**WEB APPLICATION FOR RSR FEEDS AND FOODS**

## PROJECT WORK I PHASE 1 REPORT

**Submitted by**

## PRAVIN B

### 19CSR142

## RISHI B

**19CSR159**

## ROHITH VIGNESH E

**19CSR165**

***in partial fulfilment of the requirements for the award of the degree of***

# BACHELOR OF ENGINEERING

# IN

# COMPUTER SCIENCE AND ENGINEERING

****

# KONGU ENGINEERING COLLEGE

**(Autonomous)**

**PERUNDURAI ERODE – 638 060**

**AUGUST 2022**

# DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

## KONGU ENGINEERING COLLEGE

**(Autonomous)**

**PERUNDURAI ERODE -638060**

## AUGUST 2022

**BONAFIDE CERTIFICATE**

This is to certify that the project report entitled **WEB APPLICATION FOR RSR FEEDS AND FOODS** is the bonafide record of the project work done by **PRAVIN B (Register No: 19CSR142),** **RISHI B(Register No: 19CSR159), ROHITH VIGNESH E (Register No: 19CSR165)** in the partial fulfillment of the requirements for the award of the Degree of Bachelor of Engineering in Computer Science and Engineering of Anna University Chennai during the year 2021 – 2022.

**SUPERVISOR** **HEAD OF THE DEPARTMENT**

(Signature with seal)

## Date:

Submitted for the end semester viva voce examination held on

## KONGU ENGINEERING COLLEGE

**(Autonomous)**

**PERUNDURAI ERODE – 638 060**

**AUGUST 2022**

## DECLARATION

We affirm that the project report titled **WEB APPLICATION FOR RSR FEEDS AND FOODS** being submitted in partial fulfillment of the requirements for the award of Bachelor of Engineering is the original work carried out by us. It has not formed the part of any other project report or dissertation on the basis of which a degree or award was conferred on an earlier occasion on this or any other candidate.

**PRAVIN B**

**(Reg. No: 19CSR142)**

**RISHI B**

**(Reg. No: 19CSR159)**

**ROHITH VIGNESH E**

**(Reg. No: 19CSR165)**

Date:

I certify that the declaration made by the above candidates is true to the best of my knowledge.

Date: Name and Signature of the Supervisor with seal

**ABSTRACT**

This is an application project for the leading feed production and sales company in Namakkal named **RSR FEEDS AND FOODS**. The company required a website to showcase their products and services. In the company, they produce feeds as per the requirements of the customer’s poultry farm upon successful registration. This process needs customer certificate verification module. To make this efficient, a separate module in the website is added particularly to manage the customer certificate verification process. After successful verification, a team of people will be sent to the customer’s poultry farm to decide the required feed for that farm.

The company also required a warehouse stock checking page to get daily updates on their stocks stored in the godown. This will be helpful in checking the stocks without additional manpower. It requires separate access rights for admin, office users and customers. Only admin and office users has the access to check customer certificate verification module and warehouse updates.

This web application helps to provide an overall view of the company’s products and services for the customers.

**ACKNOWLEDGEMENT**

Owing deeply to the supreme, we extend our thanks to the Almighty, who has blessed us to come out successfully with our project. We take pleasure to express out deep sense of gratitude to our beloved parents.

We express our sincere thanks and gratitude to our beloved Correspondent, **Thiru. P.SACHITHANANDAN** for giving the opportunity to pursue this course. We are extremely thankful with no words of formal nature to the dynamic Principal, **Dr.V.BALUSAMY B.E (Hons), MTech, Ph.D.,** for providing the necessary facilities to complete our work.

We would like to express our sincere gratitude to our respected Head of the Department, **Dr.N.SHANTHI B.E., M.E., Ph.D.,** for providing necessary facilities.

We also express our sincere thanks to our project coordinator, **Ms.S.MOHANAPRIYA M.E.**, Assistant Professor for her constant encouragement for the development of our project.

We take immense pleasure to express our heartful thanks to our supervisor,

**Dr.S.K.NIVETHA** **M.E.,Ph.D.,** Associate Professor for her valuable ideas and suggestions, which have been very helpful in the project. We are grateful to all the faculty members of the Computer Science and Engineering Department for their support.

### TABLE OF CONTENTS

**CHAPTER No. TITLE PAGE No.**

**ABSTRACT iv**

**LIST OF FIGURES viii**

**LIST OF ABBREVIATIONS ix**

### 1. INTRODUCTION 1

1.1 EXISTING SYSTEM 1

1.2 SYSTEM STUDY 2

1.3 OBJECTIVE 2

1.4 SCOPE 2

### 2. GENERAL DESCRIPTION 3

2.1 PROJECT PERSPECTIVE 3

2.2 USER CHARACTERISTICS 3

2.3 DESIGN AND IMPLEMENTATION 3

2.4 CONSTRAINTS

### 3. REQUIREMENTS 4

3.1 FUNCTIONAL REQUIREMENTS 4

3.2 NON-FUNCTIONAL REQIREMENTS 4

3.3 USER INTERFACES 5

### 4. DETAILED DESIGN 6

4.1 HARDWARE AND SOFTWARE

SPECIFICATION 6

4.2 TECHNOLOGIES USED 6

4.3 ARCHITECTURAL DESIGN 7

4.4 FLOWCHART DIAGRAM 8

4.5 USE CASE DIAGRAM 9

### 5. TESTING 11

5.1 SYSTEM TESTING 11

5.2 UNIT TESTING 11

5.3 USER ACCEPTANCE TESTING 11

5.4 INTEGRATION TESTING 12

### 6. CONCLUSION AND FUTURE SCOPE 14

### APPENDIX I 15

**APPENDIX II 28**

### REFERENCES 36

|  |  |  |
| --- | --- | --- |
|  | **LIST OF FIGURES** |  |
| **FIGURE No.** | **FIGURE NAME** | **PAGE No.** |
| 4.1 | ARCHITECTURE DIAGRAM | 7 |
| 4.2 | FLOWCHART DIAGRAM | 8 |
| 4.3 | USE CASE DIAGRAM | 10 |
| 5.1 | ERROR FOR WRONG INPUTS | 12 |
| 5.2 | VALIDATING THE INPUTS | 13 |
| A2.1 | HOMEPAGE | 28 |
| A2.2 | COMMON LOGIN PAGE | 28 |
| A2.3 | PROFILE PAGE | 29 |
| A2.4 | CERTIFICATION FORM PAGE | 29 |
| A2.5 | WAREHOUSE CHECKING PAGE | 30 |
| A2.6 | VERIFICATION PAGE | 30 |
| A2.7 | CONTACT PAGE | 31 |
| A2.8 | ABOUT PAGE | 32 |
| A2.9 | PRODUCT PAGE | 33 |
| A2.10 | ADMIN HOME | 34 |
| A2.11 | DJANGO DATABASE | 35 |

**LIST OF ABBREVIATIONS**

|  |  |  |
| --- | --- | --- |
|  |  |  |
| CSS | - | Cascading Style Sheet |
| HTML | - | Hyper Text Markup Language |
| HTTP | - | Hypertext Transfer Protocol |
| URL | - | Uniform Resource Locator |

**CHAPTER 1**

### INTRODUCTION

### 

This application is to maintain a neat product and services portfolio and admin management portal for RSR FEEDS AND FOODS. This is to maintain high profile and managing admin and verifying customer certificates to be uploaded in the site. To verify the certificates and intimate the customer about the verification. This to be done in manual takes a lot of time and man power. This paves a way for us to make this website to be more efficient and faster. The technologies we chose to complete this task are HTML, CSS, BOOTSTRAP and Django.

#### **EXISTING SYSTEM**

Basically, the existing system to do this task was manual. This traditionally takes more manpower to maintain records, make updates, and there is a high possibility to commit mistakes. This is also more time consuming. The customer needs to wait for weeks to get the certificates to be verified and need visit the office.

##### **1.1.1 Drawbacks of existing system**

* There is no automation
* Time consuming
* No acknowledgment to the customer
* Level of service dependent on individuals
* More effort and space to keep track of paper documents

#### **SYSTEM STUDY**

The stock maintenance system must take care of warehouse updates. It maintains the number of items that are added or removed. The admin is allowed to update information and view the database. Customer will require to login to the site or create an account if not yet have one. Customer can browse through the product catalog. Customer can submit their certificates for verification.

#### **OBJECTIVE**

The objective of this work is to automate the stock maintenance process and to verify customer certificates online. Also this work aims to promote their products and services. The main goal of this project is to reduce manual works, increase the processing speed and ensure reliability of data. This process automates the stock maintenance process.

#### **SCOPE**

#### 

* This lets admin to make the stock maintenance process simple.
* Provide a communication platform between the customer and the admin.
* To ensure our system moves with time i.e. the information of the user details are stored in the database. All the inputs should be checked for validation and warning messages should be given for the improper data. The invalid data are to be ignored and the error message should be given. This application is portable. It allows for continuous maintenance and regular upgrades.

**CHAPTER 2**

### GENERAL DESCRIPTION

#### **2.1** **PRODUCT PERSPECTIVE**

The proposed work is a web application for RSR FEEDS AND FOODS. It has a stock maintenance that provides data on available stock in warehouse. This also helps in maintaining customer certification management. This system tries to make the process as simple as possible and at the same time not risking the work of data stored.

#### **2.2** **USER CHARACTERISTICS**

**1.Customer:** The person who submits certificate for verification to get product

**2.Admin:** Maintains the stock details and verifies customer certificates.

#### **2.3** **CONSTRAINTS**

1. The customer should wait until the admin verifies the certificates.
2. After the distribution of the news about the product. The customer can submit the certificates for verification .Customer can contact the admin for queries
3. Finally after verification team is sent for verification.

#### **2.4** **DESIGN AND IMPLEMENTATION CONSTRAINTS**

The information of the stocks and user details are stored in the database. All the inputs should be checked for validation and messages should be given for the improper data. The invalid data are to be ignored and the error message should be given. The access for different types of user are secured and authenticated.

### CHAPTER 3

#### **REQUIREMENT SPECIFICATION**

##### **3.1** **FUNCTIONAL REQUIREMENTS**

Requirement analysis is as of software engineering technique that is composed of the various tasks that determine the needs or conditions that are to be met for an new or altered product, taking into consider at ion the possible conflicting requirements of the various users

**User Registration:** The system should allow new users to register their login details

**Admin Requirements**: Admin should be able to use the system to view the database, update

**Automatic update:** Once the new user registration is done, the system should be able update the database without any additional efforts from the admin.

**Authentication:** The admin should to enter the valid username and password for signing in for updating data.

##### **3.2** **NON-FUNCTIONAL REQUIREMENTS**

**Security:** The sub system should provide a high level of security and integrity of the data held by the system, only authorized personnel can gain access to the boutique’s secured page on the system and only users with valid User Id and password can login to view profile detail page.

**Performance and Response time:** The system should have high performance create when executing user’s input and should be able to provide response within a short timespan usually 50 seconds for highly complicated task and 20 to 25 seconds for less complicated task.

**Error handling:** Error is minimized and considerably should show messages for invalid inputs and wrong credentials.

.

**Availability:** This system should always be available for access at 24/7. Also, in the occurrence of any major system malfunctioning, the system should be available in 1 to 2 working days, so that the business process is not severely affected.

##### **3.3** **USER INTERFACE**

User interface defines how a person is handling an application. The component of user interface is a presentation layer, logical layer. The customer is allowed to view the profile and certificate verification status.

# CHAPTER 4

# DETAILED DESIGN

## 4.1 HARDWARE AND SOFTWARE SPECIFICATION

### 4.1.1 Hardware Requirements

Hard Disk: 50GB

RAM: 1GB

Processor: Intel(R) Xeon CPU

### 4.1.2 Software Requirements

### System: Windows

Database: SQL lite

Framework: Django

### 4.2 TECHNOLOGIES USED

### 

### 4.2.1 DJANGO

Django is a free and open source framework with sql lite database.. It is aa python framework. Python and Django are con-joined not same. This also helps developing a full stack application. Is Django a frontend or backend ? It is that a backend framework that can work with any frontend. It enables us to create secure and maintainable websites.

**4.2.2 HTML**

HTML is used to make the skeleton structure to display user visible contents.This made website easier and faster

**4.2.3 CSS and BOOTSTRAP**

CSS and BOOTSTRAP were played a important in role making the website more interactive and live . CSS is mostly used to make the website look more neat and lively it is like cloth to the html structure. BOOTSTRAP v5.2 is used . And helped for available in multiple devices web . Resizing is the first foremost use of the BOOTSTRAP.

## 4.3 ARCHITECTURE DIAGRAM

An architectural diagram is used to abstract the overall outline of the software system and the relationships, constraints, and boundaries between components. As shown in Figure 4.1.

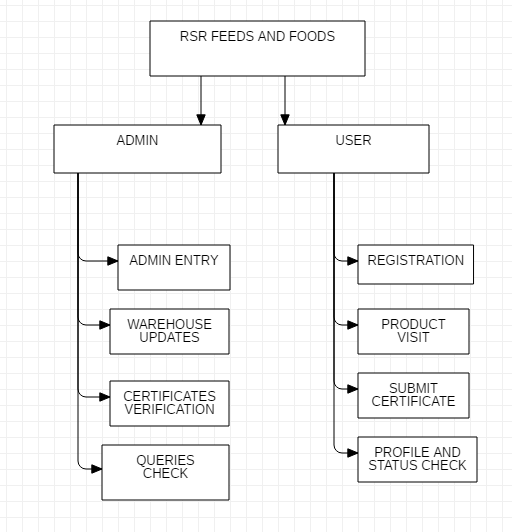
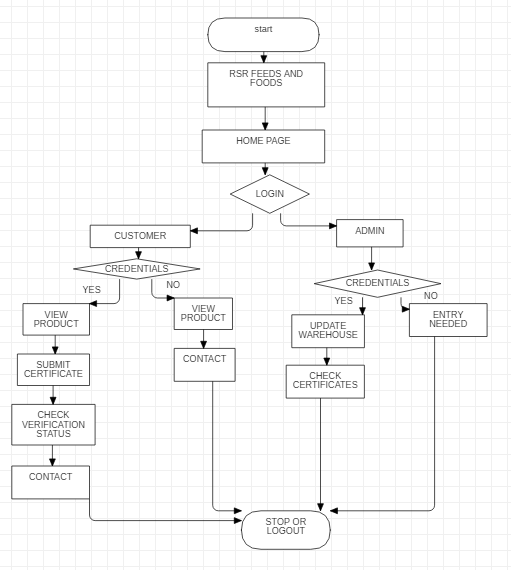


Figure 4.1 Architecture of the module

### 

### 4.4 FLOWCHART DIAGRAM

A flowchart is a type of diagram that represents a workflow or process. A flowchart can also be defined as a diagrammatic representation of an algorithm, a step-by-step approach to solving a task. As shown in Figure 4.2.



#### **Figure 4.2 flowchart of the modules**

### 4.5 USE CASE DIAGRAM

It is a graph of actors, a set of use-cases enclosed in a boundary, communication, associations between the actors and the use-cases, and generalization among the use-cases. As shown in 4.3.

The use cases used in this system are

1. LOGIN: Used for registration.
2. WAREHOUSE UPDATES: stocks check.
3. CERTIFICATE VERIFICATION: Used to verify certificates
4. QUERIES: .Manage customer queries.

**ACTORS:**

The actors used in this system are

* 1. CUSTOMER: The person who needs the product.
  2. ADMIN: The overall portal manager.

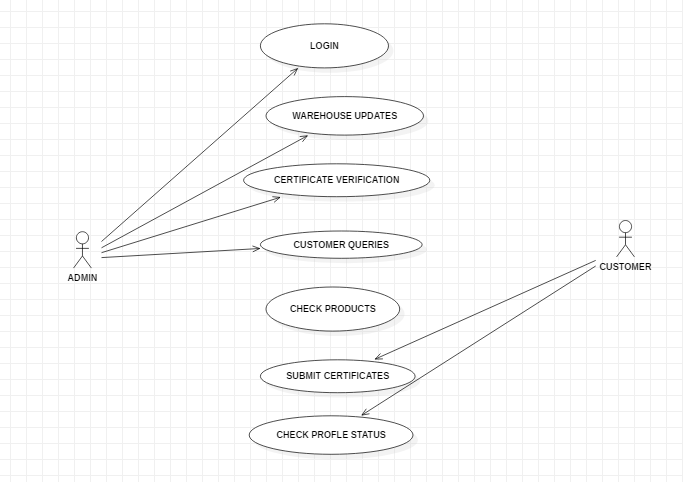


Figure 4.3 Use case of modules

**CHAPTER 5**

## 

## TESTING

### 5.1 SYSTEM TESTING

### 

After the source code has been completed, documented as related data structures. Completed the project has to undergo testing and validation where there is subtitle and definite attempt to get errors.

The project developer read slightly, designing and execution test that will demonstrates that the program works rather than uncovering errors, unfortunately errors will be present and if the project developer doesn’t find them, the user will find out. The project developer is always responsible for testing the individual units i.e. modules of the program. In many cases developer also conducts integration testing i.e. the testing step that leads to the construction of the complete program structure. This project has undergone the following testing procedures to ensure its correctness.

* Unit testing
* User Acceptance Testing

#### **5.2 Unit testing**

In unit testing, we have to test the programs making up the system. For this reason, Unit testing sometimes called as Program testing. The software units in a system are the modules and routines that are assembled and integrated to perform a specific function, Unit testing first on the modules independently of one another, to locate errors. This enables, to detect errors in coding and logic that are contained with the module alone. The testing was carried out during programming stage itself.

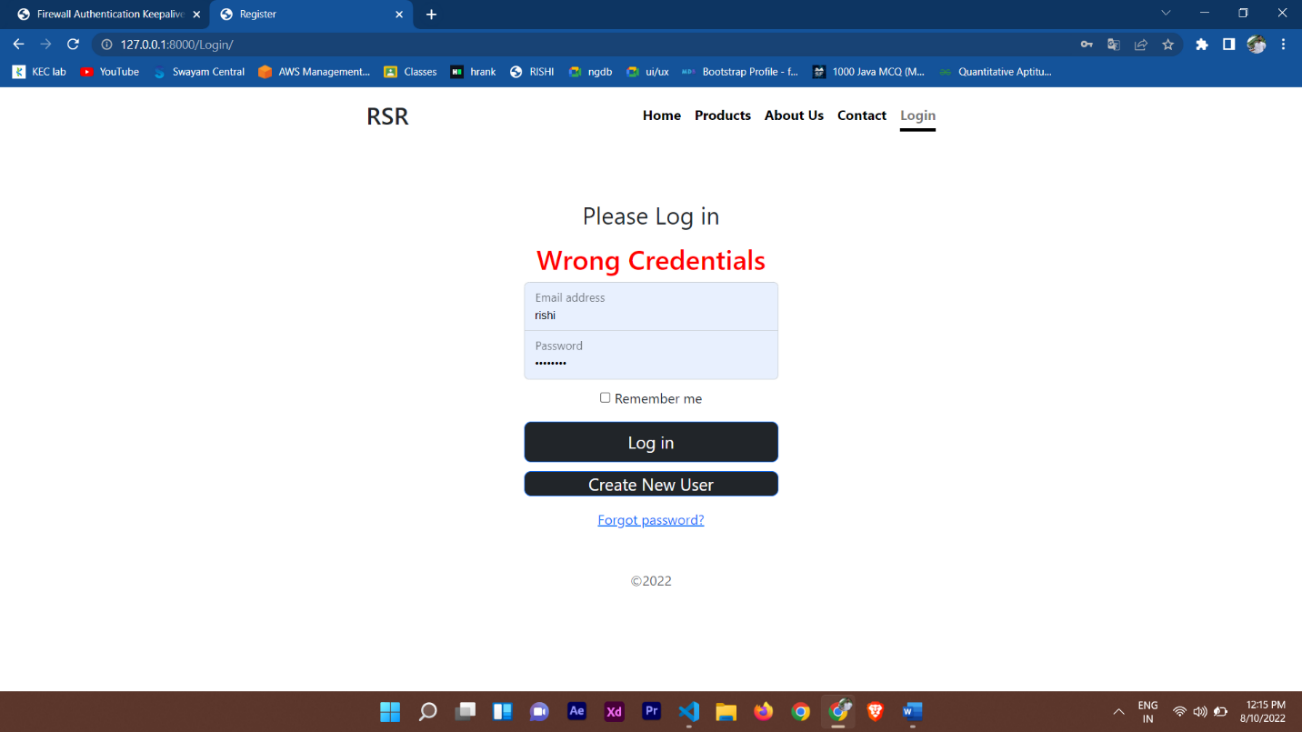
#### **5.3 User acceptance testing**

In these testing procedures the project is given to the customer to test whether all requirements have been fulfilled and after the user is fully satisfied. The project is perfectly ready. If the user makes request for any change and if they found any errors those all errors has to be taken into consideration and to be correct it to make a perfect project.

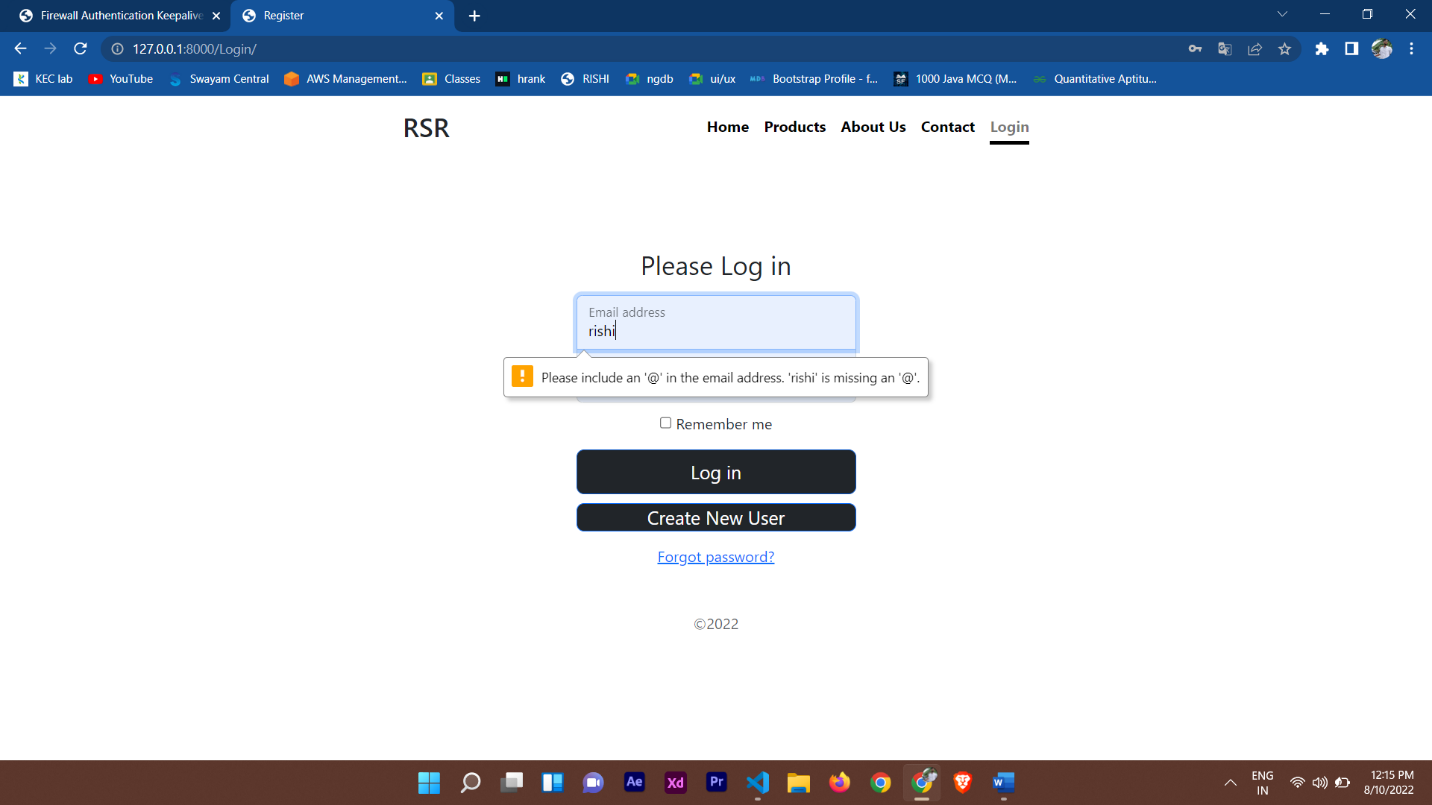
#### **5.4 Integration testing**

Integration testing is done to test if the individual modules work together as one single unit. In integration testing, the individual modules that are to be integrated are available for testing. Thus the manual test data that used to test the interfaces replaced by that which in generated automatically from the various modules. It can be used for testing how the modules would actually interact with the proposed system. The modules are integrated and tested to reveal the problem interfaces

**TESTCASES IN LOGIN MODULE:**

****

**Figure 5.1 error for wrong inputs**

****

**Figure 5.2 validating the input**

**CHAPTER 6**

## CONCLUSION AND FUTURE WORK

### 6.1 CONCLUSION

The project work entitled “WEB APPLICATION FOR RSR FEEDS AND FOODS” so has been tested and documented completely. Through this project, the warehouse management and customer certificate verification is carried out. The interface helps not only to administrator but also to customers for communication. Since the application is designed as web, any browser can be used to view the application. The change password helps to protect the accessibility of users

.

### 6.2 FUTURE WORK

### 

The application is tested well and end user’s satisfaction is found to be more. The application is designed such that minimum internet knowledge is required for to browse the website.

The application become useful if the below enhancements are made in future.

* If the application is designed as web service, it can be integrated in many websites.
* It would be good to include further features like auto checking certificates
* Integration of online payment would make this application, a standalone solution.
* The application is developed such that above said enhancements can be integrated with current modules.

## APPENDIX I

### CODING

**VIEWS.PY:**

import email, os

from itertools import product

from tokenize import String

from django.shortcuts import redirect, render

from django.http import HttpResponse,Http404,HttpResponseRedirect, FileResponse

from django import template

from django.template import loader

from cgitb import html

from django.core.mail import send\_mail

from APP.models import signin

from APP.models import Product

from APP.models import OfficeSignin

from APP.models import Adminsignin,Verification

from django.core.files.base import ContentFile

# Create your views here.

def Homepage(request):

return render (request,'RSRhomepage.html')

#Register

def register(request):

if request.method == "POST":

fullname = request.POST['fullname']

email = request.POST['email']

DOB = request.POST['dob']

Phone = request.POST['phone']

password = request.POST['password']

ins = signin(fullname=fullname,email=email,DOB=DOB,Phone=Phone,password=password)

ins.save()

return render(request,"RSRhomepage.html")

return render (request,'Register.html',)

#Login

def Login(request):

error = False

if request.method == "POST":

email = request.POST.get('email')

password = request.POST.get('password')

if email == "admin@gmail.com" and password == "adminrishi":

return HttpResponseRedirect('/Adminhome/')

for each in signin.objects.all():

if email == each.email and password == each.password:

CurrentUser = signin.objects.filter(email = email)

request.session['username'] = CurrentUser[0].fullname

request.session.modified = True

return HttpResponseRedirect('/Homepage/')

else:

error = True

return render(request, 'login.html',{'error':error})

#Logout

def Logout(request):

del request.session['username']

request.session.modified = True

return HttpResponseRedirect('/Homepage/')

def Products(request):

return render (request,'Products.html')

def About(request):

return render (request,'About.html')

def Contact(request):

return render (request,'Contact.html')

def Alogin(request):

return render (request,'Alogin.html')

def Verificationn(request):

if request.method == "POST":

fullname = request.POST.get('fullname')

email = request.POST.get('email')

phone = request.POST.get('phone')

farmAddress = request.POST.get('farmAddress')

farmRegNo = request.POST.get('farmRegNo')

certificate = request.FILES.get('farmCertificate')

print(fullname, email, phone, farmAddress, farmRegNo, certificate)

ins = Verification(fullname= fullname, email= email,phone= phone,farmAddress= farmAddress,farmRegNo= farmRegNo, certificate= certificate)

ins.save()

return HttpResponseRedirect('/Account/')

return render(request, 'verification.html')

def Account(request):

return render(request,'Account.html')

def Profileedit(request):

return render(request,'Profileedit.html')

def Forgot(request):

print("the method is",request.method)

if(request.method == "POST"):

email=request.POST['email']

npassword=request.POST['npassword']

signin.objects.filter(email=email).update(password=npassword)

print("Data updated successfully!")

else:

print("redirected")

return render(request,'Forgot.html')

def Adminhome(request):

return render (request,'Adminhome.html')

def OfficeLogin(request):

error = False

if request.method == "POST":

username = request.POST.get('username')

password = request.POST.get('password')

for each in OfficeSignin.objects.all():

if username == each.username and password == each.password:

return HttpResponseRedirect('/Adminhome/')

else:

error = True

return render(request, 'AdminLogin.html',{'error':error})

def PassCheck(request):

OldPass = request.GET['Oldpass']

CurrentUser = request.session.get('username')

record = signin.objects.filter(fullname= CurrentUser)

if record[0].password == OldPass:

return HttpResponse(1)

else:

return HttpResponse(0)

def UpdateAccount(request):

if request.method == "POST":

print("asds")

CurrentUser = request.session.get('username')

dob = request.POST.get('dob')

phone = request.POST.get('phone')

address = request.POST.get('address')

Npassword = request.POST.get('Npassword')

signin.objects.filter(fullname= CurrentUser).update(DOB= dob,Phone= phone,password= Npassword,address= address)

return HttpResponseRedirect('/Profileedit')

return render(request, 'profileedit.html')

def adminVerification(request):

return render(request, 'adminVerification.html',{

'VerficationData' : Verification.objects.all()

})

def openFile(request):

record = Verification.objects.filter(farmRegNo= request.POST['farmRegNo'])

filepath = os.path.join('', str(record.first().certificate))

return FileResponse(open(filepath, 'rb'), content\_type='application/pdf')

def makeVerify(request):

Verification.objects.filter(farmRegNo= request.GET['farmRegNo']).update(verified=1)

return HttpResponse("Verified")

def productsadmin(request):

if request.method == "POST":

pname = request.POST.get('pname')

pquantity = request.POST.get('pquantity')

print(pname,pquantity)

ins = Product(pname=pname, pquantity=pquantity)

ins.save()

return HttpResponseRedirect('/padmin')

return render(request, 'productsadmin.html')

def productsadmin2(request):

if request.method == "POST":

pname = request.POST.get('pname')

npquantity = request.POST.get('npquantity')

Product.objects.filter(pname=pname).update(pquantity=npquantity)

return HttpResponseRedirect('/padmin2')

return render(request, 'productsadmin2.html')

def office(request):

shows = []

shows = Product.objects.all()

for show in shows:

print(show.pname,show.pquantity)

return render(request, 'officechecks.html', {'shows': shows})

def adminLogout(request):

return HttpResponseRedirect('/Homepage/')

MODELS.PY:

from django.db import models

# Create your models here.

class signin(models.Model):

fullname = models.CharField(max\_length=100)

email = models.CharField(max\_length=100)

DOB = models.DateField()

Phone = models.IntegerField()

password = models.CharField(max\_length=100)

address = models.CharField(max\_length=100, default='default: India, You can update this field in edit Profile')

class OfficeSignin(models.Model):

username = models.CharField(max\_length=100)

email = models.CharField(max\_length=100)

DOB = models.DateField()

Phone = models.IntegerField()

password = models.CharField(max\_length=100)

class Adminsignin(models.Model):

Username = models.CharField(max\_length=100)

email = models.CharField(max\_length=100)

DOB = models.DateField()

Phone = models.IntegerField()

password = models.CharField(max\_length=100)

class Verification(models.Model):

fullname = models.CharField(max\_length=100)

email = models.CharField(max\_length=100)

phone = models.CharField(max\_length=100)

farmAddress = models.CharField(max\_length=100)

farmRegNo = models.IntegerField()

certificate = models.FileField()

verified = models.IntegerField(default=0)

class Product(models.Model):

pname = models.CharField(max\_length=100)

pquantity = models.CharField(max\_length=100)

def \_\_unicode\_\_(self):

return self.name

CUSTOM.PY:

from APP.models import signin, Verification

def UserAccountDetails(request):

data = signin.objects.filter(fullname = request.session.get('username'))

return {

'AccountData' : data

}

def toVerifyData(request):

data = Verification.objects.all

return {

'toVerifyData' : data

}

def userVerificationSubmitted(request):

CurrentUser = request.session.get('username')

userSubmitted =0

record = Verification.objects.filter(fullname= CurrentUser)

if(len(record) ==0):

userSubmitted = 0

else:

userSubmitted = 1

return {

'userSubmitted' : userSubmitted

}

def userIsVerified(request):

CurrentUser = request.session.get('username')

userVerified =0

record = Verification.objects.filter(fullname= CurrentUser)

try:

if record.first().verified == 0:

userVerified = 0

else:

userVerified = 1

except: pass

return {

'userVerified' : userVerified

}

ADMIN.PY:

from django.contrib import admin

from APP.models import signin

from APP.models import OfficeSignin

from APP.models import Adminsignin

from APP.models import Verification

from APP.models import Product

# Register your models here.

admin.site.register(signin)

admin.site.register(OfficeSignin)

admin.site.register(Adminsignin)

admin.site.register(Verification)

admin.site.register(Product)

**HTML PAGES:**

**HOMEAPGE:**

<!DOCTYPE html>

<html lang="en" class="h-100">

<head>

<meta charset="utf-8" />

<meta name="viewport" content="width=device-width, initial-scale=1" />

<title>Homepage</title>

<link

href="https://cdn.jsdelivr.net/npm/bootstrap@5.2.0-beta1/dist/css/bootstrap.min.css"

rel="stylesheet"

integrity="sha384-0evHe/X+R7YkIZDRvuzKMRqM+OrBnVFBL6DOitfPri4tjfHxaWutUpFmBp4vmVor"

crossorigin="anonymous>

<script

src="https://cdn.jsdelivr.net/npm/@popperjs/core@2.11.5/dist/umd/popper.min.js"

integrity="sha384-Xe+8cL9oJa6tN/veChSP7q+mnSPaj5Bcu9mPX5F5xIGE0DVittaqT5lorf0EI7Vk"

crossorigin="anonymous"

></script>

<script

src="https://cdn.jsdelivr.net/npm/bootstrap@5.2.0-beta1/dist/js/bootstrap.min.js"

integrity="sha384-kjU+l4N0Yf4ZOJErLsIcvOU2qSb74wXpOhqTvwVx3OElZRweTnQ6d31fXEoRD1Jy"

crossorigin="anonymous"

></script>

<style>

.cover-container {

max-width: 42em;

.nav-masthead .nav-link {

color: rgb(0, 0, 0);

border-bottom: 0.25rem solid transparent;}

.nav-masthead .nav-link:hover,

.nav-masthead .nav-link:focus {

border-bottom-color: rgba(23, 23, 23, 0.25); }

.nav-masthead .nav-link + .nav-link {

margin-left: 1rem;}

.nav-masthead .active {

color: rgba(23, 23, 23, 0.582);

border-bottom-color: rgb(0, 0, 0);}

#name {

position: absolute;

left: 80%;

top: 3%;}

</style>

<!-- Custom styles for this template -->

</head>

<body class="d-flex h-100 text-center text-dark bg-white">

<div class="d-flex w-100 h-100 p-3 mx-auto flex-column">

<header class="mb-auto">

<div class="cover-container mx-auto">

{% if request.session.username %}

<h6 id="name">

Welcome,

<span style="color: blue"> {{ request.session.username }} </span>

</h6>

{% endif %}

<h3 class="float-md-start mb-0">RSR</h3>

<nav class="nav nav-masthead justify-content-center float-md-end"><a

class="nav-link fw-bold py-1 px-0 active"

aria-current="page"

href="/Homepage"

>Home</a >

<a class="nav-link fw-bold py-1 px-0" href="/Products/">Products</a>

<a class="nav-link fw-bold py-1 px-0" href="/About/">About Us</a>

<a class="nav-link fw-bold py-1 px-0" href="/Contact">Contact</a>

{% if request.session.username %}

<a class="nav-link fw-bold py-1 px-0" href="/Account/">Profile</a>

<a class="nav-link fw-bold py-1 px-0" href="/Logout/">Logout</a>

<a class="nav-link fw-bold py-1 px-0" href="/Account/">

<svg

xmlns="http://www.w3.org/2000/svg"

width="30"

height="30"

fill="currentColor"

class="bi bi-person-circle"

viewBox="0 0 16 16

<path d="M11 6a3 3 0 1 1-6 0 3 3 0 0 1 6 0z" />

<path

fill-rule="evenodd"

d="M0 8a8 8 0 1 1 16 0A8 8 0 0 1 0 8zm8-7a7 7 0 0 0-5.468 11.37C3.242 11.226 4.805 10 8 10s4.757 1.225 5.468 2.37A7 7 0 0 0 8 1z" / </sv </a>

{% else %}

<a class="nav-link fw-bold py-1 px-0" href="/Login/"

>Login/Register</a >

{% endif %}

</nav>

</div>

</header>

<main class="px-3 cover-container mx-auto">

<h1>Welcome to our page.</h1>

<p class="lead">

RSR is a one of the leading poultry industry and feed production

industry in namakkal. Certified with ISO. We take care on the feeds

for your poultry.</p>

{% if request.session.username %}

<p class="lead"><a

href="/Products/"

class="btn btn-lg btn-secondary fw-bold border-dark bg-dark"

>OUR PRODUCTS</a ></p>

{% else %}

<p class="lead"><a

href="/Login/"

class="btn btn-lg btn-secondary fw-bold border-dark bg-dark"

>JOIN COMMUNITY!</></p>

{% endif %}

</main>

</body>

</html>

**APPENDIX II**

**SNAPSHOTS**

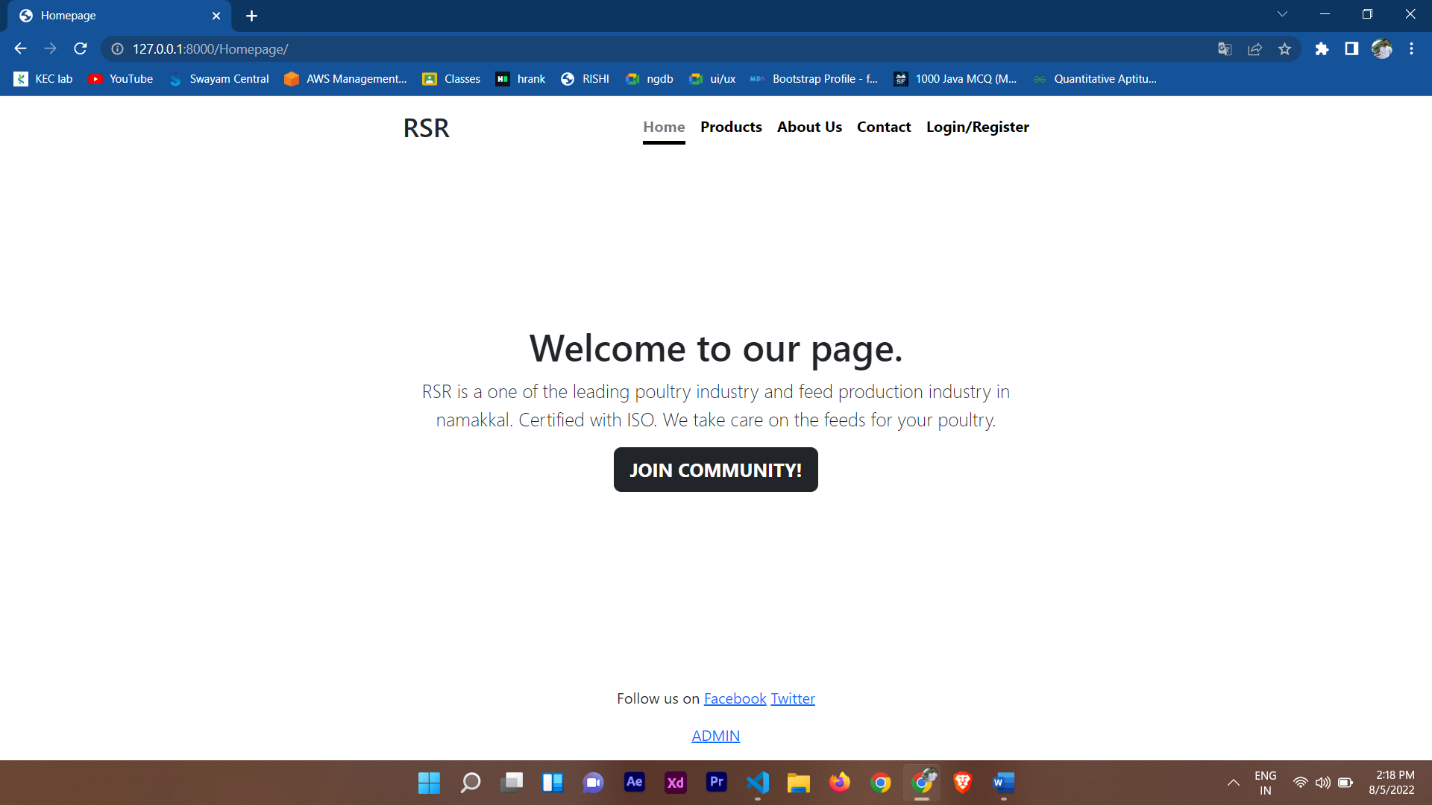


Figure A2.1 Homepage

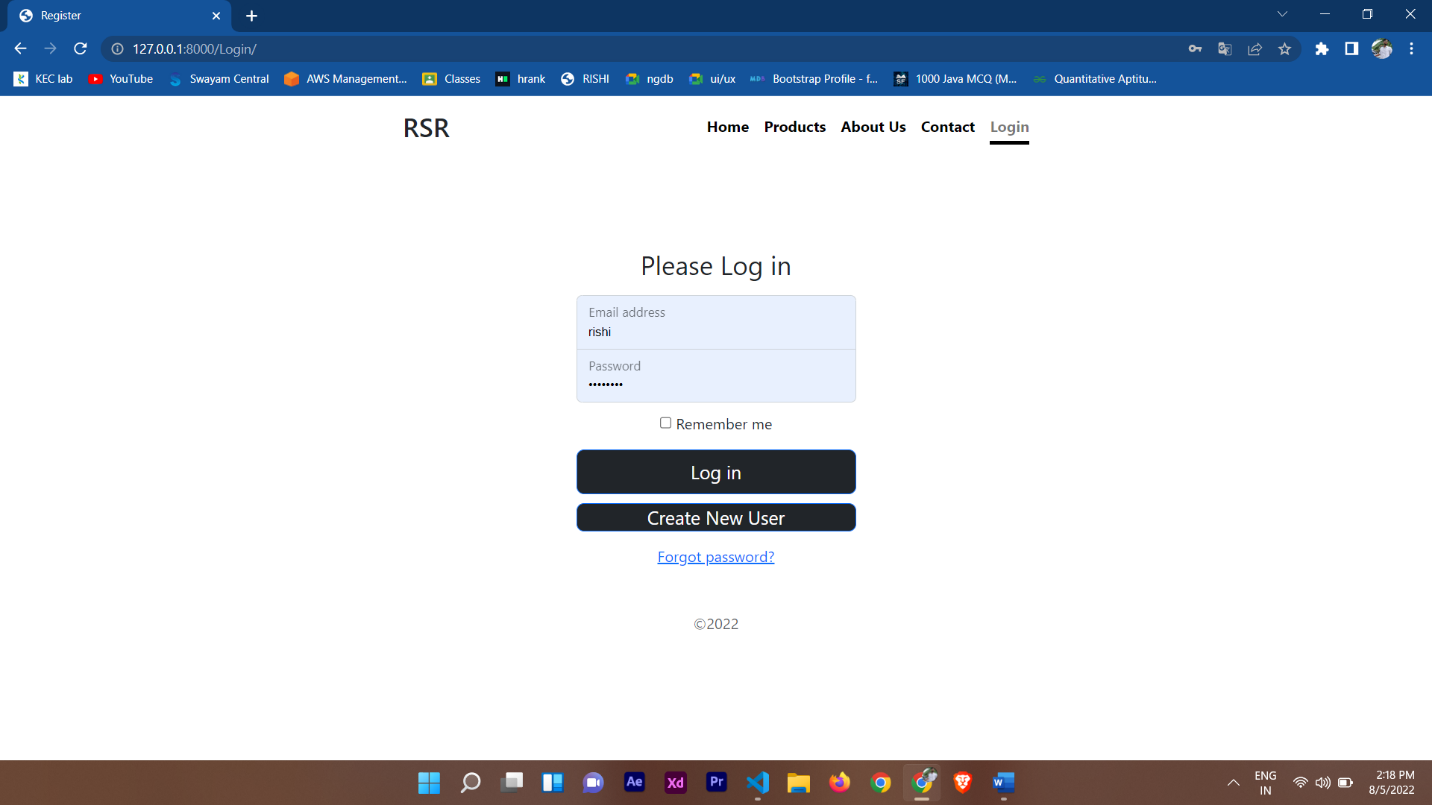


Figure A2.2 Common login page



Figure A2.3 Profile page

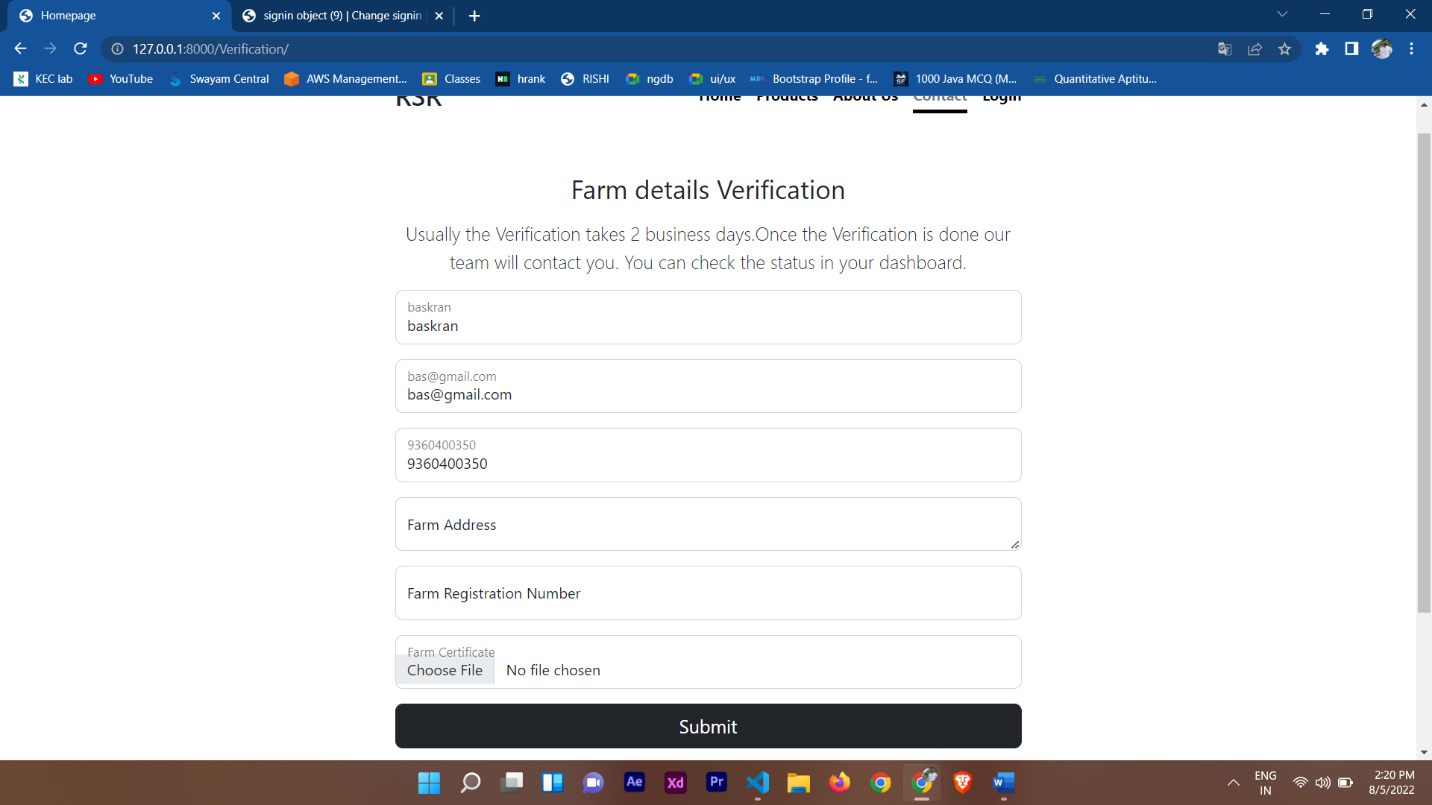


Figure A2.4 Certification form page

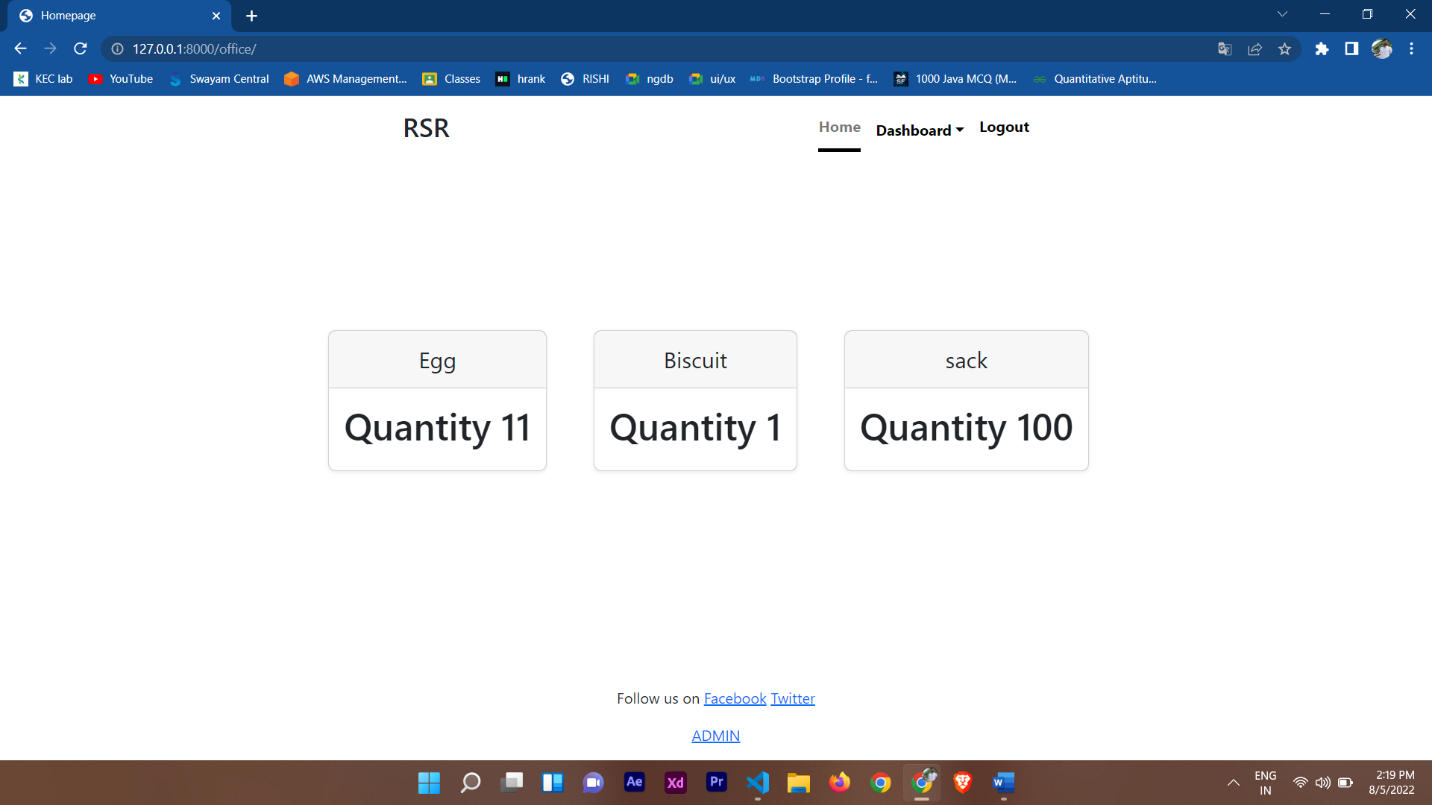


Figure A2.5 Warehouse checking page

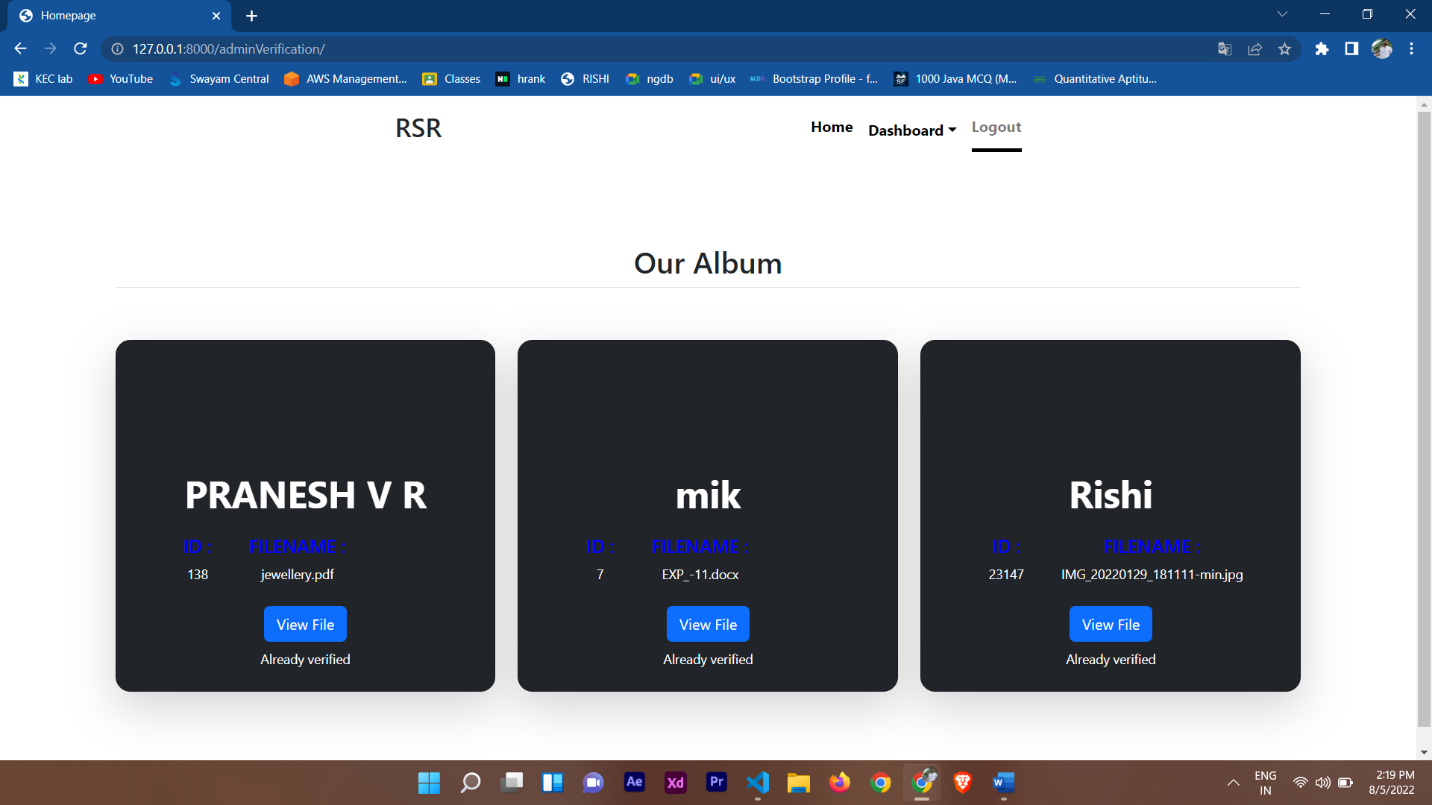
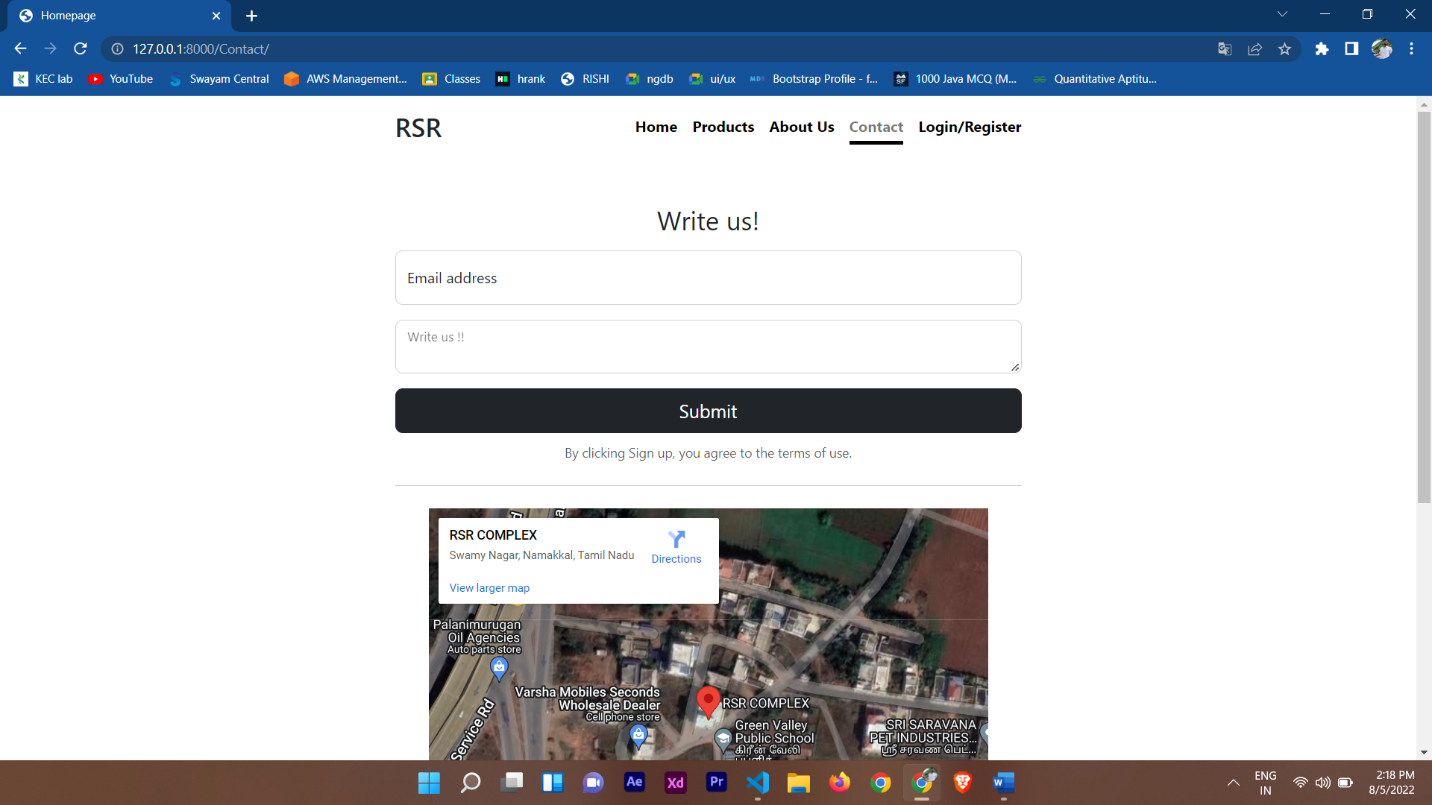


Figure A2.6 Verification page



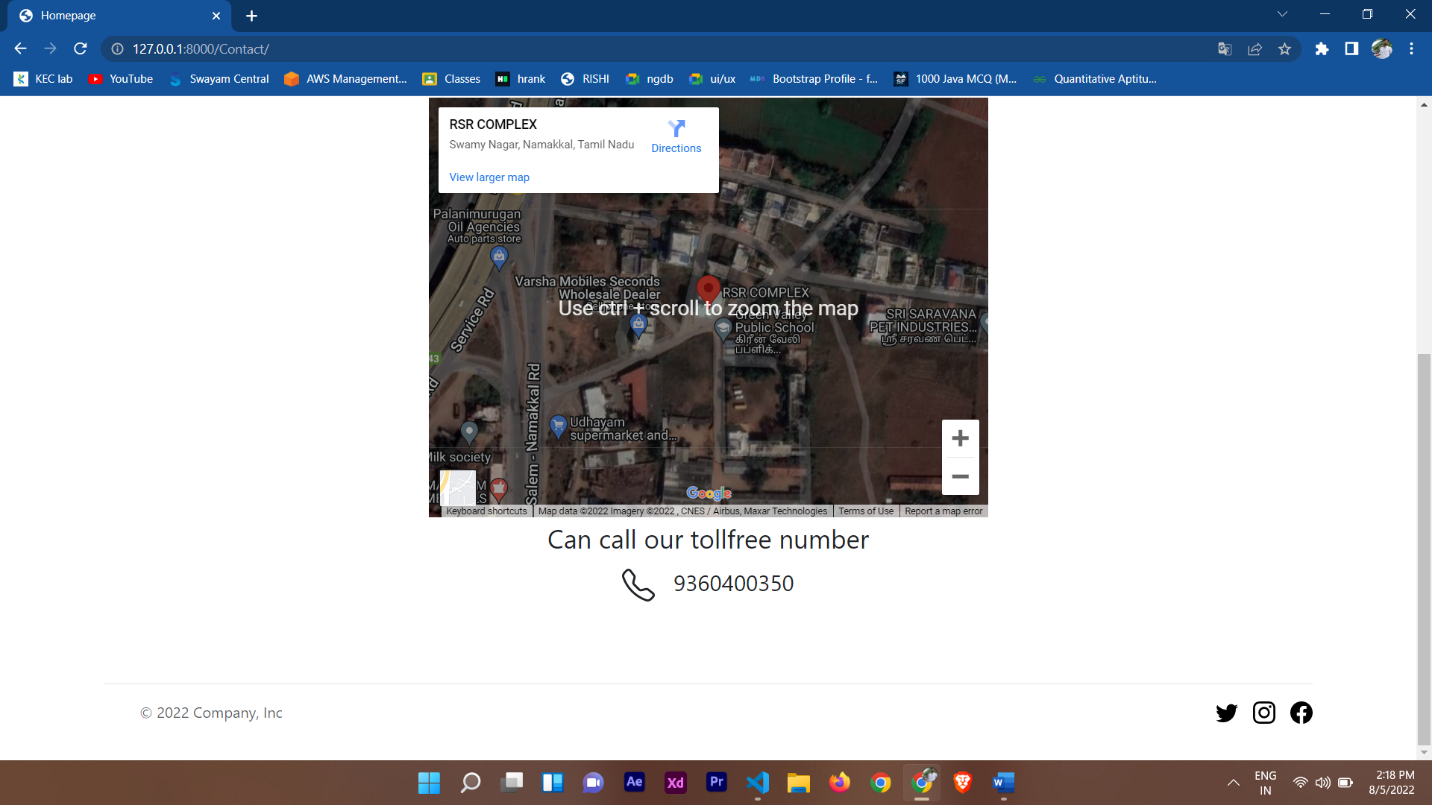
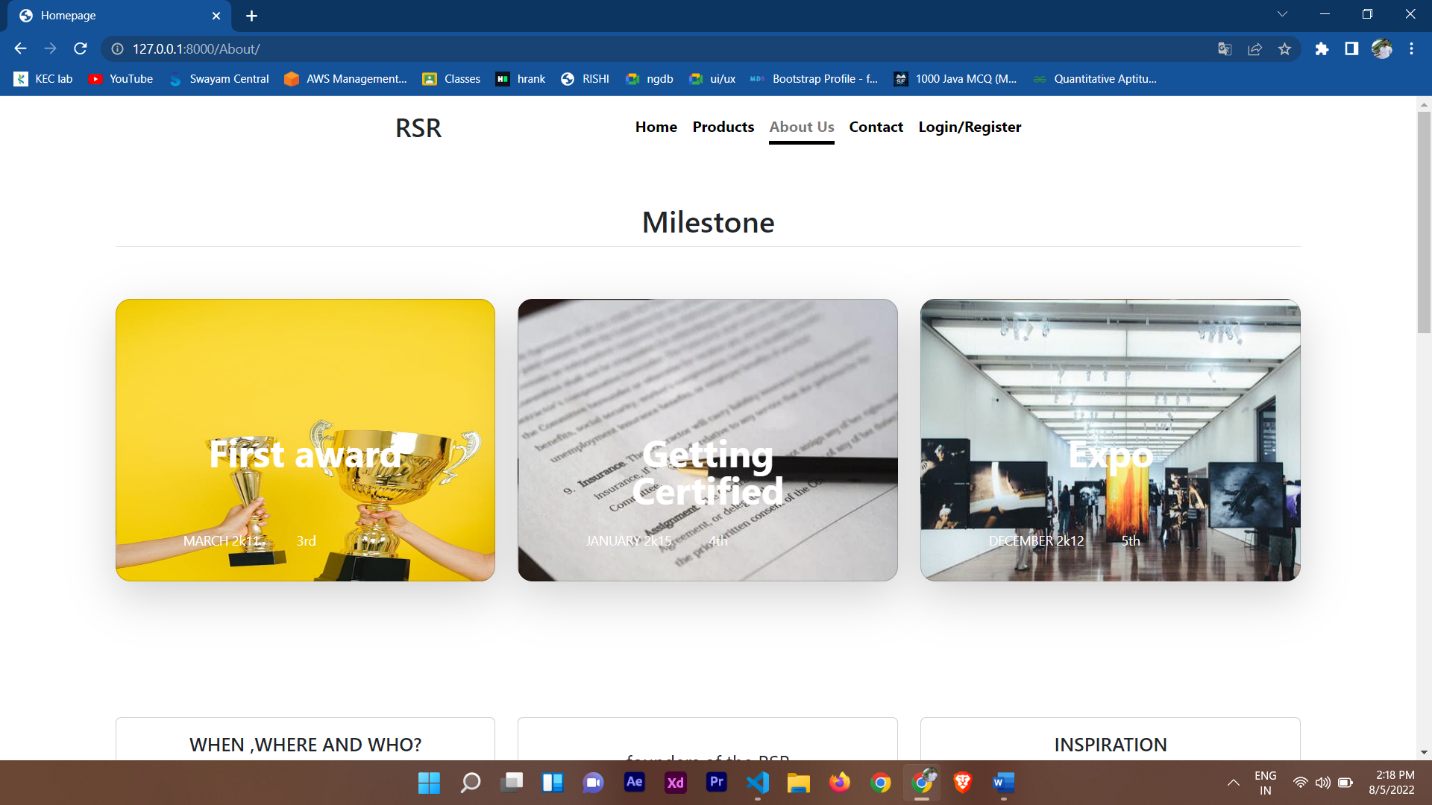


Figure A2.7 contact page



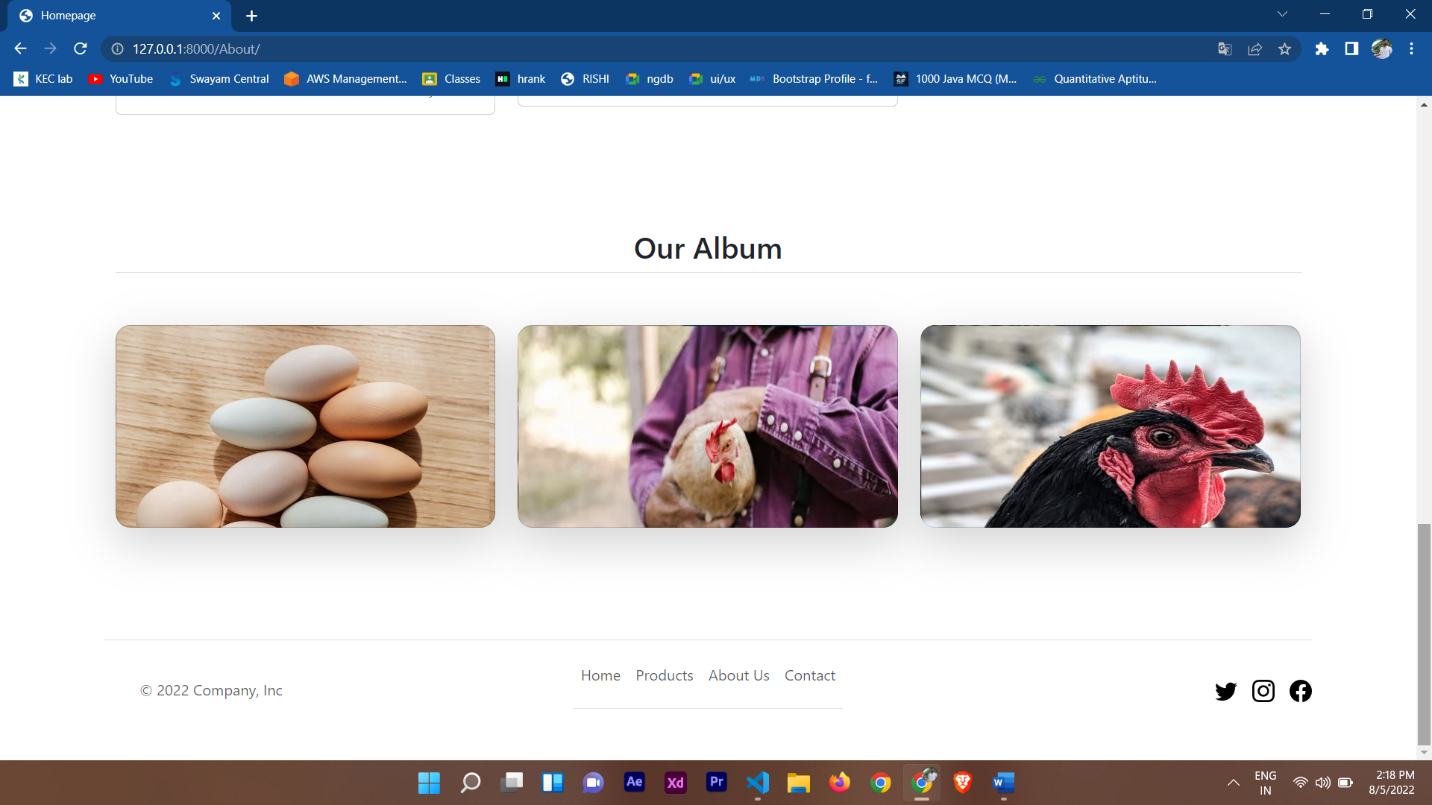


Figure A2.8 About page

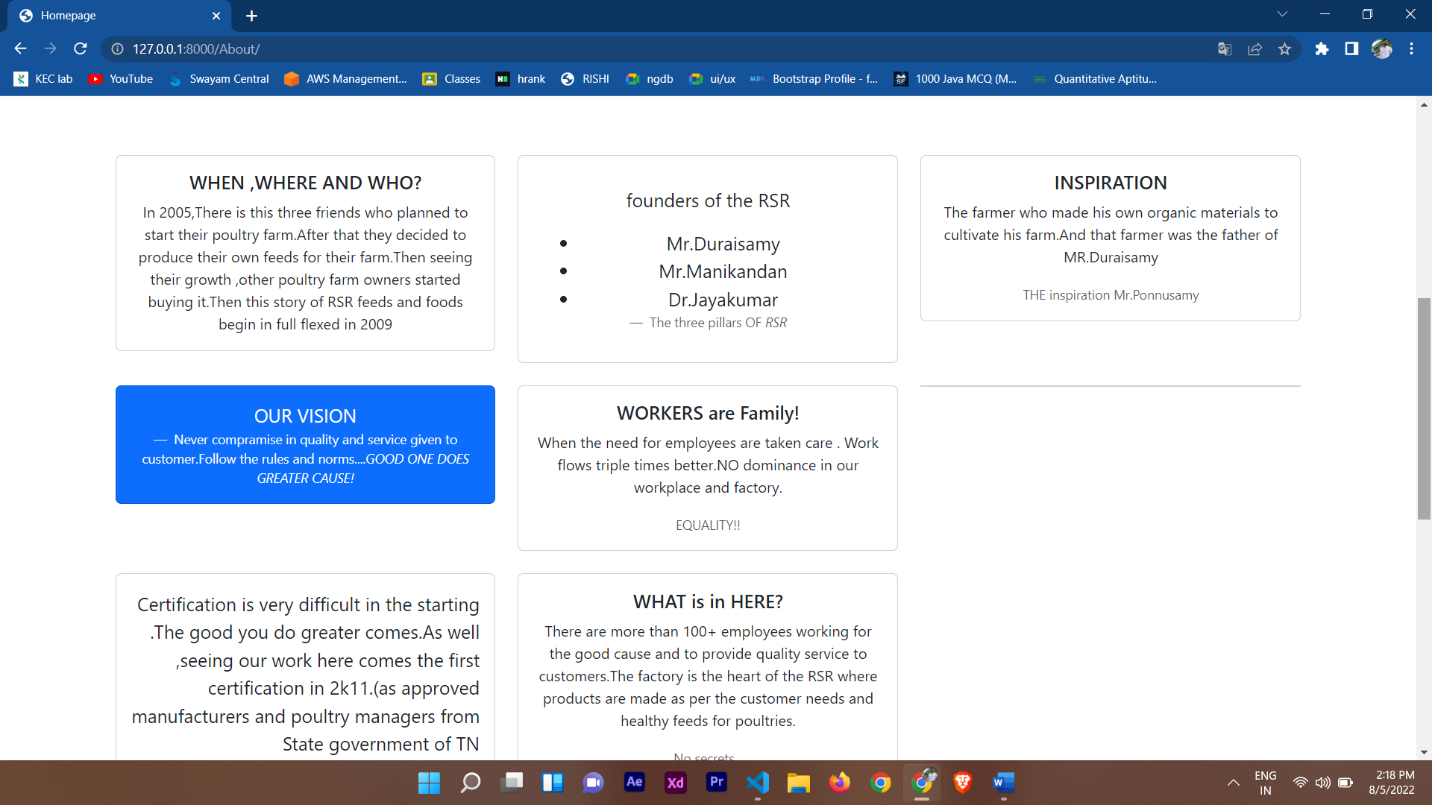


Figure A2.8 About page

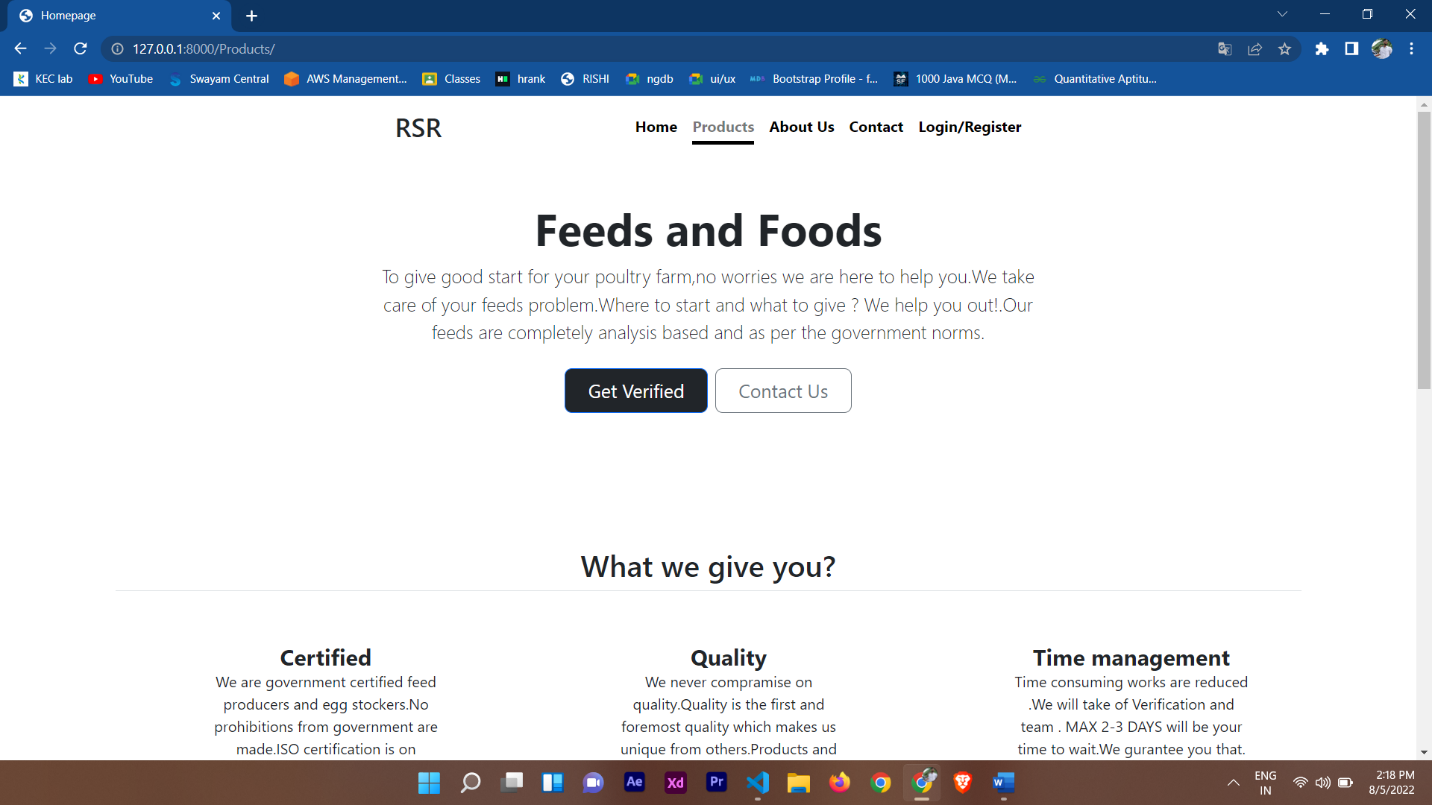


Figure A2.9 Products page

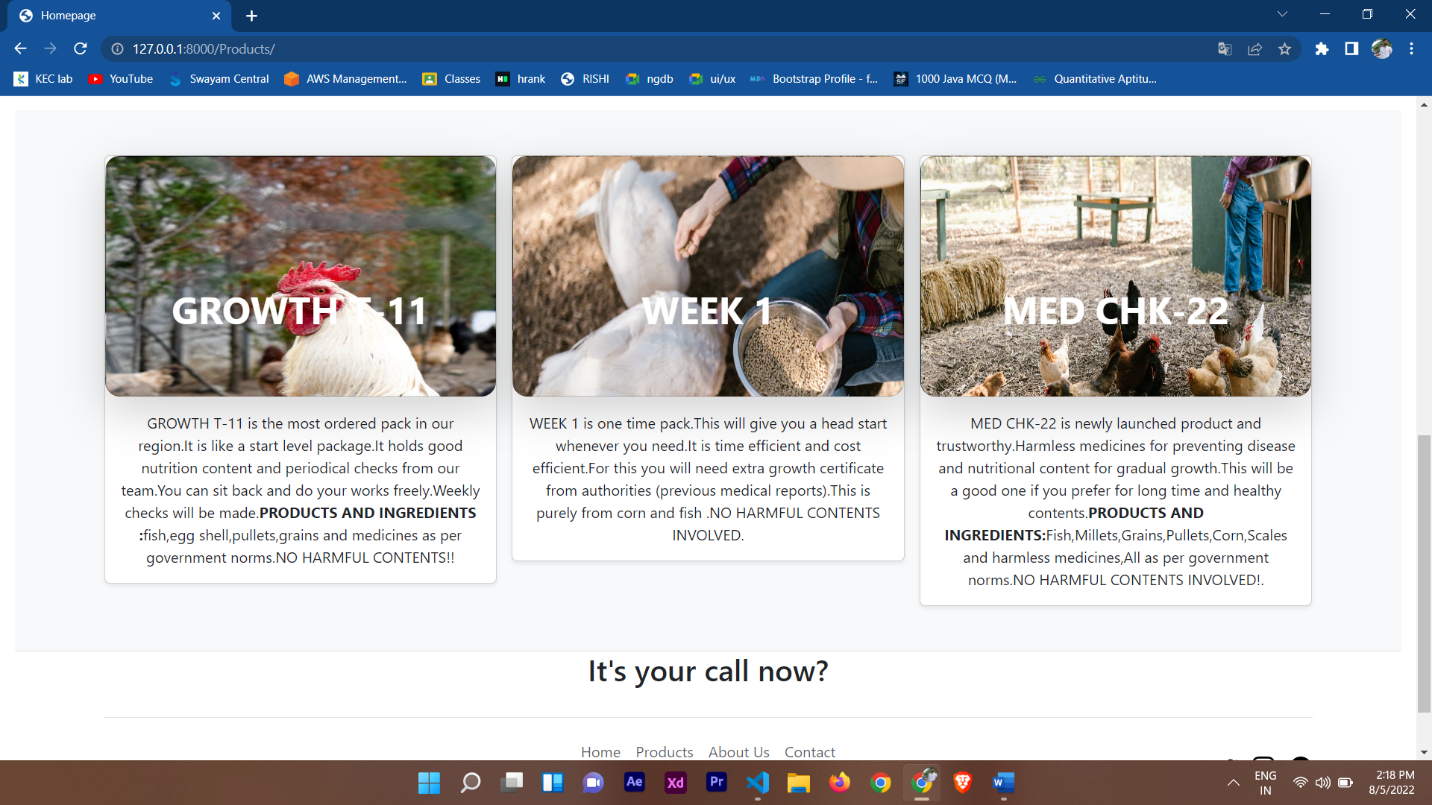


Figure A2.9 Products page

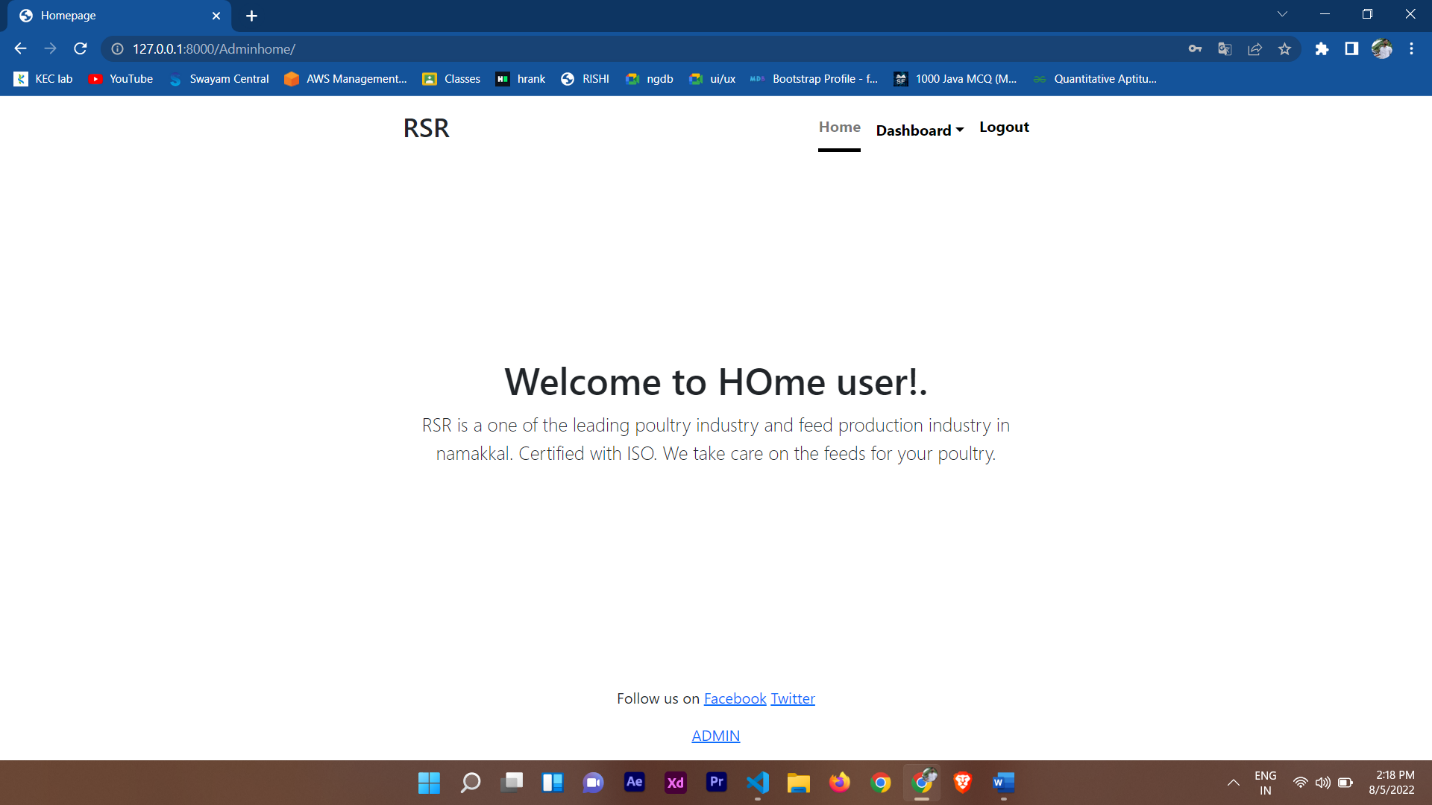


Figure A2.10 Admin home page

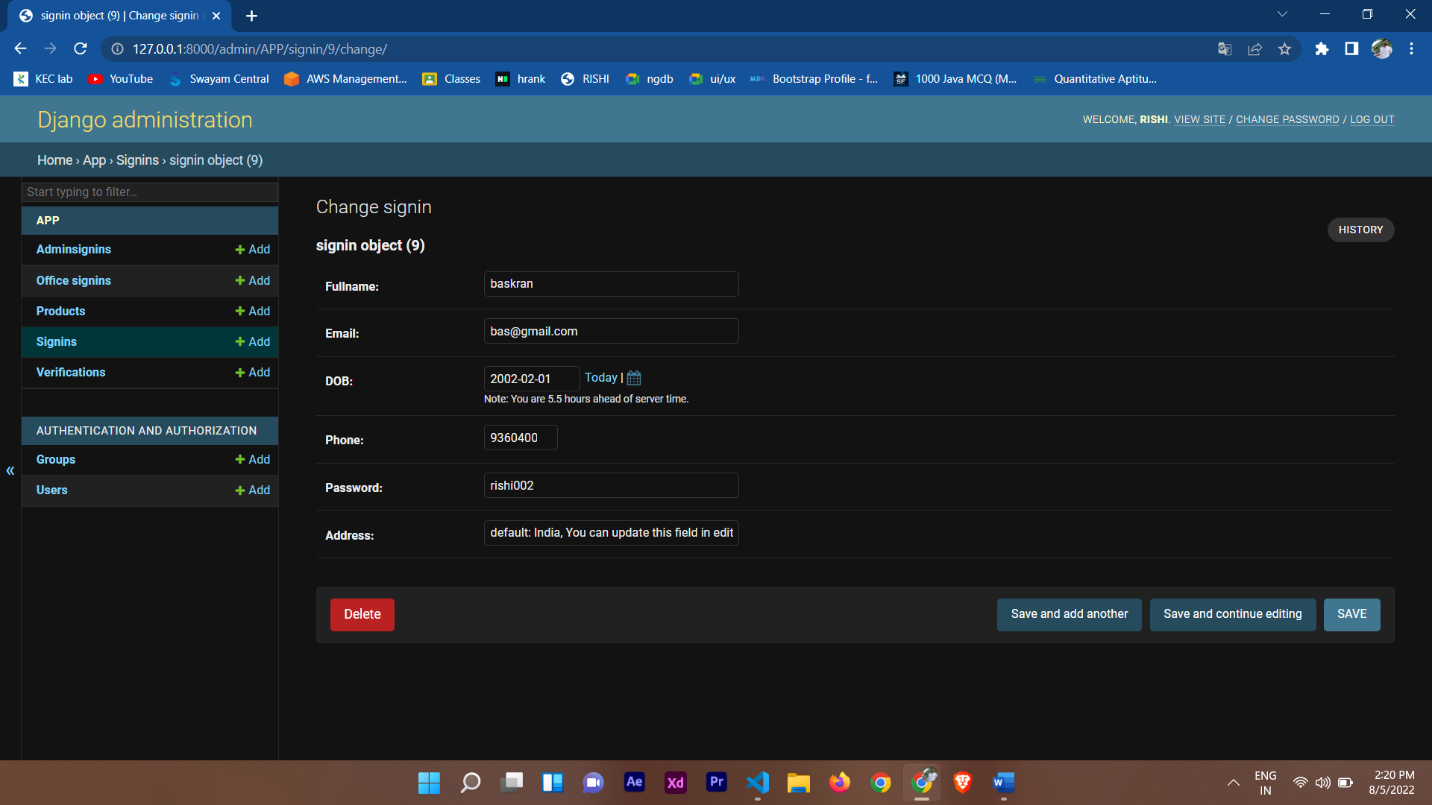


Figure A2.11 Django database

**REFERENCE**

* <https://www.w3schools.com/django/>
* <https://www.javatpoint.com/django-tutorial>
* <https://www.tutorialspoint.com/django/index.html>
* <https://www.geeksforgeeks.org/django-tutorial/>

