VISVESVARAYA TECHNOLOGICAL UNIVERSITY

“Jnana Sangama”, Belagavi – 590 018

A
Mini Project Report
on

“HOSPITAL MANAGEMENT SYSTEM”

Submitted in partial fulfillment of the requirement for the DBMS Laboratory with miniproject(18CSL58) of V Semester

Bachelor of Engineering
in
Computer Science and Engineering

Submitted By

ADITYA S NIRGUND [1GA18CS013]

Under the Guidance of
Mrs.Vanishree M L
Assistant Professor, Dept. of CSE

Department of Computer Science and Engineering
GLOBAL ACADEMY OF TECHNOLOGY
Rajarajeshwarinagar, Bengaluru - 560 098
2020 – 2021

GLOBAL ACADEMY OF TECHNOLOGY

Department of Computer Science and Engineering

CERTIFICATE

Certified that the V Semester Mini Project in DBMS Laboratory with mini project Entitled **“HOSPITAL MANAGEMENT SYSTEM”** carried out by **Mr.ADITYA S NIRGUND**, bearing **USN 1GA18CS013** a bonafide student of Global Academy of Technology, in partial fulfillment for the award of the **BACHELOR OF ENGINEERING** in Computer Science and Engineering from **Visvesvaraya Technological University, Belagavi** during the year 2020-2021. It is certified that all the corrections/suggestions indicated for Internal Assessment have been incorporated in the report submitted in the Department Library.The DBMS Mini Project report has been approved as it satisfies the academic requirements in respect of the miniproject work prescribed for the said Degree.

____________________	____________________
Mrs. Vanishree M L	Dr. Srikanta Murthy K
Assistant Professor	Professor & HOD
Dept. of CSE	Dept. of CSE
GAT,Bengaluru.	GAT,Bengaluru.

Name of the Examiners	Signature with date
1. ____________________	____________________
2. ____________________	____________________

ABSTRACT

A system to manage the activities in a hospital:

Medical care and services (medical examinations and operations, diagnostics, laboratory services, surgery and other treatment, rehabilitation, urgent medical care etc.); non-medical care (accommodation, food for hospitalised patients etc.);
Patients request for appointment for any doctor. The details of the existing patients are retrieved by the system.

New patients update their details in the system before they request for appointment with the help of assistant. The assistant confirms the appointment based on the availability of free slots for the respective doctors and the patient is informed. Assistant may cancel the appointment at any time. It is used to storing and managing the administrator information. Administrators manage & maintain the whole system. Generating the bills.

We create super user where he can give login credentials so he can create, edit and delete the records of the hospital management system

ACKNOWLEDGEMENT

The satisfaction and euphoria that accompany the successful completion of any task would be incomplete without the mention of the people who made it possible and whose constant encouragement and guidance crowned our efforts with success.

I consider myself proud, to be part of **Global Academy of Technology** family, the institution which stood by our way in endeavors.

I express my deep and sincere thanks to our Principal **Dr. N. Rana Pratap Reddy** for his support.

I am grateful to **Dr. Srikanta Murthy K,** Professor and HOD, Dept of CSE who is the source of inspiration and of invaluable help in channelizing my efforts in right direction.

I wish to thank my internal guide **Mrs. Vanishree M L**, Assistant Professor, Dept of CSE for guiding and correcting various documents of mine with attention and care. She has taken lot of pain to go through the document and make necessary corrections as and when needed.

I would like to thank the faculty members and supporting staff of the Department of CSE, GAT for providing all the support for completing the Project work.

Finally, I am grateful to my parents and friends for their unconditional support and help during the course of my Project work.

Student Name

ADITYA S NIRGUND

TABLE OF CONTENTS

CHAPTER 1

INTRODUCTION

1.1 INTRODUCTION TO SQL

SQL is a database computer language designed for the retrieval and management of data in a relational database. **SQL** stands for **Structured Query Language**. SQL is Structured Query Language, which is a computer language for storing, manipulating and retrieving data stored in a relational database.

This tutorial will give you a quick start to SQL. It covers most of the topics required for a basic understanding of SQL and to get a feel of how it works. SQL is the standard language for Relational Database System.

All the Relational Database Management Systems (RDMS) like MySQL, MS Access, Oracle, Sybase, Informix, Postgres and SQL Server use SQL as their standard database language.

Here are important elements of SQL language:

- **Keywords:** Each SQL statement contains single or multiple keywords

  .
- **Identifiers:** Identifiers are names of objects in the database, like user IDs, tables, and columns.

- **Strings:** Strings can be either literal strings or expressions with VARCHAR or CHAR data types.

- **Expressions:** Expressions are formed from several elements, like constants, SQL operators, column names, and subqueries.

- **Search Conditions:** Conditions are used to select a subset of the rows from a table or used to control statements like an IF statement to determine control of flow.

- **Special Values:** Special values should be used in expressions and as column defaults when building tables.

- **Variables:** Sybase IQ supports local variables, global variables, and connection-level variables.

- **Comments:** Comment is another SQL element which is used to attach explanatory text to SQL statements or blocks of statements. The database server does not execute any comment.

- **NULL Value:** Use NULL, which helps you to specify a value that is unknown, missing, or not applicable.

1.2 INTRODUCTION TO FRONT END SOFTWARE

Web development is the building and maintenance of websites; it’s the work that happens behind the scenes to make a website look great, work fast and perform well with a seamless user experience.

Web developers, or ‘devs’, do this by using a variety of coding languages. The languages they use depends on the types of tasks they are preforming and the platforms on which they are working.

Web development skills are in high demand worldwide and well paid too – making development a great career option. It is one of the easiest accessible higher paid fields as you do not need a traditional university degree to become qualified.

A front-end dev takes care of layout, design and interactivity using HTML, CSS and JavaScript. They take an idea from the drawing board and turn it into reality.

The backend developer engineers what is going on behind the scenes. This is where the data is stored, and without this data, there would be no frontend. The backend of the web consists of the server that hosts the website, an application for running it and a database to contain the data.

The backend dev uses computer programmes to ensure that the server, the application and the database run smoothly together. This type of dev need to analyse what a company’s needs are and provide efficient programming solutions. To do all this amazing stuff they use a variety of server-side languages, like PHP, Ruby, Python and Java.

HTML

HTML is at the core of every web page, regardless the complexity of a site or number of technologies involved. It's an essential skill for any web professional. It's the starting point for anyone learning how to create content for the web. And, luckily for us, it's surprisingly easy to learn.

HTML stands for HyperText Markup Language. "Markup language" means that, rather than using a programming language to perform functions, HTML uses tags to identify different types of content and the purposes they each serve to the webpage.

CSS

CSS stands for Cascading Style Sheets. This programming language dictates how the HTML elements of a website should actually appear on the frontend of the page.

HTML provides the raw tools needed to structure content on a website. CSS, on the other hand, helps to style this content so it appears to the user the way it was intended to be seen. These languages are kept separate to ensure websites are built correctly before they're reformatted.

JavaScript

JavaScript is a more complicated language than HTML or CSS, and it wasn't released in beta form until 1995. Nowadays, JavaScript is supported by all modern web browsers and is used on almost every site on the web for more powerful and complex functionality.

JavaScript is a logic-based programming language that can be used to modify website content and make it behave in different ways in response to a user's actions. Common uses for JavaScript include confirmation boxes, calls-to-action, and adding new identities to existing information.

1.3 PROJECT REPORT OUTLINE

The report is arranged in the following way:

Chapter 1: INTRODUCTION

Chapter 2: REQUIREMENT SPECIFICATION

Chapter 3: OBJECTIVE OF THE PROJECT

Chapter 4: IMPLEMENTATION

Chapter 5: FRONT END DESIGN

Chapter 6: TESTING

Chapter 7: RESULT

CHAPTER 2

REQUIREMENT SPECIFICATION

2.1 SOFTWARE REQUIREMENTS

Operating System : Microsoft Windows

Database : db.sqlite3

Tools : Django(Front design ,Back end,Python),Html,Css,Javascript

2.2 HARDWARE REQUIREMENTS

Processor : Any Processor above 500 MHz

RAM : 2 GB

Hard Disk : 40 GB

Compact Disk : NA

Input device : Keyboard and Mouse

Output device : Monitor

CHAPTER 3

OBJECTIVE OF THE PROJECT

Main objectives of a Hospital Management System

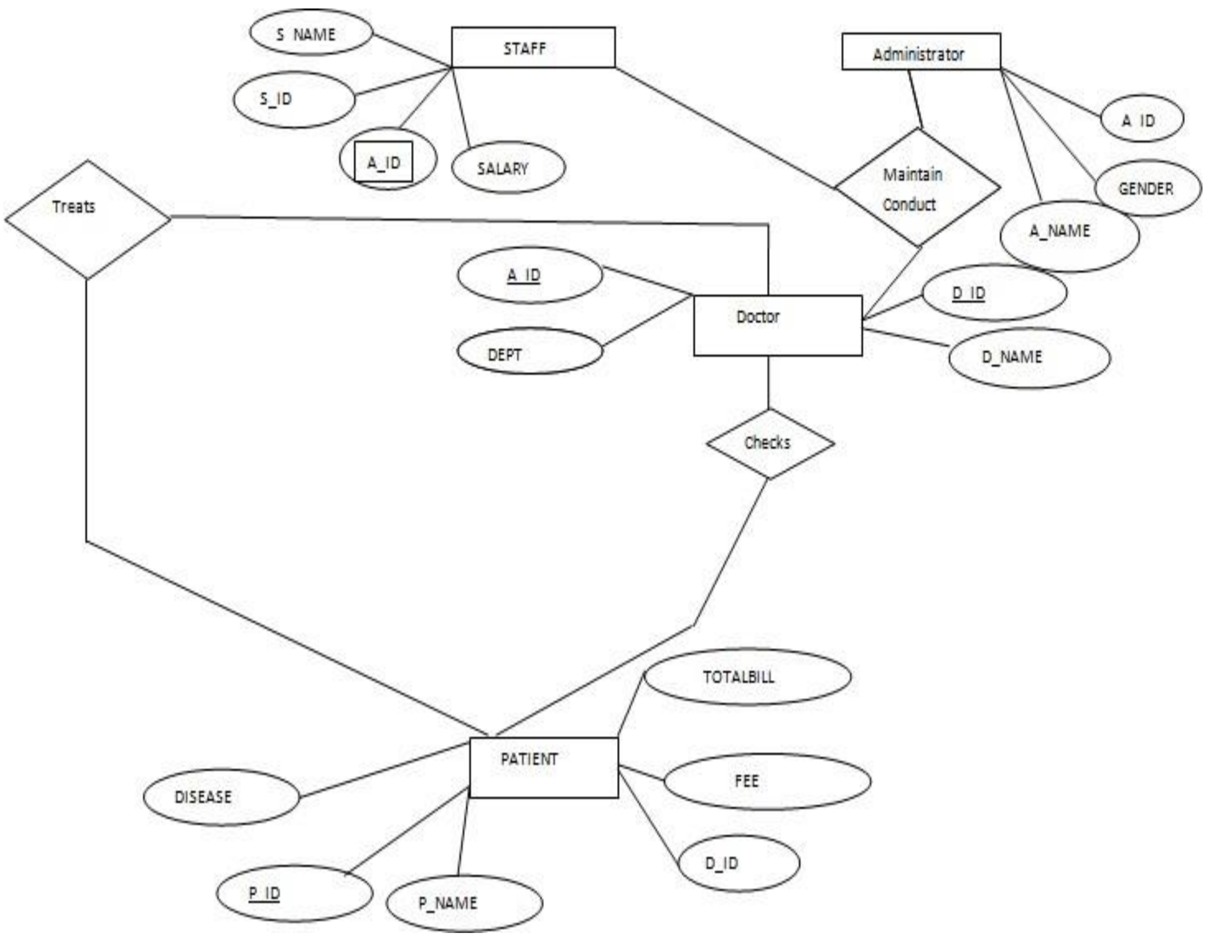
Design a system for better patient care. Reduce hospital operating costs. Provide MIS report on demand to management for better decision making. Better co-ordination among the different departments. Provide top management a single point of control.

To computerize all details regarding patient details & hospital details. Scheduling the appointment of patient with doctors to make it convenient for both. Scheduling the services of specialized doctors and emergency properly so that facilities provided by hospital are fully utilized in effective and efficient manner.

If the medical store issues medicines to patients, it should reduce the stock status of the medical store and vice-versa.The inventory should be updated automatically whenever a transaction is made.The information of the patients should be kept up to date and there record should be kept in the system for historical purposes.

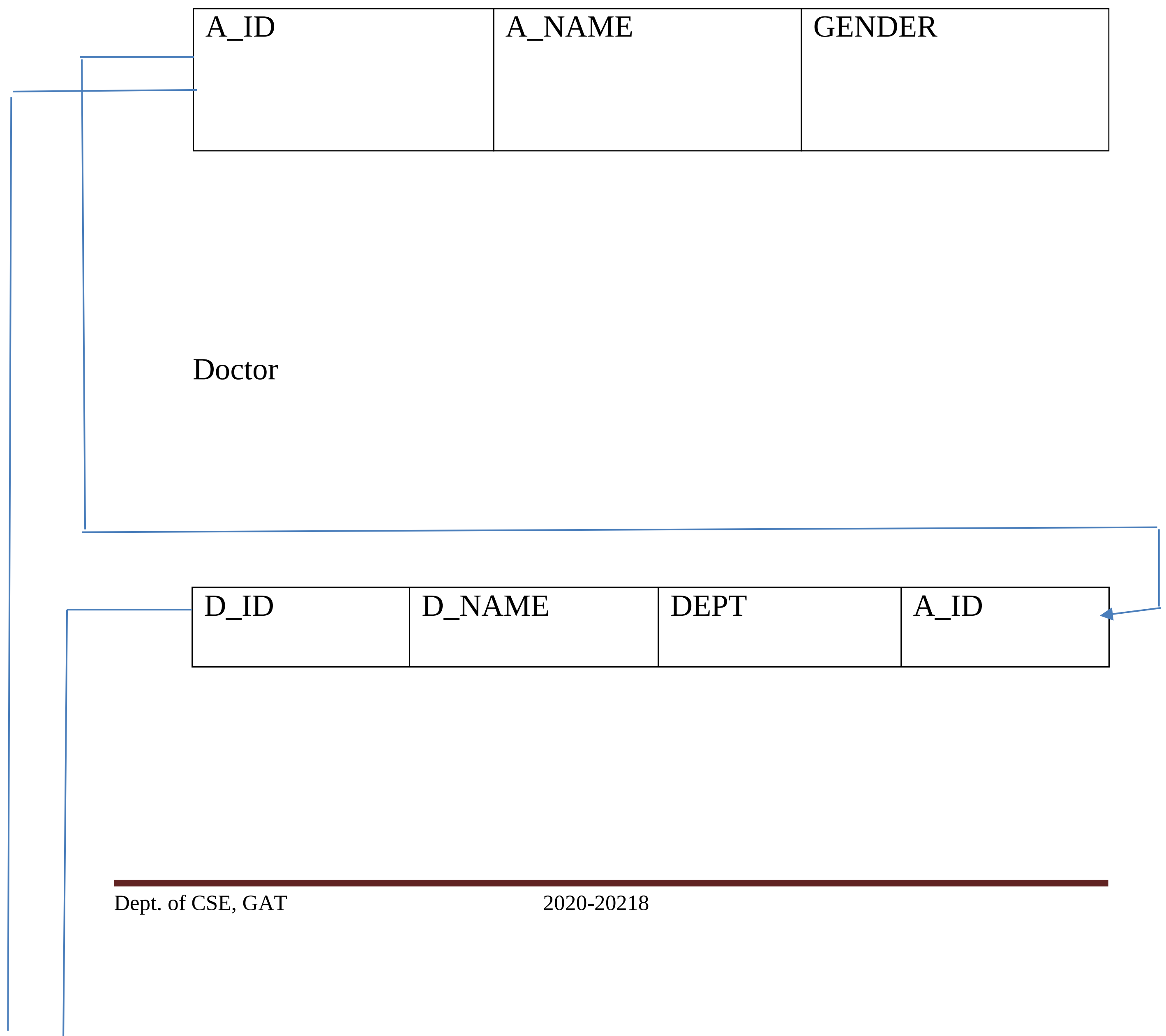
CHAPTER 4 IMPLEMENTATION

4.1 ER DIAGRAM



4.2 MAPPING OF THE ER DIAGRAM TO SCHEMA DIAGRAM

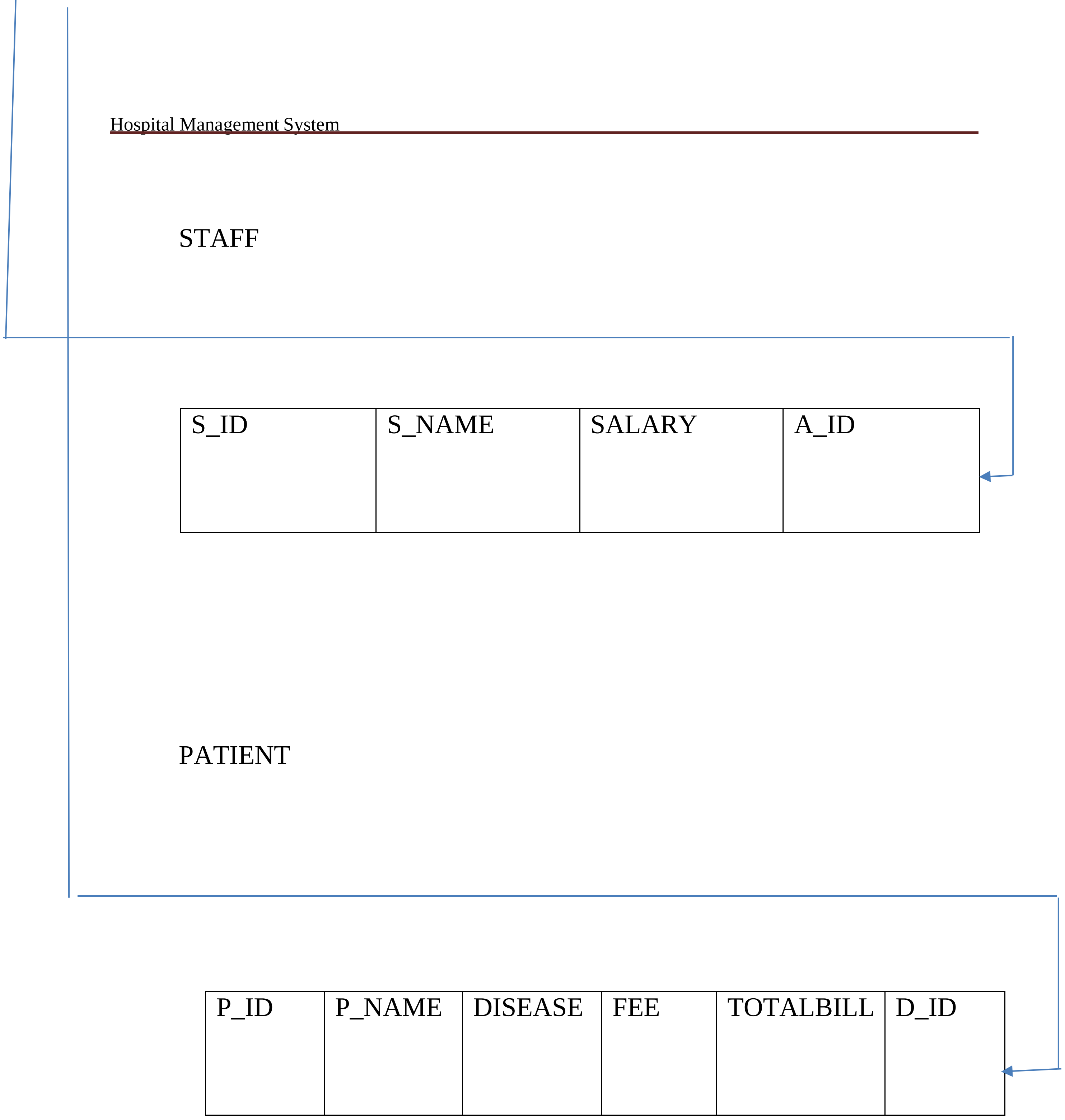
Administration



A_ID	A_NAME	GENDER

Doctor

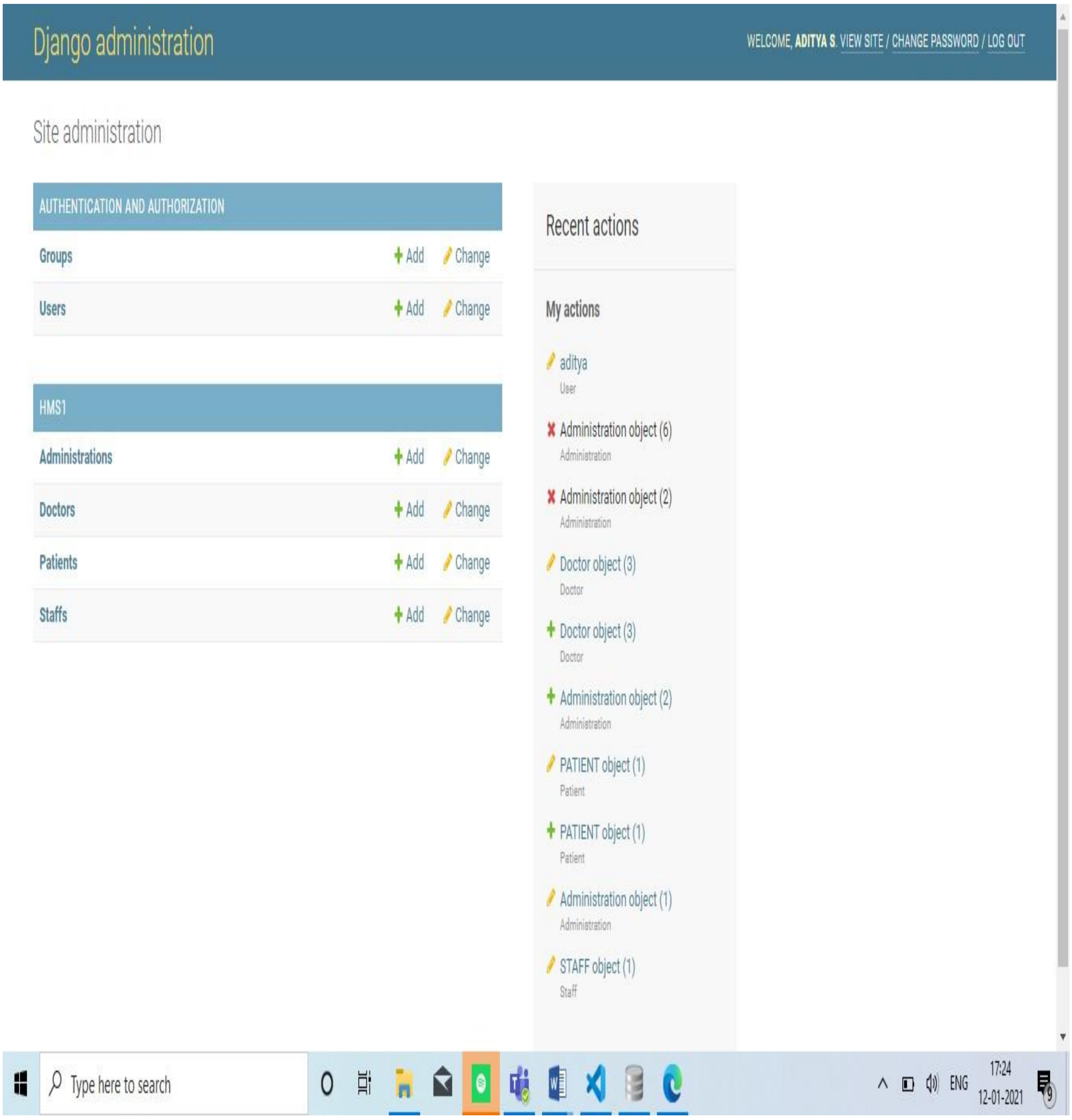
D_ID	D_NAME	DEPT	A_ID


STAFF
S_ID
S_NAME
SALARY
A_ID
PATIENT
P_ID
P_NAME
DISEASE
FEE
TOTALBILL
D_ID

auth_user (TRIGGER TABLE)

id	password	last_login	is_superuser	username	last_name	email	is_staff	is_active	date_joined	first_name

4.3 MAPPING OF THE ER SCHEMA TO RELATIONS



Django administration

WELCOME, **ADITYA S.** VIEW SITE / CHANGE PASSWORD / LOG OUT

Site administration

AUTHENTICATION AND AUTHORIZATION		
Groups	Add	Change
Users	Add	Change

HMS1		
Administrations	Add	Change
Doctors	Add	Change
Patients	Add	Change
Staffs	Add	Change

Recent actions

My actions

- aditya
  User
- Administration object (6)
  Administration
- Administration object (2)
  Administration
- Doctor object (3)
  Doctor
- Doctor object (3)
  Doctor
- Administration object (2)
  Administration
- PATIENT object (1)
  Patient
- PATIENT object (1)
  Patient
- Administration object (1)
  Administration
- STAFF object (1)
  Staff

4.4 CREATION OF TABLES

Administration Table

```
CREATE TABLE "hms1_administration" ("A_ID"
integer NOT NULL PRIMARY KEY AUTOINCREMENT,
"A_NAME" varchar(1000) NOT NULL, "GENDER"
varchar(100) NOT NULL)
```

Doctor Table

```
CREATE  TABLE  "hms1_doctor"  ("D_ID  "  integer
NOT NULL PRIMARY KEY AUTOINCREMENT,
 "D_NAME"  varchar(1000)  NOT  NULL,  "DEPT"
varchar(100) NOT NULL, "A_ID"
REFERENCES Administration(A_ID) On DELETE CASCADE)
```

Staff Table

```
CREATE TABLE "hms1_staff" ("S_ID " integer NOT
NULL PRIMARY KEY AUTOINCREMENT,
"S_NAME" varchar(1000) NOT NULL, "SALARY"
integer NOT NULL, "A_ID"
REFERENCES Administration(A_ID) On DELETE CASCADE)
```

PATIENT TABLE

```
CREATE TABLE "hms1_patient" ("P_ID " integer
NOT NULL PRIMARY KEY AUTOINCREMENT,
"P_NAME" varchar(1000) NOT NULL, "DISEASE"
varchar(100) NOT NULL, "FEE"
integer NOT NULL, "TOTALBILL" integer NOT
NULL, "D_ID" REFERENCES DOCTOR(D_ID) On DELETE CASCADE)
```

auth_user TABLE

```
CREATE TABLE "auth_user" ("id" integer NOT
NULL PRIMARY KEY AUTOINCREMENT, "password"
varchar(128) NOT NULL, "last_login" datetime
```

NULL, "is_superuser" bool NOT NULL, "username" varchar(150) NOT NULL UNIQUE, "last_name" varchar(150) NOT NULL, "email" varchar(254) NOT NULL, "is_staff" bool NOT NULL, "is_active" bool NOT NULL, "date_joined" datetime NOT NULL, "first_name" varchar(150) NOT NULL)

4.5 INSERTION OF TUPLES

Administration TABLE

```
INSERT INTO Administration VALUES(1,’Ajay’,’Male’);

INSERT INTO Administration VALUES(3,’Anjali’,’Female’);

INSERT INTO Administration VALUES(4,’Ajay Kumar’,’Male’);

INSERT INTO Administration VALUES(9,’Kumar’,’Male’);

INSERT INTO Administration VALUES(10,’Shraddha’,’Female’);
```

Doctor TABLE

```
INSERT INTO Doctor VALUES(12,’Govind’,’eye’,673);
```

INSERT INTO Doctor VALUES(122,’Cale’,’heart’,679);

INSERT INTO Doctor VALUES(10,’Stenz’,’eye’,671);

INSERT INTO Doctor VALUES(89,’Sawyer’,’ear’,679);

INSERT INTO Doctor VALUES(90,’Carol Finn’,’eye’,89);

Staff TABLE

INSERT INTO STAFF VALUES(189,’Rama’,10800,9);

INSERT INTO STAFF VALUES(145,’Krishna T’,10800,15);

INSERT INTO STAFF VALUES(167,’Rama A S’,10800,96);

INSERT INTO STAFF VALUES(1891,’Shobha’,10800,971);

INSERT INTO STAFF VALUES(122,’Ted’,10800,978);

PATIENT TABLE

INSERT INTO PATIENT VALUES(789,’Aditi’,’Lung
Cancer’,800,780,154);

```
INSERT INTO PATIENT VALUES(79,’Aditya G’,’Skin
Cancer’,1800,780,152);

INSERT INTO PATIENT VALUES(457,’Shekhar,’Heart
surgery’,11800,12780,154);

INSERT INTO PATIENT VALUES(767,’Vivek’,’Colour
blindness’,1800,1780,167);



INSERT INTO PATIENT
VALUES(709,’Golu’,’Leukemai’,11800,11780,194);
```

auth_user TABLE

```
python manage.py createsuperuser

Username (leave blank to use 'npsrinivas'): nirgund

Email address: nirgund.ps@gmail.com

Password:

Password (again):

This password is too short. It must contain at least 8 characters.

This password is too common.
```

```
Bypass password validation and create user anyway? [y/N]: y

Superuser created successfully.

python manage.py createsuperuser

Username (leave blank to use 'npsrinivas'): aravindan

Email address: aravindanspark@gmail.com

Password:

Password (again):

This password is too short. It must contain at least 8 characters.

This password is too common.
```

```
Bypass password validation and create user anyway? [y/N]: y

Superuser created successfully.

python manage.py createsuperuser

Username (leave blank to use 'npsrinivas'): aditya

Email address: adityasnirgund@gmail.com

Password:

Password (again):

This password is too short. It must contain at least 8 characters.
This password is too common.

Bypass password validation and create user anyway? [y/N]: y

Superuser created successfully.

python manage.py createsuperuser

Username (leave blank to use 'npsrinivas'): vanishreearun

Email address: vanishreearun@gmail.com

Password:

Password (again):
```

```
This password is too short. It must contain at least 8 characters.

This password is too common.

Bypass password validation and create user anyway? [y/N]: y

Superuser created successfully.

python manage.py createsuperuser

Username (leave blank to use 'npsrinivas'): ajay

Email address: ajay@gmail.com

Password:
```

```
Password (again):

This password is too short. It must contain at least 8 characters.

This password is too common.

Bypass password validation and create user anyway? [y/N]: y

Superuser created successfully.
```

4.6 CREATION OF TRIGGERS

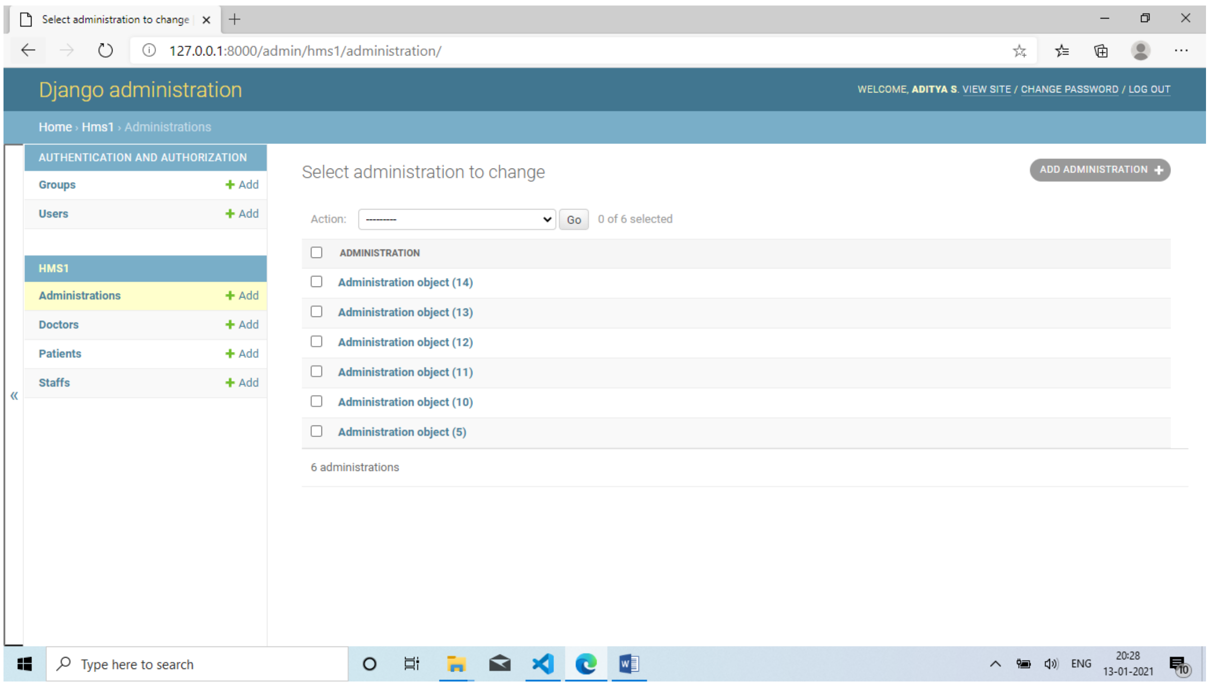
```
CREATE TRIGGER insertA_ID AFTER INSERT ON `Administration` FOR EACH
ROW
INSERT INTO logs VALUES (null, NEW.A_ID, "Inserted", NOW());
CREATE TRIGGER updateA_ID AFTER INSERT ON ` Administration ` FOR EACH
ROW
INSERT INTO logs VALUES (null, NEW. A_ID, "Updated", NOW());
CREATE TRIGGER deleteA_ID BEFORE INSERT ON ` Administration ` FOR EACH
ROW INSERT INTO logs VALUES (null, OLD. A_ID, "Deleted", NOW());
```

```
CREATE TRIGGER insertD_ID AFTER INSERT ON `Doctor` FOR EACH ROW
INSERT INTO logs VALUES (null, NEW.D_ID, "Inserted", NOW());
CREATE TRIGGER updateD_ID AFTER INSERT ON ` Doctor ` FOR EACH ROW
INSERT INTO logs VALUES (null, NEW. D_ID, "Updated", NOW());
CREATE TRIGGER deleteD_ID BEFORE INSERT ON `Doctor` FOR EACH
ROW INSERT INTO logs VALUES (null, OLD. D_ID, "Deleted", NOW());
```

```
CREATE TRIGGER insertS_ID AFTER INSERT ON `STAFF ` FOR EACH ROW
INSERT INTO logs VALUES (null, NEW.S_ID, "Inserted", NOW());
CREATE TRIGGER updateS_ID AFTER INSERT ON ` STAFF ` FOR EACH ROW
INSERT INTO logs VALUES (null, NEW. S_ID, "Updated", NOW());
CREATE TRIGGER deleteS_ID BEFORE INSERT ON ‘STAFF’ FOR EACH
ROW INSERT INTO logs VALUES (null, OLD. S_ID, "Deleted", NOW());
```

```
CREATE TRIGGER insertP_ID AFTER INSERT ON `PATIENT` FOR EACH ROW
INSERT INTO logs VALUES (null, NEW.P_ID, "Inserted", NOW());
CREATE TRIGGER updateP_ID AFTER INSERT ON ` PATIENT ` FOR EACH ROW
INSERT INTO logs VALUES (null, NEW. P_ID, "Updated", NOW());
CREATE TRIGGER deleteP_ID BEFORE INSERT ON ` PATIENT ` FOR EACH
ROW INSERT INTO logs VALUES (null, OLD. P_ID, "Deleted", NOW());
```

4.7 CREATION OF STORED PROCEDURES


Select administration to change

127.0.0.1:8000/admin/hms1/administration/

Django administration

WELCOME, ADITYA S. VIEW SITE / CHANGE PASSWORD / LOG OUT

Home › Hms1 › Administrations

AUTHENTICATION AND AUTHORIZATION	
Groups	+ Add
Users	+ Add

HMS1	
Administrations	+ Add
Doctors	+ Add
Patients	+ Add
Staffs	+ Add

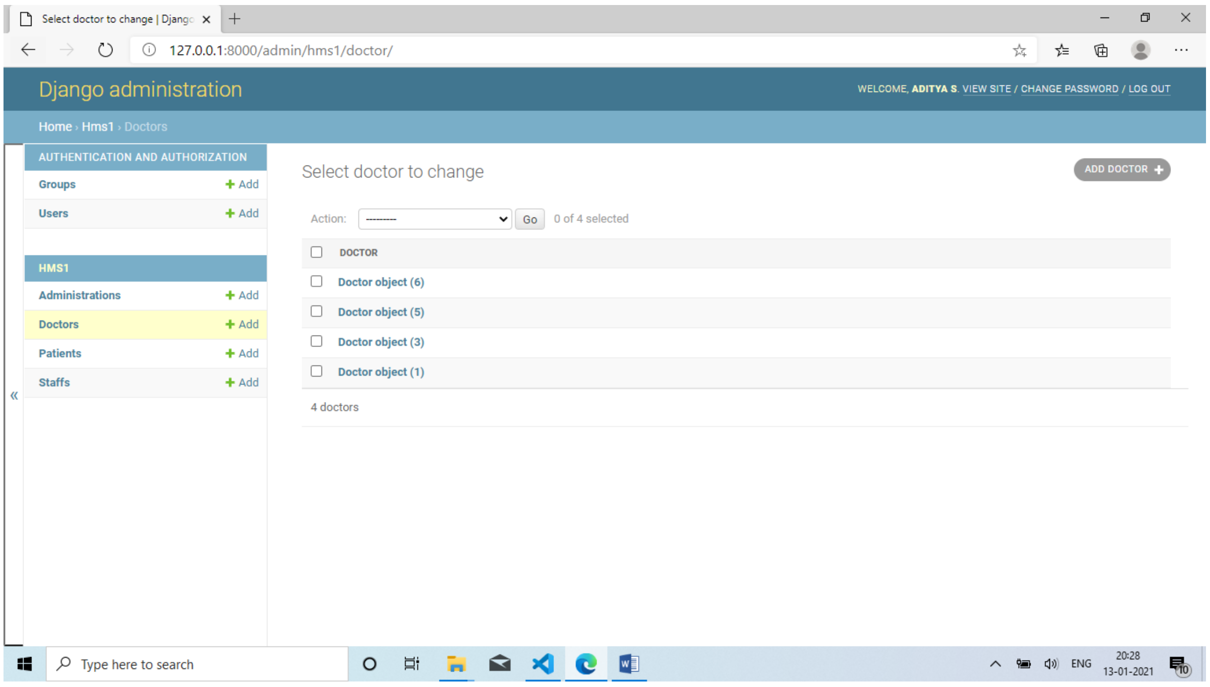
Select administration to change

ADD ADMINISTRATION +

Action: --------- Go 0 of 6 selected

ADMINISTRATION
Administration object (14)
Administration object (13)
Administration object (12)
Administration object (11)
Administration object (10)
Administration object (5)

6 administrations


Select doctor to change | Django

127.0.0.1:8000/admin/hms1/doctor/

Django administration

WELCOME, ADITYA S. VIEW SITE / CHANGE PASSWORD / LOG OUT

Home › Hms1 › Doctors

AUTHENTICATION AND AUTHORIZATION	
Groups	+ Add
Users	+ Add

HMS1	
Administrations	+ Add
Doctors	+ Add
Patients	+ Add
Staffs	+ Add

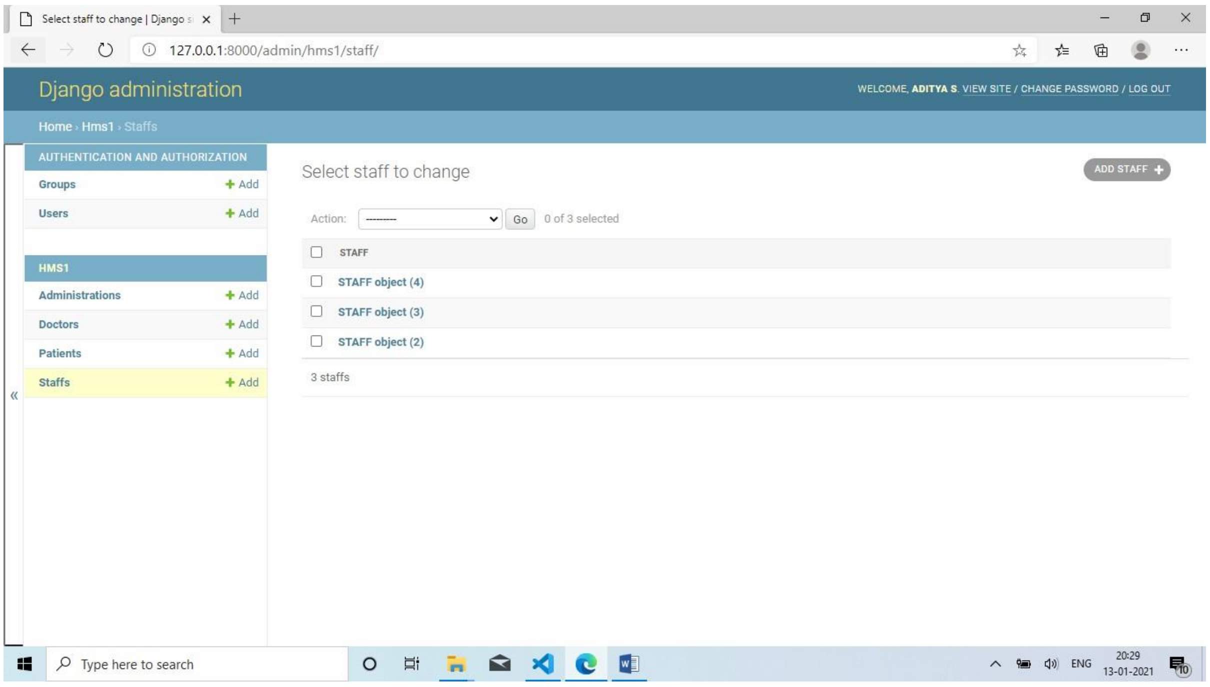
Select doctor to change

ADD DOCTOR +

Action: --------- Go 0 of 4 selected

DOCTOR
Doctor object (6)
Doctor object (5)
Doctor object (3)
Doctor object (1)

4 doctors



Select staff to change | Django

127.0.0.1:8000/admin/hms1/staff/

Django administration

WELCOME, ADITYA S. VIEW SITE / CHANGE PASSWORD / LOG OUT

Home › Hms1 › Staffs

AUTHENTICATION AND AUTHORIZATION
- Groups + Add
- Users + Add

HMS1
- Administrations + Add
- Doctors + Add
- Patients + Add
- Staffs + Add

Select staff to change

ADD STAFF +

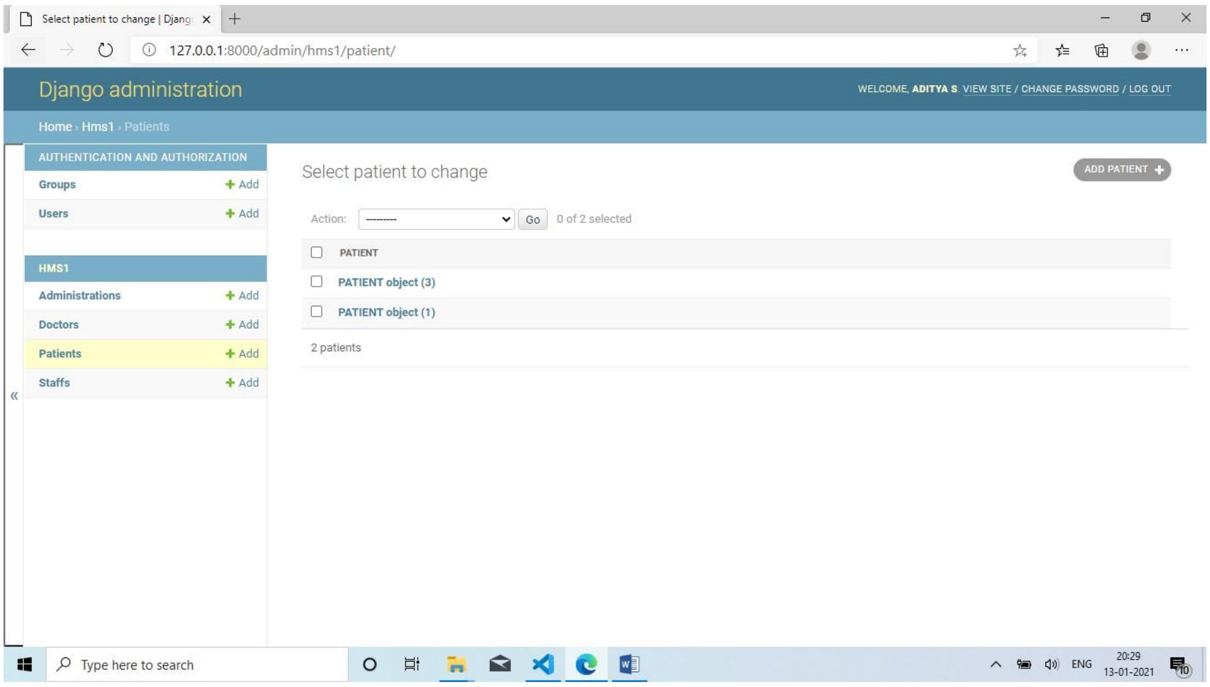
Action: --------- Go 0 of 3 selected

STAFF
STAFF object (4)
STAFF object (3)
STAFF object (2)

3 staffs

Type here to search

ENG 20:29 13-01-2021



Select patient to change | Django

127.0.0.1:8000/admin/hms1/patient/

Django administration

WELCOME, ADITYA S. VIEW SITE / CHANGE PASSWORD / LOG OUT

Home › Hms1 › Patients

AUTHENTICATION AND AUTHORIZATION
- Groups + Add
- Users + Add

HMS1
- Administrations + Add
- Doctors + Add
- Patients + Add
- Staffs + Add

Select patient to change

ADD PATIENT +

Action: --------- Go 0 of 2 selected

PATIENT
PATIENT object (3)
PATIENT object (1)

2 patients

Type here to search

ENG 20:29 13-01-2021

CHAPTER 5 FRONT END DESIGN

5.1 CONNECTIVITY TO DATABASE

Settings.py

```
"""
Django settings for dbms1 project.

Generated by 'django-admin startproject' using Django 3.1.4.

For more information on this file, see
https://docs.djangoproject.com/en/3.1/topics/settings/

For the full list of settings and their values, see
https://docs.djangoproject.com/en/3.1/ref/settings/
"""

from pathlib import Path

# Build paths inside the project like this: BASE_DIR / 'subdir'.
BASE_DIR = Path(__file__).resolve().parent.parent


# Quick-start development settings - unsuitable for production
# See https://docs.djangoproject.com/en/3.1/howto/deployment/checklist/

# SECURITY WARNING: keep the secret key used in production secret!
SECRET_KEY = 'p*4zewsa4rfh)koo=xe#1!=*0^nbvo+akr8lsa*!e)0*%q&)1y'

# SECURITY WARNING: don't run with debug turned on in production!
DEBUG = True

ALLOWED_HOSTS = []


# Application definition

INSTALLED_APPS = [
    'django.contrib.admin',
    'django.contrib.auth',
    'django.contrib.contenttypes',
    'django.contrib.sessions',
    'django.contrib.messages',
    'django.contrib.staticfiles',
    'hms1'
```

```
]

MIDDLEWARE = [
    'django.middleware.security.SecurityMiddleware',
    'django.contrib.sessions.middleware.SessionMiddleware',
    'django.middleware.common.CommonMiddleware',
    'django.middleware.csrf.CsrfViewMiddleware',
    'django.contrib.auth.middleware.AuthenticationMiddleware',
    'django.contrib.messages.middleware.MessageMiddleware',
    'django.middleware.clickjacking.XFrameOptionsMiddleware',
]

ROOT_URLCONF = 'dbms1.urls'

TEMPLATES = [
    {
        'BACKEND': 'django.template.backends.django.DjangoTemplates',
        'DIRS': [],
        'APP_DIRS': True,
        'OPTIONS': {
            'context_processors': [
                'django.template.context_processors.debug',
                'django.template.context_processors.request',
                'django.contrib.auth.context_processors.auth',
                'django.contrib.messages.context_processors.messages',
            ],
        },
    },
]

WSGI_APPLICATION = 'dbms1.wsgi.application'


# Database
# https://docs.djangoproject.com/en/3.1/ref/settings/#databases

DATABASES = {
    'default': {
        'ENGINE': 'django.db.backends.sqlite3',
        'NAME': BASE_DIR / 'db.sqlite3',
    }
}


# Password validation
# https://docs.djangoproject.com/en/3.1/ref/settings/#auth-password-validators

AUTH_PASSWORD_VALIDATORS = [
    {
        'NAME': 'django.contrib.auth.password_validation.UserAttributeSimilarityValidator',
    },
    {
        'NAME': 'django.contrib.auth.password_validation.MinimumLengthValidator',
    },
    {
        'NAME': 'django.contrib.auth.password_validation.CommonPasswordValidator',
    },
    {
        'NAME': 'django.contrib.auth.password_validation.NumericPasswordValidator',
    },
]


# Internationalization
# https://docs.djangoproject.com/en/3.1/topics/i18n/

LANGUAGE_CODE = 'en-us'

TIME_ZONE = 'UTC'

USE_I18N = True

USE_L10N = True

USE_TZ = True


# Static files (CSS, JavaScript, Images)
# https://docs.djangoproject.com/en/3.1/howto/static-files/

STATIC_URL = '/static/'
```

5.1 FRONT END CODE

Systems design is the process of defining the architecture, components, modules, interfaces, and data for a system to satisfy specified requirements. Systems design could see it as the application of systems theory to product development. There is some overlap with the disciplines of systems analysis, systems architecture and systems engineering. If the broader topic of product development "blends the perspective of marketing, design, and manufacturing into a single approach to product development," then design is the act of taking the marketing information and creating the design of the product to be manufactured. Systems design is therefore the process of defining and developing systems to satisfy specified requirements of the user.

Until the 1990s systems design had a crucial and respected role in the data processing industry. In the 1990s standardization of hardware and software resulted in the ability to build modular systems. The increasing importance of software running on generic platforms has enhanced the discipline of software engineering.

Object-oriented analysis and design methods are becoming the most widely used methods for computer systems design.[citation needed] The UML has become the standard language in object-oriented analysis and design.[citation needed] It is widely used for modeling software systems and is increasingly used for high designing non-software systems and organizations.[citation needed]

System design is one of the most important phases of software development process. The purpose of the design is to plan the solution of a problem specified by the requirement documentation. In other words the first step in the solution to the problem is the design of the project.

asgi.py

```python
"""
ASGI config for dbms1 project.

It exposes the ASGI callable as a module-level variable named ``application``.

For more information on this file, see
https://docs.djangoproject.com/en/3.1/howto/deployment/asgi/
"""

import os

from django.core.asgi import get_asgi_application

os.environ.setdefault('DJANGO_SETTINGS_MODULE', 'dbms1.settings')

application = get_asgi_application()
```

urls.py

```python
"""dbms1 URL Configuration

The `urlpatterns` list routes URLs to views. For more information please see:
    https://docs.djangoproject.com/en/3.1/topics/http/urls/
Examples:
Function views
    1. Add an import:  from my_app import views
    2. Add a URL to urlpatterns:  path('', views.home, name='home')
Class-based views
    1. Add an import:  from other_app.views import Home
    2. Add a URL to urlpatterns:  path('', Home.as_view(), name='home')
Including another URLconf
    1. Import the include() function: from django.urls import include, path
    2. Add a URL to urlpatterns:  path('blog/', include('blog.urls'))
"""
from django.contrib import admin
from django.urls import path
from hms1.views import edit_doctor, home, create_admin, create_doctor, create_staff, create
_patient, Administration_post, Doctor_post, Staff_post, Patient_post, all_posts_administrat
ion, all_posts_doctor, all_posts_staff, all_posts_patient, delete_post_admin, delete_post_d
octor, delete_post_staff, delete_post_patient, edit_admin, edit_staff, edit_patient
```

```
urlpatterns = [
    path('admin/', admin.site.urls),
    path('',home),
    path('Administ/',create_admin),
    path('Doct/',create_doctor),
    path('STA/',create_staff),
    path('PA/',create_patient),
    path('post1/<int:post_id>/',Administration_post),
    path('post2/<int:post_id>/',Doctor_post),
    path('post3/<int:post_id>/',Staff_post),
    path('post4/<int:post_id>/',Patient_post),
    path('allposts1/',all_posts_administration),
    path('allposts2/',all_posts_doctor),
    path('allposts3/',all_posts_staff),
    path('allposts4/',all_posts_patient),
    path('delete1/<int:post_id>/',delete_post_admin),
    path('delete2/<int:post_id>/',delete_post_doctor),
    path('delete3/<int:post_id>/',delete_post_staff),
    path('delete4/<int:post_id>/',delete_post_patient),
    path('edit1/<int:post_id>/',edit_admin),
    path('edit2/<int:post_id>/',edit_doctor),
    path('edit3/<int:post_id>/',edit_staff),
    path('edit4/<int:post_id>/',edit_patient)
]
```

wsgi.py

```
"""
WSGI config for dbms1 project.

It exposes the WSGI callable as a module-level variable named ``application``.

For more information on this file, see
https://docs.djangoproject.com/en/3.1/howto/deployment/wsgi/
"""

import os

from django.core.wsgi import get_wsgi_application

os.environ.setdefault('DJANGO_SETTINGS_MODULE', 'dbms1.settings')

application = get_wsgi_application()
```

001_initial.py

```
# Generated by Django 3.1.4 on 2021-01-29 11:45

from django.db import migrations, models
import django.db.models.deletion


class Migration(migrations.Migration):

    initial = True

    dependencies = [
    ]

    operations = [
        migrations.CreateModel(
            name='Administration',
            fields=[
                ('A_ID', models.IntegerField(primary_key=True, serialize=False)),
                ('A_NAME', models.CharField(max_length=1000)),
                ('GENDER', models.CharField(max_length=100)),
            ],
        ),
        migrations.CreateModel(
            name='Doctor',
            fields=[
                ('D_ID', models.IntegerField(primary_key=True, serialize=False)),
                ('D_NAME', models.CharField(max_length=1000)),
                ('DEPT', models.CharField(max_length=100)),
                ('A_ID', models.ForeignKey(on_delete=django.db.models.deletion.CASCADE, to=
'hms2.administration')),
            ],
        ),
        migrations.CreateModel(
            name='STAFF',
            fields=[
                ('S_ID', models.IntegerField(primary_key=True, serialize=False)),
                ('S_NAME', models.CharField(max_length=1000)),
                ('SALARY', models.IntegerField()),
                ('A_ID', models.ForeignKey(on_delete=django.db.models.deletion.CASCADE, to=
'hms2.administration')),
            ],
        ),
        migrations.CreateModel(
            name='PATIENT',
            fields=[
                ('P_ID', models.IntegerField(primary_key=True, serialize=False)),
                ('P_NAME', models.CharField(max_length=1000)),
                ('DISEASE', models.CharField(max_length=100)),
                ('FEE', models.IntegerField()),
                ('TOTALBILL', models.IntegerField()),
                ('D_ID', models.ForeignKey(on_delete=django.db.models.deletion.CASCADE, to=
'hms2.doctor')),
            ],
        ),
    ]
```

All_posts_administration.html

```
<html>
    <head>
        <title>All Posts</title>
        <meta charset="utf-8">
        <meta http-equiv="X-UA-Compatible" content="IE=edge">
        <title>Welcome to Hospital Management System</title>
        <meta name="description" content="An interactive getting Hospital Management System
.">
        <link rel="stylesheet" href="/static/main.css">
    </head>
    <body>
        <div class="container">
            {% for post in posts %}
            <a href="/post1/{{post.id}}/">
                <h1>{{post.A_ID}}</h1>
            </a>
            <p>{{post.A_ID}}</p>
            <p>{{post.A_NAME}}</p>
            <p>{{post.GENDER}}</p>
            <hr> {% endfor %}
        </div>
    </body>
</html>
```

All_posts_doctor.html

```
<html>
    <head>
        <title>All Posts</title>
        <meta charset="utf-8">
        <meta http-equiv="X-UA-Compatible" content="IE=edge">
        <title>Welcome to Hospital Management System</title>
        <meta name="description" content="An interactive getting Hospital Management System
.">
        <link rel="stylesheet" href="/static/main.css">
    </head>
    <body>
        <div class="container">
            {% for post in posts %}
            <a href="/post2/{{post.id}}/">
                <h1>{{post.D_ID}}</h1>
            </a>
            <p>{{post.D_ID}}</p>
            <p>{{post.D_NAME}}</p>
            <p>{{post.DEPT}}</p>
            <p>{{post.A_ID_id}}</p>
            <hr> {% endfor %}
        </div>
    </body>
</html>
```

All_posts_Staff.html

```
<html>
    <head>
        <title>All Posts</title>
        <meta charset="utf-8">
        <meta http-equiv="X-UA-Compatible" content="IE=edge">
        <title>Welcome to Hospital Management System</title>
        <meta name="description" content="An interactive getting Hospital Management System
.">
        <link rel="stylesheet" href="/static/main.css">
    </head>
    <body>
```

```
        <div class="container">
            {% for post in posts %}
            <a href="/post3/{{post.id}}/">
                <h1>{{post.S_ID}}</h1>
            </a>
            <p>{{post.S_ID}}</p>
            <p>{{post.S_NAME}}</p>
            <p>{{post.SALARY}}</p>
            <p>{{post.A_ID_id}}</p>
            <hr> {% endfor %}
        </div>
    </body>
</html>
```

All_posts_Patient.html

```
<html>
    <head>
        <title>All Posts</title>
        <meta charset="utf-8">
        <meta http-equiv="X-UA-Compatible" content="IE=edge">
        <title>Welcome to Hospital Management System</title>
        <meta name="description" content="An interactive getting Hospital Management System
.">
        <link rel="stylesheet" href="/static/main.css">
    </head>
    <body>
        <div class="container">
            {% for post in posts %}
            <a href="/post4/{{post.id}}/">
                <h1>{{post.P_ID}}</h1>
            </a>
            <p>{{post.P_ID}}</p>
            <p>{{post.P_NAME}}</p>
            <p>{{post.DISEASE}}</p>
            <p>{{post.FEE}}</p>
            <p>{{post.TOTALBILL}}</p>
            <p>{{post.D_ID_id}}</p>
            <hr> {% endfor %}
        </div>
    </body>
</html>
```

Post_Admin.html

```
<html>
<head>
<title>{{selected_post.A_ID}}</title>
<link href="https://cdn.jsdelivr.net/npm/bootstrap@5.0.0-
beta1/dist/css/bootstrap.min.css" rel="stylesheet" integrity="sha384-
giJF6kkoqNQ00vy+HMDP7azOuL0xtbfIcaT9wjKHr8RbDVddVHyTfAAsrekwKmP1" crossorigin="anonymous">


</head>
<body>
    <a href="/allposts1/">Back</a>
    <br>
    <a href="/delete1/{{selected_post.id}}">Delete</a>
    <br>
    <a href="/edit1/{{selected_post.id}}">Edit</a>
    <br>
    <h1>{{selected_post.A_ID}}</h1>

    <p>Published {{selected_post.created}}</p>
    <hr>
    <p>{{selected_post.A_ID_id}}</p>
    <p>{{selected_post.A_NAME}}</p>
    <p>{{selected_post.GENDER}}</p>

</body>



</html>
```

Post_Doctor.html

```
<html>
<head>
<title>{{selected_post.D_ID}}</title>
<link href="https://cdn.jsdelivr.net/npm/bootstrap@5.0.0-
beta1/dist/css/bootstrap.min.css" rel="stylesheet" integrity="sha384-
giJF6kkoqNQ00vy+HMDP7azOuL0xtbfIcaT9wjKHr8RbDVddVHyTfAAsrekwKmP1" crossorigin="anonymous">
```

```
</head>
<body>
    <a href="/allposts2/">Back</a>
    <br>
    <a href="/delete2/{{selected_post.id}}">Delete</a>
    <br>
    <a href="/edit2/{{selected_post.id}}">Edit</a>
    <br>
    <h1>{{selected_post.D_ID}}</h1>
    <p>Published {{selected_post.created}}</p>
    <hr>
    <p>{{selected_post.D_ID}}</p>
    <p>{{selected_post.D_NAME}}</p>
    <p>{{selected_post.DEPT}}</p>
    <p>{{selected_post.A_ID_id}}</p>


</body>



</html>
```

Post_Staff.html

```
<html>
<head>
<title>{{selected_post.S_ID}}</title>
<link href="https://cdn.jsdelivr.net/npm/bootstrap@5.0.0-
beta1/dist/css/bootstrap.min.css" rel="stylesheet" integrity="sha384-
giJF6kkoqNQ00vy+HMDP7azOuL0xtbfIcaT9wjKHr8RbDVddVHyTfAAsrekwKmP1" crossorigin="anonymous">

</head>
<body>
    <a href="/allposts3/">Back</a>
    <br>
    <a href="/delete3/{{selected_post.id}}">Delete</a>
    <br>
    <a href="/edit3/{{selected_post.id}}">Edit</a>
    <br>
    <h1>{{selected_post.S_ID}}</h1>
    <p>Published {{selected_post.created}}</p>
    <hr>
    <p>{{selected_post.S_ID}}</p>
    <p>{{selected_post.S_NAME}}</p>
    <p>{{selected_post.SALARY}}</p>
    <p>{{selected_post.A_ID_id}}</p>
```

```
</body>



</html>
```

Post_Patient.html

```
<html>

    <head>

    <title>Create new Patient</title>
    <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.0.0-
beta1/dist/css/bootstrap.min.css" rel="stylesheet" integrity="sha384-
giJF6kkoqNQ00vy+HMDP7azOuL0xtbfIcaT9wjKHr8RbDVddVHyTfAAsrekwKmP1" crossorigin="anonymous">
    </head>
    <body>
        <br><br>
        <div class="container">

            <h3>Create new Patient</h3>
            <hr>
            <img src="/static/dd.jpeg">
            <form action="/PA/"  method="POST">
            {% csrf_token %}
            <p>P_ID</p>
            <input type="text" class="form-control" name="P_ID">
            <br>
            <p>P_NAME</p>
            <input type="text" class="form-control" name="P_NAME">
            <br>
            <p>DISEASE</p>
            <input type="text" class="form-control" name="DISEASE">
            <br>
            <p>FEE</p>
            <input type="text" class="form-control" name="FEE">
            <br>
            <p>TOTALBILL</p>
            <input type="text" class="form-control" name="TOTALBILL">
            <br>
            <p>D_ID</p>
            <input type="text" class="form-control" name="D_ID_id">
            <br>
            <button type="submit" class="btn btn-success">Submit</button>
            </form>
        </div>
    </body>


</html>
```

edit_administration.html

```
<html>

    <head>

    <title>Create  Update Administration</title>

    <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.0.0-
beta1/dist/css/bootstrap.min.css" rel="stylesheet" integrity="sha384-
giJF6kkoqNQ00vy+HMDP7azOuL0xtbfIcaT9wjKHr8RbDVddVHyTfAAsrekwKmP1" crossorigin="anonymous">

    </head>
    <body>
        <br><br>
        <div class="container">

            <h3>Create Update Administration</h3>
            <img src="/static/OIP.jpeg">
            <hr>
            <form action="/edit1/{{post.id}}/" method="POST">
            {% csrf_token %}
            <p>A_ID</p>
            <input type="text" class="form-control" name="A_ID" value="{{post.A_ID}}">
            <br>
            <p>A_NAME</p>
            <input type="text" class="form-control" name="A_NAME" value="{{post.A_NAME}}">
            <br>
            <p>GENDER</p>
            <input type="text" class="form-control" name="GENDER" value="{{post.GENDER}}">
            <br>
            <button type="submit" class="btn btn-success">Update</button>
            </form>
        </div>
    </body>


</html>
```

edit_Doctor.html

```
<html>

    <head>

    <title> Update Doctor</title>

    <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.0.0-
beta1/dist/css/bootstrap.min.css" rel="stylesheet" integrity="sha384-
giJF6kkoqNQ00vy+HMDP7azOuL0xtbfIcaT9wjKHr8RbDVddVHyTfAAsrekwKmP1" crossorigin="anonymous">

    </head>
    <body>
        <br><br>
        <div class="container">

            <h3>Update Doctor</h3>
            <img src="/static/doctor-supply.jpeg">
            <hr>
            <form action="/edit2/{{post.id}}/" method="POST">
            {% csrf_token %}
            <p>D_ID</p>
            <input type="text" class="form-control" name="D_ID" value="{{post.D_ID}}">
            <br>
            <p>D_NAME</p>
            <input type="text" class="form-control" name="D_NAME" value="{{post.D_NAME}}">
            <br>
            <p>DEPT</p>
            <input type="text" class="form-control" name="DEPT" value="{{post.DEPT}}">
            <br>
            <p>A_ID</p>
            <input type="text" class="form-
control" name="A_ID_id" value="{{post.A_ID_id}}">
            <br>
            <button type="submit" class="btn btn-success">Update</button>
            </form>
        </div>
    </body>


</html>
```

edit_Staff.html

```
<html>

    <head>

    <title> Update Staff</title>

    <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.0.0-
beta1/dist/css/bootstrap.min.css" rel="stylesheet" integrity="sha384-
giJF6kkoqNQ00vy+HMDP7azOuL0xtbfIcaT9wjKHr8RbDVddVHyTfAAsrekwKmP1" crossorigin="anonymous">

    </head>
    <body>
        <br><br>
        <div class="container">

            <h3>Update Staff</h3>
            <img src="/static/20200630_HT_Web_MonITor_Tech-Organizations-Should-
Consider.jpg">
            <hr>
            <form action="/edit3/{{post.id}}/" method="POST">
            {% csrf_token %}
            <p>S_ID</p>
            <input type="text" class="form-control" name="S_ID" value="{{post.S_ID}}">
            <br>
            <p>S_NAME</p>
            <input type="text" class="form-control" name="S_NAME" value="{{post.S_NAME}}">
            <br>
            <p>SALARY</p>
            <input type="text" class="form-control" name="SALARY" value="{{post.SALARY}}">
            <br>
            <p>A_ID</p>
            <input type="text" class="form-
control" name="A_ID_id" value="{{post.A_ID_id}}">
            <br>
            <button type="submit" class="btn btn-success">Update</button>
            </form>
        </div>
    </body>


</html>
```

edit_Patient.html

```
<html>

    <head>

    <title> Update Doctor</title>

    <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.0.0-
beta1/dist/css/bootstrap.min.css" rel="stylesheet" integrity="sha384-
giJF6kkoqNQ00vy+HMDP7azOuL0xtbfIcaT9wjKHr8RbDVddVHyTfAAsrekwKmP1" crossorigin="anonymous">

    </head>
    <body>
        <br><br>
        <div class="container">

            <h3>Update Doctor</h3>
            <img src="/static/doctor-supply.jpeg">
            <hr>
            <form action="/edit2/{{post.id}}/" method="POST">
            {% csrf_token %}
            <p>D_ID</p>
            <input type="text" class="form-control" name="D_ID" value="{{post.D_ID}}">
            <br>
            <p>D_NAME</p>
            <input type="text" class="form-control" name="D_NAME" value="{{post.D_NAME}}">
            <br>
            <p>DEPT</p>
            <input type="text" class="form-control" name="DEPT" value="{{post.DEPT}}">
            <br>
            <p>A_ID</p>
            <input type="text" class="form-
control" name="A_ID_id" value="{{post.A_ID_id}}">
            <br>
            <button type="submit" class="btn btn-success">Update</button>
            </form>
        </div>
    </body>


</html>
```

Create_admin.html

```
<html>

    <head>

    <title>Create new Administration</title>

    <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.0.0-
beta1/dist/css/bootstrap.min.css" rel="stylesheet" integrity="sha384-
giJF6kkoqNQ00vy+HMDP7azOuL0xtbfIcaT9wjKHr8RbDVddVHyTfAAsrekwKmP1" crossorigin="anonymous">

    </head>
    <body>
        <br><br>
        <div class="container">

            <h3>Create new Administration</h3>
            <img src="/static/OIP.jpeg">
            <hr>
            <form action="/Administ/" method="POST">
            {% csrf_token %}
            <p>A_ID</p>
            <input type="text" class="form-control" name="A_ID">
            <br>
            <p>A_NAME</p>
            <input type="text" class="form-control" name="A_NAME">
            <br>
            <p>GENDER</p>
            <input type="text" class="form-control" name="GENDER">
            <br>
            <button type="submit" class="btn btn-success">Submit</button>
            </form>
        </div>
    </body>


</html>
```

Create_Doctor.html

```
<html>

    <head>

    <title>Create new Doctor</title>
    <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.0.0-
beta1/dist/css/bootstrap.min.css" rel="stylesheet" integrity="sha384-
giJF6kkoqNQ00vy+HMDP7azOuL0xtbfIcaT9wjKHr8RbDVddVHyTfAAsrekwKmP1" crossorigin="anonymous">
    </head>
    <body>
        <br><br>
        <div class="container">

            <h3>Create new Doctor</h3>
            <hr>
            <img src="/static/doctor-supply.jpeg">
            <form action="/Doct/" method="POST">
            {% csrf_token %}
            <p>D_ID</p>
            <input type="text" class="form-control" name="D_ID">
            <br>
            <p>D_NAME</p>
            <input type="text" class="form-control" name="D_NAME">
            <br>
            <p>DEPT</p>
            <input type="text" class="form-control" name="DEPT">
            <br>
            <p>A_ID</p>
            <input type="text" class="form-control" name="A_ID_id">
            <br>
            <button type="submit" class="btn btn-success">Submit</button>
            </form>
        </div>
    </body>


</html>
```

Create_Staff.html

```
<html>

    <head>

    <title>Create new Staff</title>
    <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.0.0-
beta1/dist/css/bootstrap.min.css" rel="stylesheet" integrity="sha384-
giJF6kkoqNQ00vy+HMDP7azOuL0xtbfIcaT9wjKHr8RbDVddVHyTfAAsrekwKmP1" crossorigin="anonymous">
    </head>
    <body>
        <br><br>
        <div class="container">

            <h3>Create new Staff</h3>
            <hr>
            <img src="/static/20200630_HT_Web_MonITor_Tech-Organizations-Should-
Consider.jpg">
            <form action="/STA/" method="POST">
            {% csrf_token %}
            <p>S_ID</p>
            <input type="text" class="form-control" name="S_ID">
            <br>
            <p>S_NAME</p>
            <input type="text" class="form-control" name="S_NAME">
            <br>
            <p>SALARY</p>
            <input type="text" class="form-control" name="SALARY">
            <br>
            <p>A_ID</p>
            <input type="text" class="form-control" name="A_ID_id">
            <br>
            <button type="submit" class="btn btn-success">Submit</button>
            </form>
        </div>
    </body>


</html>
```

Create_Patient.html

```
<html>

    <head>

    <title>Create new Patient</title>
    <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.0.0-
beta1/dist/css/bootstrap.min.css" rel="stylesheet" integrity="sha384-
giJF6kkoqNQ00vy+HMDP7azOuL0xtbfIcaT9wjKHr8RbDVddVHyTfAAsrekwKmP1" crossorigin="anonymous">
    </head>
    <body>
        <br><br>
        <div class="container">

            <h3>Create new Patient</h3>
            <hr>
            <img src="/static/dd.jpeg">
            <form action="/PA/"  method="POST">
            {% csrf_token %}
            <p>P_ID</p>
            <input type="text" class="form-control" name="P_ID">
            <br>
            <p>P_NAME</p>
            <input type="text" class="form-control" name="P_NAME">
            <br>
            <p>DISEASE</p>
            <input type="text" class="form-control" name="DISEASE">
            <br>
            <p>FEE</p>
            <input type="text" class="form-control" name="FEE">
            <br>
            <p>TOTALBILL</p>
            <input type="text" class="form-control" name="TOTALBILL">
            <br>
            <p>D_ID</p>
            <input type="text" class="form-control" name="D_ID_id">
            <br>
            <button type="submit" class="btn btn-success">Submit</button>
            </form>
        </div>
    </body>


</html>
```

Index.html

```
<!DOCTYPE html>
<html>

    <head>
        <meta charset="utf-8">
        <meta http-equiv="X-UA-Compatible" content="IE=edge">
        <title>Welcome to Hospital Management System</title>
        <meta name="description" content="An interactive getting Hospital Management System
.">
        <link rel="stylesheet" href="/static/main.css">

    </head>
    <body>

        <h1> Hospital Management System</h1>
        <h2>This is your guide!</h2>

        <!--
            MADE WITH <3 AND JAVASCRIPT
        -->

        <p>
        A hospital information system (HIS) is an element of health informatics that focuse
s mainly on the administrational needs of hospitals. In many implementations, a HIS is a co
mprehensive, integrated information system designed to manage all the aspects of a hospital
's operation, such as medical, administrative, financial, and legal issues and the correspo
nding processing of services. Hospital information system is also known as hospital managem
ent software (HMS) or hospital management system.

        Hospital information systems provide a common source of information about a patient
's health history. The system has to keep data in a secure place and controls who can reach
 the data in certain circumstances. These systems enhance the ability of health care profes
sionals to coordinate care by providing a patient's health information and visit history at
 the place and time that it is needed. Patient's laboratory test information also includes
visual results such as X-
ray, which may be reachable by professionals. HIS provide internal and external communicati
on among health care providers. Portable devices such as smartphones and tablet computers m
ay be used at the bedside.

        Hospital information systems are often composed of one or several software componen
ts with specialty-specific extensions, as well as of a large variety of sub-
systems in medical specialties from a multi-
vendor market. Specialized implementations name for example laboratory information system (
LIS), Policy and Procedure Management System, radiology information system (RIS) or picture
 archiving and communication system (PACS).

        Potential benefits of hospital information systems include:

        Efficient and accurate administration of finance, diet of patient, engineering, and
 distribution of medical aid. It helps to view a broad picture of hospital growth
        Improved monitoring of drug usage, and study of effectiveness. This leads to the re
duction of adverse drug interactions while promoting more appropriate pharmaceutical utiliz
ation.
        Enhances information integrity, reduces transcription errors, and reduces duplicati
on of information entries.
        Hospital software is easy to use and eliminates error caused by handwriting. New te
chnology computer systems give perfect performance to pull up information from server or cl
oud servers.
        </p>

        <img src="/static/800px-NHS_NNUH_entrance.jpeg">
        <!--
            LET US KNOW WHAT YOU THINK
        -->
        <h2>Get involved</h2>
        <p>
        A hospital is a health care institution providing patient treatment with specialize
d medical and nursing staff and medical equipment. The best-
known type of hospital is the general hospital, which typically has an emergency department
 to treat urgent health problems ranging from fire and accident victims to a sudden illness
.
        As long as each stage implementation needs to be accurate and explicit, the clinic
management system provides certain automation of many vital daily processes. The hospital s
ystem software covers the services that unify and simplify the work of healthcare professio
nals as well as their interactions with patients.

        There is always the wide choice of features that can be included in the system. Mor
eover, the most important thing they are created to streamline various procedures that meet
 the needs of all the users. The hospital management system feature list is concentrated on
 providing the smooth experience of patients, staff and hospital authorities. It might seem
 that their expectations differ, they still are covered by components of the hospital infor
mation system. Quality and security still remain the main criteria of the medical industry.
 It is also known for the constant and rapid changes to improve the efficiency of medical s
ervices and satisfaction of the patients.

        Hospital management has greatly changed over the last decades. Business expertise,
modern technologies, connected devices, mobile apps, and knowledge of healthcare are key el
ements for the implementation of hospital management system project. The number of healthca
re providers has increased and the patients have a wide choice of medical specialists. The
interactions between the hospital and the patient can be simplified for the convenience of
both sides. Each institution has the opportunity to create the efficient, clear and fast de
livering healthcare model.
```

```
        </p>
       <ul>

            <li><a href="http://127.0.0.1:8000/Administ/">Administration.io</a></li>
            <li><a href="http://127.0.0.1:8000/Doct/">Doctor.io</a></li>
            <li><a href="http://127.0.0.1:8000/STA/">STAFF.io</a></li>
            <li><a href="http://127.0.0.1:8000/PA/">PATIENT.io</a></li>


            <li><a href="https://en.wikipedia.org/wiki/Hospital">Hospital Management System
</a></li>



        </ul>

        <h3>Hospital Definition</h3>
        <p>
        A hospital contains one or more wards that house hospital beds for inpatients. It m
ay also have acute services such as an emergency department, operating theatre, and intensi
ve care unit, as well as a range of medical specialty departments. A well-
equipped hospital may be classified as a trauma center. They may also have other services s
uch a hospital pharmacy, radiology, pathology and medical laboratories. Some hospitals have
 outpatient departments such as behavioral health services, dentistry, and rehabilitation s
ervices.

        A hospital may also have a department of nursing, headed by a chief nursing officer
 or director of nursing. This department is responsible for the administration of professio
nal nursing practice, research, and policy for the hospital.

        Many units have both a nursing and a medical director that serve as administrators
for their respective disciplines within that unit. For example, within an intensive care nu
rsery, a medical director is responsible for physicians and medical care, while the nursing
 manager is responsible for all of the nurses and nursing care.

        Support units may include a medical records department, release of information depa
rtment, technical support, clinical engineering, facilities management, plant operations, d
ining services, and security departments.

        The COVID-
19 pandemic stimulated the development of virtual wards across the British NHS. Patients ar
e managed at home, monitoring their own oxygen levels using an oxygen saturation probe if n
ecessary and supported by telephone. West Hertfordshire Hospitals NHS Trust managed around
1200 patients at home between March and June 2020 and planned to continue the system after
covid-19, initially for respiratory patients.
        </p>

    </body>
</html>
```

Admin.py

```
from django.contrib import admin
from hms1.models import Administration
from hms1.models import Doctor
from hms1.models import STAFF
from hms1.models import PATIENT



# Register your models here.

admin.site.register(Administration)
admin.site.register(Doctor)
admin.site.register(STAFF)
admin.site.register(PATIENT)
```

Apps.py

```
from django.apps import AppConfig


class HmsConfig(AppConfig):
    name = 'hms1'
```

Models.py

```
from django.db import models

# Create your models here.
from django.db import models

# Create your models here.
class Administration(models.Model):
    A_ID=models.IntegerField(primary_key=True)
    A_NAME=models.CharField(max_length=1000)
    GENDER=models.CharField(max_length=100)

    def _str(self):
        return self.A_ID
```

```
class Doctor(models.Model):
    D_ID=models.IntegerField(primary_key=True)
    D_NAME=models.CharField(max_length=1000)
    DEPT=models.CharField(max_length=100)
    A_ID=models.ForeignKey(Administration, on_delete=models.CASCADE)

    def _str(self):
        return self.D_ID


class STAFF(models.Model):
    S_ID=models.IntegerField(primary_key=True)
    S_NAME=models.CharField(max_length=1000)
    SALARY=models.IntegerField()
    A_ID=models.ForeignKey(Administration, on_delete=models.CASCADE)

    def _str(self):
        return self.S_ID


class PATIENT(models.Model):
    P_ID=models.IntegerField(primary_key=True)
    P_NAME=models.CharField(max_length=1000)
    DISEASE=models.CharField(max_length=100)
    FEE=models.IntegerField()
    TOTALBILL=models.IntegerField()
    D_ID=models.ForeignKey(Doctor, on_delete=models.CASCADE)
    def _str(self):
        return self.P_ID
```

tests.py

```
from django.test import TestCase

# Create your tests here.
```

views.py

```
from django.shortcuts import render, redirect
from django.http import HttpResponse
from hms1.models import Administration
from hms1.models import Doctor
from hms1.models import STAFF
from hms1.models import PATIENT


# Create your views here.
def home(request):
    return render(request,"index1.html")
# Create your views here.



def Administration_post(request, post_id):
    print(post_id)
    post1 = Administration.objects.get(pk=post_id)
    return render(request,"create_admin_post.html",{"selected_post": post1})

def Doctor_post(request, post_id):
    print(post_id)
    post2 = Doctor.objects.get(pk=post_id)
    return render(request,"creaate_doctor_post.html",{"selected_post": post2})

def Staff_post(request, post_id):
    print(post_id)
    post3 = STAFF.objects.get(pk=post_id)
    return render(request,"create_staff_post.html",{"selected_post": post3})

def Patient_post(request, post_id):
    print(post_id)
    post4 = PATIENT.objects.get(pk=post_id)
    return render(request,"create_patient_post.html",{"selected_post": post4})




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

        A_ID = request.POST["A_ID"]
        A_NAME = request.POST["A_NAME"]
        GENDER = request.POST["GENDER"]

        new_post1 = Administration.objects.create(A_ID=A_ID, A_NAME=A_NAME, GENDER=GENDER)

        return redirect("/allposts1/")
    return render(request,"create_admin.html")

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

        D_ID = request.POST["D_ID"]
        D_NAME = request.POST["D_NAME"]
        DEPT = request.POST["DEPT"]
        A_ID_id = request.POST["A_ID_id  "]

        new_post2 = Doctor.objects.create(D_ID= D_ID, D_NAME=D_NAME, DEPT=DEPT, A_ID_id  =A
_ID_id )

        return redirect("/allposts2/")
    return render(request,"create_doctor.html")

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

        S_ID = request.POST["S_ID"]
        S_NAME = request.POST["S_NAME"]
        SALARY = request.POST["SALARY"]
        A_ID_id  = request.POST["A_ID_id  "]

        new_post3 = STAFF.objects.create(S_ID= S_ID, S_NAME=S_NAME, SALARY=SALARY, A_ID_id
=A_ID_id )

        return redirect("/allposts3/")
    return render(request,"create_staff.html")

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

        P_ID = request.POST["P_ID"]
        P_NAME = request.POST["P_NAME"]
        DISEASE = request.POST["DISEASE"]
        FEE = request.POST["FEE"]
        TOTALBILL = request.POST["TOTALBILL"]
        D_ID_id  = request.POST["D_ID"]

        new_post4 = PATIENT.objects.create(P_ID= P_ID, P_NAME=P_NAME, DISEASE= DISEASE, FEE
= FEE, TOTALBILL= TOTALBILL, D_ID_id  =D_ID_id )

        return redirect("/allposts4/")
    return render(request,"create_patient.html")

def all_posts_administration(request):
    all_posts_administration = Administration.objects.all()
    return render(request,"all_posts_administration.html",{"posts": all_posts_administratio
n})

def all_posts_doctor(request):
    all_posts_doctor = Doctor.objects.all()
    return render(request,"all_posts_doctor.html",{"posts": all_posts_doctor})

def all_posts_staff(request):
    all_posts_staff = STAFF.objects.all()
    return render(request,"all_posts_staff.html",{"posts": all_posts_staff})

def all_posts_patient(request):
    all_posts_patient = PATIENT.objects.all()
    return render(request,"all_posts_patient.html",{"posts": all_posts_patient})

def delete_post_admin(request, post_id):
    post1 = Administration.objects.get(pk=post_id)
    post1.delete()
    return redirect("/allposts1/")

def delete_post_doctor(request, post_id):
    post2 = Doctor.objects.get(pk=post_id)
    post2.delete()
    return redirect("/allposts2/")

def delete_post_staff(request, post_id):
    post3 = STAFF.objects.get(pk=post_id)
    post3.delete()
    return redirect("/allposts3/")

def delete_post_patient(request, post_id):
    post4 = PATIENT.objects.get(pk=post_id)
    post4.delete()
    return redirect("/allposts4/")

def edit_admin(request, post_id):
    post1 = Administration.objects.get(pk=post_id)
    if request.method == "POST":

        A_ID = request.POST["A_ID"]
        A_NAME = request.POST["A_NAME"]
        GENDER = request.POST["GENDER"]

        post1.A_ID = A_ID
        post1.A_NAME = A_NAME
        post1.GENDER = GENDER
        post1.save()
        return redirect(f"/post1/{post1.id}/")

    return render(request,"edit_admin.html",{"post": post1})

def edit_doctor(request, post_id):
    post2 = Doctor.objects.get(pk=post_id)
    if request.method == "POST":

        D_ID = request.POST["D_ID"]
        D_NAME = request.POST["D_NAME"]
        DEPT = request.POST["DEPT"]
        A_ID_id  = request.POST["A_ID_id  "]

        post2.D_ID = D_ID
        post2.D_NAME = D_NAME
        post2.DEPT = DEPT
        post2.A_ID_id  = A_ID_id
        post2.save()
        return redirect(f"/post2/{post2.id}/")

    return render(request,"edit_doctor.html",{"post": post2})

def edit_staff(request, post_id):
    post3 = STAFF.objects.get(pk=post_id)
    if request.method == "POST":

        S_ID = request.POST["S_ID"]
        S_NAME = request.POST["S_NAME"]
        SALARY = request.POST["SALARY"]
        A_ID_id  = request.POST["A_ID"]

        post3.S_ID = S_ID
        post3.S_NAME = S_NAME
        post3.SALARY = SALARY
        post3.A_ID_id  = A_ID_id
        post3.save()
        return redirect(f"/post3/{post3.id}/")

    return render(request,"edit_staff.html",{"post": post3})


def edit_patient(request, post_id):
    post4 = PATIENT.objects.get(pk=post_id)
    if request.method == "POST":

        P_ID = request.POST["P_ID"]
        P_NAME = request.POST["P_NAME"]
        DISEASE = request.POST["DISEASE"]
        FEE = request.POST["FEE"]
        TOTALBILL = request.POST["TOTALBILL"]
        D_ID = request.POST["D_ID_id  "]

        post4.P_ID = P_ID
        post4.P_NAME = P_NAME
        post4.DISEASE = DISEASE
        post4.FEE = FEE
        post4.TOTALBILL = TOTALBILL
        post4.D_ID_id  = D_ID_id
        post4.save()
        return redirect(f"/post4/{post4.id}/")

    return render(request,"edit_patient.html",{"post": post4})
```

manage.py

```
#!/usr/bin/env python
"""Django's command-line utility for administrative tasks."""
import os
import sys


def main():
    """Run administrative tasks."""
    os.environ.setdefault('DJANGO_SETTINGS_MODULE', 'dbms1.settings')
    try:
        from django.core.management import execute_from_command_line
    except ImportError as exc:
        raise ImportError(
            "Couldn't import Django. Are you sure it's installed and "
            "available on your PYTHONPATH environment variable? Did you "
            "forget to activate a virtual environment?"
        ) from exc
    execute_from_command_line(sys.argv)


if __name__ == '__main__':
    main()
```

CHAPTER 6

TESTING

This chapter gives the outline of all testing methods that are carried out to get a bug free system. Quality can be achieved by testing the product using different techniques at different phases of the project development. 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 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 test. Each test type addresses a specific testing requirement.

6.1 TESTING PROCESS

Testing is an integral part of software development. Testing process certifies whether the product that is developed compiles with the standards that it was designed to. Testing process involves building of test cases against which the product has to be tested.

6.2 TESTING OBJECTIVES

The main objectives of testing process are as follows.

- Testing is a process of executing a program with the intent of finding an error.
- A good test case is one that has high probability of finding undiscovered error.
- A successful test is one that uncovers the undiscovered error.

6.3 TEST CASES

The test cases provided here test the most important features of the project.

6.3.1 Test cases for the project

Table 6.1 ------- Test Case

Sl No	Test Input	Expected Results	Observed Results	Remarks

1	Wrong Password	Invalid Credentials	Invalid Credentials	
2				
3				

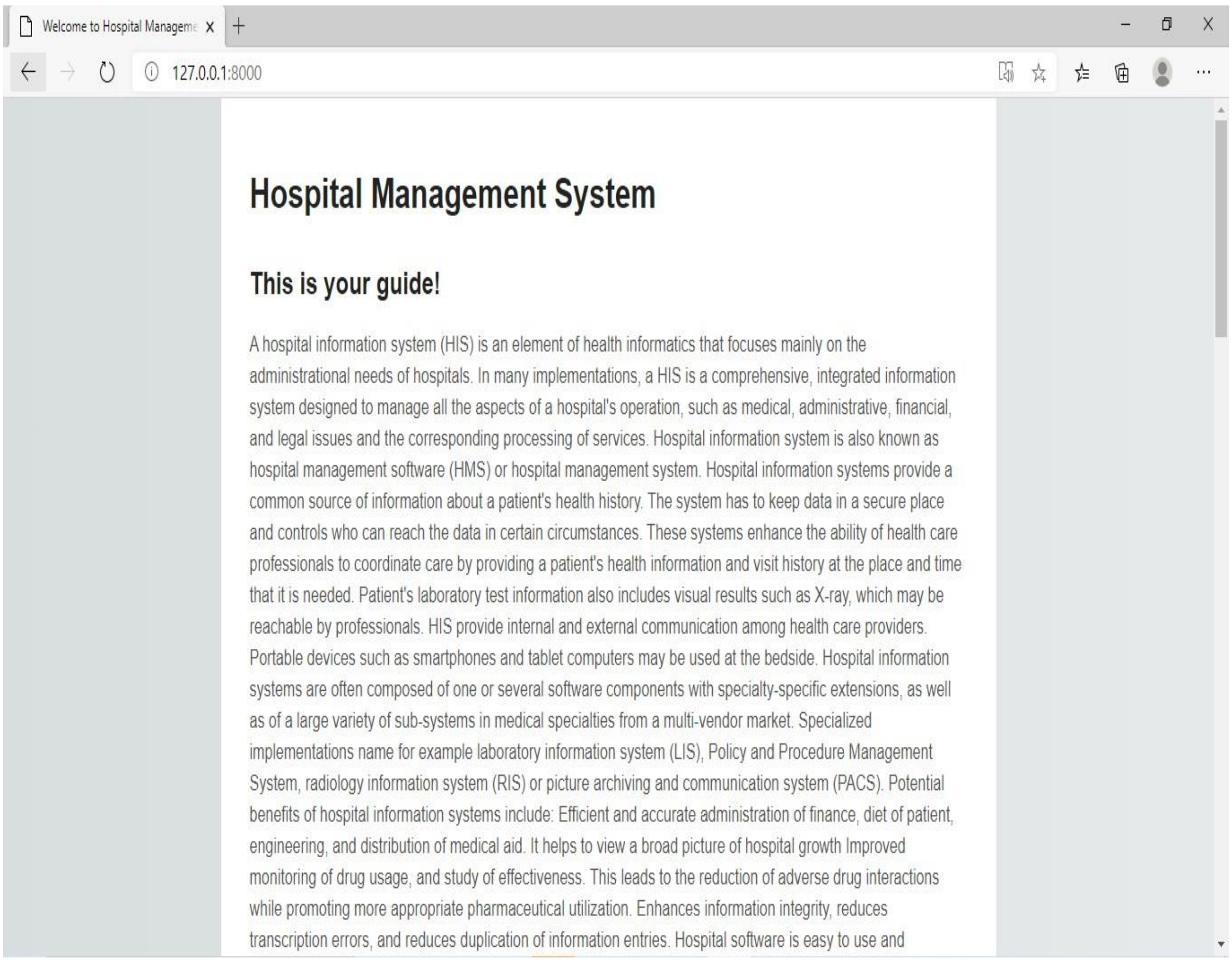
CHAPTER 7

RESULTS

This section describes the screens of the "Hospital Management System". The snapshots are shown below for each module.

7.1 SNAPSHOTS

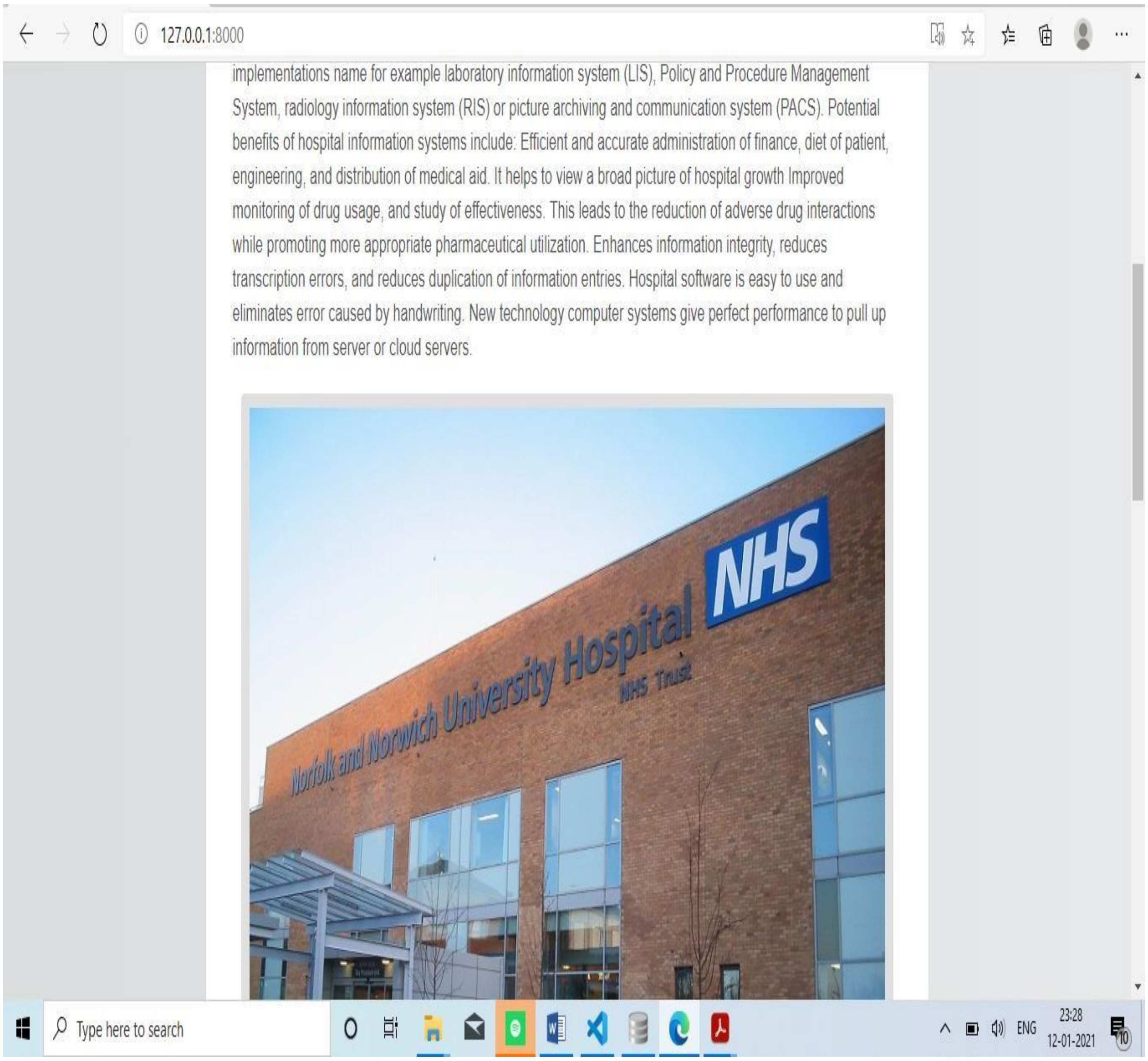
Snapshot1 :----------Layout



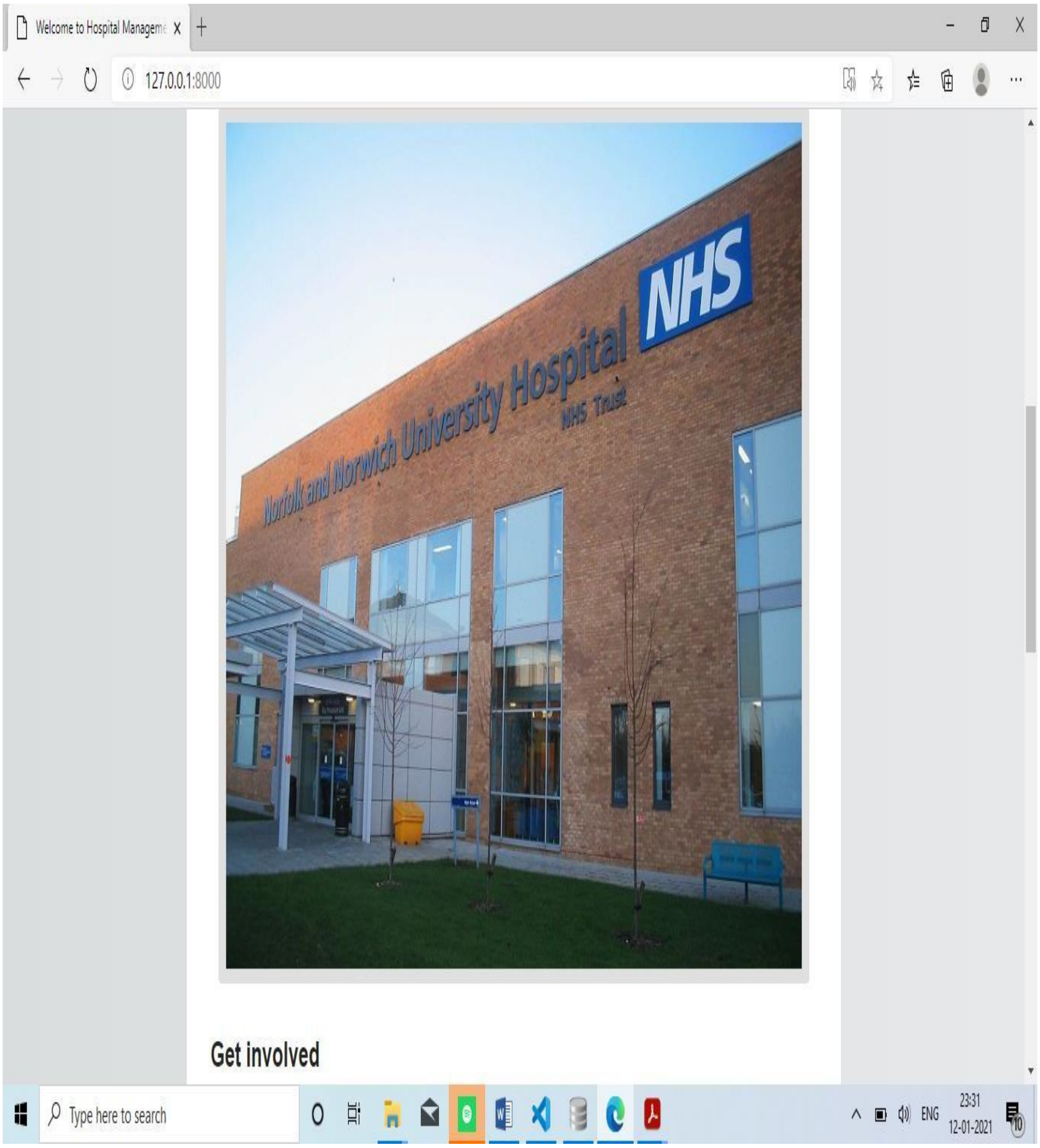
Hospital Management System

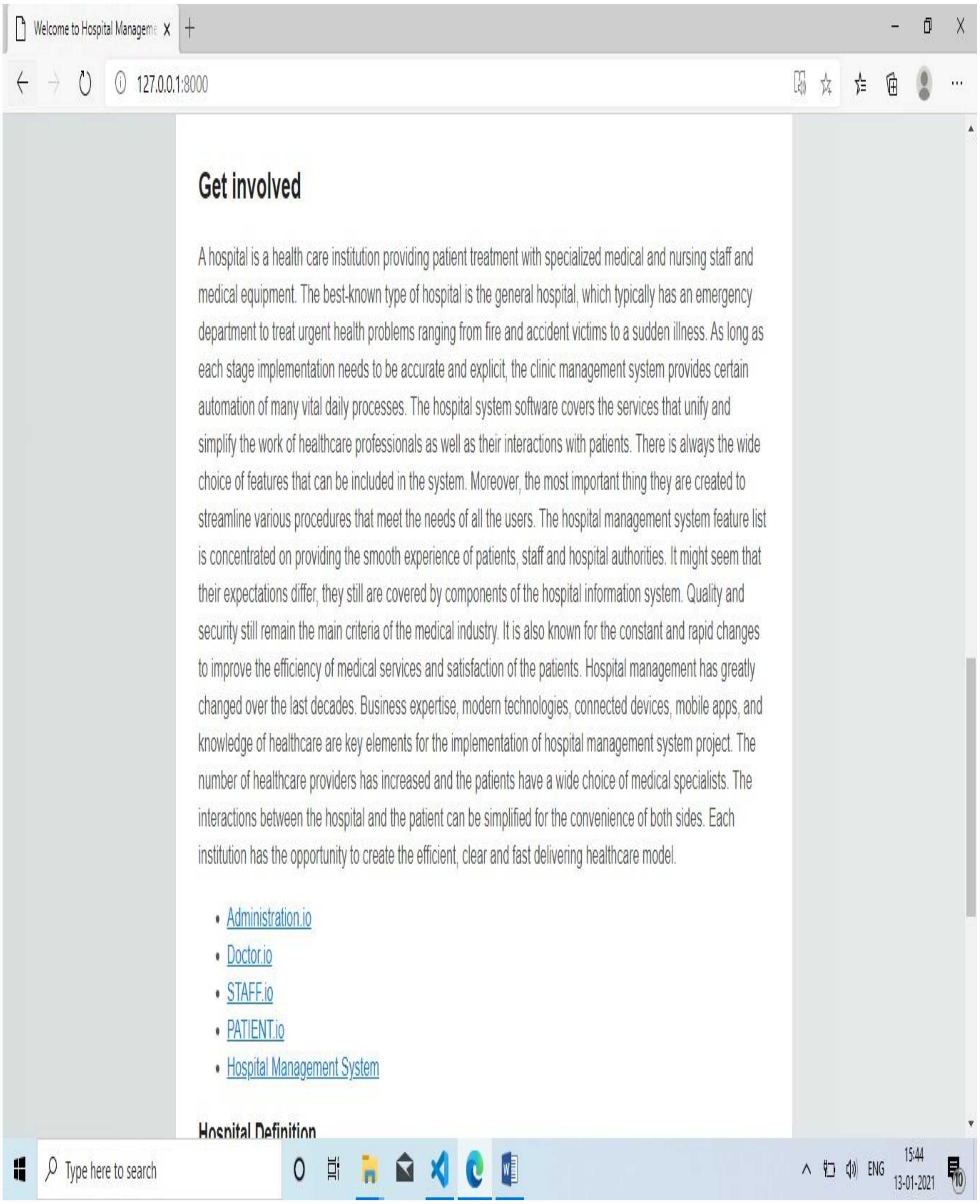
This is your guide!

A hospital information system (HIS) is an element of health informatics that focuses mainly on the administrational needs of hospitals. In many implementations, a HIS is a comprehensive, integrated information system designed to manage all the aspects of a hospital's operation, such as medical, administrative, financial, and legal issues and the corresponding processing of services. Hospital information system is also known as hospital management software (HMS) or hospital management system. Hospital information systems provide a common source of information about a patient's health history. The system has to keep data in a secure place and controls who can reach the data in certain circumstances. These systems enhance the ability of health care professionals to coordinate care by providing a patient's health information and visit history at the place and time that it is needed. Patient's laboratory test information also includes visual results such as X-ray, which may be reachable by professionals. HIS provide internal and external communication among health care providers. Portable devices such as smartphones and tablet computers may be used at the bedside. Hospital information systems are often composed of one or several software components with specialty-specific extensions, as well as of a large variety of sub-systems in medical specialties from a multi-vendor market. Specialized implementations name for example laboratory information system (LIS), Policy and Procedure Management System, radiology information system (RIS) or picture archiving and communication system (PACS). Potential benefits of hospital information systems include: Efficient and accurate administration of finance, diet of patient, engineering, and distribution of medical aid. It helps to view a broad picture of hospital growth Improved monitoring of drug usage, and study of effectiveness. This leads to the reduction of adverse drug interactions while promoting more appropriate pharmaceutical utilization. Enhances information integrity, reduces transcription errors, and reduces duplication of information entries. Hospital software is easy to use and



implementations name for example laboratory information system (LIS), Policy and Procedure Management System, radiology information system (RIS) or picture archiving and communication system (PACS). Potential benefits of hospital information systems include: Efficient and accurate administration of finance, diet of patient, engineering, and distribution of medical aid. It helps to view a broad picture of hospital growth Improved monitoring of drug usage, and study of effectiveness. This leads to the reduction of adverse drug interactions while promoting more appropriate pharmaceutical utilization. Enhances information integrity, reduces transcription errors, and reduces duplication of information entries. Hospital software is easy to use and eliminates error caused by handwriting. New technology computer systems give perfect performance to pull up information from server or cloud servers.


Welcome to Hospital Manageme
127.0.0.1:8000
Norfolk and Norwich University Hospital
NHS Trust
NHS
Get involved
Type here to search
ENG
23:31
12-01-2021

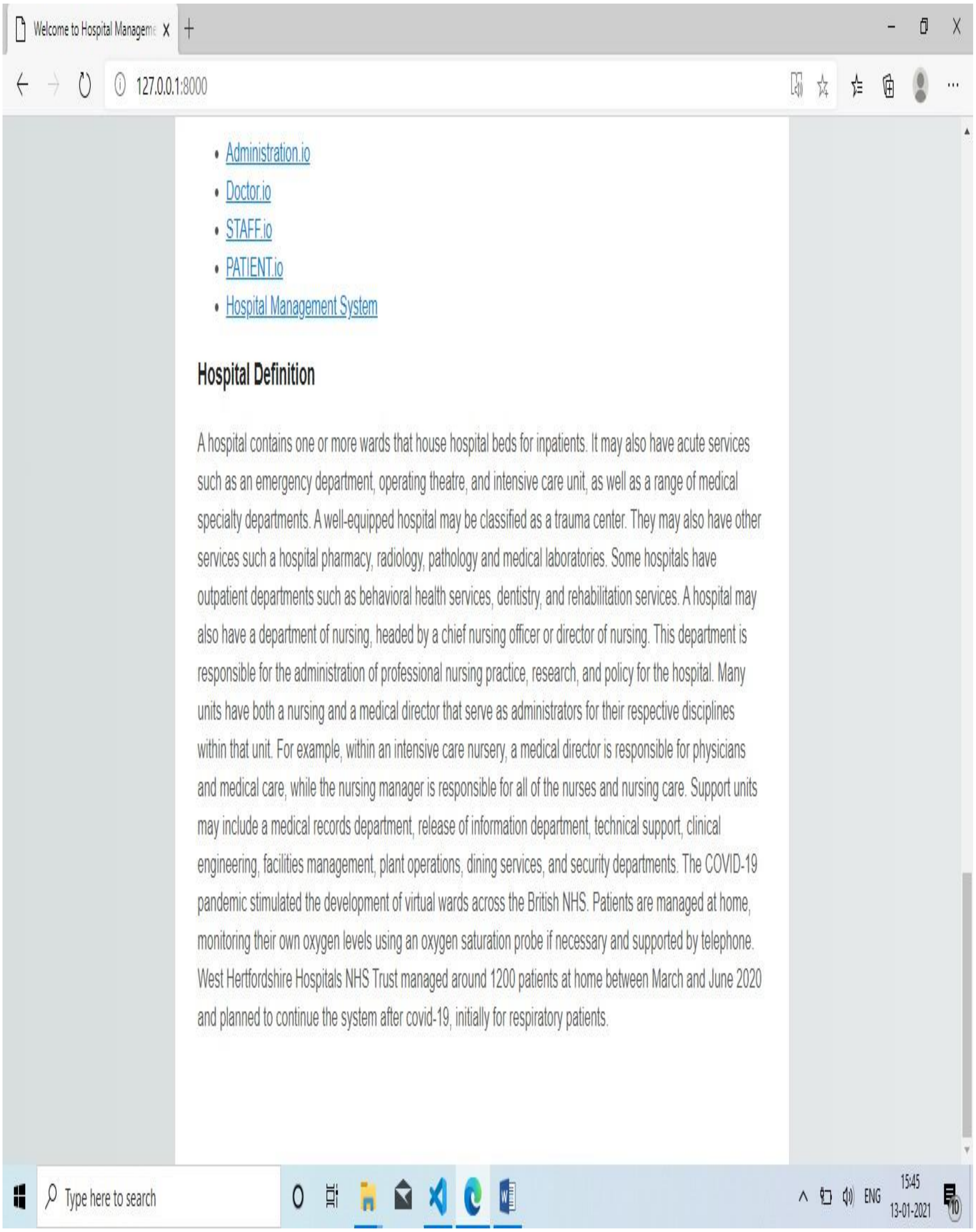


Get involved

A hospital is a health care institution providing patient treatment with specialized medical and nursing staff and medical equipment. The best-known type of hospital is the general hospital, which typically has an emergency department to treat urgent health problems ranging from fire and accident victims to a sudden illness. As long as each stage implementation needs to be accurate and explicit, the clinic management system provides certain automation of many vital daily processes. The hospital system software covers the services that unify and simplify the work of healthcare professionals as well as their interactions with patients. There is always the wide choice of features that can be included in the system. Moreover, the most important thing they are created to streamline various procedures that meet the needs of all the users. The hospital management system feature list is concentrated on providing the smooth experience of patients, staff and hospital authorities. It might seem that their expectations differ, they still are covered by components of the hospital information system. Quality and security still remain the main criteria of the medical industry. It is also known for the constant and rapid changes to improve the efficiency of medical services and satisfaction of the patients. Hospital management has greatly changed over the last decades. Business expertise, modern technologies, connected devices, mobile apps, and knowledge of healthcare are key elements for the implementation of hospital management system project. The number of healthcare providers has increased and the patients have a wide choice of medical specialists. The interactions between the hospital and the patient can be simplified for the convenience of both sides. Each institution has the opportunity to create the efficient, clear and fast delivering healthcare model.

- Administration.io
- Doctor.io
- STAFF.io
- PATIENT.io
- Hospital Management System

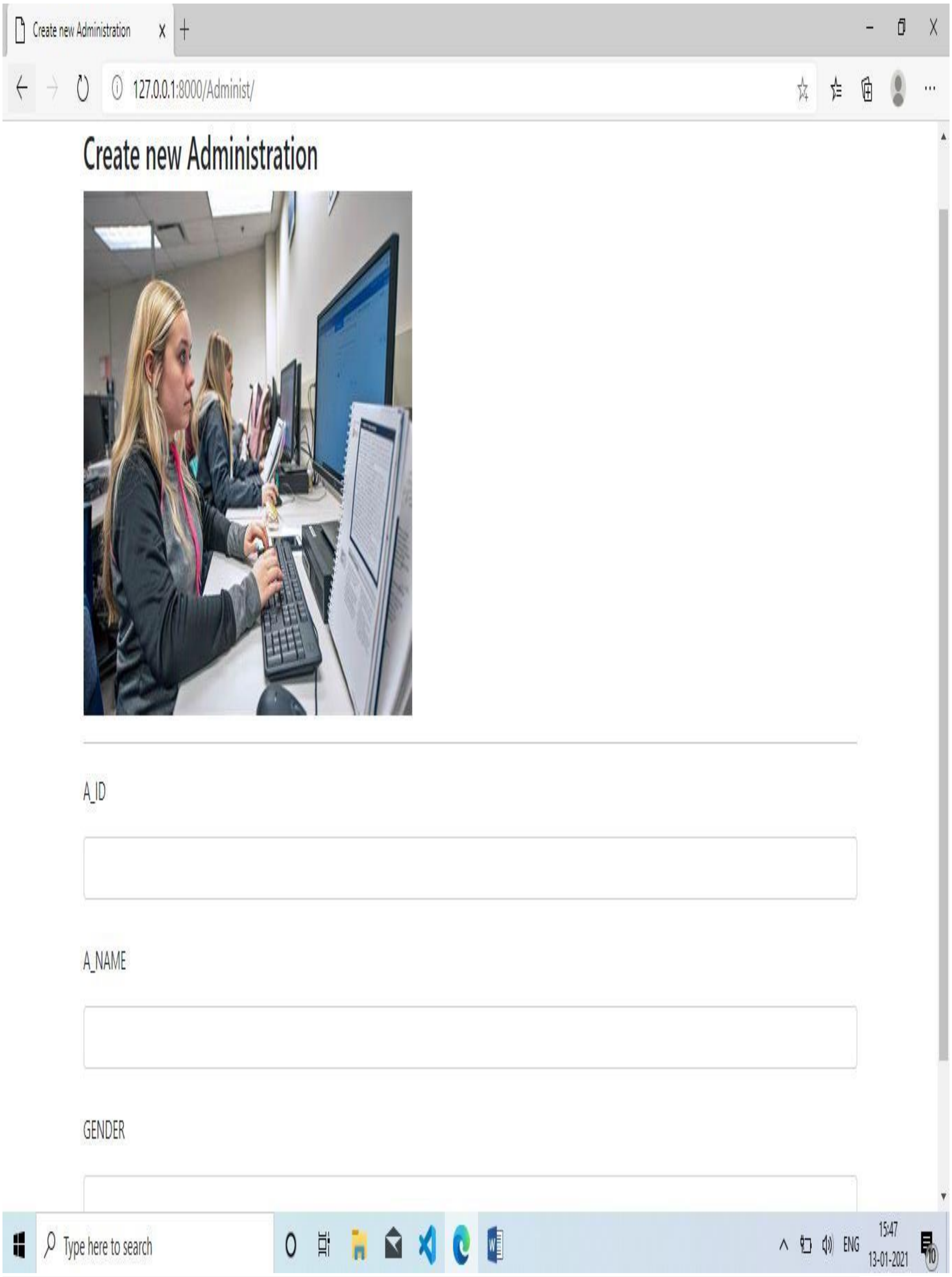
Hospital Definition



- Administration.io
- Doctor.io
- STAFF.io
- PATIENT.io
- Hospital Management System

Hospital Definition

A hospital contains one or more wards that house hospital beds for inpatients. It may also have acute services such as an emergency department, operating theatre, and intensive care unit, as well as a range of medical specialty departments. A well-equipped hospital may be classified as a trauma center. They may also have other services such a hospital pharmacy, radiology, pathology and medical laboratories. Some hospitals have outpatient departments such as behavioral health services, dentistry, and rehabilitation services. A hospital may also have a department of nursing, headed by a chief nursing officer or director of nursing. This department is responsible for the administration of professional nursing practice, research, and policy for the hospital. Many units have both a nursing and a medical director that serve as administrators for their respective disciplines within that unit. For example, within an intensive care nursery, a medical director is responsible for physicians and medical care, while the nursing manager is responsible for all of the nurses and nursing care. Support units may include a medical records department, release of information department, technical support, clinical engineering, facilities management, plant operations, dining services, and security departments. The COVID-19 pandemic stimulated the development of virtual wards across the British NHS. Patients are managed at home, monitoring their own oxygen levels using an oxygen saturation probe if necessary and supported by telephone. West Hertfordshire Hospitals NHS Trust managed around 1200 patients at home between March and June 2020 and planned to continue the system after covid-19, initially for respiratory patients.


Create new Administration

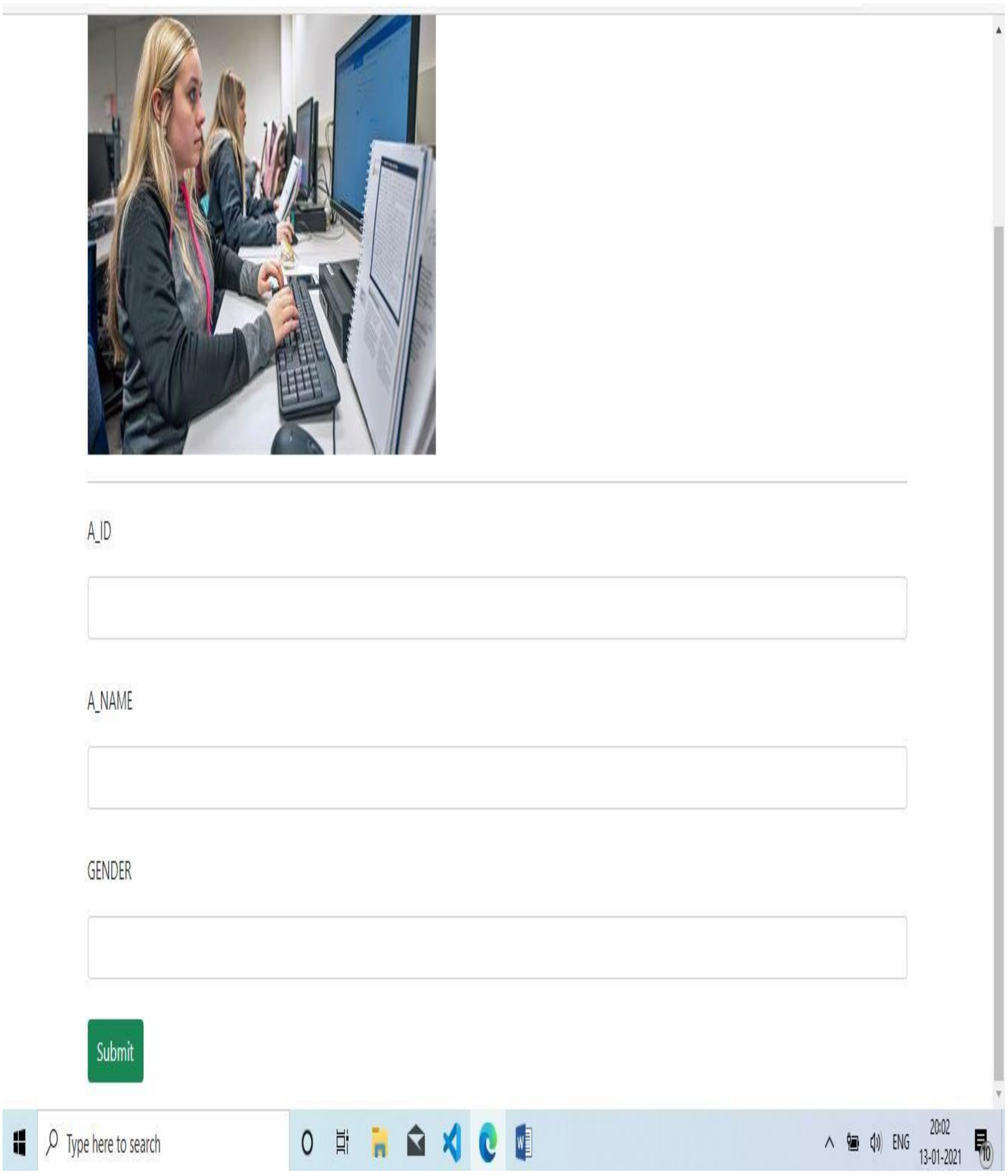
127.0.0.1:8000/Administ/

Create new Administration

A_ID

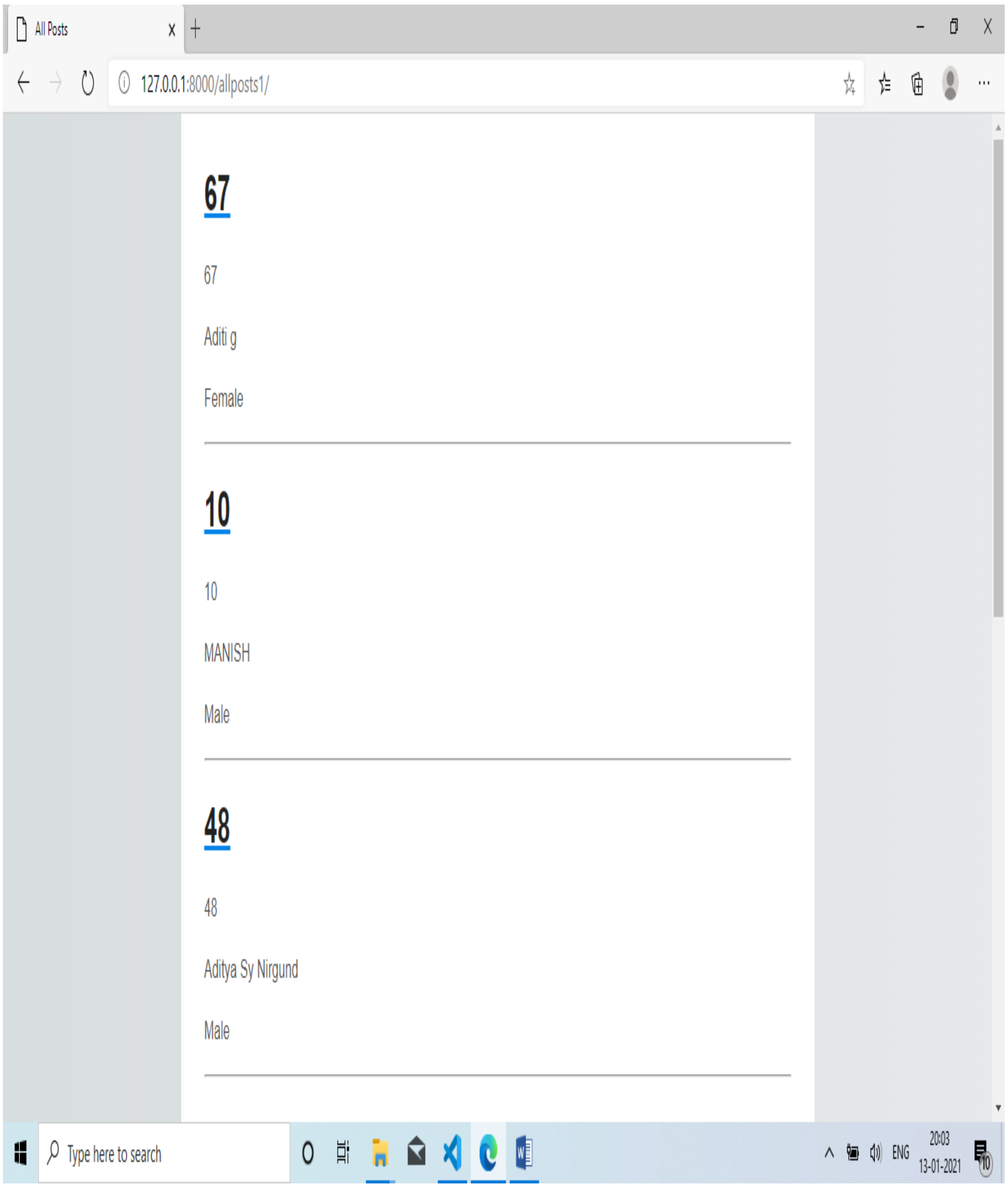
A_NAME

GENDER


A_ID

A_NAME

GENDER


All Posts

127.0.0.1:8000/allposts1/

67

67

Aditi g

Female

10

10

MANISH

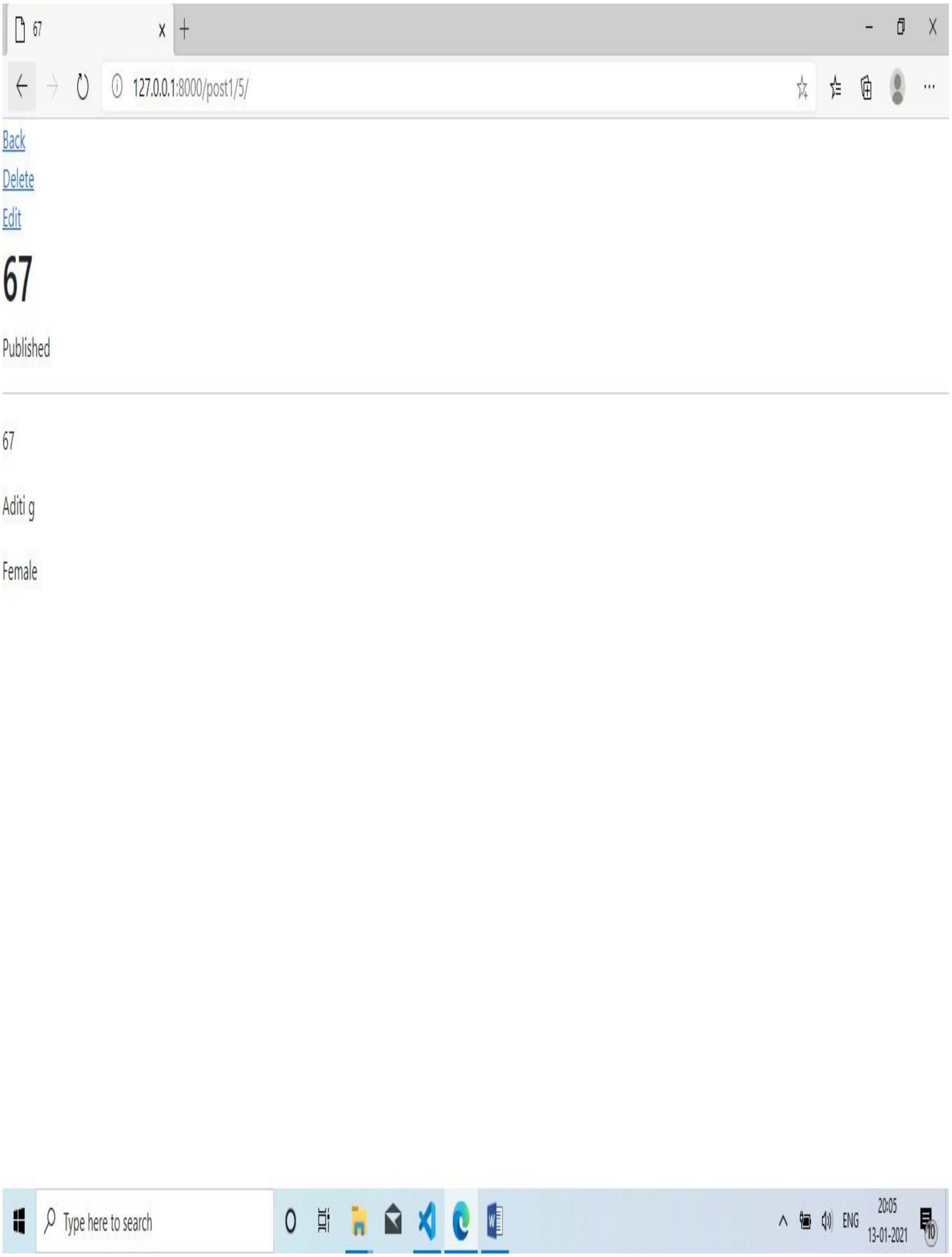
Male

48

48

Aditya Sy Nirgund

Male



67

127.0.0.1:8000/post1/5/

Back

Delete

Edit

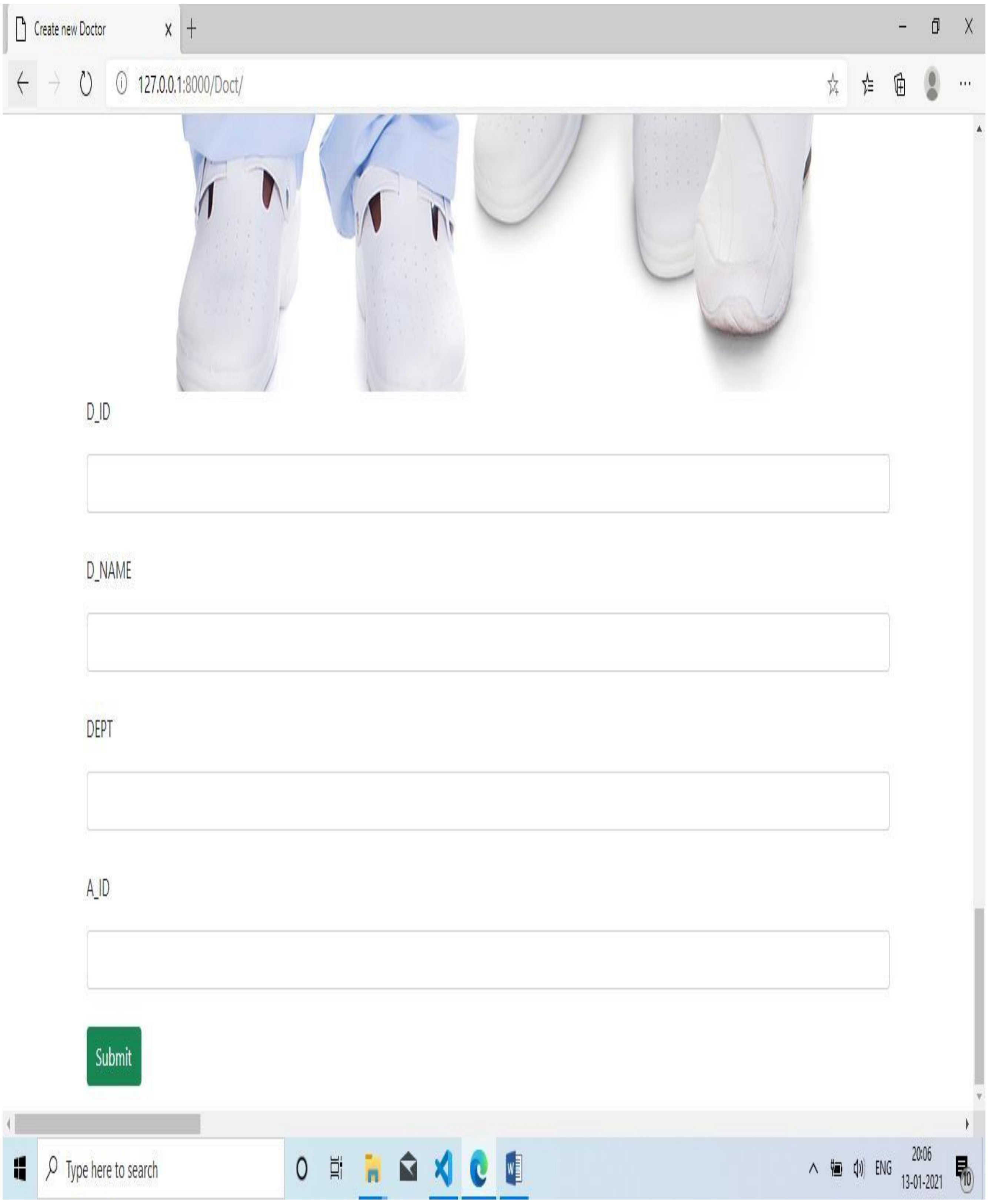
67

Published

67

Aditi g

Female


Create new Doctor

127.0.0.1:8000/Doct/

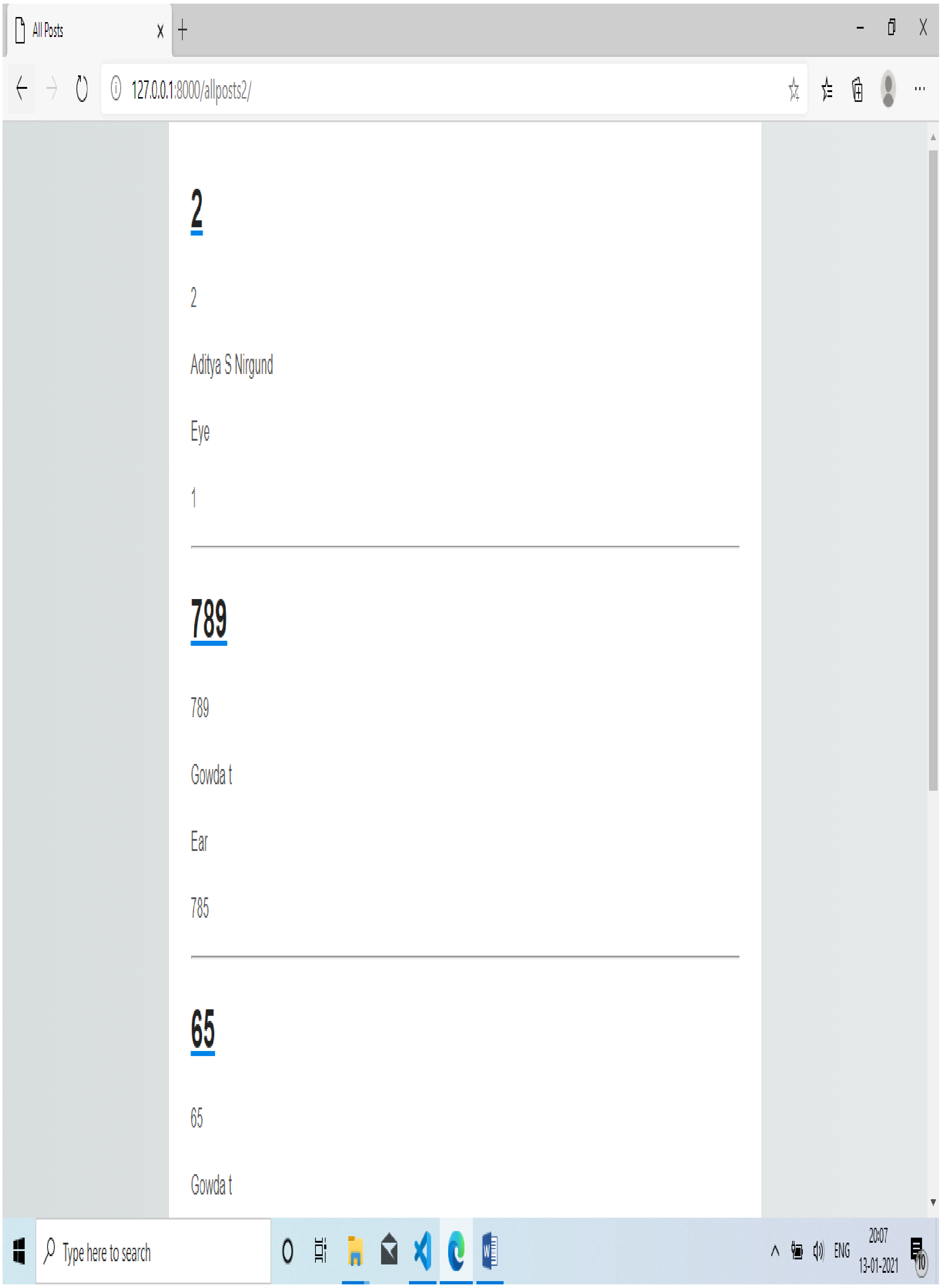
D_ID

D_NAME

DEPT

A_ID

Submit


All Posts

127.0.0.1:8000/allposts2/

2

2

Aditya S Nirgund

Eye

1

789

789

Gowda t

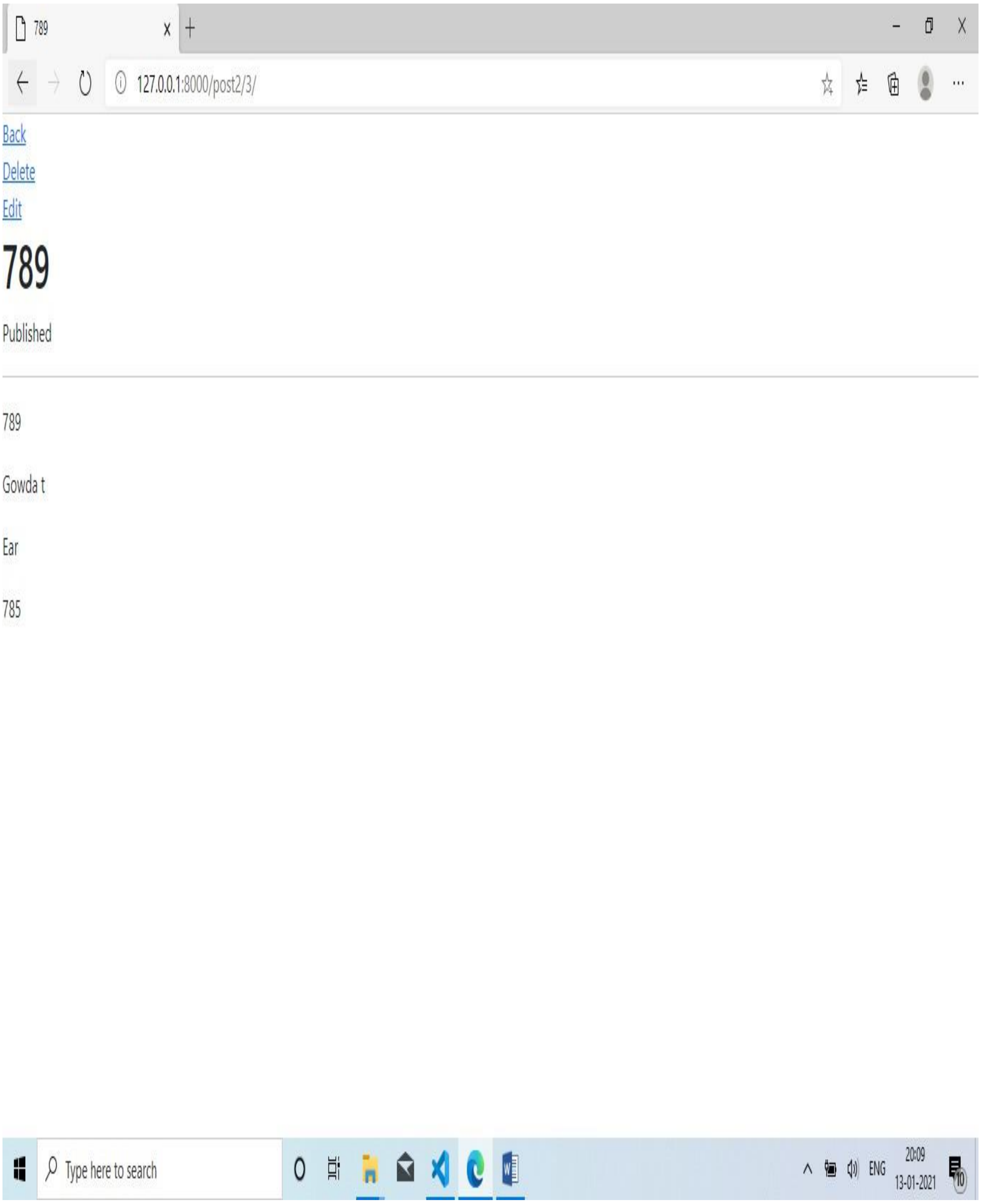
Ear

785

65

65

Gowda t



Back

Delete

Edit

789

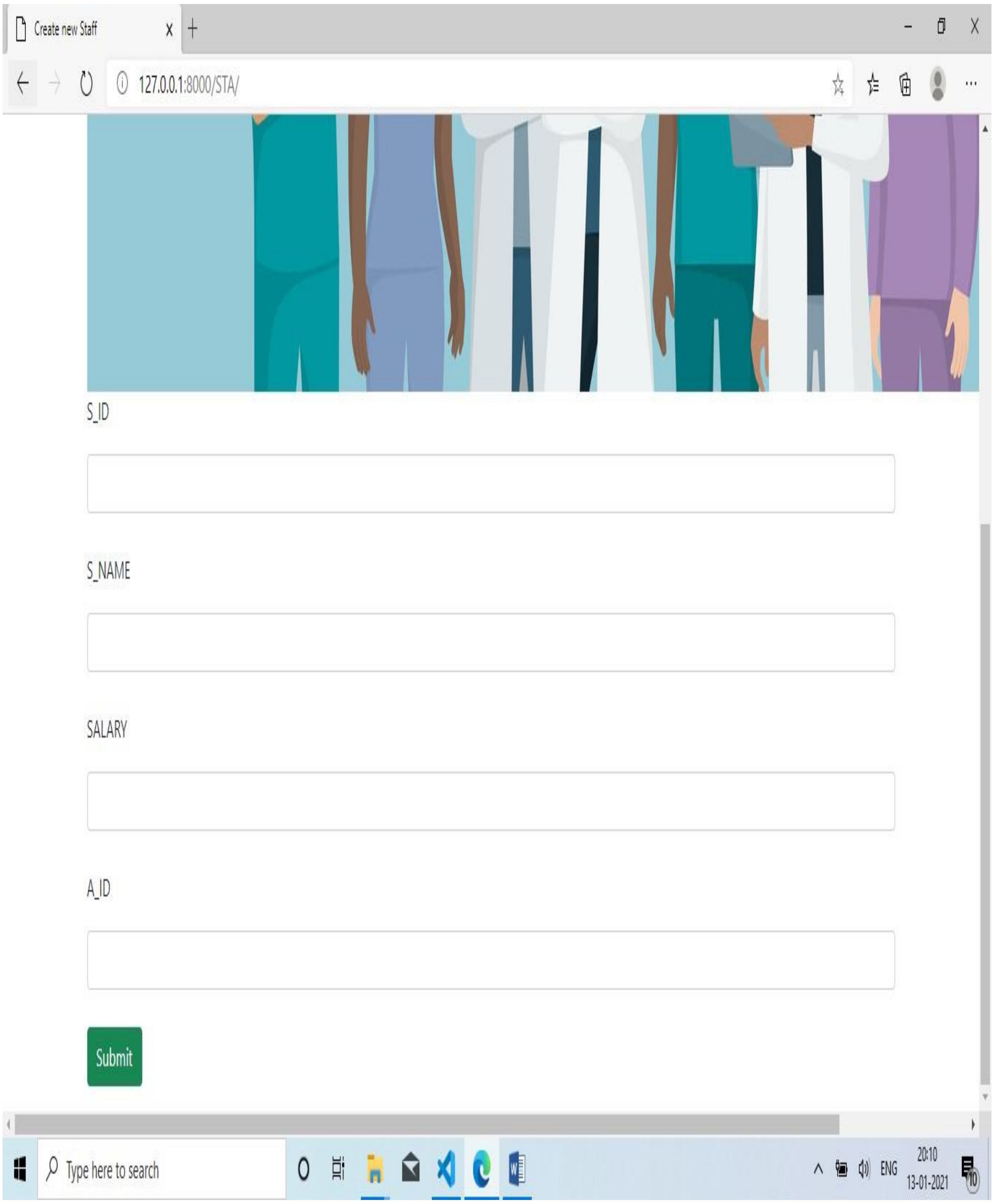
Published

789

Gowda t

Ear

785



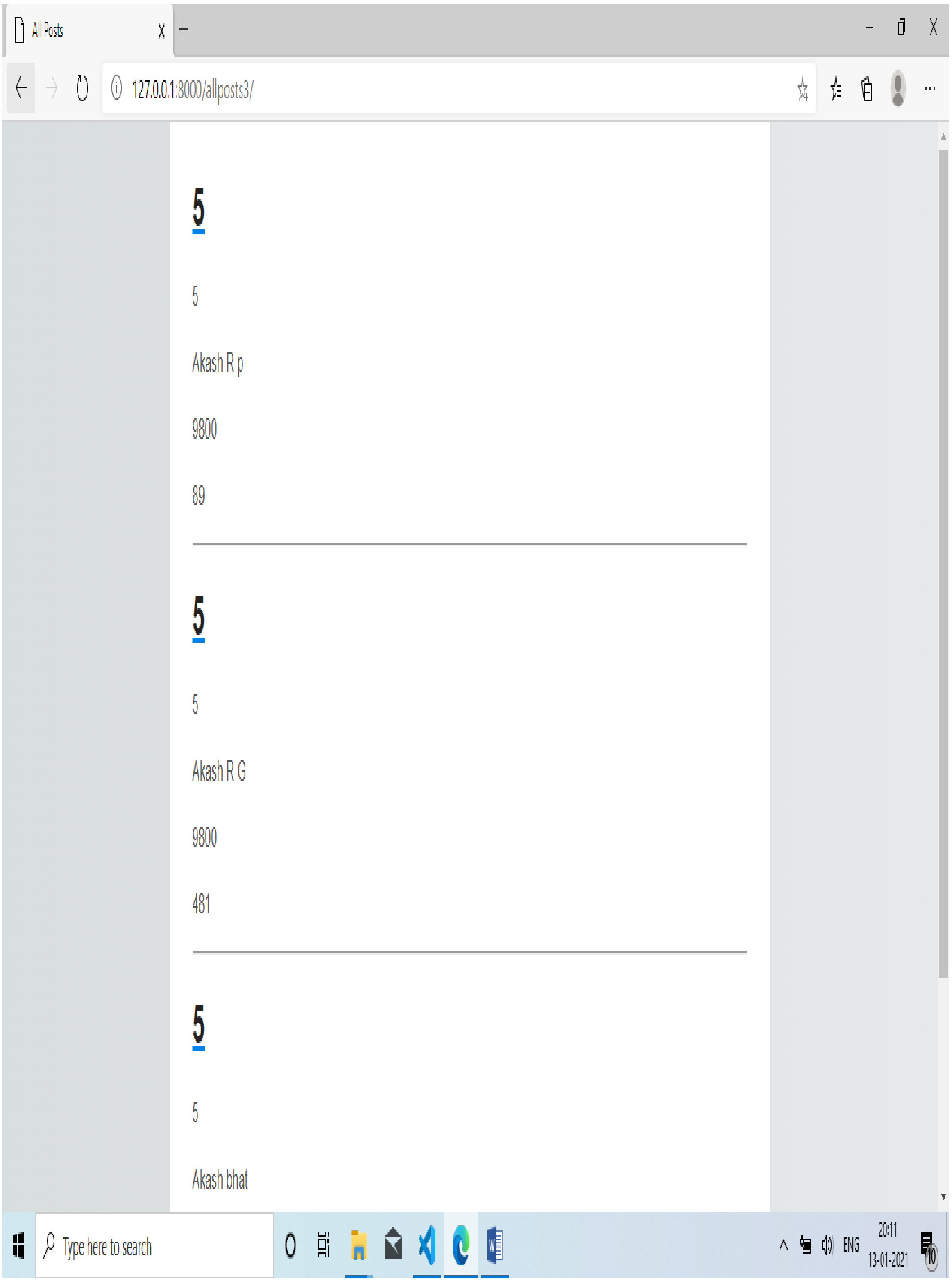
S_ID

S_NAME

SALARY

A_ID

Submit



5

5

Akash R p

9800

89

5

5

Akash R G

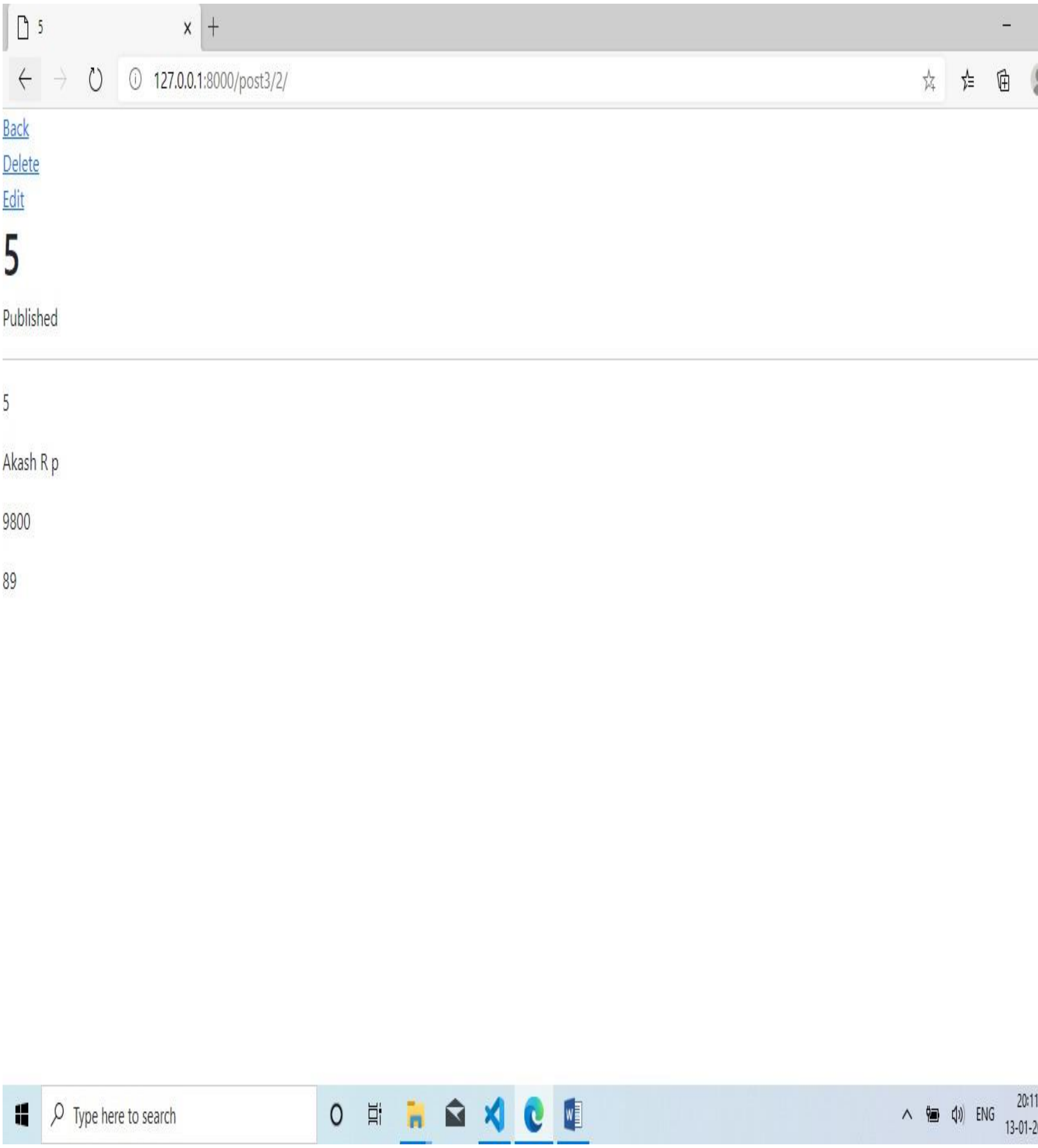
9800

481

5

5

Akash bhat



Back

Delete

Edit

5

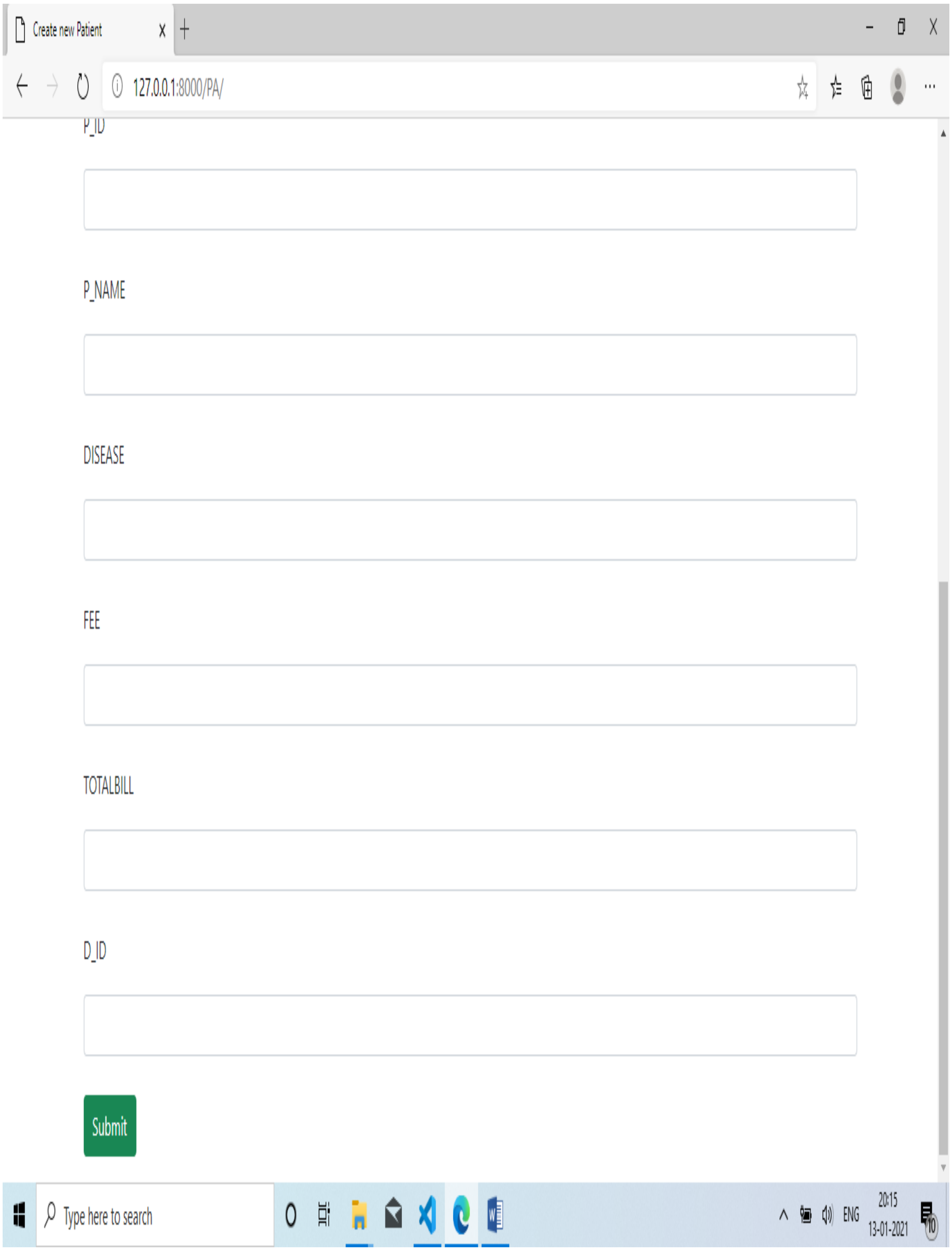
Published

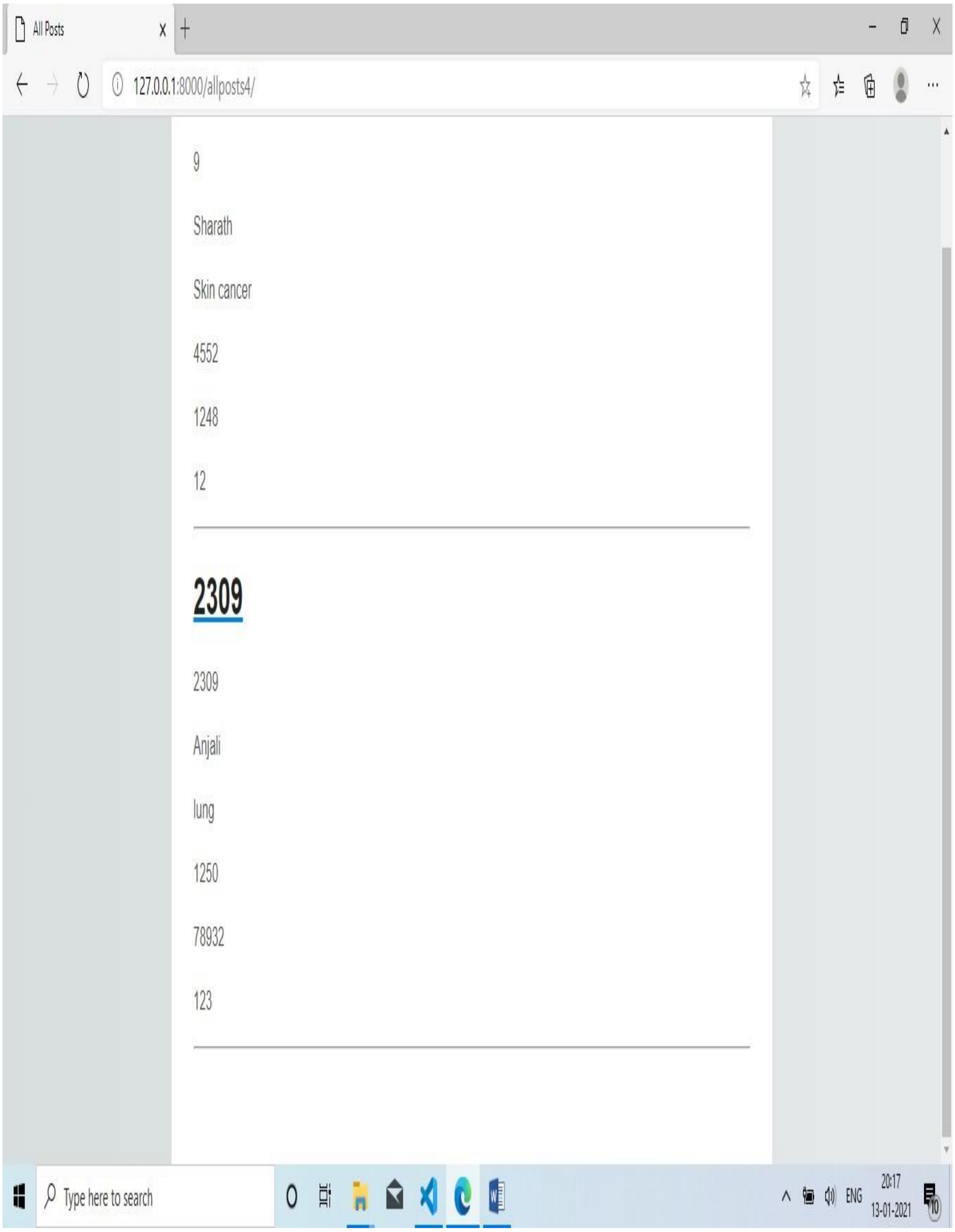
5

Akash R p

9800

89


Create new Patient
127.0.0.1:8000/PA/
P_ID
P_NAME
DISEASE
FEE
TOTALBILL
D_ID
Submit
Type here to search
ENG
20:15
13-01-2021



9

Sharath

Skin cancer

4552

1248

12

2309

2309

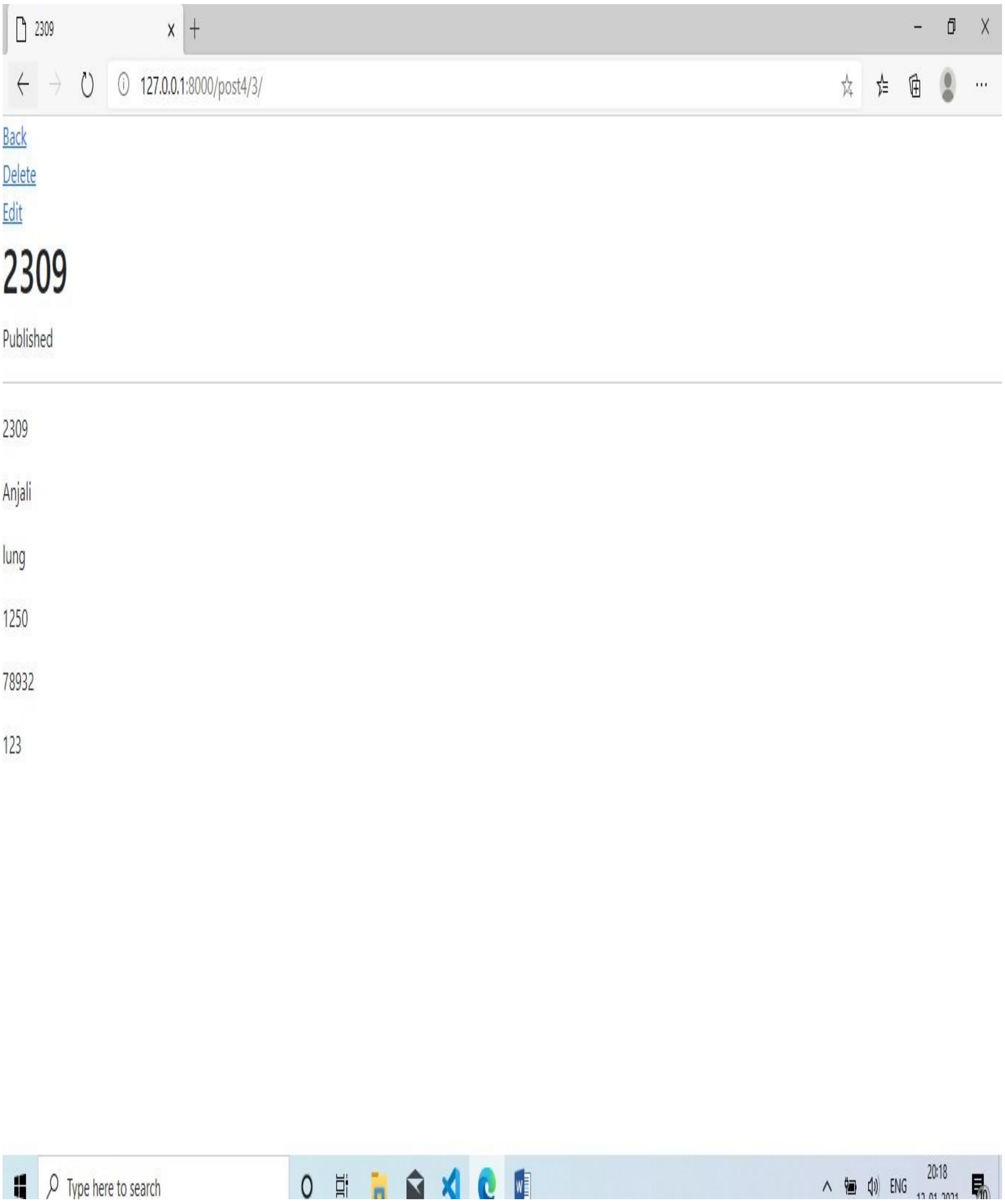
Anjali

lung

1250

78932

123



2309

127.0.0.1:8000/post4/3/

Back

Delete

Edit

2309

Published

2309

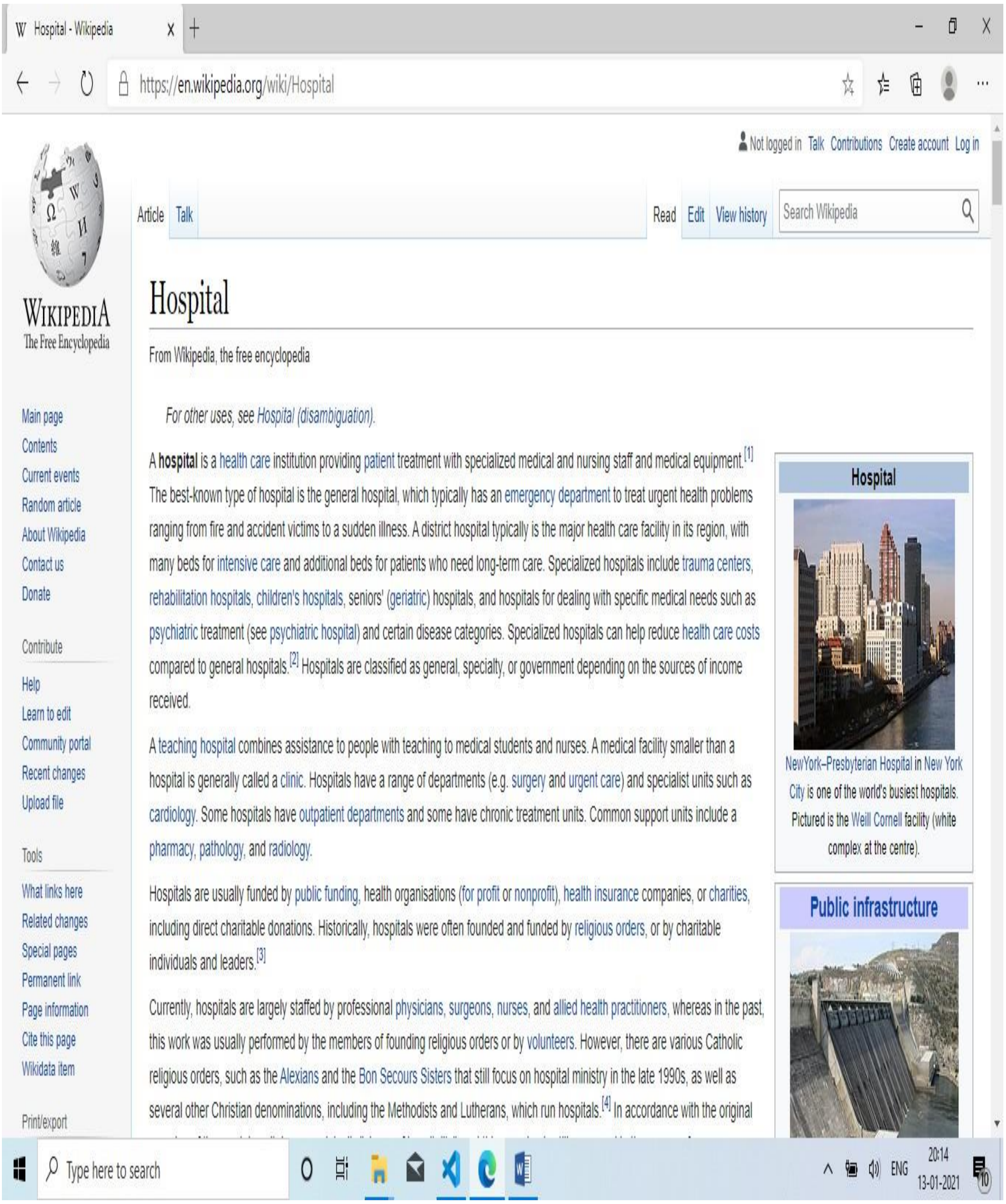
Anjali

lung

1250

78932

123



Hospital

From Wikipedia, the free encyclopedia

For other uses, see Hospital (disambiguation).

A **hospital** is a health care institution providing patient treatment with specialized medical and nursing staff and medical equipment.[1] The best-known type of hospital is the general hospital, which typically has an emergency department to treat urgent health problems ranging from fire and accident victims to a sudden illness. A district hospital typically is the major health care facility in its region, with many beds for intensive care and additional beds for patients who need long-term care. Specialized hospitals include trauma centers, rehabilitation hospitals, children's hospitals, seniors' (geriatric) hospitals, and hospitals for dealing with specific medical needs such as psychiatric treatment (see psychiatric hospital) and certain disease categories. Specialized hospitals can help reduce health care costs compared to general hospitals.[2] Hospitals are classified as general, specialty, or government depending on the sources of income received.

A teaching hospital combines assistance to people with teaching to medical students and nurses. A medical facility smaller than a hospital is generally called a clinic. Hospitals have a range of departments (e.g. surgery and urgent care) and specialist units such as cardiology. Some hospitals have outpatient departments and some have chronic treatment units. Common support units include a pharmacy, pathology, and radiology.

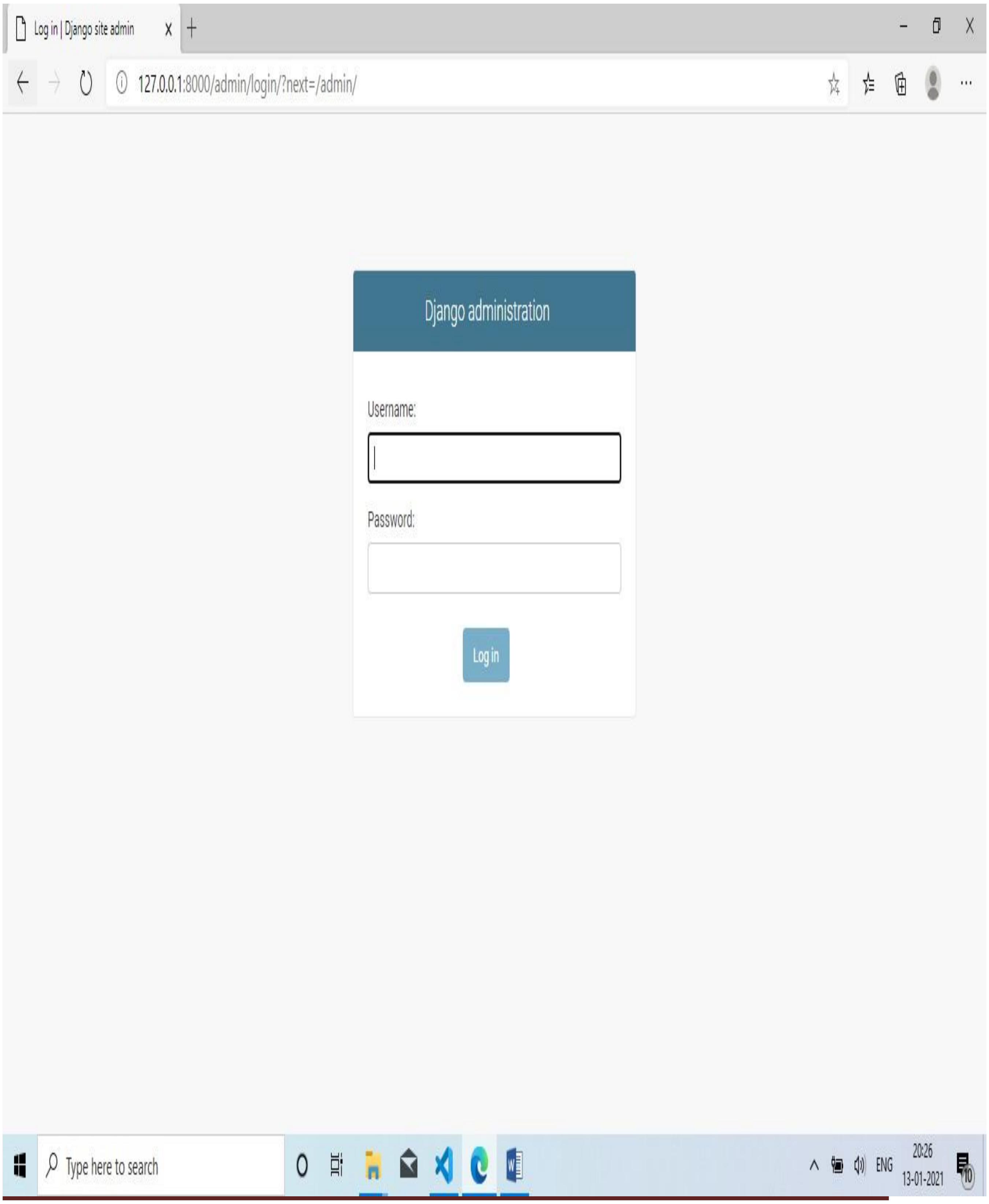
Hospitals are usually funded by public funding, health organisations (for profit or nonprofit), health insurance companies, or charities, including direct charitable donations. Historically, hospitals were often founded and funded by religious orders, or by charitable individuals and leaders.[3]

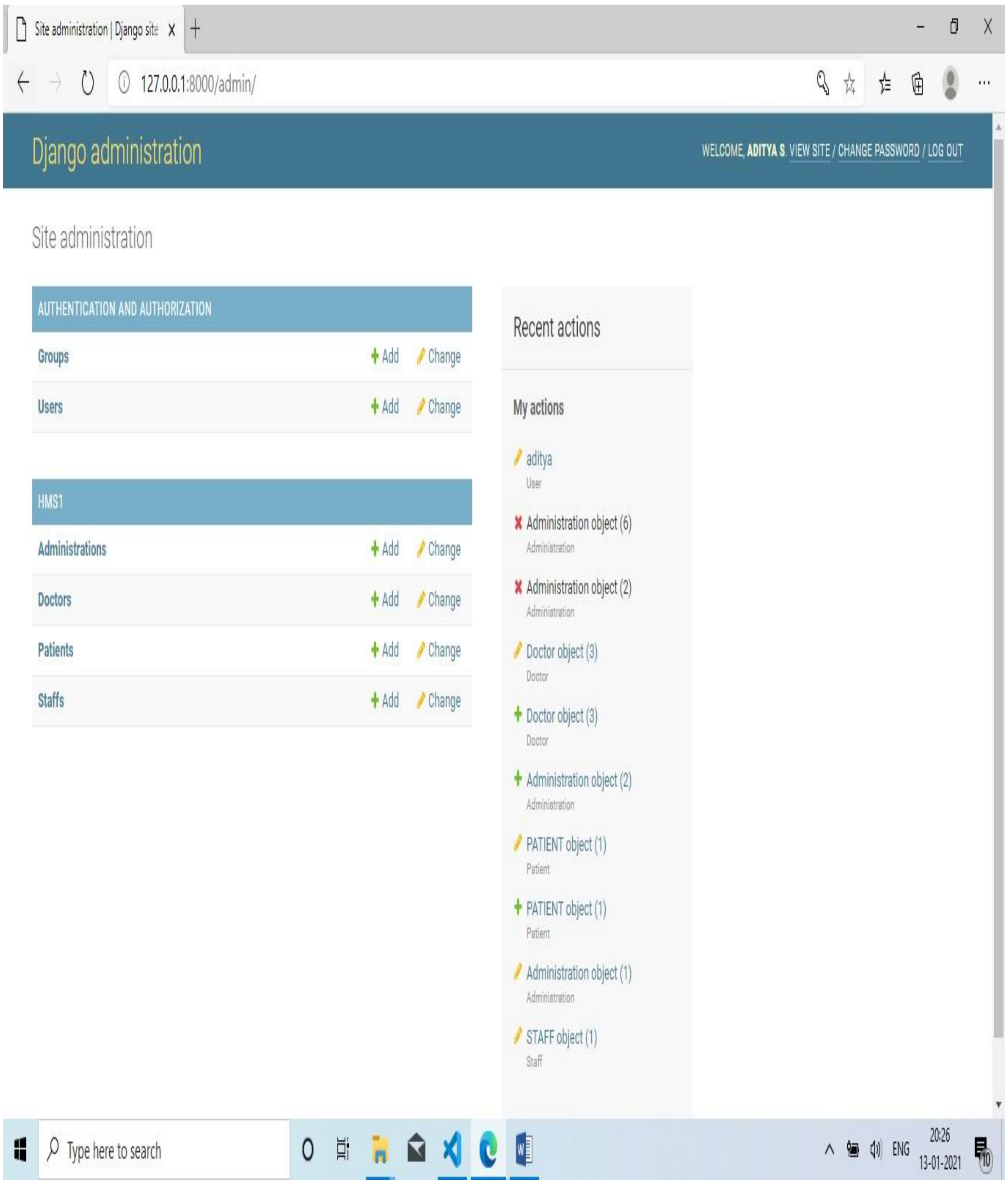
Currently, hospitals are largely staffed by professional physicians, surgeons, nurses, and allied health practitioners, whereas in the past, this work was usually performed by the members of founding religious orders or by volunteers. However, there are various Catholic religious orders, such as the Alexians and the Bon Secours Sisters that still focus on hospital ministry in the late 1990s, as well as several other Christian denominations, including the Methodists and Lutherans, which run hospitals.[4] In accordance with the original

Hospital

NewYork–Presbyterian Hospital in New York City is one of the world's busiest hospitals. Pictured is the Weill Cornell facility (white complex at the centre).

Public infrastructure


Log in | Django site admin
127.0.0.1:8000/admin/login/?next=/admin/
Django administration
Username:
Password:
Log in
Type here to search
ENG
20:26
13-01-2021



Django administration

WELCOME, **ADITYA S**. VIEW SITE / CHANGE PASSWORD / LOG OUT

Site administration

AUTHENTICATION AND AUTHORIZATION		
Groups	Add	Change
Users	Add	Change

HMS1		
Administrations	Add	Change
Doctors	Add	Change
Patients	Add	Change
Staffs	Add	Change

Recent actions

My actions

- aditya
  User
- Administration object (6)
  Administration
- Administration object (2)
  Administration
- Doctor object (3)
  Doctor
- Doctor object (3)
  Doctor
- Administration object (2)
  Administration
- PATIENT object (1)
  Patient
- PATIENT object (1)
  Patient
- Administration object (1)
  Administration
- STAFF object (1)
  Staff

CONCLUSION

Taking into account all the mentioned details, we can make the conclusion that the hospital management system is the inevitable part of the lifecycle of the modern medical institution. It automates numerous daily operations and enables smooth interactions of the users.

Developing the hospital system software is a great opportunity to create the distinct, efficient and fast delivering healthcare model. Implementation of hospital management system project helps to store all the kinds of records, provide coordination and user communication, implement policies, improve day-to-day operations, arrange the supply chain, manage financial and human resources, and market hospital services.

This beneficial decision covers the needs of the patients, staff and hospital authorities and simplifies their interactions. It has become the usual approach to manage the hospital. Many clinics have already experienced its advantages and continue developing new hospital management system project modules.

REFERENCES

[1] Django-Speckbit videos

[2] Full Stackment videos