**TABLE OF CONTENTS**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Sr. No** |  |  | **CONTENT** | **PG. NO** |
| **1** |  |  | **INTRODUCTION** | 2 |
|  | **1.1** |  | **Problem Introduction** | 3 |
|  | **1.2** |  | **Objectives** | 4 |
|  | **1.3** |  | **Purpose** | 4 |
|  | **1.4** |  | **Scope** | 4 |
|  |  |  |  |  |
| **2** |  |  | **REQUIREMENTS SPECIFICATION** | 5 |
|  | **2.1** |  | **Hardware Requirement** | 5 |
|  | **2.2** |  | **Software Requirement** | 5 |
|  |  |  |  |  |
| **3** |  |  | **SOFTWARE ANALYSIS** | 6 |
|  | **3.1** |  | **Software Specification** | 6 |
|  |  | 3.1.1 | HTML | 6 |
|  |  | 3.1.2 | CSS | 6 |
|  |  | 3.1.3 | JavaScript | 6 |
|  |  | 3.3.4 | API | 7 |
|  |  | 3.3.5 | Google Sheet | 7 |
|  | **3.2** |  | **GANTT CHART** | 8 |
|  |  |  |  |  |
| **4** |  |  | **SYSTEM DESIGN** | 8 |
|  | **4.1** |  | **Block Diagram** | 8 |
|  | **4.2** |  | **Use Case Diagram** | 9 |
|  | **4.3** |  | **Deployment Diagram** | 10 |
|  |  |  |  |  |
| **5** |  |  | **SOFTWARE TESTING** | 11 |
|  | **5.1** |  | **Introduction of Software Testing** | 11 |
|  | **5.2** |  | **Module Testing** | 11 |
|  | **5.3** |  | **Testing Result** | 13 |
|  |  |  |  |  |
| **6** |  |  | **CODING AND SNAPSHOTS** | 14 |
|  | **6.1** |  | **Implementation approaches** | 14 |
|  |  | 6.1.1 | Code | 14 |
|  |  | 6.1.2 | Snapshots | 28 |
|  |  |  |  |  |
| **7** |  |  | **CONCLUSION** | 31 |
| **8** |  |  | **FUTURE ENHANCEMENT** | 32 |
|  | **7.1** |  | **Scope of Improvement** | 32 |
| **9** |  |  | **REFERENCES** | 33 |

**CHAPTER 1**

**INTRODUCTION**

NB Metha a seamlessly integrated collage website that harmoniously connects with Google Sheets to effortlessly display dynamic data. This comprehensive guide will walk you through the straightforward process of leveraging our API to sync your website with a Google Spreadsheet, ensuring a smooth and efficient flow of information. Discover the simplicity of data retrieval and presentation as you embark on a journey of creating a synchronized and interactive experience between NB Metha and your Google Sheet.

The magic of NB Metha lies in its real-time data synchronization with Google Sheets. As you update your spreadsheet, watch your collage transform dynamically, creating an immersive and responsive user experience.

One of the standout features of NB Metha is its ability to connect seamlessly with Google Sheets, leveraging the power of cloud-based data storage and collaboration. This integration is made possible through a robust API (Application Programming Interface), allowing for the effortless transfer of data between the Google Sheet and NB Metha.

* 1. **PROBLEM INTRODUCTION**

**Connectivity Challenges**

Users may encounter difficulties establishing a connection between NB Metha and their Google Sheets accounts. This can hinder the seamless data transfer required for the real-time collaboration features.

**Data Synchronization Concerns**

Real-time data synchronization is a core feature of NB Metha. However, users might experience delays in data updates between the Google Sheet and the website, impacting the dynamic nature of collages.

**User Experience Hurdles**

Addressing issues related to navigation and feature discoverability will enhance the overall user experience.

**Collaboration and Sharing Complexities**

Some users may face challenges when attempting to collaborate on a single collage. Enhancements in collaborative editing functionalities can facilitate smoother teamwork and communication.

**Performance and Technical Concerns**

Users may experience slow loading times when accessing NB Metha, affecting the overall responsiveness of the website. Identifying and addressing the factors contributing to sluggish performance is crucial for an optimal user experience.

* 1. **OBJECTIVES**

1. User friendly

2. Simple fast

3. Low cost and effective

4. Optimize Data Synchronization

5. Address Performance and Technical Concerns

* 1. **PURPOSE**

At the core of NB Metha purpose is the belief that creativity should not be hindered by technical complexities. The platform strives to bridge the gap between artistic expression and efficient functionality, offering a versatile canvas for users to create captivating collages effortlessly.

* 1. **SCOPE**

The scope of NB Metha encompasses a wide range of features and functionalities, providing users with a versatile platform for creative expression and data visualization. Users can personalize collages with themes, layouts, colors, and design elements, ensuring a unique and tailored visual representation. The platform supports the seamless integration of images, allowing users to enhance their collages with visual content. NB Metha connects directly to Google Sheets, enabling real-time data synchronization for dynamic and up-to-date collages.

**CHAPTER 2**

**REQUIREMENTS SPECIFICATION**

To ensure the successful implementation and functionality of NB Metha, a set of comprehensive requirements have been outlined. These requirements cover various aspects of the website, from core features to security measures, and are crucial for providing users with a seamless and reliable experience.

* 1. **Hardware Requirement**

The physical computer resources required are as follow

* **PROCESSOR**: Intel dual Core i3
* **RAM**: 4 GB
* **HARD DISK**: 100 GB
  1. **Software Requirement**

.

* **OPERATING SYSTEM**: Windows 11/ 10 / 7/ XP/ 8
* **FRONT END**: HTML, CSS, Java script.
* **SOFTWARE**: Browser like (Google Chrome, Mozilla Firefox, etc.

**CHAPTER 3**

**SOFTWARE ANALYSIS**

* 1. **Software Specification**
     1. **HTML**

HTML, or Hyper Text Markup Language, is the standard markup language used to create and design documents on the World Wide Web. It is the backbone of web pages and is essential for structuring the content on the internet. HTML provides a set of tags and attributes that define the structure and layout of a web document. HTML has gone through several versions, with the latest major version as of my knowledge cutoff being HTML5. HTML5 introduced new features, improved support for multimedia, and enhanced the capabilities of web applications. HTML helps organize content hierarchically. Elements like headings, paragraphs, lists, tables, images, and links are used to structure and present information.

* + 1. **CSS**

CSS, or Cascading Style Sheets, is a style sheet language used to describe the presentation and formatting of a document written in HTML or XML (including XML dialects such as SVG or XHTML). In the context of web development, CSS is crucial for styling and designing the visual presentation of web pages. One of the core principles of web development is the separation of content (HTML) and presentation (CSS). HTML is used to define the structure and content of a web page, while CSS is responsible for styling and layout. CSS uses selectors to target HTML elements and declarations to define the style rules for those elements.

* + 1. **JavaScript**

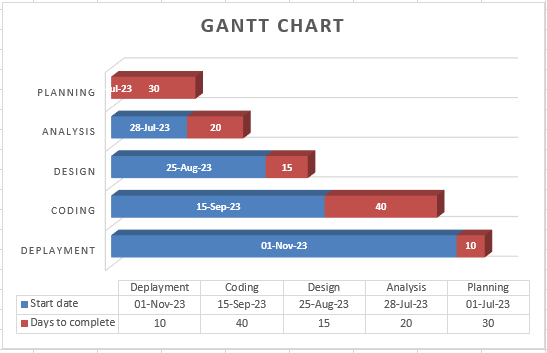
JavaScript is a high-level, versatile, and dynamically typed programming language primarily used for building dynamic, interactive, and client-side functionality on websites. It allows developers to enhance the user experience by adding interactivity, manipulating the Document Object Model (DOM), handling asynchronous events, and communicating with servers. JavaScript is widely used for manipulating the DOM, which represents the structure of a web page. With JavaScript, you can dynamically change content, update styles, handle user input, and create interactive elements.

* + 1. **API**

API stands for Application Programming Interface. It is a set of rules and protocols that allows different software applications to communicate with each other. APIs define the methods and data formats that applications can use to request and exchange information. APIs are used to enable the integration of different software systems, allowing them to work together and share data seamlessly.

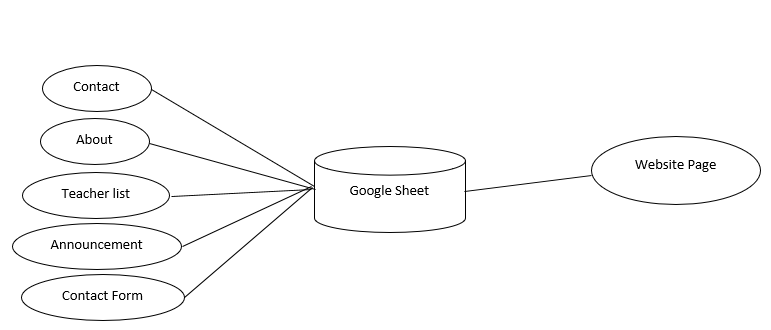
* + 1. **Google Sheet**

Google Sheets is a cloud-based spreadsheet application developed by Google as part of the Google Workspace (formerly G Suite) productivity suite. It allows users to create, edit, and collaborate on spreadsheets online, without the need for desktop software. Google Sheets provides many features similar to traditional spreadsheet applications like Microsoft Excel, but it operates entirely in a web browser.

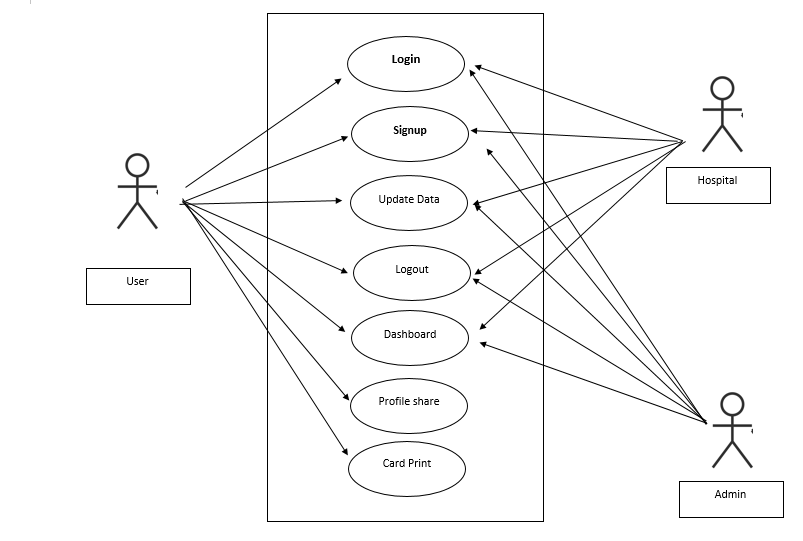
* 1. **Gantt Chart**

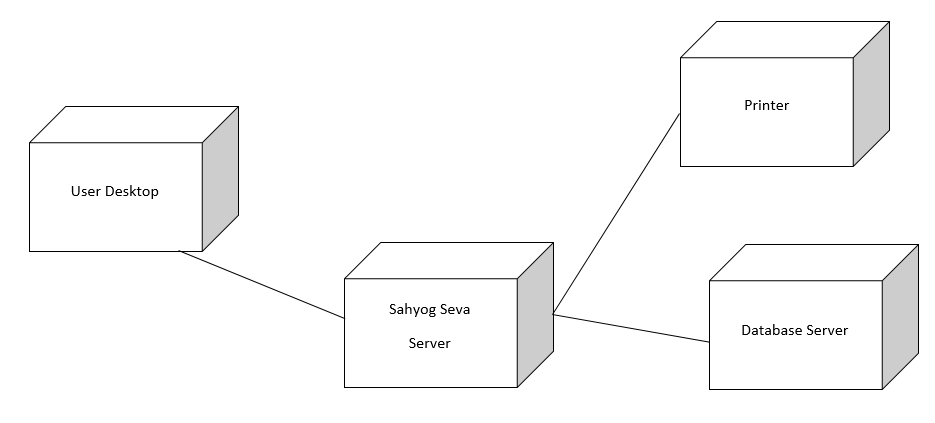
**CHAPTER 4**

**SYSTEM DESIGN**

**4.1 Block Diagram:**

**4.2 Use Case Diagram:**



**4.3 Deployment Diagram:**

**CHAPTER 5**

**Software Testing**

**5.1 Introduction to System Testing:**

The purpose of testing is to discover errors. Testing is the process of trying to discover every conceivable fault or weakness in a work product. It provides a way to check the functionality of components, sub-assemblies, assemblies and/or a finished product. It is the process of exercising software with the intent of ensuring that the Software system meets its requirements and user expectations and does not fail in an unacceptable manner. There are various types of tests. Each test type addresses a specific testing requirement.

**5.2 Module Testing.**

**Admin Model.**

**SignUP**

|  |  |
| --- | --- |
| First Name | Deepak |
| Last Name | Sharma |
| Username | dipu |
| Password | **123** |

**Login**

|  |  |
| --- | --- |
| Username | dipu |
| Password | 123 |

**Hospital Model.**

**SignUP:**

|  |  |
| --- | --- |
| First Name | Manish |
| Last Name | Varma |
| Username | Manish01 |
| Password | 112233 |
| Hospital Name | Mahima Hospital |
| Mobile | 9898989898 |
| Address | Gujrat |
| Choose File | image |

**Login**

|  |  |
| --- | --- |
| Username | Manish01 |
| Password | 112233 |

**User Signup:**

|  |  |
| --- | --- |
| First Name | Suresh |
| Last Name | Jha |
| Username | suru |
| Password | 909090 |
| email | suraj@gmail.com |
| Mobile 1 | 1234567890 |
| Mobile 2 | 9874563210 |
| Address | Gujrat |
| Language | Hindi |
| Gender | Male |
| Symptoms | No |
| Hobby | cricket |
| Choose file | Photo |
| city | valsad |
| Allergy | no |

**User Login:**

|  |  |
| --- | --- |
| Username | suru |
| Password | 909090 |

**Features to be tested**

* Verify that the entries are of the correct format
* No duplicate entries should be allowed
* All links should take the user to the correct page.

**5.3 Test Results:**

All the test cases mentioned above passed successfully. No defects encountered.

**CHAPTER 6**

**CODING AND SNAPSHOTS**

**6.1 Implementation Approaches:**

The Software Design Description Document has been used as input in the implementation process. The actual implementation has been done using Python (Django). Django has been used to interact with the backend database. In this implementation, SQLite Server has been used as the backend DBMS.

**6.1.1 Code:**

from django.shortcuts import render,redirect,reverse

from . import forms,models

from django.db.models import Sum

from django.contrib.auth.models import Group

from django.http import HttpResponseRedirect

from django.core.mail import send\_mail

from django.contrib.auth.decorators import login\_required,user\_passes\_test

from datetime import datetime,timedelta,date

from django.conf import settings

from django.db.models import Q

import time

# Create your views here.

def home\_view(request):

if request.user.is\_authenticated:

return HttpResponseRedirect('afterlogin')

return render(request,'hospital/index.html')

#for showing signup/login button for admin(by sumit)

def adminclick\_view(request):

if request.user.is\_authenticated:

return HttpResponseRedirect('afterlogin')

return render(request,'hospital/adminclick.html')

#for showing signup/login button for doctor(by sumit)

def doctorclick\_view(request):

if request.user.is\_authenticated:

return HttpResponseRedirect('afterlogin')

return render(request,'hospital/doctorclick.html')

#for showing signup/login button for patient(by sumit)

def patientclick\_view(request):

if request.user.is\_authenticated:

return HttpResponseRedirect('afterlogin')

return render(request,'hospital/patientclick.html')

def admin\_signup\_view(request):

form=forms.AdminSigupForm()

if request.method=='POST':

form=forms.AdminSigupForm(request.POST)

if form.is\_valid():

user=form.save()

user.set\_password(user.password)

user.save()

my\_admin\_group = Group.objects.get\_or\_create(name='ADMIN')

my\_admin\_group[0].user\_set.add(user)

return HttpResponseRedirect('adminlogin')

return render(request,'hospital/adminsignup.html',{'form':form})

@login\_required(login\_url='adminlogin')

@user\_passes\_test(is\_admin)

def admin\_appointment\_view(request):

return render(request,'hospital/admin\_appointment.html')

@login\_required(login\_url='adminlogin')

@user\_passes\_test(is\_admin)

def admin\_view\_appointment\_view(request):

return render(request,'hospital/admin\_view\_appointment.html')

@login\_required(login\_url='adminlogin')

@user\_passes\_test(is\_admin)

def admin\_approve\_appointment\_view(request):

#those whose approval are needed

appointments=models.Appointment.objects.all().filter(status=False)

return render(request,'hospital/admin\_approve\_appointment.html',{'appointments':appointments})

@login\_required(login\_url='adminlogin')

@user\_passes\_test(is\_admin)

def approve\_appointment\_view(request,pk):

appointment=models.Appointment.objects.get(id=pk)

appointment.status=True

appointment.save()

return redirect(reverse('admin-approve-appointment'))

@login\_required(login\_url='adminlogin')

@user\_passes\_test(is\_admin)

def reject\_appointment\_view(request,pk):

appointment=models.Appointment.objects.get(id=pk)

appointment.delete()

return redirect('admin-approve-appointment')

@login\_required(login\_url='doctorlogin')

@user\_passes\_test(is\_doctor)

def doctor\_patient\_view(request):

mydict={

'doctor':models.Doctor.objects.get(user\_id=request.user.id), #for profile picture of doctor in sidebar

}

patients=models.Patient.objects.all().filter(status=True)

return render(request,'hospital/aadmin\_view\_patient.html',{'patients':patients,'mydict':mydict})

@login\_required(login\_url='doctorlogin')

@user\_passes\_test(is\_doctor)

@login\_required(login\_url='doctorlogin')

@user\_passes\_test(is\_doctor)

def search\_view(request):

doctor=models.Doctor.objects.get(user\_id=request.user.id) #for profile picture of doctor in sidebar

# whatever user write in search box we get in query

query = request.GET['query']

patients=models.Patient.objects.all().filter(status=True,assignedDoctorId=request.user.id).filter(Q(symptoms\_\_icontains=query)|Q(user\_\_first\_name\_\_icontains=query))

return render(request,'hospital/doctor\_view\_patient.html',{'patients':patients,'doctor':doctor})

@login\_required(login\_url='doctorlogin')

@user\_passes\_test(is\_doctor)

def doctor\_view\_discharge\_patient\_view(request):

dischargedpatients=models.PatientDischargeDetails.objects.all().distinct().filter(assignedDoctorName=request.user.first\_name)

doctor=models.Doctor.objects.get(user\_id=request.user.id) #for profile picture of doctor in sidebar

return render(request,'hospital/doctor\_view\_discharge\_patient.html',{'dischargedpatients':dischargedpatients,'doctor':doctor})

@login\_required(login\_url='doctorlogin')

@user\_passes\_test(is\_doctor)

def doctor\_appointment\_view(request):

doctor=models.Doctor.objects.get(user\_id=request.user.id) #for profile picture of doctor in sidebar

return render(request,'hospital/doctor\_appointment.html',{'doctor':doctor})

def user\_data\_profile(request,pk):

try:

user=models.User.objects.get(username=pk)

patient=models.Patient.objects.get(user\_id=user.id)

data={

'patient':user,

'fname':user.first\_name,

'lname':user.last\_name,

'mobile':patient.mobile,

'profile\_pic':patient.profile\_pic,

'aadhar':patient.aadhar,

'address\_nd':patient.address\_nd,

'address':patient.address,

'mobile\_nd':patient.mobile\_nd,

'email':patient.email,

'aadhar':patient.aadhar,

'state':patient.state,

'city':patient.city,

'pincode':patient.pincode,

'bloodGroup':patient.bloodGroup,

'dob':patient.dob,

'gender':patient.gender,

'language':patient.language,

'motherName':patient.motherName,

'fatherName':patient.fatherName,

'symptoms':patient.symptoms,

'other':patient.other,

'hobby':patient.hobby,

'allergy':patient.allergy,

'pastMediacalRec':patient.pastMediacalRec,

'allergy':patient.allergy,

'uname':pk

}

return render(request,'hospital/mm.html',{'data':data})

except:

return render(request,'hospital/notfound.html')

@login\_required(login\_url='patientlogin')

@user\_passes\_test(is\_patient)

def patient\_dashboard\_view(request):

patient=models.Patient.objects.get(user\_id=request.user.id)

user=models.User.objects.get(username=request.user)

data={

'patient':user,

'fname':user.first\_name,

'lname':user.last\_name,

'mobile':patient.mobile,

'profile\_pic':patient.profile\_pic,

'aadhar':patient.aadhar,

'address\_nd':patient.address\_nd,

'address':patient.address,

'mobile\_nd':patient.mobile\_nd,

'email':patient.email,

'aadhar':patient.aadhar,

'state':patient.state,

'city':patient.city,

'pincode':patient.pincode,

'bloodGroup':patient.bloodGroup,

'dob':patient.dob,

'gender':patient.gender,

'language':patient.language,

'motherName':patient.motherName,

'fatherName':patient.fatherName,

'symptoms':patient.symptoms,

'other':patient.other,

'hobby':patient.hobby,

'allergy':patient.allergy,

'pastMediacalRec':patient.pastMediacalRec,

'allergy':patient.allergy,

'uname':user

}

if request.method=='POST':

userForm=forms.PatientUserForm(request.POST,instance=user)

patientForm=forms.PatientForm(request.POST,request.FILES,instance=patient)

user=models.User.objects.get(username=request.user)

print("mthid call")

# print(patientForm.is\_valid())

# print(patientForm.errors.as\_data())

if patientForm.is\_valid():

patient.user=user

print(patient.user)

patient=patientForm.save(commit=False)

patient.status=True

mess="update sucessfully"

patient=patient.save()

return render(request,'hospital/patient\_dashboard.html',{'data':data,'mess':mess})

return render(request,'hospital/patient\_dashboard.html',{'data':data})

@login\_required(login\_url='patientlogin')

@user\_passes\_test(is\_patient)

def patient\_visiting\_card(request,pk,jk):

print(pk)

print(jk)

data={

"uname":pk,

'name':jk

}

return render(request,'hospital/visiting.html',{'data':data})

@login\_required(login\_url='patientlogin')

@user\_passes\_test(is\_patient)

def patient\_appointment\_view(request):

patient=models.Patient.objects.get(user\_id=request.user.id) #for profile picture of patient in sidebar

return render(request,'hospital/patient\_appointment.html',{'patient':patient})

@login\_required(login\_url='patientlogin')

@user\_passes\_test(is\_patient)

def patient\_book\_appointment\_view(request):

appointmentForm=forms.PatientAppointmentForm()

patient=models.Patient.objects.get(user\_id=request.user.id) #for profile picture of patient in sidebar

message=None

mydict={'appointmentForm':appointmentForm,'patient':patient,'message':message}

if request.method=='POST':

appointmentForm=forms.PatientAppointmentForm(request.POST)

if appointmentForm.is\_valid():

print(request.POST.get('doctorId'))

desc=request.POST.get('description')

doctor=models.Doctor.objects.get(user\_id=request.POST.get('doctorId'))

appointment=appointmentForm.save(commit=False)

appointment.doctorId=request.POST.get('doctorId')

appointment.patientId=request.user.id #----user can choose any patient but only their info will be stored

appointment.doctorName=models.User.objects.get(id=request.POST.get('doctorId')).first\_name

appointment.patientName=request.user.first\_name #----user can choose any patient but only their info will be stored

appointment.status=False

appointment.save()

return HttpResponseRedirect('patient-view-appointment')

return render(request,'hospital/patient\_book\_appointment.html',context=mydict)

def patient\_view\_doctor\_view(request):

doctors=models.Doctor.objects.all().filter(status=True)

patient=models.Patient.objects.get(user\_id=request.user.id) #for profile picture of patient in sidebar

return render(request,'hospital/patient\_view\_doctor.html',{'patient':patient,'doctors':doctors})

def search\_doctor\_view(request):

patient=models.Patient.objects.get(user\_id=request.user.id) #for profile picture of patient in sidebar

# whatever user write in search box we get in query

query = request.GET['query']

doctors=models.Doctor.objects.all().filter(status=True).filter(Q(department\_\_icontains=query)| Q(user\_\_first\_name\_\_icontains=query))

return render(request,'hospital/patient\_view\_doctor.html',{'patient':patient,'doctors':doctors})

@login\_required(login\_url='patientlogin')

@user\_passes\_test(is\_patient)

def patient\_view\_appointment\_view(request):

patient=models.Patient.objects.get(user\_id=request.user.id) #for profile picture of patient in sidebar

appointments=models.Appointment.objects.all().filter(patientId=request.user.id)

return render(request,'hospital/patient\_view\_appointment.html',{'appointments':appointments,'patient':patient})

@login\_required(login\_url='patientlogin')

@user\_passes\_test(is\_patient)

def patient\_discharge\_view(request):

patient=models.Patient.objects.get(user\_id=request.user.id) #for profile picture of patient in sidebar

dischargeDetails=models.PatientDischargeDetails.objects.all().filter(patientId=patient.id).order\_by('-id')[:1]

patientDict=None

if dischargeDetails:

patientDict ={

'is\_discharged':True,

'patient':patient,

'patientId':patient.id,

'patientName':patient.get\_name,

'assignedDoctorName':dischargeDetails[0].assignedDoctorName,

'address':patient.address,

'mobile':patient.mobile,

'symptoms':patient.symptoms,

'admitDate':patient.admitDate,

'releaseDate':dischargeDetails[0].releaseDate,

'daySpent':dischargeDetails[0].daySpent,

'medicineCost':dischargeDetails[0].medicineCost,

'roomCharge':dischargeDetails[0].roomCharge,

'doctorFee':dischargeDetails[0].doctorFee,

'OtherCharge':dischargeDetails[0].OtherCharge,

'total':dischargeDetails[0].total,

}

print(patientDict)

else:

patientDict={

'is\_discharged':False,

'patient':patient,

'patientId':request.user.id,

}

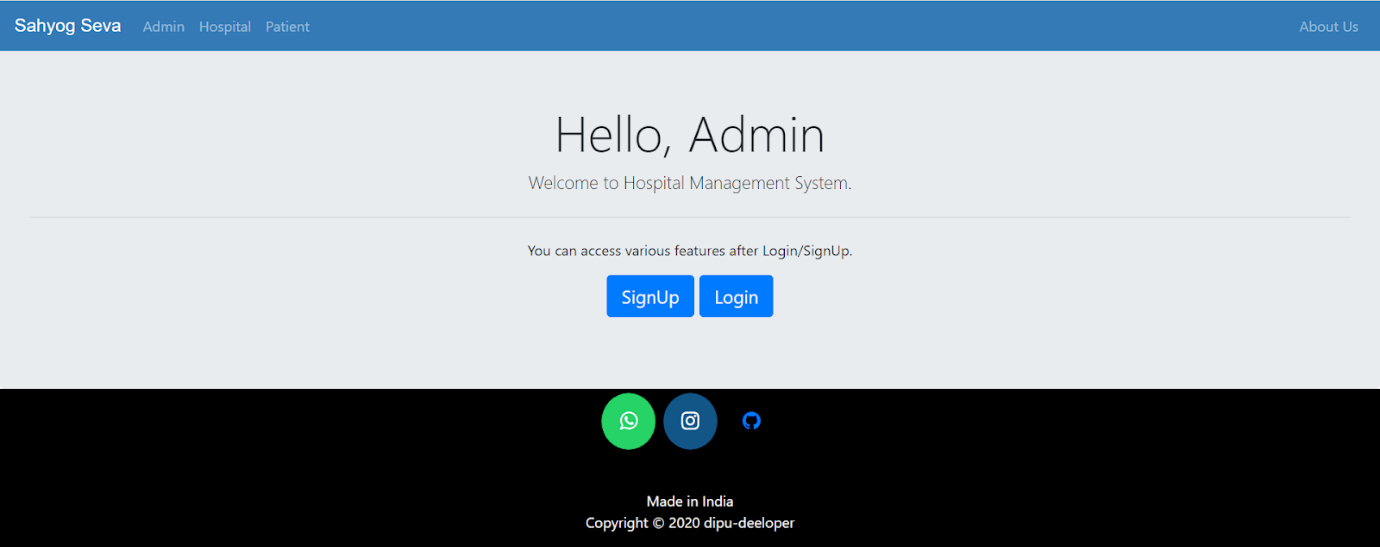
return render(request,'hospital/patient\_discharge.html',context=patientDict)

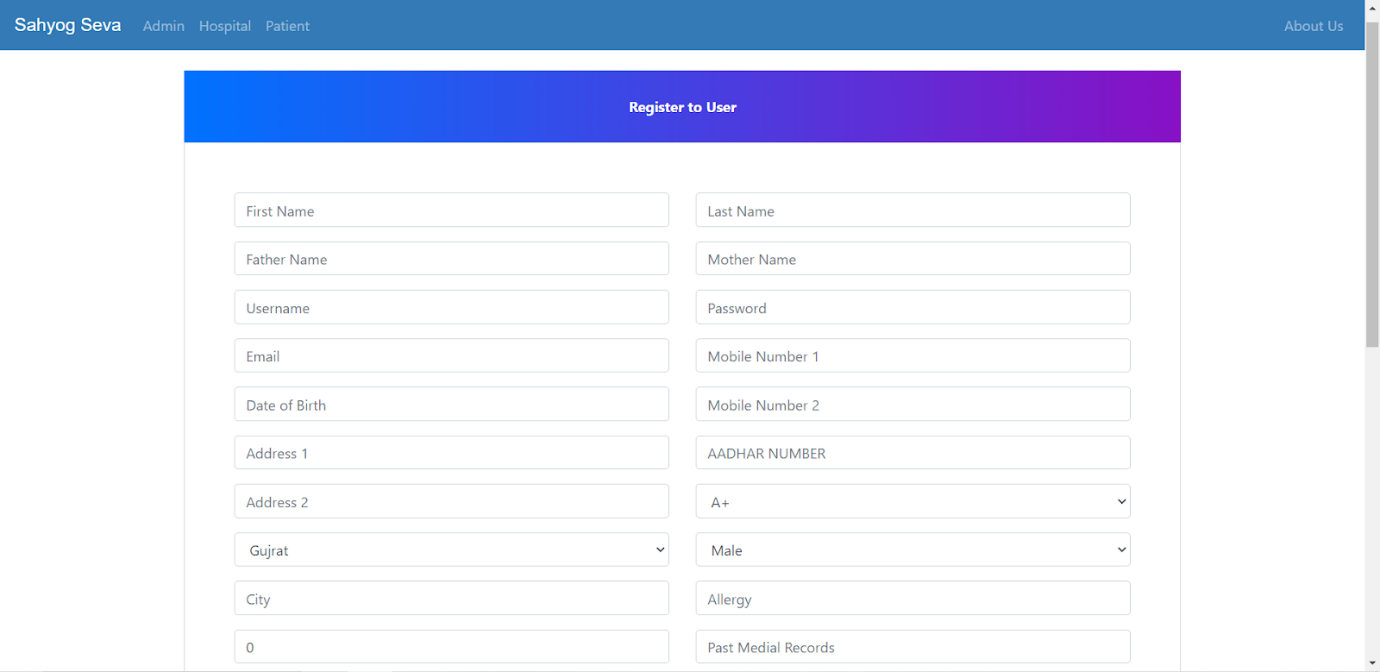
def aboutus\_view(request):

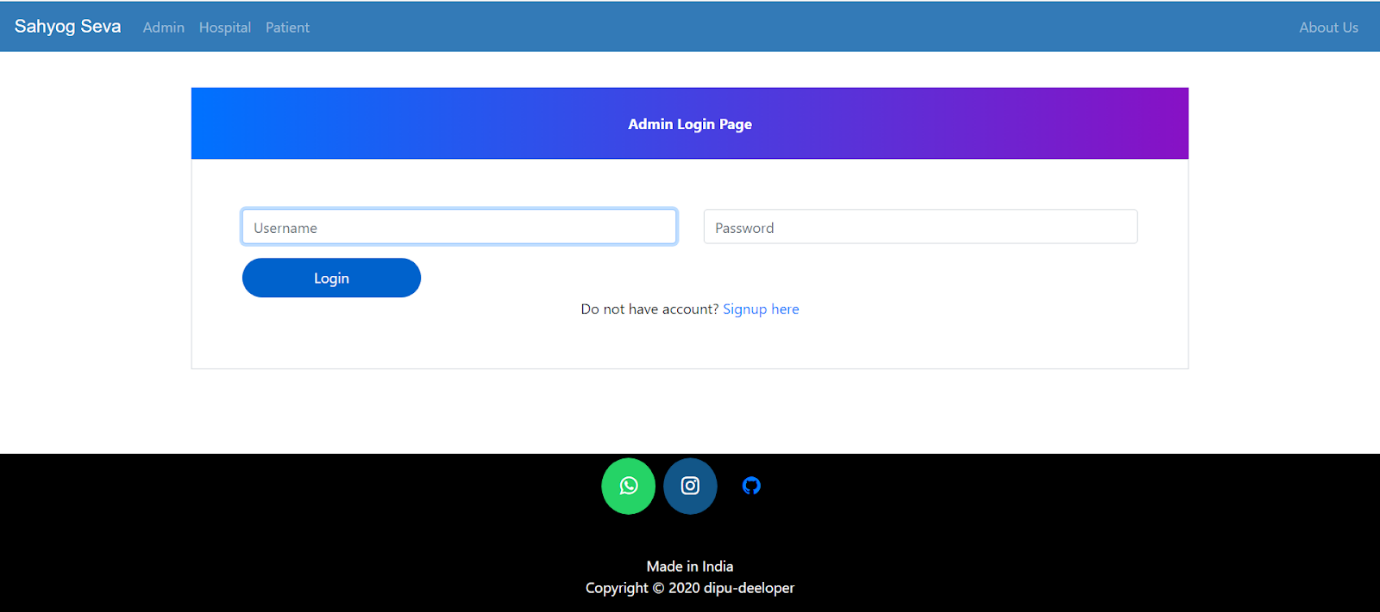
return render(request,'hospital/aboutus.html')

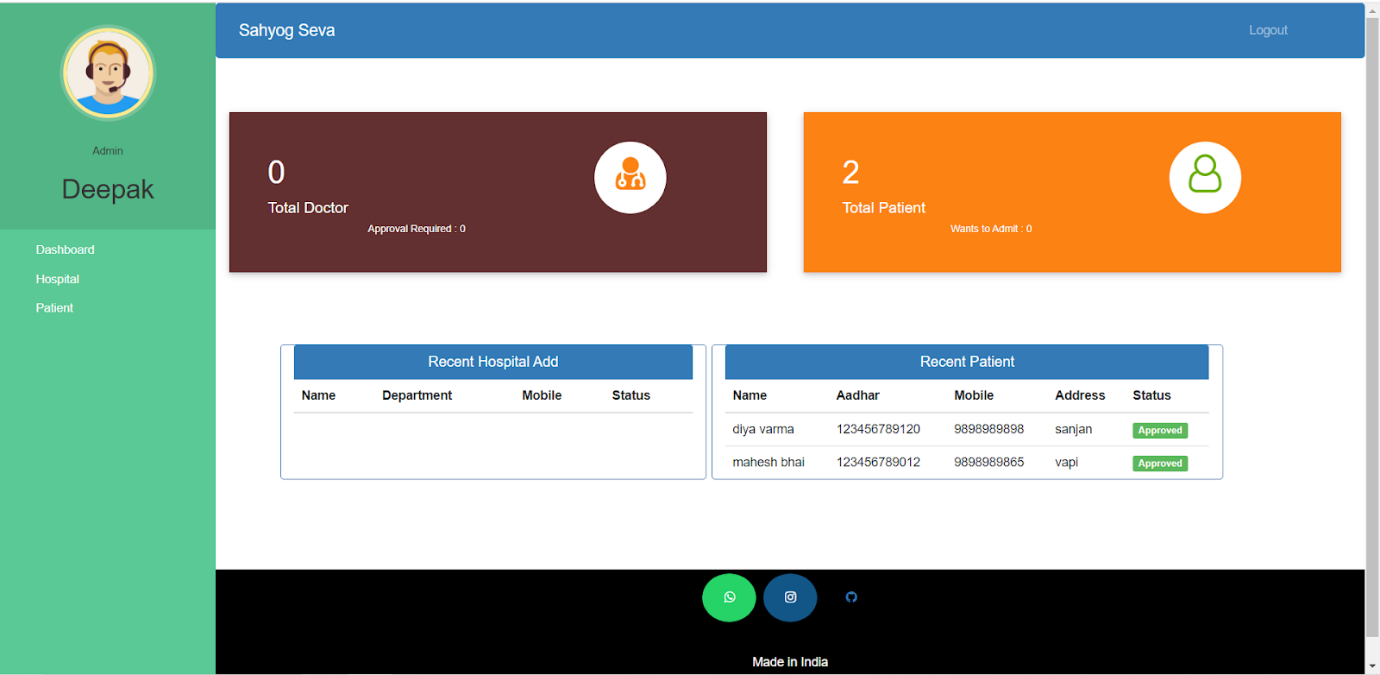
**6.1.2 Snapshot**

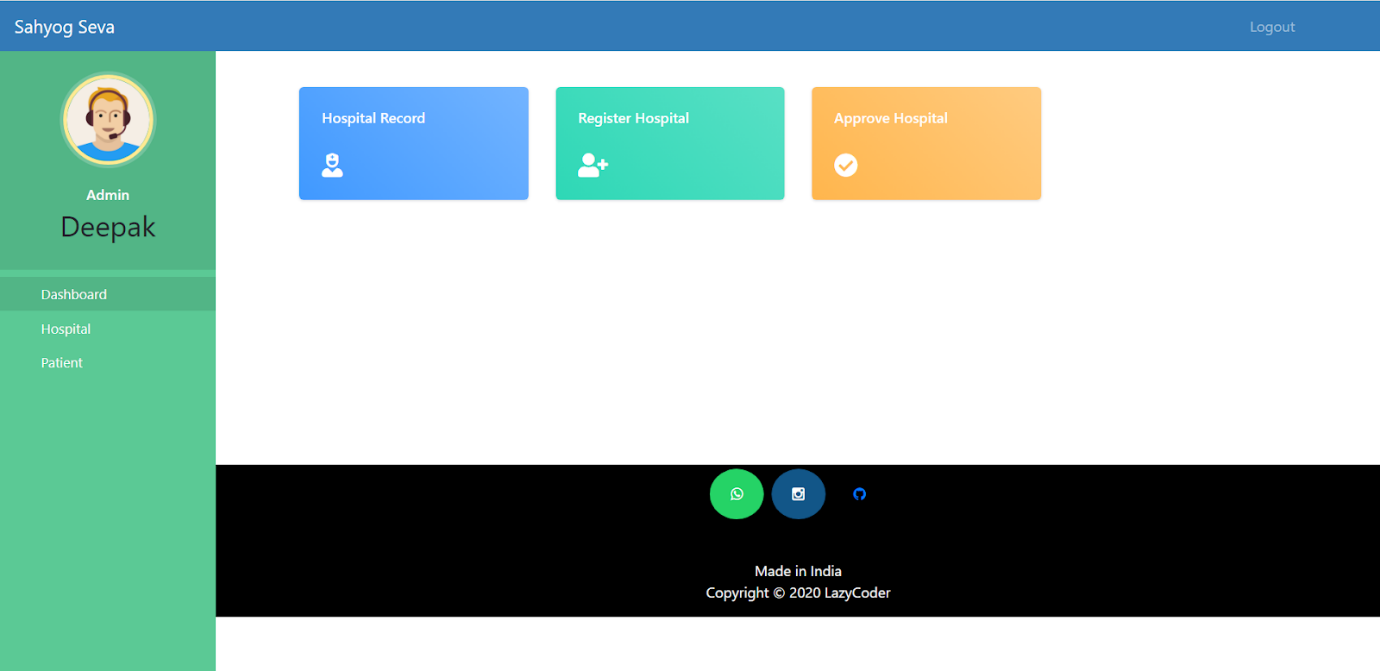


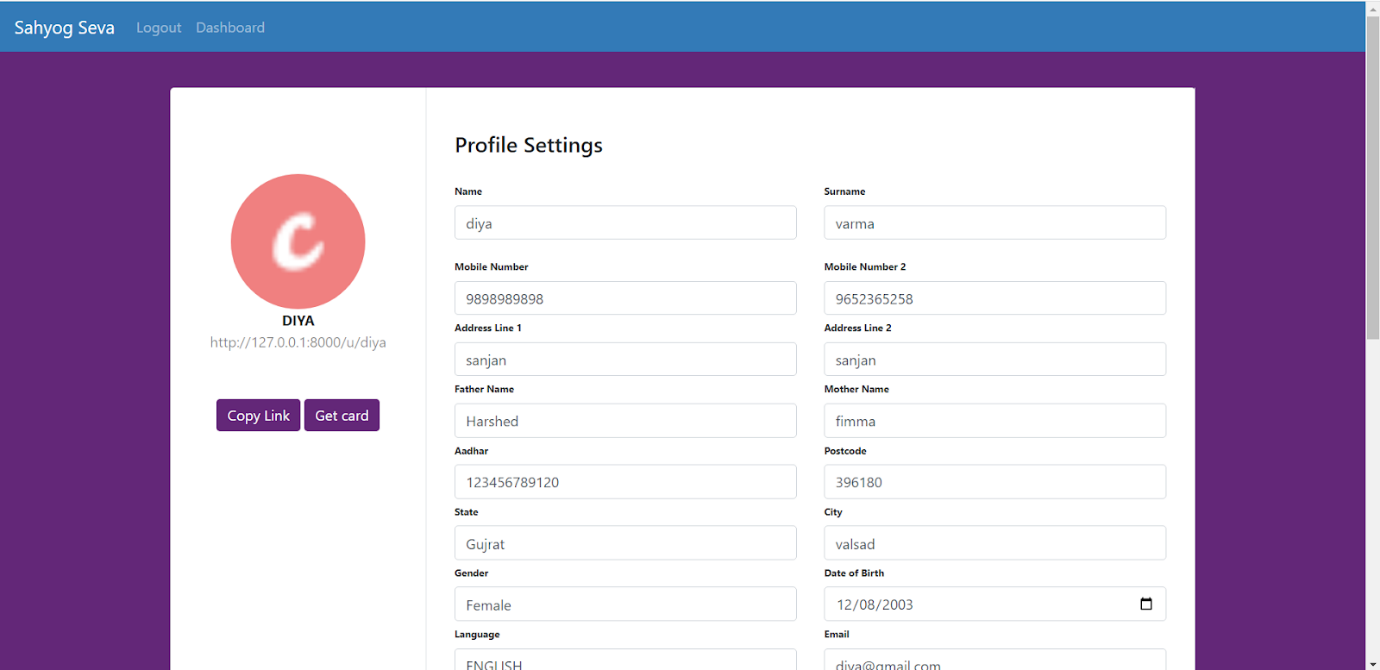


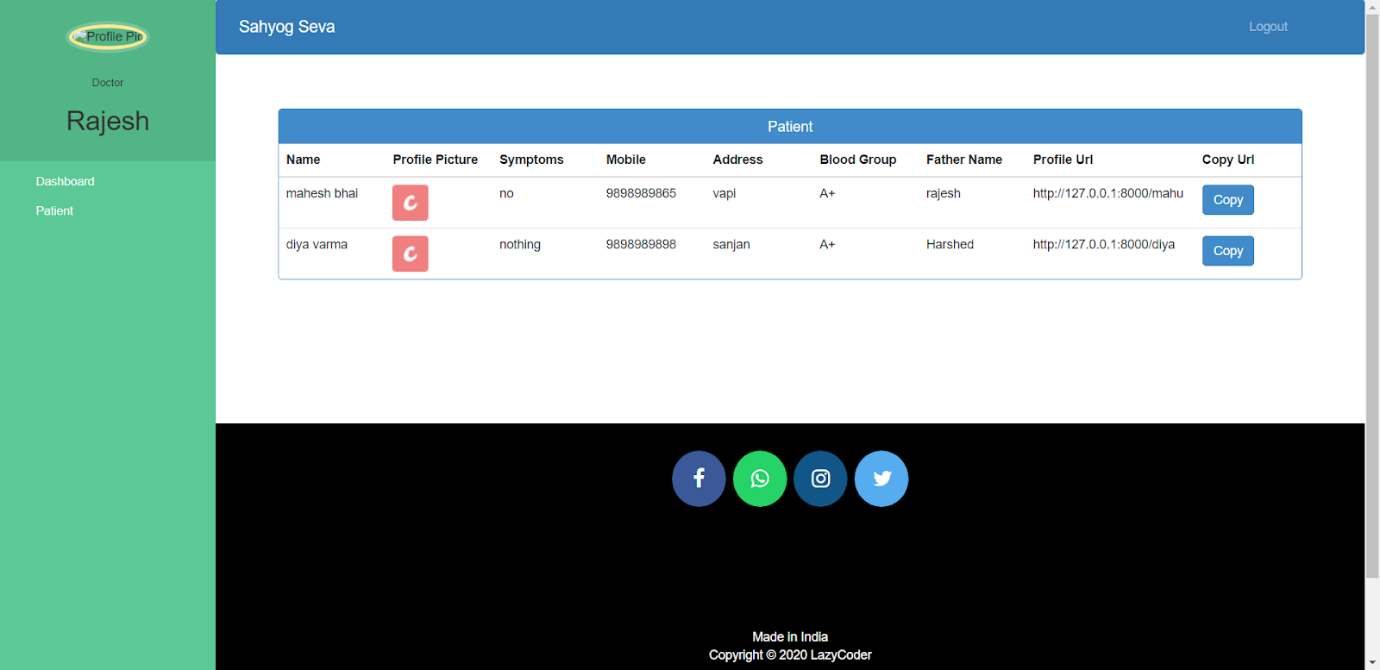
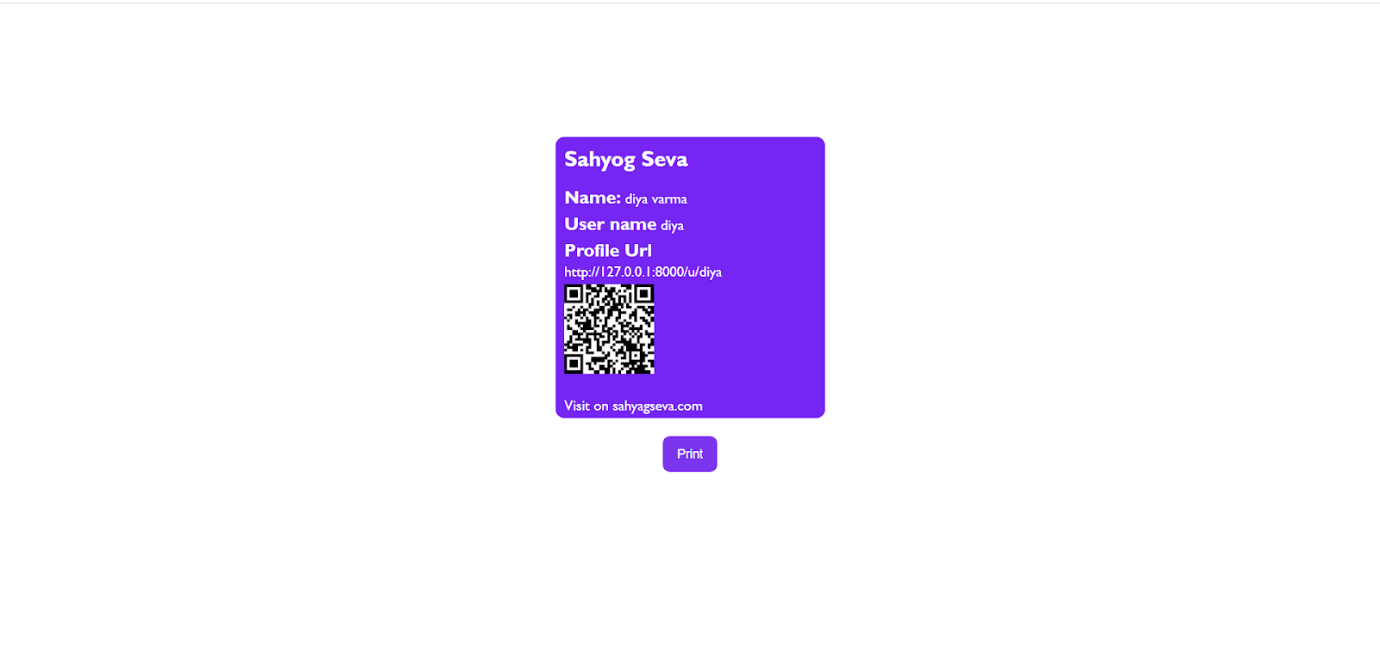












**CHAPTER 7**

**CONCLUSION**

Working on the project was an excellent experience. It helped us to understand the importance of planning, designing and implementation so far we have learnt in our theory books. It helped us unleashing our creativity while working in a team. It also realized the importance of team working, communication as a part of this project.

The project was successfully completed after a lot of effort and work hours. This project underwent a number of compiling, debugging, removing errors, making it bugs free, adding more facilities in Sahyog Seva System and interactivity making it more reliable and useful.

The entire project has been developed and deployed as per the requirements stated by the user.

Finally, we like to conclude that we put all our efforts throughout the development of our project and tried to fulfill most of the requirements of the user.

**CHAPTER 8**

**FUTURE ENHANCEMENT**

This project can be used in the hospitals and personal use and after adding some more useful modules in the project for which services providing.

Utmost care and back-up procedures must be established to ensure 100% successful implementation of the computerized this system. In the case of system failure, the organization should be in a position to process the transaction with another organization or if the worst comes to the worst, it should be in a position to complete it manually.

**8.1 Scope of Improvement.**

* Adding facial reorganization
* Adding auto verification.
* Adding live location share.
* Adding auto share accident case.

**CHAPTER 9**

**REFERENCES**

[1] “How to start a new Django project,” *Stack Overflow*, 2018. <https://stackoverflow.com/questions/28842901/how-to-start-a-new-django-project> (accessed Dec. 17, 2023).

[2] CodeWithHarry, “Django Tutorial In Hindi,” *YouTube*. Apr. 01, 2020. Accessed: Jul. 25, 2020. [Online]. Available: <https://www.youtube.com/watch?v=JxzZxdht-XY>

[3] OpenAI, “ChatGPT,” *chat.openai.com*, 2023. <https://chat.openai.com/>

[4] W3Schools, “HTML Tutorial,” *W3schools.com*, 2018. <https://www.w3schools.com/html/>

[5] “Bootstrap 3 or Bootstrap 4,” *W3schools.com*, 2019. <https://www.w3schools.com/bootstrap/bootstrap_ver.asp>

[6] W3Schools, “SQL Tutorial,” *W3schools.com*, 2019. <https://www.w3schools.com/sql/default.asp>

[7] “Hospital Management System,” *GitHub*, Oct. 05, 2023. https://github.com/IT-Department-Projects/OOAD-Project (accessed Dec. 17, 2023).

[8] “MySQL Tutorial - Tutorialspoint,” *Tutorialspoint.com*, 2019. <https://www.tutorialspoint.com/mysql/index.htm>

[9] D. Gavor, “The Best CSS Button Hover Effects You Can Use Too,” *Slider Revolution*, Feb. 25, 2021. <https://www.sliderrevolution.com/resources/css-button-hover-effects/>

[10] MDN, “JavaScript — Dynamic client-side Scripting - Learn Web Development | MDN,” *developer.mozilla.org*, 2019. <https://developer.mozilla.org/en-US/docs/Learn/JavaScript>