**SMARTINTERNZ**

VMware Build-A-Thon

# CUSTOMER CARE REGISTRY

**SUBMITTED BY:**

**SANDESH TIWARI S**

B.E in Computer Science

Don Bosco Institute of Technology, Bengaluru - 560074

email: sandeshtiwari16@gmail.com

**INTRODUCTION**

Overview

This Application has been developed to help the customer in processing their complaints. The customers can raise the ticket with a detailed description of the issue. An Agent will be assigned to the Customer to solve the problem. Whenever the agent is assigned to the customer they will be notified with an email alert. Customers can view the status of the ticket till the service is provided.

Admin: The main roles and responsibilities of the admin is to take care of the whole process. Starting from Admin login followed by the agent creation and assigning the customers complaints. Finally, He will be able to track the work assigned to the agent and notification will be sent to the customer.

User: They can register for an account. After the login, they can create a complaint with a description of the problem they are facing. Each user will be assigned an agent. They can view the status of their complaint.

Purpose

The main use of this project is to help the customer in processing their complaints. The customers can raise the ticketof their issues and the problem will be solved by the organization.

**LITERATURE SURVEY**

Existing Problem

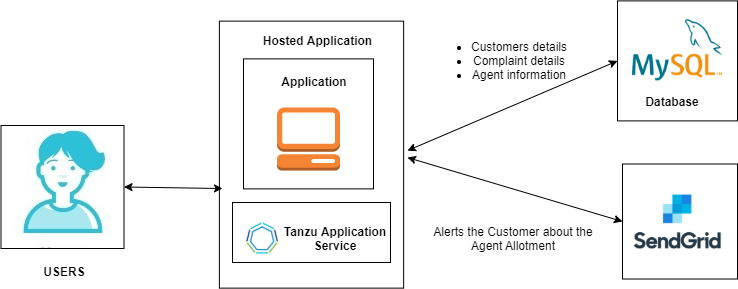
The Customers dealing with problems are not able to approach the organization for the solution. They are also not able to contact the organization. And, finally they are left over with a contact us form or a feedback form, where they are unable to report their problems.

Proposed Solution

Instead of this old school solution, I have come up with an idea of Customer Care Registry where the Customer can make their account and register their complaints by raising ticket. All the compplaints will be stored in a registry and the admin will assign agents to each customer to solve their problems. In this manner, every Customer's problem will be heard and also solved.

**THEORITICAL ANALYSIS**

Block Diagram



Hardware / Software Designing

Minimum RAM: 4GB

Hard Disk: 40GB

Processor: Intel Core i5

Operating System: Windows 10

Library: flask, flask\_mysqldb

Programming Language: Python

Tools: Spyder

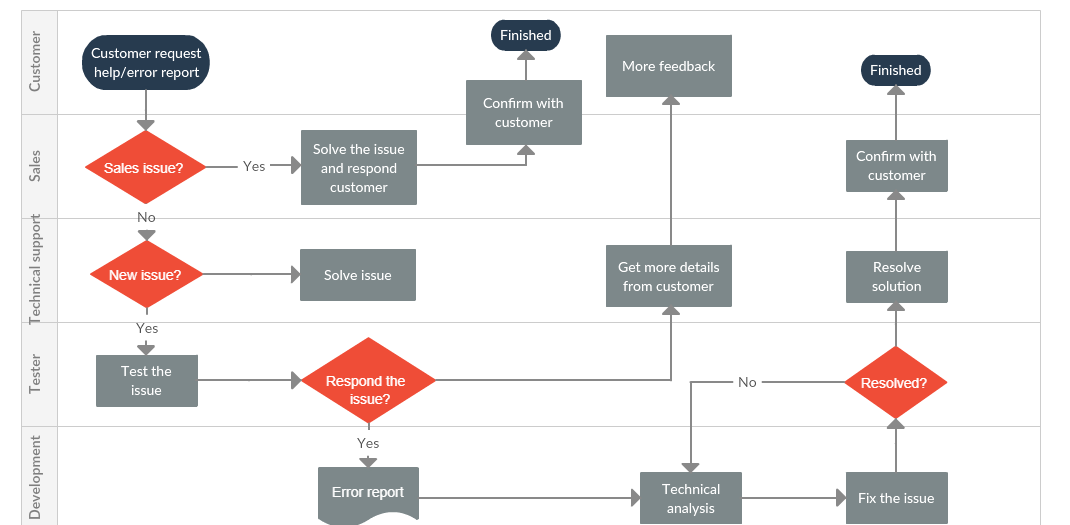
Skills Required: HTML, Bootstrap, MySQL, Python-Flask, Cloud Foundry, Tanzu Application Service, REST API's

**EXPERIMENTAL INVESTIGATION**

Some of the common investigations or analysis that were made while working on solution were:

* How well our product or service matches customer needs
* The value for money we offer
* Our efficiency and reliability in fulfilling orders
* The professionalism, friendliness and expertise of our employees
* How well we keep our customers informed
* User-friendly GUI
* How to deal with Customer complaints

**FLOWCHART**

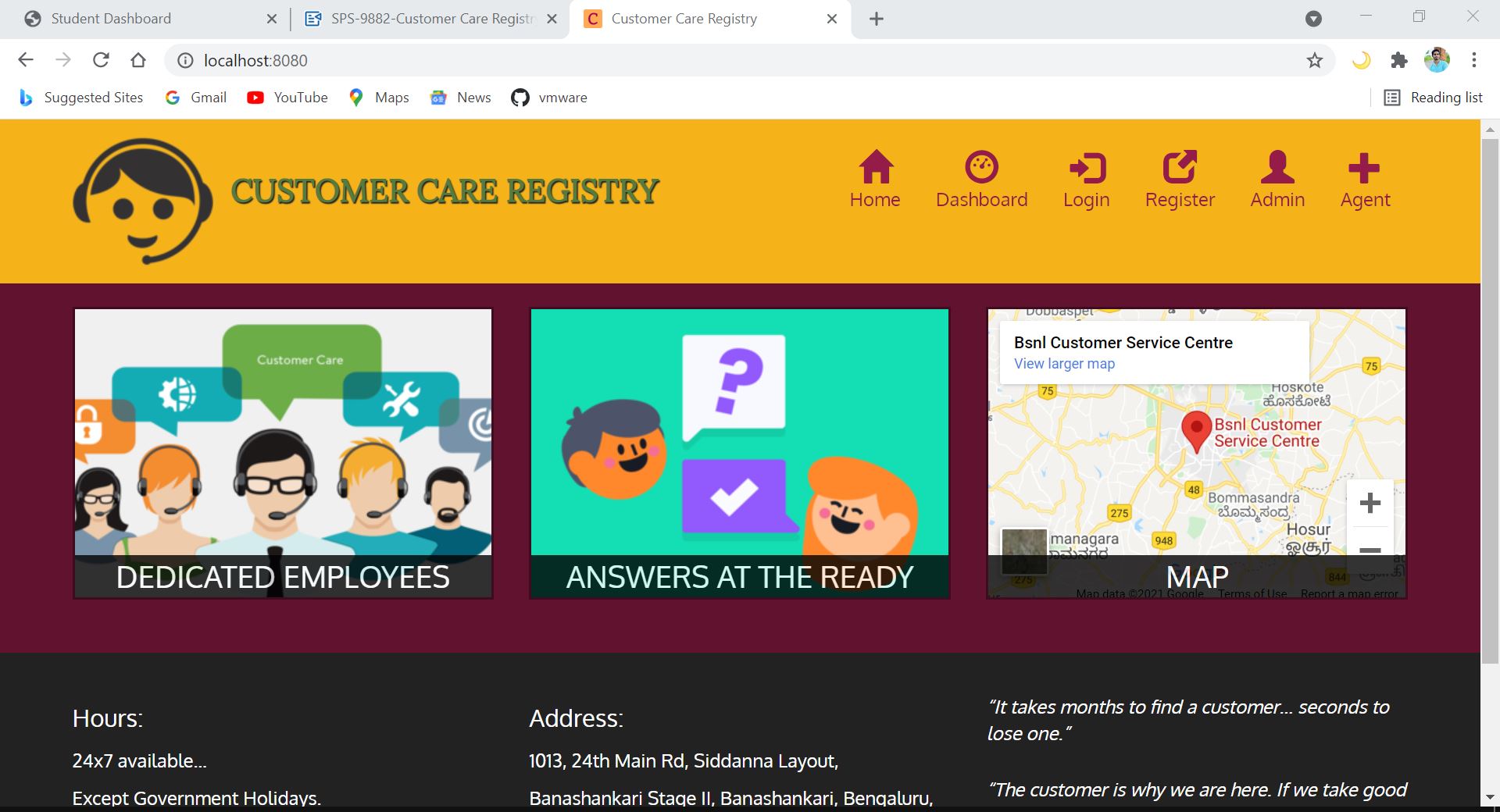


**RESULT**

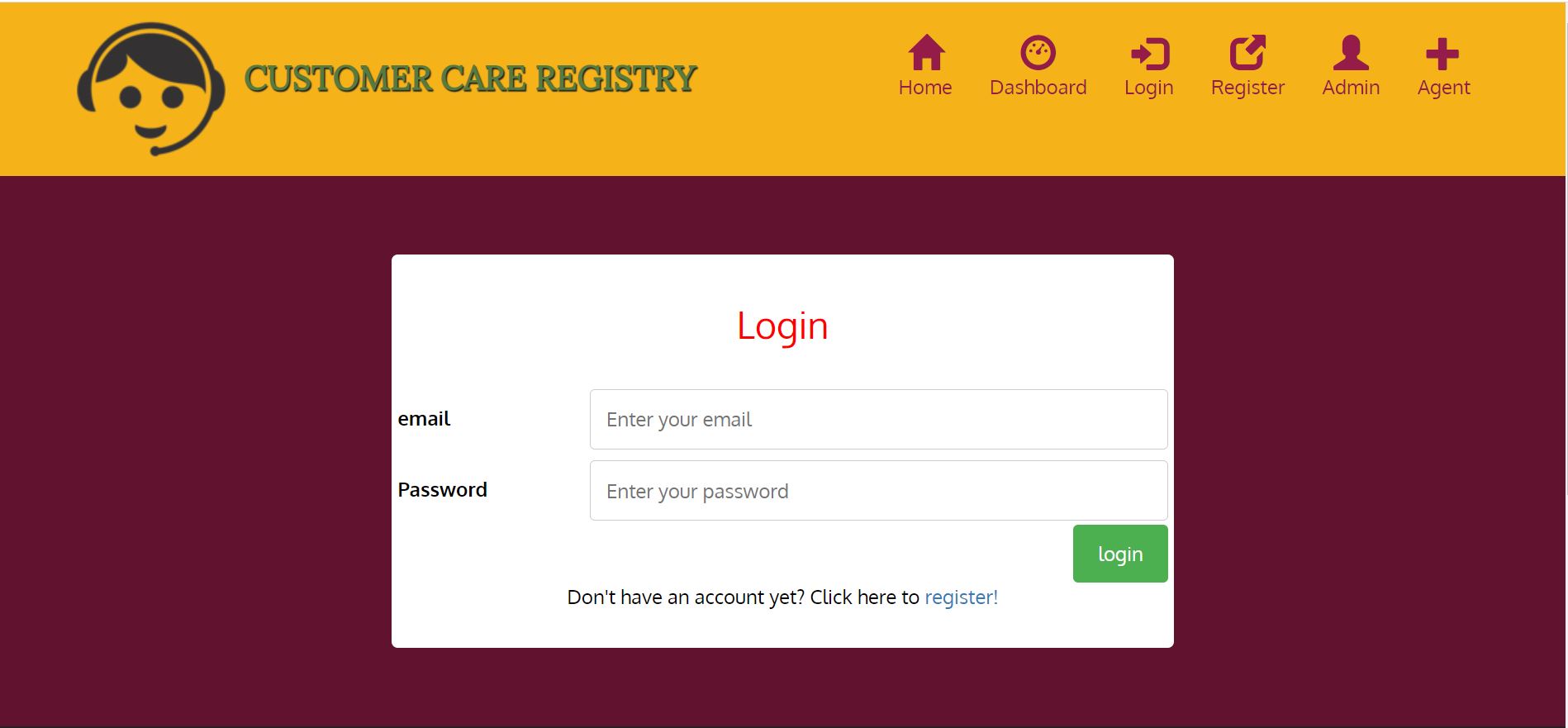
This Application helps the customer in processing their complaints. The customers can raise the ticket with a detailed description of the issue. An Agent will be assigned to the Customer to solve the problem. Whenever the agent is assigned to the customer they will be notified with an email alert. Customers can view the status of the ticket till the service is provided.

Screenshots

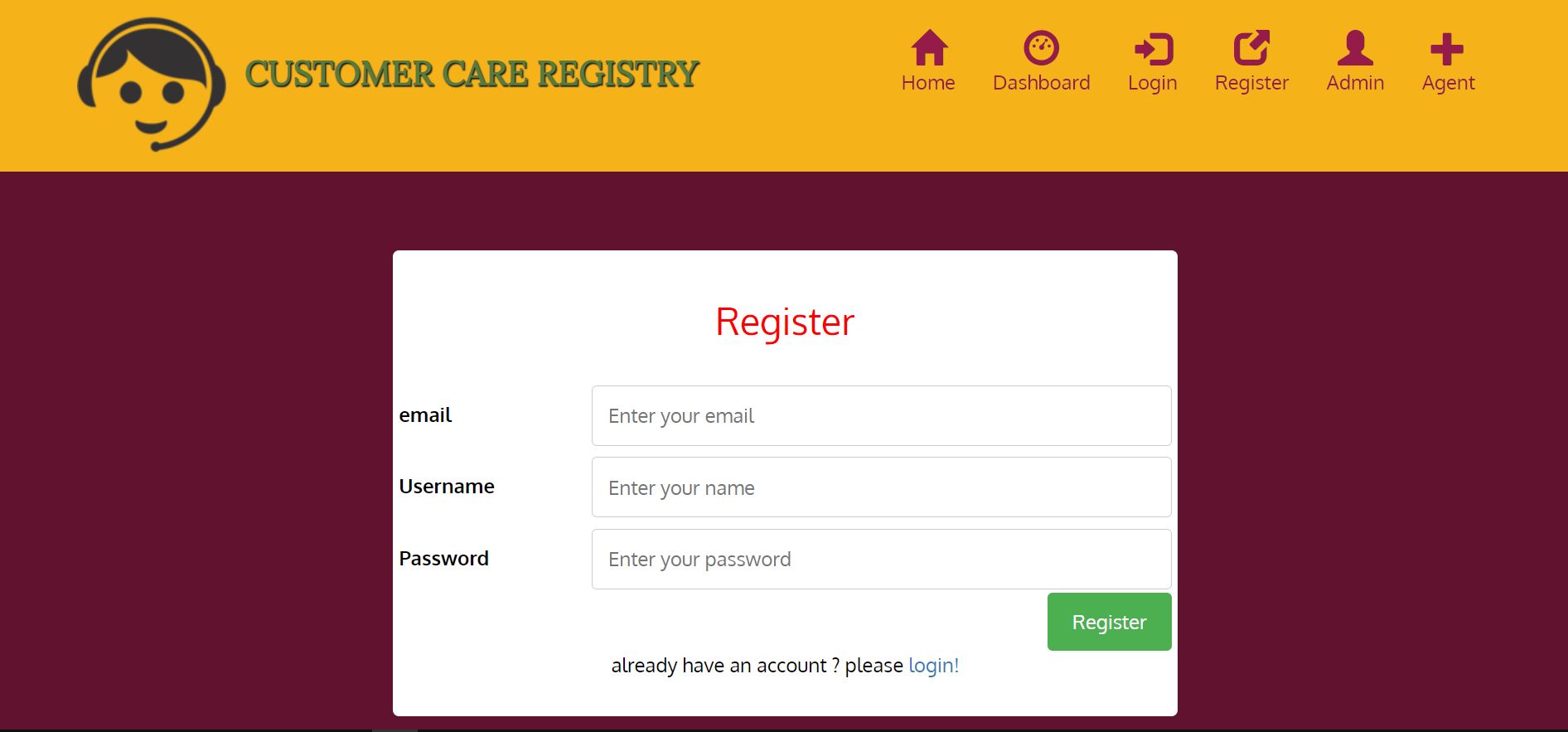
Home Page:



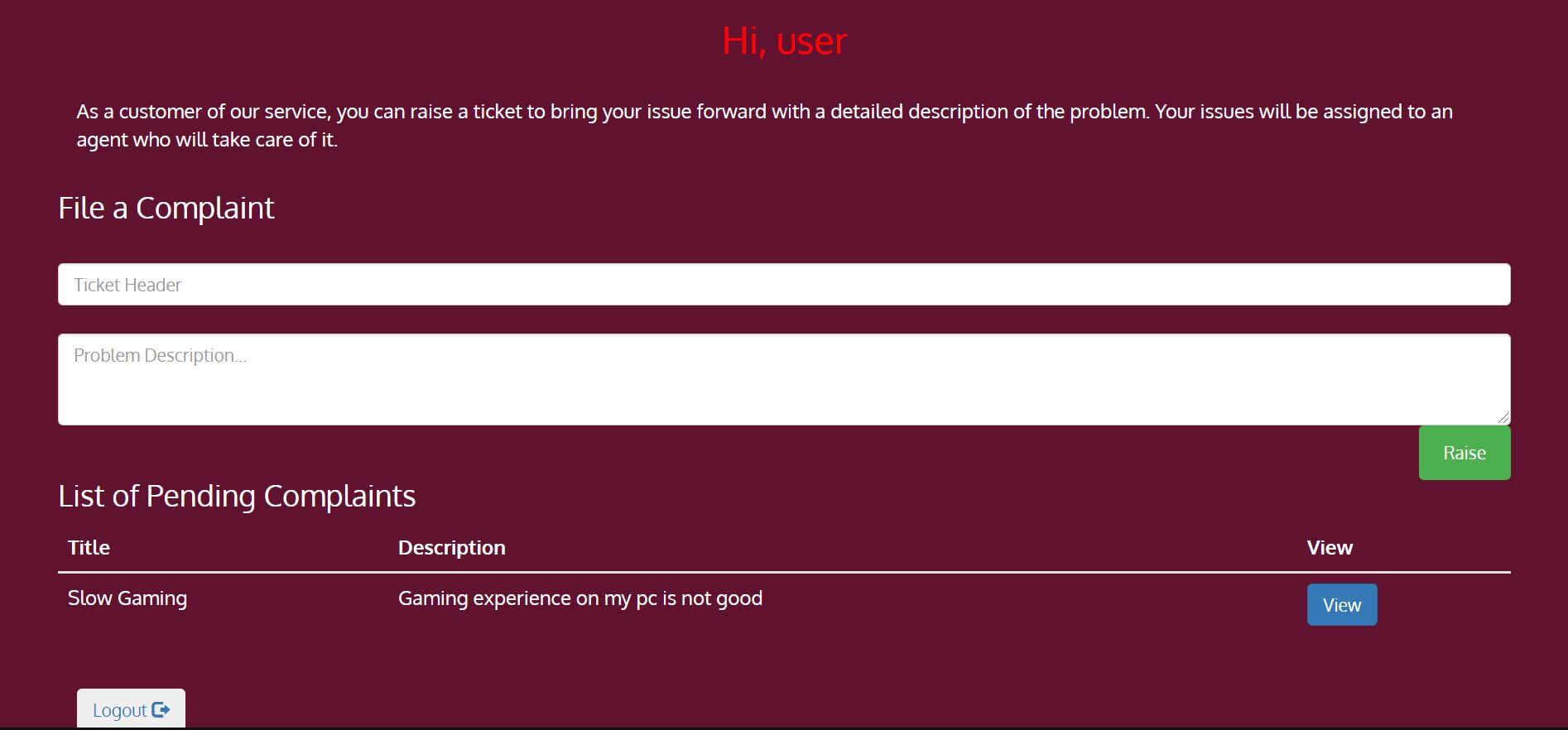
Login Page:



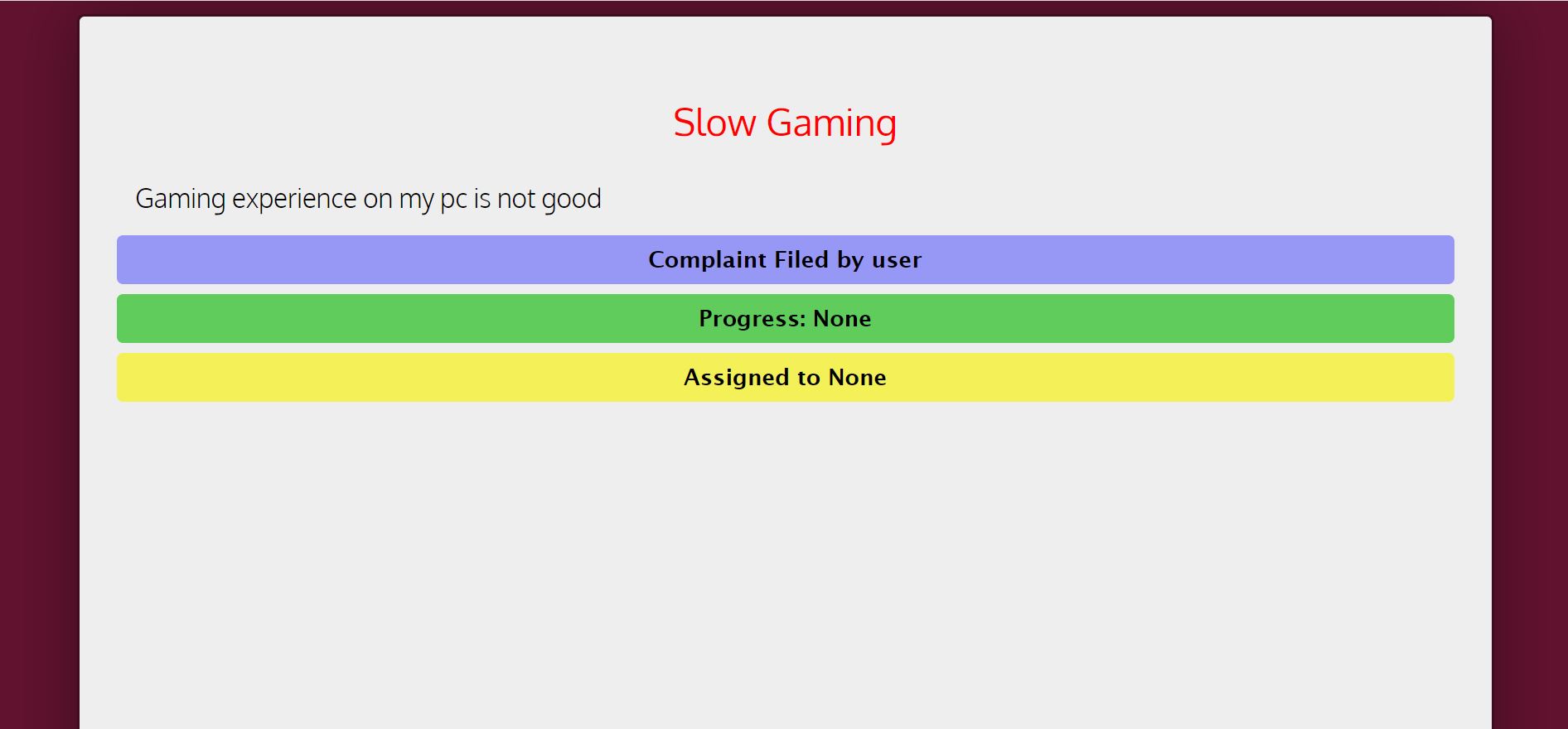
Register Page:



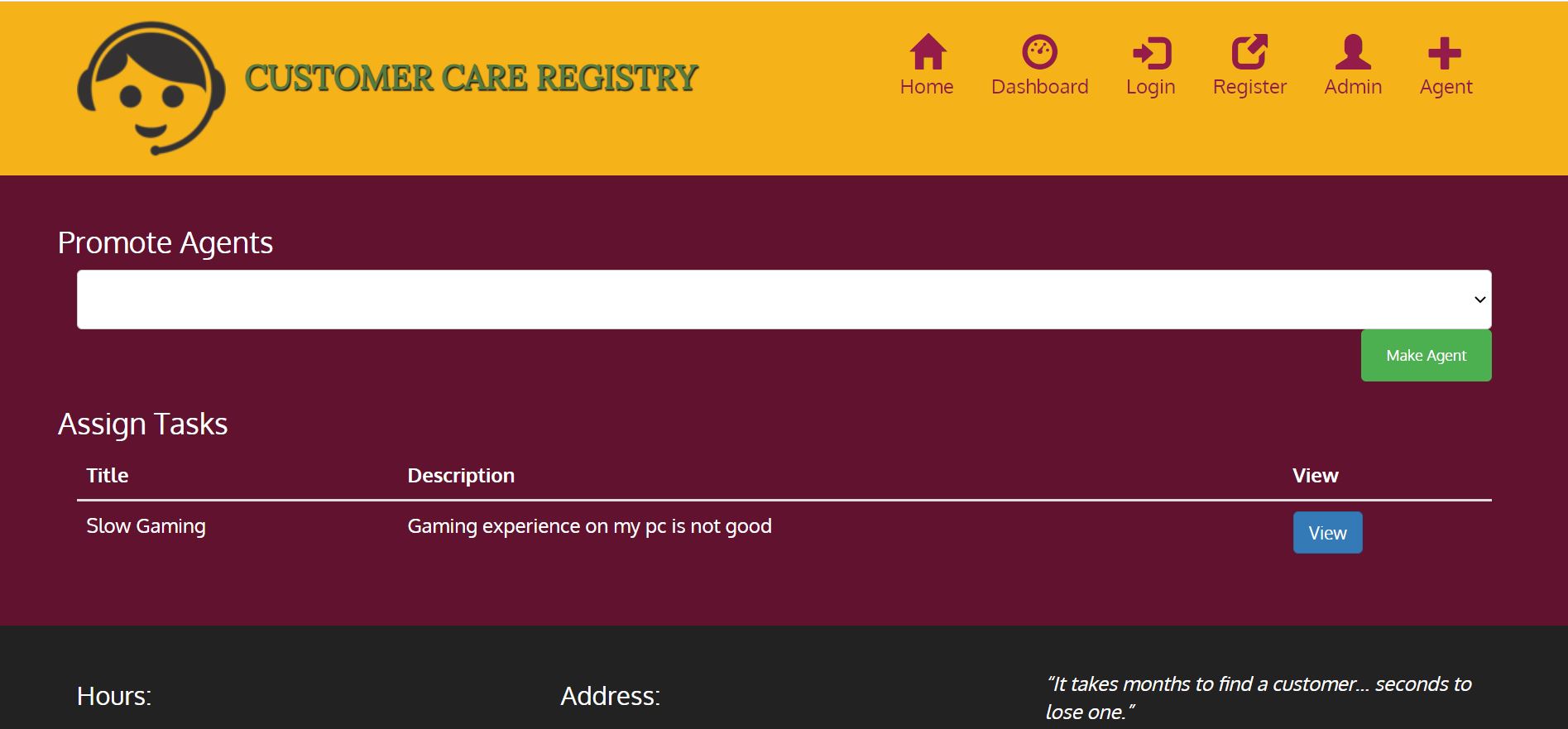
User Dashboard Page:



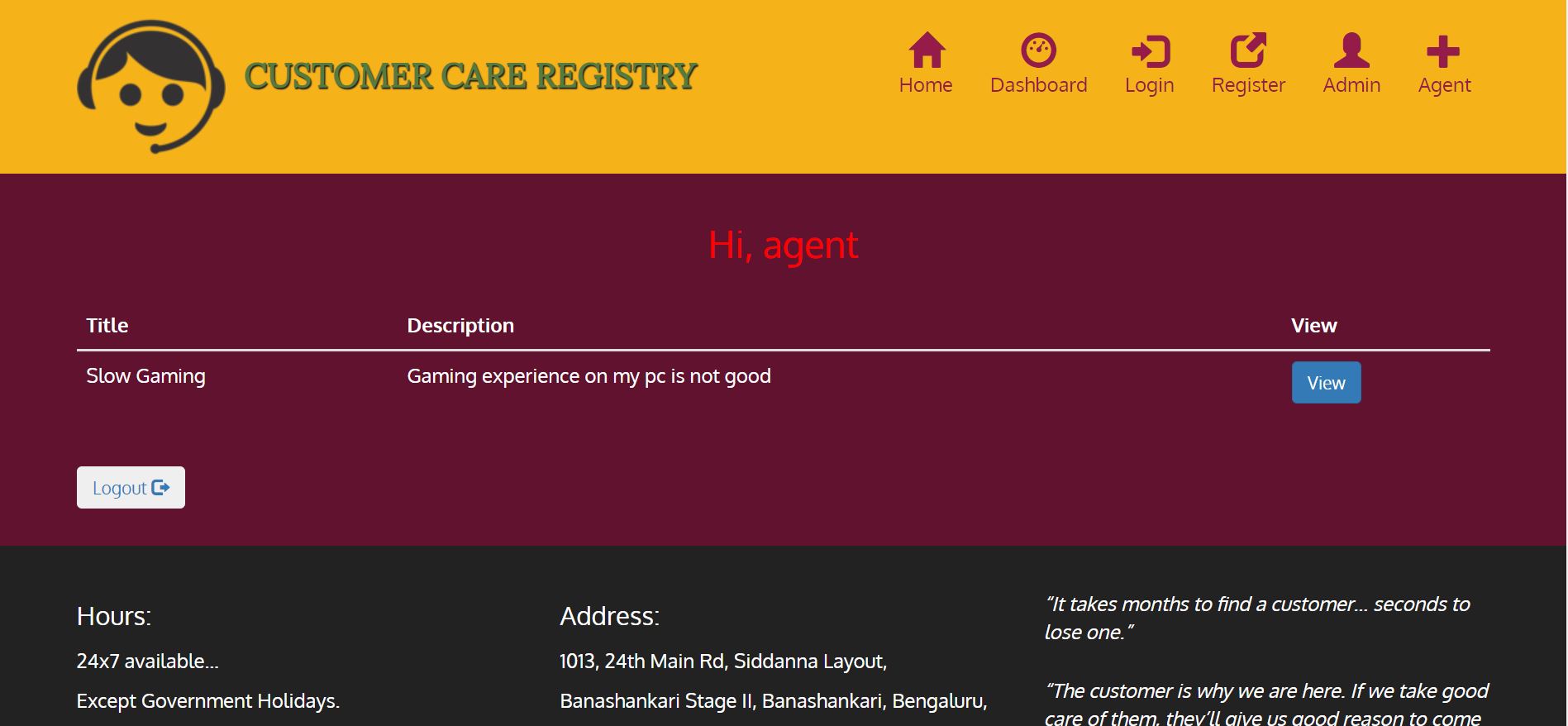
Tickets View Page:



Admin Panel Page:



Agent Dashboard Page:



**ADVANTAGES & DISADVANTAGES**

Advantages:

* Build Loyalty
* Increase Referrals
* Celebrated for Customer service
* Can improve Survey of Customer

Disadvantages:

* Lack of Innovation
* Ever-changing Customer needs
* May become self-serving

**APPLICATIONS**

Customer Care Registry is a fully scalable customer support system that is easy to setup, supports all customer communication channels and saves time and money. Its simplicity, ease of set-up and clean UI are all features which can be praised by users. It can help ensure that customer support tickets are dealt with promptly, minimising response times. Also, can be used in varied sectors of Organization's with different functionalities.

**CONCLUSION**

In Conclusion, Customer care, involves the use of basic ethics and any company who wants to have success and grow, needs to remember, that in order to do so, it must begin with establishing a code of ethics in regards to how each employee is to handle the dealing with customers. Customers are the heart of the company. Customer care involves, the treatment, care, loyalty, trust - the employee should extend to the consumer, as well in life. This concept can be applied to so much more than just customer care. People need to treat others with respect and kindness, people should try to take others into consideration when making any decision. If more people were to practice this policy, chances are the world would be a better, more understanding place for all to exist.

**FUTURE SCOPE**

Enhancements can be made in the future. New features like Customer call facility can be added. Customer thermometer can be added in the app, which will tell how satisfied customer is with the app. AI powered self service can also be offered in the future.

**BIBILOGRAPHY**

Websites

* [remotemysql](https://www.remotemysql.com)
* [geeksforgeeks](https://geeksforgeeks.org)

**APPENDIX**

1. Source Code

#importing modules

from flask import Flask, render\_template, request, redirect, url\_for, session

from flask\_mysqldb import MySQL

import MySQLdb.cursors

import re

import smtplib

#app config

app = Flask(\_\_name\_\_)

app.config['MYSQL\_HOST'] = 'remotemysql.com'

app.config['MYSQL\_USER'] = 'XnUpVja8h7'

app.config['MYSQL\_PASSWORD'] = 'Hmh4Iwk4H7'

app.config['MYSQL\_DB'] = 'XnUpVja8h7'

mysql = MySQL(app)

app.secret\_key = 'apple'

#routes

# home

@app.route("/")

def index():

return render\_template("index.html")

#dashboard

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

def home():

if ('user' not in session.keys()) or (session['user'] == None):

return redirect(url\_for('login'))

else:

cursor = mysql.connection.cursor()

cursor.execute("SELECT \* FROM User WHERE id = % s",[session['user']])

userdetails = cursor.fetchone()

if userdetails[4] == 2:

return render\_template("home.html",user=userdetails)

if userdetails[4] == 5:

return render\_template("home.html",user=userdetails)

elif userdetails[4] == 1:

cursor.execute("SELECT \* FROM Tickets WHERE agent=%s",[session['user']])

tickets = cursor.fetchall()

return render\_template("home.html",user=userdetails,tickets=tickets)

else:

if request.method == "POST":

title = request.form['title']

description = request.form['description']

cust\_id = session['user']

cursor = mysql.connection.cursor()

cursor.execute("SELECT username FROM User WHERE id = % s",[session['user']])

username = cursor.fetchone()

cursor.execute("INSERT INTO Tickets(customer,customer\_name,title,description) VALUES(%s,%s,%s,%s)",(cust\_id,username,title,description))

mysql.connection.commit()

cursor.execute("SELECT \* FROM User WHERE id = % s",[session['user']])

userdetails = cursor.fetchone()

cursor.execute("SELECT \* FROM Tickets WHERE customer = %s",[session['user']])

tickets = cursor.fetchall()

return render\_template("home.html",msg="Ticket Filed",user=userdetails,tickets=tickets)

cursor = mysql.connection.cursor()

cursor.execute("SELECT \* FROM User WHERE id = % s",[session['user']])

userdetails = cursor.fetchone()

cursor.execute("SELECT \* FROM Tickets WHERE customer = %s",[session['user']])

tickets = cursor.fetchall()

return render\_template("home.html",user=userdetails,tickets=tickets)

# user account registration

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

def register\_account():

msg=''

if request.method == "POST":

username = request.form['username']

email = request.form['email']

password = request.form['password']

cursor = mysql.connection.cursor()

cursor.execute('SELECT \* FROM User WHERE email = % s', (email, ))

userdetails = cursor.fetchone()

print(userdetails)

if userdetails:

msg = 'Account already exists !'

elif not re.match(r'[^@]+@[^@]+\.[^@]+', email):

msg = 'Invalid email address !'

elif not re.match(r'[A-Za-z0-9]+', username):

msg = 'name must contain only characters and numbers !'

else:

cursor.execute("INSERT INTO User(username,email,password,role) VALUES(% s,% s,% s,% s)", (username,email,password,0))

mysql.connection.commit()

msg = 'You have successfully registered !'

mysql.connection.commit()

return redirect(url\_for("login"))

elif request.method == 'POST':

msg = 'Please fill out the form !'

return render\_template('register.html', msg = msg)

# login

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

def login():

msg=''

if request.method == "POST":

email = request.form['email']

password = request.form['password']

cursor = mysql.connection.cursor()

cursor.execute('SELECT \* FROM User WHERE email = % s AND password = % s', (email, password ))

userdetails = cursor.fetchone()

print (userdetails)

if userdetails:

session['loggedin'] = True

session['user'] = userdetails[0]

session['username'] = userdetails[1]

msg = 'Logged in successfully !'

return redirect(url\_for("home"))

else:

msg = 'Incorrect username / password !'

return render\_template("login.html",msg=msg)

return render\_template('login.html', msg = msg)

# ticket detail

@app.route("/ticket/<int:id>",methods=["GET","POST"])

def ticket\_detail(id):

cursor = mysql.connection.cursor()

cursor.execute("SELECT \* FROM Tickets WHERE id = % s",[id])

ticket = cursor.fetchone()

cursor.execute("SELECT \* FROM User WHERE id = % s",[session['user']])

user = cursor.fetchone()

cursor.execute("SELECT \* FROM User WHERE role = 1")

all\_users = cursor.fetchall()

if user is None:

return redirect(url\_for("login"))

if request.method == "POST":

agent = request.form['agent']

print(agent,id)

cursor.execute("SELECT username FROM User WHERE id= % s",(agent,))

agent\_name = cursor.fetchone()

agt=str(agent\_name)

cursor.execute("SELECT customer\_name FROM Tickets WHERE id= % s",[id])

customer\_name = cursor.fetchone()

cust=str(customer\_name)

cursor.execute("UPDATE Tickets SET agent= %s,agent\_name= % s WHERE id = %s",(agent,agent\_name,id))

cursor.execute("UPDATE Tickets SET progress='assigned' WHERE id = %s",[id])

mysql.connection.commit()

cursor.execute("SELECT email FROM User WHERE id=(SELECT customer FROM Tickets WHERE id= % s)",[id])

email = cursor.fetchall()

TEXT = "Hello "+cust+",\n\n"+"Agent "+agt+" is successfully assigned to you.The Agent will contact you soon. To keep a track of your ticket, please visit our website dashboard."

SUBJECT = "Agent Assigned"

message = 'Subject: {}\n\n{}'.format(SUBJECT, TEXT)

server = smtplib.SMTP("smtp.gmail.com",587)

server.starttls()

server.login("customercaretanzu@gmail.com","tanzu@123")

server.sendmail("customercaretanzu@gmail.com", email, message)

return redirect(url\_for("panel"))

return render\_template("details.html",ticket=ticket,user=user,all\_users=all\_users)

# admin register

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

def admin\_register():

if request.method == "POST":

username = request.form['username']

email = request.form['email']

password = request.form['password']

secret\_key = request.form['secret']

if secret\_key == "12345":

cursor = mysql.connection.cursor()

cursor.execute("INSERT INTO User(username,email,password,role) VALUES(% s,% s,% s,% s)",(username,email,password,2))

mysql.connection.commit()

return redirect(url\_for("login"))

else:

return render\_template("admin\_register.html",msg="Invlaid Secret")

return render\_template("admin\_register.html")

#accept ticket

@app.route("/accept/<int:ticket\_id>/<int:user\_id>")

def accept(ticket\_id,user\_id):

cursor = mysql.connection.cursor()

cursor.execute("SELECT \* FROM User WHERE id = % s",[user\_id])

agent = cursor.fetchone()

cursor.execute("SELECT \* FROM Tickets WHERE id = % s",[ticket\_id])

ticket = cursor.fetchone()

if agent[0] == ticket[3]:

cursor.execute("UPDATE Tickets SET progress='accepted' WHERE id = % s",[ticket\_id])

mysql.connection.commit()

return redirect(url\_for("home"))

#delete ticket

@app.route("/delete/<int:ticket\_id>/<int:user\_id>")

def delete(ticket\_id,user\_id):

cursor = mysql.connection.cursor()

cursor.execute("SELECT \* FROM User WHERE id = % s",[user\_id])

agent = cursor.fetchone()

cursor.execute("SELECT \* FROM Tickets WHERE id=% s",[ticket\_id])

ticket = cursor.fetchone()

if agent[0] == ticket[3]:

cursor.execute("DELETE FROM Tickets WHERE id=%s",[ticket\_id])

mysql.connection.commit()

return redirect(url\_for("home"))

# logout

@app.route("/logout")

def logout():

session['user'] = None

return redirect(url\_for("home"))

# run server

if \_\_name\_\_ == "\_\_main\_\_":

app.run(host='0.0.0.0',port=8080,debug=True)

\*\*\*\*\*Some part of the code is skipped, due to page limit issues\*\*\*\*\*