A logo with black text

Description automatically generated

**NIT6150 Advanced Project**

**Quality Assurance Report**

**HealthCare Chatbot System**

**A hand holding a heart with a pulse line

Description automatically generated**

**Team Leader:** Jitendra Shrestha (s8104215)

**Team Member:** Pranish Acharya (s8100698)

**Client:** Holroyd Private Hospital

**Supervisor:** Fakhra Jabeen

VU Sydney

**Table of Contents**

[Manual Testing 1](#_Toc177122992)

[User Management System 1](#_Toc177122993)

[Test Case UMS 01: Create Admin User 1](#_Toc177122994)

[Test Case UMS 02: Login As admin User 2](#_Toc177122995)

[Test Case UMS 03. Add Doctor 3](#_Toc177122996)

[Personal Health Management System 5](#_Toc177122997)

[Test Case PHMS 01: Register Patient and Log In 5](#_Toc177122998)

[Test Case PHMS 02. Book Appointment 6](#_Toc177122999)

[Test Case PHMS 03. Add Prescription by Doctor 8](#_Toc177123000)

[Test Case PHMS 03. Check health record of patient 9](#_Toc177123001)

[Chatbot System 11](#_Toc177123002)

[Test Case CS 01. Accessing Chatbot 11](#_Toc177123003)

[Automated Testing 12](#_Toc177123004)

[UI Testing 12](#_Toc177123005)

[Unit Testing 16](#_Toc177123006)

# Manual Testing

## User Management System

### Test Case UMS 01: Create Admin User

|  |  |  |
| --- | --- | --- |
| S.N. | Action required | Screenshots |
| 1 | Open browser and go to the URL: [***http://localhost:8000/admin***](http://localhost:8000/admin) |  |
| 2 | Use the Credentials of Superuser and click on login |  |
| 3 | Click on the **Users** and **Add User**  Fill the username and password  Fill the first name, last name and email  In permissions section add the admin in chosen groups as shown in screenshot |  |
| 4 | Click on **Save** |  |
|  |  | **PASS** |

### Test Case UMS 02: Login As admin User

|  |  |  |
| --- | --- | --- |
| S.N. | Action required | Screenshots |
| 1 | Open browser and go to the URL: [***http://localhost:8000/admin***](http://localhost:8000/admin) |  |
| 2 | Click on **Account** on Navbar which should redirect to the Login page  Add email and password of admin user  Click on **Log In** |  |
| 3 | User can see the admin homepage  Go to URL  <http://localhost:8000/home/> |  |
|  |  | **PASS** |

### Test Case UMS 03. Add Doctor

|  |  |  |
| --- | --- | --- |
| S.N. | Action required | Screenshots |
| 1 | Open browser and go to the URL: [***http://localhost:8000/home***](http://localhost:8000/home) |  |
| 2 | Click on **Add Doctor**  Fill all the Details |  |
| 3 | Click on **Add Doctor** |  |
| 4 | Log in using created email and password |  |
|  |  | **PASS** |

## Personal Health Management System

### Test Case PHMS 01: Register Patient and Log In

|  |  |  |
| --- | --- | --- |
| S.N. | Action required | Screenshots |
| 1 | Open browser and go to the URL: [***http://localhost:8000***](http://localhost:8000) |  |
| 2 | Click on **Account**  Click on **New user**  Popup will be available and fill the details  Click on **Create Account** |  |
| 3 | Log in using created email and password |  |
|  |  | **PASS** |

### Test Case PHMS 02. Book Appointment

|  |  |  |
| --- | --- | --- |
| S.N. | Action required | Screenshots |
| 1 | Open browser and go to the URL: [***http://localhost:8000***](http://localhost:8000) |  |
| 2 | Click on **Account**  Add email and Password  Click on **Log In** |  |
| 3 | Logged in and show the homepage |  |
| 4 | Click on **Book Appointments** on sidebar  Choose Doctor  Choose Date  Choose Time Slot  Provide Symptoms  Click on **Book Appointment** |  |
| 5 | Click on **view Appointments** in sidebar  Recently booked appointment is shown |  |
|  |  | **PASS** |

### Test Case PHMS 03. Add Prescription by Doctor

|  |  |  |
| --- | --- | --- |
| S.N. | Action required | Screenshots |
| 1 | Open browser and go to the URL: [***http://localhost:8000***](http://localhost:8000) |  |
| 2 | Click on **Account**  Add email and Password  Click on **Log In** |  |
| 3 | Logged in and show the homepage |  |
| 4 | Click on **View Appointments** on sidebar  Click on Appointment  Add prescription and follow up date if required  Click on **Add prescription** |  |
| 5 | After prescription it will be move to previous appointments.  Click on recent appointment. |  |
|  |  | **PASS** |

### Test Case PHMS 03. Check health record of patient

|  |  |  |
| --- | --- | --- |
| S.N. | Action required | Screenshots |
| 1 | Open browser and go to the URL: [***http://localhost:8000***](http://localhost:8000) |  |
| 2 | Click on **Account**  Add email and Password  Click on **Log In** |  |
| 3 | Logged in and show the homepage |  |
| 4 | Click on **View Health Records** on sidebar |  |
| 5 | We will be able to see the health record and if required we can print in provided formats(excel, csv and pdf)  PDF is shown |  |
|  |  | **PASS** |

## Chatbot System

### Test Case CS 01. Accessing Chatbot

|  |  |  |
| --- | --- | --- |
| S.N. | Action required | Screenshots |
| 1 | Open browser and go to the URL: [***http://localhost:8000***](http://localhost:8000) |  |
| 2 | Click on **Chatroom** on navbar |  |
| 3 | Start the conversation using ‘**Hi’**  Should get response from chatbot  Exit the bot using ‘**exit’** |  |
|  |  | **PASS** |

# Automated Testing

## UI Testing

**Required Libraries import for both UI and Unit testing**

#import required libraries

from selenium import webdriver

from selenium.webdriver.common.by import By

from selenium.webdriver.common.keys import Keys

from selenium.webdriver.chrome.service import Service as ChromeService

from webdriver\_manager.chrome import ChromeDriverManager

from selenium.webdriver.support.ui import WebDriverWait

from selenium.webdriver.support import expected\_conditions as EC

from frontend.models import Doctor,Appointment

from django.urls import reverse

from django.test import LiveServerTestCase, TestCase, Client

import requests

from django.contrib.auth.models import User, Group

from django.contrib.auth import authenticate

from django.utils import timezone

from datetime import datetime, timedelta

"""

    Testing UI using Selenium

    Given: We will be giving the parameters same as in browser

    When: Correct Credentials used in same as UI

    Then: User can log in and see the successfully

"""

class SeleniumUITestCase(LiveServerTestCase):

    def setUp(self):

        self.browser = webdriver.Chrome(service=ChromeService(ChromeDriverManager().install()))

        self.browser.implicitly\_wait(10)  # Wait up to 10 seconds for elements to be available

        self.server\_url = 'http://localhost:8000'

    def tearDown(self):

        self.browser.quit()

    # check the index is loaded correctly or not

    def test\_homepage(self):

        self.browser.get(f'{self.server\_url}/')

        self.assertIn('HealthCare', self.browser.title)

        self.assertIn('What Does Pink Eye Look Like?', self.browser.page\_source)

    # check the home page is loaded correctly or not for patient

    def test\_patienthome(self):

        self.browser.get(f'{self.server\_url}/login/')

        self.browser.find\_element(By.NAME, 'email').send\_keys('patient1@gmail.com')

        self.browser.find\_element(By.NAME, 'password').send\_keys('patient1', Keys.RETURN)

        WebDriverWait(self.browser, 10).until(EC.url\_changes(f'{self.server\_url}/login/'))

        self.browser.get(f'{self.server\_url}/home/')

        self.assertIn('Patient Home', self.browser.title)

        self.assertIn('Healthy Tips', self.browser.page\_source)

    # check the about us page

    def test\_aboutpage(self):

        self.browser.get(f'{self.server\_url}/about/')

        self.assertIn('HealthCare', self.browser.title)

        self.assertIn('The main motive', self.browser.page\_source)

    # check the login using valid username and password

    def test\_login\_valid(self):

        self.browser.get(f'{self.server\_url}/login/')

        self.browser.find\_element(By.NAME, 'email').send\_keys('patient1@gmail.com')

        self.browser.find\_element(By.NAME, 'password').send\_keys('patient1', Keys.RETURN)

        WebDriverWait(self.browser, 10).until(EC.url\_changes(f'{self.server\_url}/login/'))

        self.assertIn('Patient Home', self.browser.title)

        self.assertIn('Healthy Tips', self.browser.page\_source)

    # check the login using invalid username and password

    def test\_login\_invalid(self):

        self.browser.get(f'{self.server\_url}/login/')

        self.browser.find\_element(By.NAME, 'email').send\_keys('invalid@gmail.com')

        self.browser.find\_element(By.NAME, 'password').send\_keys('password', Keys.RETURN)

        self.assertIn('HealthCare', self.browser.title)

        self.assertIn('Invalid credentials.', self.browser.page\_source)

    # check the chatroom page popup

    def test\_chatroom\_view(self):

        self.browser.get(f'{self.server\_url}/chatroom/')

        self.assertIn('HealthCare', self.browser.title)

        self.assertIn('Personal Health Assistant!', self.browser.page\_source)

    # check the View Doctor page from admin perspective

    def test\_adminviewDoctor(self):

        self.browser.get(f'{self.server\_url}/login/')

        self.browser.find\_element(By.NAME, 'email').send\_keys('tjiten123@gmail.com')

        self.browser.find\_element(By.NAME, 'password').send\_keys('zeyrox1@', Keys.RETURN)

        WebDriverWait(self.browser, 10).until(EC.url\_changes(f'{self.server\_url}/login/'))

        self.browser.get(f'{self.server\_url}/adminviewDoctor/')

        self.assertIn('HealthCare', self.browser.title)

        self.assertIn('ALL DOCTORS', self.browser.page\_source)

    # check the View appointment from admin perspective

    def test\_adminviewAppointment(self):

        self.browser.get(f'{self.server\_url}/login/')

        self.browser.find\_element(By.NAME, 'email').send\_keys('tjiten123@gmail.com')

        self.browser.find\_element(By.NAME, 'password').send\_keys('zeyrox1@', Keys.RETURN)

        WebDriverWait(self.browser, 10).until(EC.url\_changes(f'{self.server\_url}/login/'))

        self.browser.get(f'{self.server\_url}/adminviewAppointment/')

        self.assertIn('HealthCare', self.browser.title)

        self.assertIn('PREVIOUS APPIONTMENTS', self.browser.page\_source)

    # check the View profile page from patient perspective

    def test\_profile(self):

        self.browser.get(f'{self.server\_url}/login/')

        self.browser.find\_element(By.NAME, 'email').send\_keys('patient1@gmail.com')

        self.browser.find\_element(By.NAME, 'password').send\_keys('patient1', Keys.RETURN)

        WebDriverWait(self.browser, 10).until(EC.url\_changes(f'{self.server\_url}/login/'))

        self.browser.get(f'{self.server\_url}/profile/')

        self.assertIn('HealthCare', self.browser.title)

        self.assertIn('Phone Number', self.browser.page\_source)

    # check the View appointment page from patient perspective

    def test\_view\_appointments\_patient(self):

        self.browser.get(f'{self.server\_url}/login/')

        self.browser.find\_element(By.NAME, 'email').send\_keys('patient1@gmail.com')

        self.browser.find\_element(By.NAME, 'password').send\_keys('patient1', Keys.RETURN)

        WebDriverWait(self.browser, 10).until(EC.url\_changes(f'{self.server\_url}/login/'))

        self.browser.get(f'{self.server\_url}/viewappointments/')

        self.assertIn('HealthCare', self.browser.title)

        self.assertIn('APPOINTMENTS', self.browser.page\_source)

    # check the View appointment page from doctor perspective

    def test\_view\_appointments\_doctor(self):

        self.browser.get(f'{self.server\_url}/login/')

        self.browser.find\_element(By.NAME, 'email').send\_keys('alhan@gmail.com')

        self.browser.find\_element(By.NAME, 'password').send\_keys('alhan', Keys.RETURN)

        WebDriverWait(self.browser, 10).until(EC.url\_changes(f'{self.server\_url}/login/'))

        self.browser.get(f'{self.server\_url}/viewappointments/')

        self.assertIn('HealthCare', self.browser.title)

        self.assertIn('YOU HAVE APPOINTMENTS WITH:', self.browser.page\_source)

    # check the View health record page from patient perspective

    def test\_view\_health\_records\_patient(self):

        self.browser.get(f'{self.server\_url}/login/')

        self.browser.find\_element(By.NAME, 'email').send\_keys('patient1@gmail.com')

        self.browser.find\_element(By.NAME, 'password').send\_keys('patient1', Keys.RETURN)

        WebDriverWait(self.browser, 10).until(EC.url\_changes(f'{self.server\_url}/login/'))

        self.browser.get(f'{self.server\_url}/viewhealthrecords/')

        self.assertIn('HealthCare', self.browser.title)

        self.assertIn('HEALTH RECORDS', self.browser.page\_source)

    # check the contact page

    def test\_contactus(self):

        self.browser.get(f'{self.server\_url}/contact/')

        self.assertIn('HealthCare', self.browser.title)

        self.assertIn('ll never share your email with anyone else.', self.browser.page\_source)

    # checking the available time slot returned on api

    def test\_get\_available\_time\_slots(self):

        date = '2024-09-12'

        doctor\_email = 'baburam@gmail.com'

        url = f'{self.server\_url}/get-available-time-slots/?date={date}&doctor\_email={doctor\_email}'

        response = requests.get(url)

        self.assertEqual(response.status\_code, 200)

        response\_data = response.json()

        expected\_slots = ['10:30', '11:00', '11:30', '12:00', '12:30', '13:00', '13:30']

        self.assertEqual(response\_data['timeSlots'], expected\_slots)

## Unit Testing

"""

    Functional Testing of views methods

    Given: We will be giving the parameters required for the methods

    When: Correct paramters provided

    Then: User can get the correct template and peform tasks

"""

class FunctionalTestCase(TestCase):

    def setUp(self):

        self.client = Client()

        # Create groups

        self.patient\_group = Group.objects.create(name='Patient')

        self.doctor\_group = Group.objects.create(name='Doctor')

        self.admin\_group = Group.objects.create(name='Admin')

        # Create users

        self.patient\_user = User.objects.create\_user(username='patient@gmail.com', password='password', email='patient@gmail.com', first\_name='Patient')

        self.patient\_group.user\_set.add(self.patient\_user)

        self.doctor\_user = User.objects.create\_user(username='doctor@gmail.com', password='password', email='doctor@gmail.com', first\_name='Doctor')

        self.doctor\_group.user\_set.add(self.doctor\_user)

        self.admin\_user = User.objects.create\_user(username='admin@gmail.com', password='password', email='admin@gmail.com', first\_name='Admin', is\_staff=True)

        self.admin\_group.user\_set.add(self.admin\_user)

    def test\_homepage(self):

        response = self.client.get(reverse('homepage'))

        self.assertEqual(response.status\_code, 200)

        self.assertTemplateUsed(response, 'index.html')

    def test\_patienthome(self):

        self.client.login(username='patient@gmail.com', password='password')

        response = self.client.get(reverse('patienthome'))

        self.assertEqual(response.status\_code, 200)

        self.assertTemplateUsed(response, 'patienthome.html')

    def test\_aboutpage(self):

        response = self.client.get(reverse('aboutpage'))

        self.assertEqual(response.status\_code, 200)

        self.assertTemplateUsed(response, 'about.html')

    def test\_loginpage\_valid(self):

        response = self.client.post(reverse('loginpage'), {

            'email': 'patient@gmail.com',

            'password': 'password'

        })

        self.assertEqual(response.status\_code, 302)

        self.assertRedirects(response, reverse('patienthome'))

    def test\_loginpage\_invalid(self):

        response = self.client.post(reverse('loginpage'), {

            'email': 'invalid@gmail.com',

            'password': 'password'

        })

        self.assertEqual(response.status\_code, 302)

        self.assertRedirects(response, reverse('loginpage'))

    def test\_createaccountpage(self):

        response = self.client.post(reverse('createaccountpage'), {

            'name': 'New Patient',

            'email': 'newpatient@gmail.com',

            'password': 'password',

            'repeatpassword': 'password',

            'gender': 'Male',

            'phonenumber': '1234567890',

        })

        self.assertEqual(response.status\_code, 302)

        self.assertRedirects(response, reverse('loginpage'))

    def test\_chatroom\_view(self):

        url = reverse('chatroom')

        response = self.client.get(url)

        self.assertEqual(response.status\_code, 200)

        self.assertTemplateUsed(response, 'chatroom.html')

        self.assertContains(response, "Chatroom")

    def test\_adminaddDoctor(self):

        self.client.login(username='admin@gmail.com', password='password')

        response = self.client.post(reverse('adminaddDoctor'), {

            'name': 'Dr. Smith',

            'email': 'drsmith@gmail.com',

            'password': 'password',

            'repeatpasssword': 'password',

            'gender': 'Male',

            'phonenumber': '1234567890',

            'address': '123 Street',

            'licenseNo': 'ABC123',

            'specialization': 'Cardiology',

        })

        self.assertEqual(response.status\_code, 302)

        self.assertRedirects(response, reverse('adminviewDoctor'))

    def test\_adminviewDoctor(self):

        self.client.login(username='admin@gmail.com', password='password')

        response = self.client.get(reverse('adminviewDoctor'))

        self.assertEqual(response.status\_code, 200)

        self.assertTemplateUsed(response, 'adminviewDoctors.html')

    def test\_admin\_delete\_doctor(self):

        doctor = Doctor.objects.create(name='Dr. Smith', email='drsmith@gmail.com', gender='Male', phonenumber='1234567890', address='123 Street', licenseNo='ABC123', specialization='Cardiology')

        user = User.objects.create\_user(username='drsmith@gmail.com', email='drsmith@gmail.com', password='password')

        self.client.login(username='admin@gmail.com', password='password')

        response = self.client.get(reverse('admin\_delete\_doctor', args=[doctor.id, doctor.email]))

        self.assertEqual(response.status\_code, 302)

        self.assertRedirects(response, reverse('adminviewDoctor'))

    def test\_patient\_delete\_appointment(self):

        appointment = Appointment.objects.create(doctorname='Dr. Smith', doctoremail='drsmith@gmail.com', patientname='Patient', patientemail='patient@gmail.com', appointmentdate=timezone.now().date(), appointment\_time='10:00', symptoms='Cough', status=True)

        self.client.login(username='patient@gmail.com', password='password')

        response = self.client.get(reverse('patient\_delete\_appointment', args=[appointment.id]))

        self.assertEqual(response.status\_code, 302)

        self.assertRedirects(response, reverse('viewappointments'))

    def test\_adminviewAppointment(self):

        self.client.login(username='admin@gmail.com', password='password')

        response = self.client.get(reverse('adminviewAppointment'))

        self.assertEqual(response.status\_code, 200)

        self.assertTemplateUsed(response, 'adminviewappointments.html')

    def test\_profile(self):

        self.client.login(username='patient@gmail.com', password='password')

        response = self.client.get(reverse('profile'))

        self.assertEqual(response.status\_code, 200)

        self.assertTemplateUsed(response, 'pateintprofile.html')

    def test\_make\_appointments(self):

        self.client.login(username='patient@gmail.com', password='password')

        response = self.client.post(reverse('makeappointments'), {

            'doctoremail': 'drsmith@gmail.com',

            'doctorname': 'Dr. Smith',

            'patientname': 'Patient',

            'patientemail': 'patient@gmail.com',

            'appointmentdate': timezone.now().date(),

            'timeslot': '10:00',

            'symptoms': 'Cough',

        })

        self.assertEqual(response.status\_code, 302)

        self.assertRedirects(response, reverse('makeappointments'))

    def test\_view\_appointments\_patient(self):

        self.client.login(username='patient@gmail.com', password='password')

        response = self.client.get(reverse('viewappointments'))

        self.assertEqual(response.status\_code, 200)

        self.assertTemplateUsed(response, 'patientviewappointments.html')

    def test\_view\_appointments\_doctor(self):

        self.client.login(username='doctor@gmail.com', password='password')

        response = self.client.get(reverse('viewappointments'))

        self.assertEqual(response.status\_code, 200)

        self.assertTemplateUsed(response, 'doctorviewappointment.html')

    def test\_view\_health\_records\_patient(self):

        self.client.login(username='patient@gmail.com', password='password')

        response = self.client.get(reverse('viewhealthrecords'))

        self.assertEqual(response.status\_code, 200)

        self.assertTemplateUsed(response, 'pateintviewrecord.html')

    def test\_view\_health\_records\_doctor(self):

        self.client.login(username='doctor@gmail.com', password='password')

        response = self.client.get(reverse('viewhealthrecords'))

        self.assertEqual(response.status\_code, 200)

        self.assertTemplateUsed(response, 'doctorviewappointment.html')

    def test\_contactus(self):

        response = self.client.post(reverse('contactus'), {

            'contactname': 'John Doe',

            'contactphonenumber': '1234567890',

            'contactemail': 'johndoe@gmail.com',

            'message': 'This is a test message.',

        })

        self.assertEqual(response.status\_code, 200)

        self.assertTemplateUsed(response, 'contactus.html')

"""

    Functional Testing of get\_available\_time\_slots API

    Given: We will be giving the parameters required for the methods

    When: Correct paramters provided

    Then: User can get the JSON response or not

"""

class FunctionalTestForGetAvailableSlot(TestCase):

    def setUp(self):

        # Set up the test client

        self.client = Client()

        # Create test data

        self.date = timezone.now().date() + timedelta(days=1)

        self.doctor\_email = 'alhan@gmail.com'

        self.booked\_times = ['10:00', '11:00']

        # Create appointments for the doctor

        for time\_slot in self.booked\_times:

            Appointment.objects.create(

                doctorname='Alhan',

                doctoremail=self.doctor\_email,

                patientname='Patient Test',

                patientemail='patient@gmail.com',

                appointmentdate=self.date,

                appointment\_time=datetime.strptime(time\_slot, '%H:%M').time(),

                symptoms='Test Symptoms',

                status=True,

                prescription=''

            )

    # Check the method to get the JSON response

    def test\_get\_available\_time\_slots(self):

        url = reverse('get\_available\_time\_slots')

        response = self.client.get(url, {'date': self.date, 'doctor\_email': self.doctor\_email})

        self.assertEqual(response['Content-Type'], 'application/json')

        response\_data = response.json()

        expected\_slots = ['10:30', '11:30', '12:00', '12:30', '13:00', '13:30']

        self.assertEqual(response\_data['timeSlots'], expected\_slots)

    def tearDown(self):

        # Clean up any data

        Appointment.objects.all().delete()

**Screenshot of Test Ran Successfully**

A screenshot of a computer program

Description automatically generated