn, sql)
    ibm_db.execute(stmt)          dictionary =
    ibm_db.fetch_assoc(stmt)          while
dictionary != False:
        complaints.append(dictionary)
    dictionary = ibm_db.fetch_assoc(stmt)          sql =
    "select username from agents where status <>
1"          freeagents = []          stmt =
    ibm_db.prepare(conn, sql)
    ibm_db.execute(stmt)          dictionary
= ibm_db.fetch_assoc(stmt)          while
dictionary != False:
        freeagents.append(dictionary)
    dictionary = ibm_db.fetch_assoc(stmt)
    print(freeagents)
    return render_template('tickets.html', complaints=complaints,
freeagents=freeagents)

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

```



```
print(ccid)
```




```

        print(agent)
    try:
        sql = "update complaints set assigned_agent =? where c_id =
        ?"
        stmt = ibm_db.prepare(conn, sql)
        ibm_db.bind_param(stmt, 1, agent)
        ibm_db.bind_param(stmt, 2, ccid)
        ibm_db.execute(stmt)
        sql = "update
        agents set status =1 where username = ?"
        stmt =
        ibm_db.prepare(conn, sql)
        ibm_db.bind_param(stmt, 1,
        userid)
        ibm_db.execute(stmt)
    except:
        print("cant update")
        return redirect(url_for('tickets'))

@app.route('/about') def about():
    return render_template('about.html')

@app.route('/privacyterms') def
privacyterms():
    return render_template('privacyterms.html')

if __name__ == "__main__":
    app.run(debug=True)

```

