DOMAIN-BASED WEB DEVELOPMENT PROJECT

HOSPITAL MANAGEMENT WEBSITE

Web Technology

DSC302-4

Vishal Das

(2341350)

Department of Statistics and Data Science

Date: 14-03-2025

1. Introduction

The **Hospital Management Django Project** is a web application designed to provide a comprehensive platform for managing hospital services, patient appointments, and healthcare information. The project is built using Django, a high-level Python web framework, and MySQL as the database backend. The application includes multiple pages, each dedicated to specific services offered by the hospital.

2. Project Structure

The project is organized into the following key components:

2.1. Django App: hospital_app

- **Models**: Define the database schema (e.g., Appointment model for storing patient appointments).
- **Views**: Handle the logic for rendering templates and processing form submissions.
- **Templates**: Contain HTML files for each page of the website.
- **Static Files**: Include CSS, JavaScript, and images used in the templates.
- **URLs**: Map URLs to views for routing requests.

2.2. Folder Structure

The folder structure of the project is as follows:

```
hospital_management/
├── hospital_app/
│   ├── migrations/                # Database migration files
│   ├── static/
│   │   └── hospital_app/          # Static files (CSS, JS,
images)
│   │       ├── styles.css
│   │       ├── script.js
│   │       ├── image/
│   │       │   ├── home.gif
│   │       │   ├── about.png
```

```
│   │       │   └── ...
│   ├── templates/
│   │   └── hospital_app/            # HTML templates
│   │       ├── home.html
│   │       ├── advanced_bed_facilities.html
│   │       ├── ambulance_service.html
│   │       ├── cardiology.html
│   │       ├── comprehensive_care.html
│   │       ├── dental_care.html
│   │       ├── experts_consultancy.html
│   │       ├── free_checkup.html
│   │       ├── pharmacy_medicine.html
│   │       └── physiotherapy.html
│   ├── admin.py                     # Django admin configuration
│   ├── apps.py                      # App configuration
│   ├── models.py                    # Database models
│   ├── tests.py                     # Unit tests
│   ├── views.py                     # View functions
│   ├── urls.py                      # URL routing for the app
├── hospital_management/
│   ├── settings.py                  # Django project settings
│   ├── urls.py                      # Main URL routing
│   ├── wsgi.py                      # WSGI configuration
│   ├── asgi.py                      # ASGI configuration
├── manage.py                        # Django management script
```

2.3. Templates

The templates/hospital_app folder contains the following HTML files:

Page	Description
home.html	The homepage of the website, featuring an overview of the hospital and services.
advanced_bed_facilities.html	Details about the advanced bed facilities available at the hospital.
ambulance_service.html	Information about the 24/7 ambulance service.

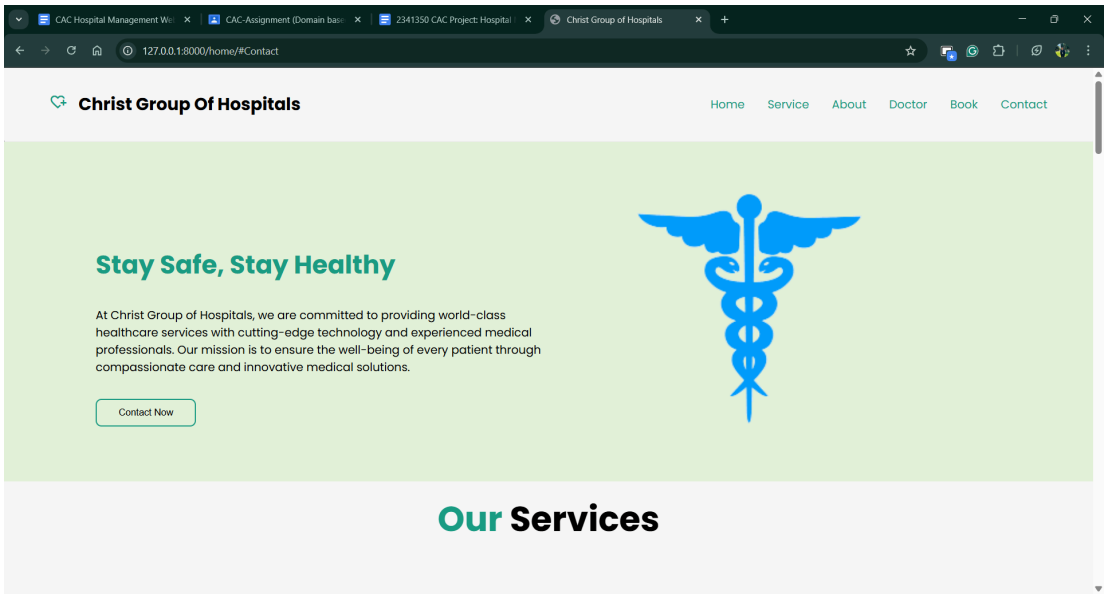
Page	Description
home.html	The homepage of the website, featuring an overview of the hospital and services.
cardiology.html	Details about cardiology services and specialists.
comprehensive_care.html	Overview of comprehensive healthcare services.
dental_care.html	Information about dental care services.
experts_consultancy.html	Details about expert consultancy services.
free_checkup.html	Information about free health checkup services.
pharmacy_medicine.html	Details about the pharmacy and medicine services.
physiotherapy.html	Information about physiotherapy services.

---

3. Pages and Their Functionality

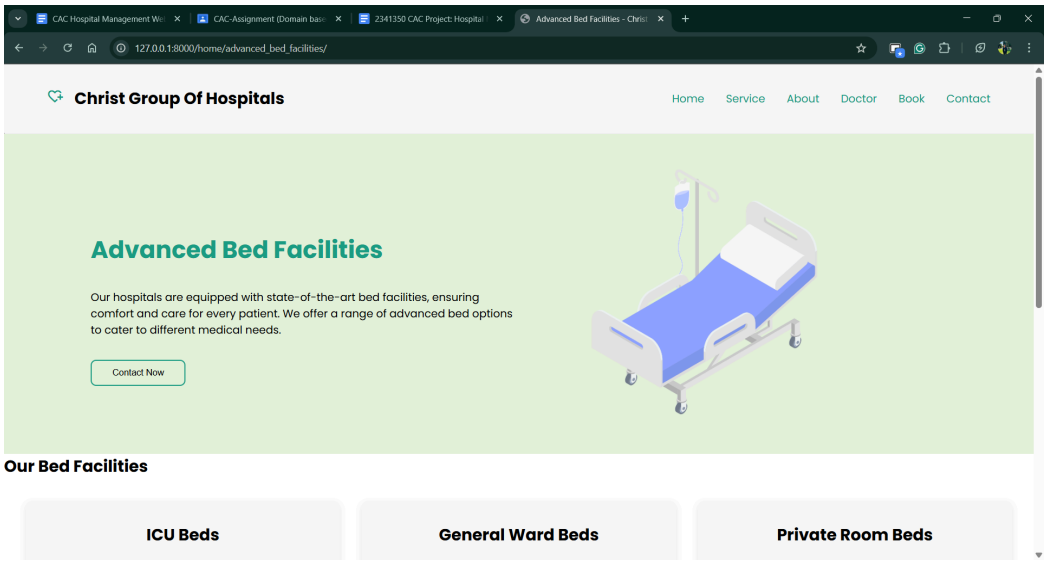
3.1. Home Page (home.html)

- **Purpose**: Serves as the landing page for the website.
- **Features**:
    - Welcome message and hospital overview.
    - Links to all other pages.
    - A form for booking appointments (connected to the Appointment model in the database).



3.2. Advanced Bed Facilities (advanced_bed_facilities.html)

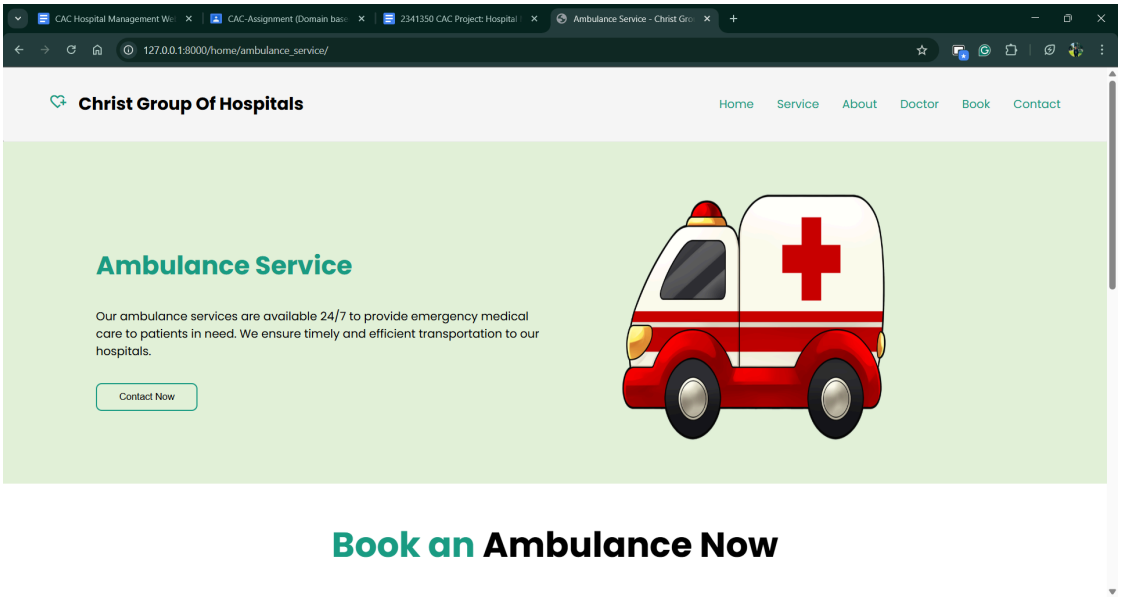
- **Purpose**: Provides details about the hospital's advanced bed facilities.
- **Features**:
    - Description of ICU beds, general ward beds, private room beds, maternity beds, and pediatric beds.



3.3. Ambulance Service (ambulance_service.html)

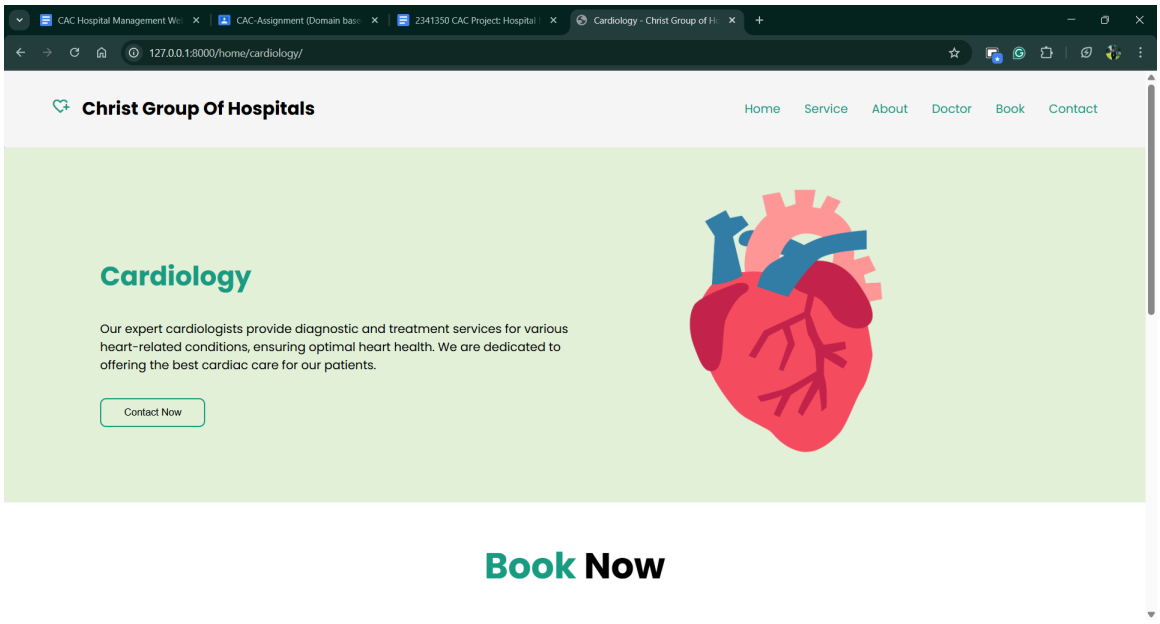
- **Purpose**: Highlights the 24/7 ambulance service.
- **Features**:

- Information about emergency response times and contact details.



3.4. Cardiology (cardiology.html)

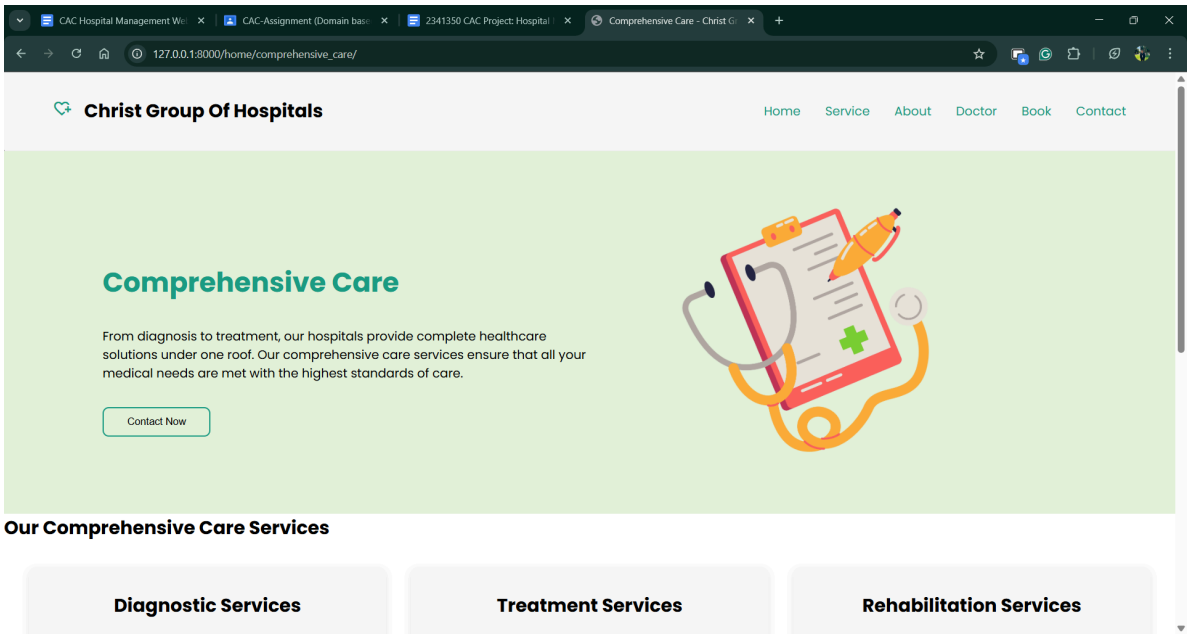
- **Purpose**: Showcases cardiology services.
- **Features**:
  - Details about heart-related treatments and specialists.



3.5. Comprehensive Care (comprehensive_care.html)

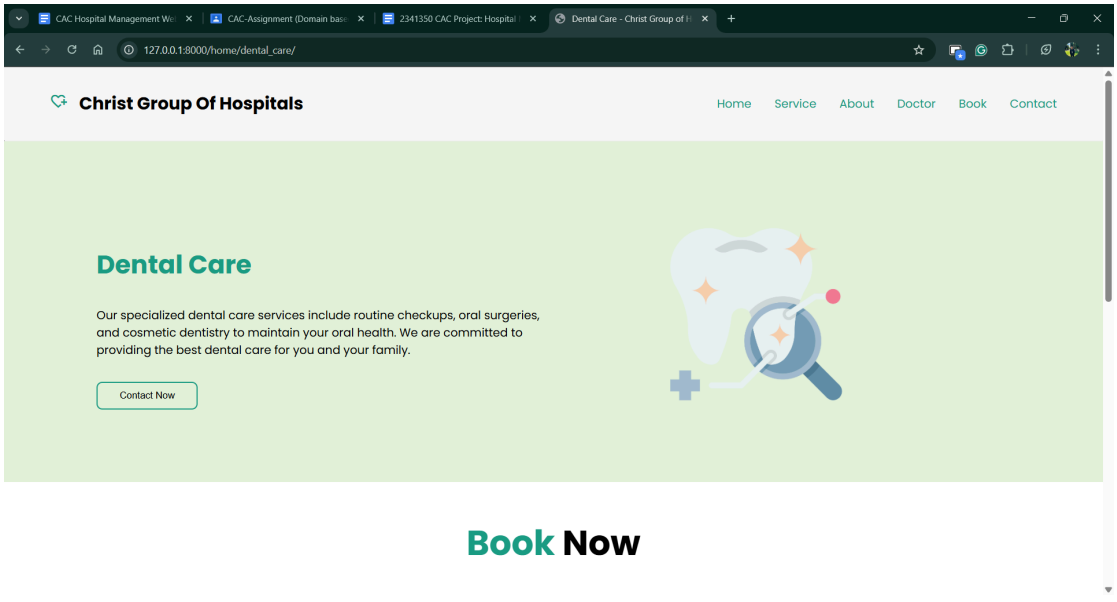
- **Purpose**: Describes comprehensive healthcare services.

- **Features**:
    - Overview of integrated care solutions.



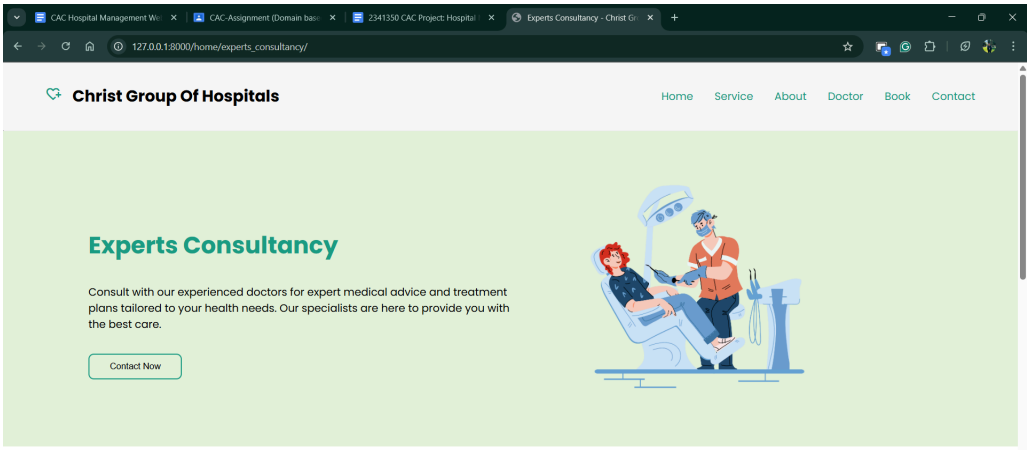
3.6. Dental Care (dental_care.html)

- **Purpose**: Provides information about dental care services.
- **Features**:
    - Details about routine checkups, oral surgeries, and cosmetic dentistry.



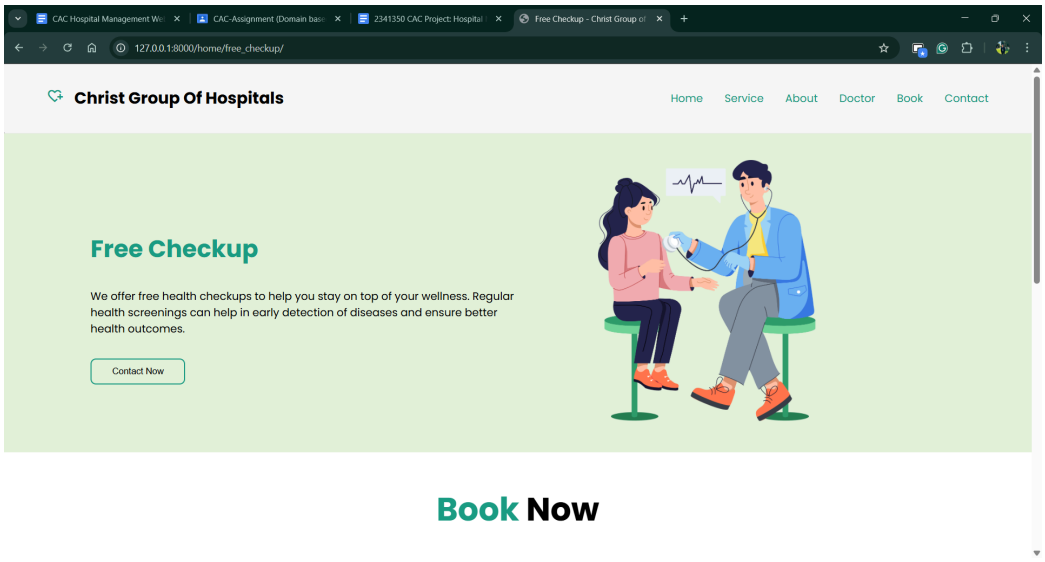
3.7. Experts Consultancy (experts_consultancy.html)

- **Purpose**: Highlights expert consultancy services.
- **Features**:
    - Information about specialized medical advice and treatment plans.



3.8. Free Checkup (free_checkup.html)

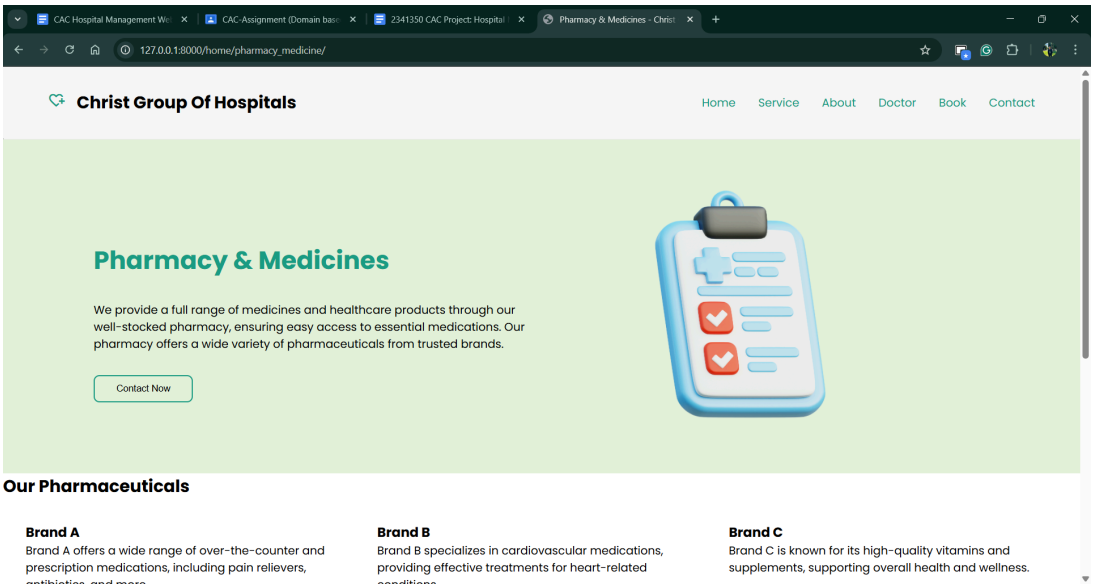
- **Purpose**: Promotes free health checkup services.
- **Features**:
    - Details about free screenings and early disease detection.



3.9. Pharmacy & Medicine (pharmacy_medicine.html)

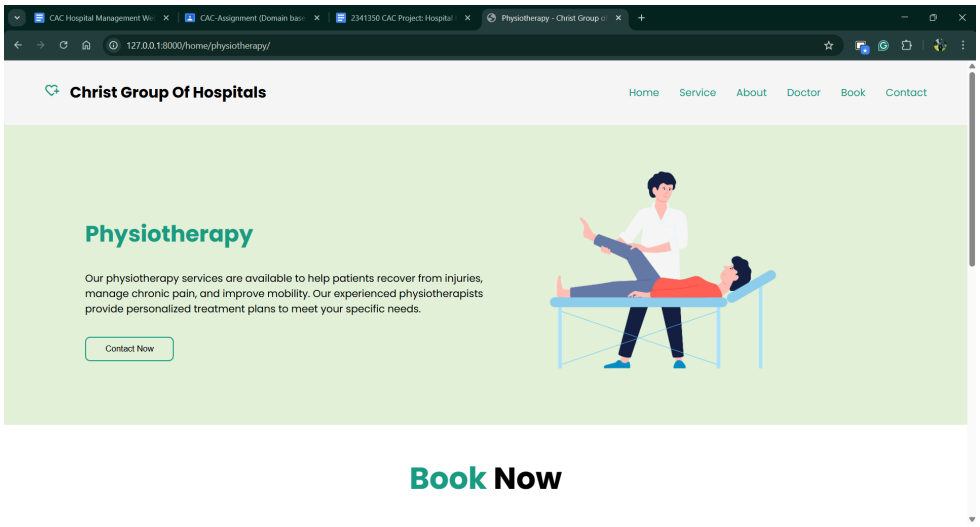
- **Purpose**: Provides information about the hospital pharmacy.

- **Features**:
    - Details about available medicines and healthcare products.



3.10. Physiotherapy (physiotherapy.html)

- **Purpose**: Describes physiotherapy services.
- **Features**:
    - Information about injury recovery, chronic pain management, and mobility improvement.



4. Styles Used in the Project

The Hospital Management Django Project uses a custom CSS file to style the web application, ensuring a visually appealing and user-friendly interface. The styles are designed to be responsive, consistent, and aligned with the hospital's branding. Below is an explanation of the key styles and design choices used in the project.

4.1. Global Styles

The global styles define the base layout, typography, and color scheme for the entire application.

```
:root {
  --clr-one: #f6f5f5;       /* Light background color */
  --clr-two: #e1f0da;       /* Secondary background color */
  --clr-font: #1b9c85;      /* Primary accent color (green) */
  --clr-text: #677483;      /* Text color */
}

* {
  padding: 0;
  margin: 0;
  box-sizing: border-box;   /* Ensures padding and border are
included in element dimensions */
}

body {
  font-family: "Poppins", sans-serif; /* Modern and clean font */
}

button, i {
  cursor: pointer;          /* Changes cursor to pointer for buttons
and icons */
}
```

- **Color Scheme**: The project uses a consistent color palette defined in the :root pseudo-class. The primary accent color (--clr-font) is a shade of green, which is commonly associated with health and wellness.
- **Typography**: The "Poppins" font is used for its modern and clean appearance, ensuring readability across all devices.

4.2. Navigation Bar

The navigation bar is fixed at the top of the page and provides easy access to all sections of the website.

```
.nav__bar {
  display: flex;
  justify-content: space-between;
  position: fixed;
  align-items: center;
  width: 100%;
  background: var(--clr-one); /* Light background */
  padding: 2rem 4rem;         /* Spacing around the content */
}

.nav__bar .logo span {
  color: var(--clr-font);     /* Accent color for the logo */
}

.nav__bar ul {
  display: flex;
  list-style: none;
  gap: 2rem;                  /* Spacing between navigation items */
}

ul li a {
  color: var(--clr-font);     /* Accent color for links */
  text-decoration: none;      /* Removes underline from links */
}
```

- **Fixed Positioning**: The navigation bar remains visible at the top of the page as the user scrolls.
- **Flexbox Layout**: Ensures the logo and navigation items are aligned and spaced evenly.

4.3. Home Page

The home page features a two-column layout with a welcoming message and an image.

```
.home {
```

```
  background: var(--clr-two); /* Secondary background color */
  display: grid;
  grid-template-columns: repeat(2, 1fr); /* Two-column layout */
  padding: 2rem 8rem;        /* Spacing around the content */
}

.home .home__first h2 {
  color: var(--clr-font);    /* Accent color for headings */
  font-size: 2.2em;          /* Large font size for emphasis */
}

.home .home__first button {
  border: 2px solid var(--clr-font); /* Accent color for button
border */
  border-radius: 0.5rem;     /* Rounded corners */
  background: transparent;   /* Transparent background */
}

.home .home__first button:hover {
  background: var(--clr-font); /* Accent color on hover */
  color: var(--clr-one);     /* Light text color on hover */
}
```

- **Grid Layout**: The two-column layout ensures a balanced design.
- **Interactive Buttons**: Buttons change color on hover to provide visual feedback.

4.4. Service Sections

The service sections are designed to highlight key features of the hospital.

```
.service {
  background: var(--clr-one); /* Light background */
  display: grid;
  grid-template-columns: repeat(4, 1fr); /* Four-column layout */
  padding: 8rem;             /* Spacing around the content */
  text-align: center;        /* Centered text */
}
```

```
.service .service__first {
  border: 2px solid var(--clr-font); /* Accent color for borders */
  border-radius: 0.3rem;     /* Rounded corners */
}

.service .service__first i {
  font-size: 3em;            /* Large icon size */
  color: var(--clr-font);    /* Accent color for icons */
}
```

- **Grid Layout**: The four-column layout ensures all services are displayed prominently.
- **Icons**: Large icons are used to visually represent each service.

4.5. About Page

The About page uses a grid layout to display information in a structured manner.

```
.about__page {
  background: var(--clr-two); /* Secondary background color */
  display: grid;
  grid-template-columns: repeat(3, 1fr); /* Three-column layout */
  padding: 8rem;             /* Spacing around the content */
}

.about__page .about__one {
  border: 2px solid var(--clr-font); /* Accent color for borders */
  padding-left: 2rem;        /* Spacing inside the box */
}

.about__page .about__one button {
  background: var(--clr-font); /* Accent color for buttons */
  color: var(--clr-one);     /* Light text color */
}
```

- **Structured Layout**: The three-column grid ensures information is easy to read and navigate.

- **Consistent Design**: Buttons and borders use the accent color for consistency.

4.6. Responsive Design

The project includes media queries to ensure the design adapts to different screen sizes.

```
@media (max-width: 830px) {
  .nav__bar .navbar {
    display: none;           /* Hides the navigation bar on smaller
screens */
  }

  .home {
    grid-template-columns: repeat(1, 1fr); /* Single-column layout
for mobile */
  }

  .service {
    grid-template-columns: repeat(2, 1fr); /* Two-column layout for
tablets */
  }
}

@media (max-width: 580px) {
  .service {
    grid-template-columns: repeat(1, 1fr); /* Single-column layout
for mobile */
  }
}
```

- **Mobile-First Approach**: The design prioritizes smaller screens, ensuring usability on all devices.
- **Flexible Grids**: The grid layout adjusts based on screen size to maintain readability.

4.7. Additional Features

- **Google Fonts:** The project uses the Poppins font from Google Fonts to ensure a modern and clean typography. The font is imported in the CSS file using the following line:

```
@import
url("https://fonts.googleapis.com/css2?family=Poppins:wght@40
0;500;600;700;800&display=swap");
```

  This font is applied globally to the body element, ensuring consistency across all pages.
- **Font Awesome Icons:** Icons used throughout the project, such as in service sections and navigation, are sourced from Font Awesome. These icons are lightweight, scalable, and easy to customize. For example:

```
<i class="fas fa-heartbeat"></i> <!-- Example of a Font
Awesome icon -->
```

- **Back-to-Top Button**: A hidden button appears when the user scrolls down, allowing them to quickly return to the top of the page.
- **Hover Effects**: Buttons and icons change color on hover to provide interactive feedback.
- **Box Shadows**: Subtle shadows are used to create depth and highlight elements like service boxes.

5. JavaScript Functionality

The Hospital Management Django Project includes JavaScript to enhance interactivity and improve the user experience. The JavaScript code handles navigation, form validation, smooth scrolling, and a back-to-top button. Below is a detailed explanation of the JavaScript functionality implemented in the project.

5.1. Navigation Menu Toggle

The navigation menu is designed to be responsive, collapsing into a toggleable menu on smaller screens. This functionality is achieved using JavaScript.

```
const menu = document.querySelector("#menu__btn");
const navbar = document.querySelector(".navbar");

menu.onclick = () => {
  menu.classList.toggle("fa-times"); // Toggles the menu icon (e.g., hamburger to close icon)
  navbar.classList.toggle("active"); // Toggles the visibility of the navigation menu
};

window.onscroll = () => {
  menu.classList.remove("fa-times"); // Resets the menu icon when scrolling
  navbar.classList.remove("active"); // Hides the navigation menu when scrolling
};
```

- **Menu Toggle**: When the menu button is clicked, it toggles between a hamburger icon and a close icon (fa-times from Font Awesome) and shows or hides the navigation menu.
- **Scroll Behavior**: The menu automatically collapses when the user scrolls, ensuring a clean and unobstructed view of the page content.

5.2. Smooth Scrolling

Smooth scrolling is implemented to provide a seamless navigation experience when users click on anchor links.

```
document.querySelectorAll('a[href^="#"]').forEach(anchor => {
  anchor.addEventListener("click", function (e) {
    e.preventDefault(); // Prevents the default jump behavior

document.querySelector(this.getAttribute("href")).scrollIntoView({
      behavior: "smooth" // Enables smooth scrolling
    });
  });
});
```

- **Anchor Links**: Smooth scrolling is applied to all anchor links (<a href="#section">) on the page.
- **User Experience**: This feature enhances the user experience by providing a smooth transition to the target section instead of an abrupt jump.

5.3. Form Validation

The appointment booking form includes client-side validation to ensure all required fields are filled before submission.

```
document.querySelector("form").addEventListener("submit", function
(e) {
  const name =
document.querySelector('input[placeholder="Name"]').value;
  const number =
document.querySelector('input[placeholder="Number"]').value;
  const email =
document.querySelector('input[placeholder="Email"]').value;
  const date = document.querySelector('input[type="date"]').value;

  if (!name || !number || !email || !date) {
    e.preventDefault(); // Prevents form submission if any field is
empty
    alert("Please fill in all fields.");
  } else {
    e.preventDefault(); // Prevents default form submission
    alert(`Appointment booked successfully!\n\nDetails:\nName:
${name}\nNumber: ${number}\nEmail: ${email}\nDate: ${date}`);
  }
});
```

- **Validation**: The script checks if all form fields (name, phone number, email, and date) are filled out. If any field is empty, it prevents form submission and displays an alert.
- **Success Message**: If all fields are valid, a success message is displayed with the submitted details.

5.4. Back-to-Top Button

A dynamic back-to-top button is created and displayed when the user scrolls down the page.

```
const backToTopButton = document.createElement("button");
backToTopButton.innerHTML = "↑";
backToTopButton.id = "backToTop";
backToTopButton.style.position = "fixed";
backToTopButton.style.bottom = "40px";
backToTopButton.style.right = "40px";
backToTopButton.style.padding = "20px";
backToTopButton.style.border = "none";
backToTopButton.style.borderRadius = "5px";
backToTopButton.style.backgroundColor = "#1b9c85";
backToTopButton.style.color = "#fff";
backToTopButton.style.cursor = "pointer";
backToTopButton.style.display = "none";
document.body.appendChild(backToTopButton);

backToTopButton.addEventListener("click", () => {
  window.scrollTo({
    top: 0,
    behavior: "smooth" // Smooth scroll to the top of the page
  });
});

window.addEventListener("scroll", () => {
  if (window.scrollY > 300) {
    backToTopButton.style.display = "block"; // Show the button when scrolled down
  } else {
    backToTopButton.style.display = "none"; // Hide the button when at the top
  }
});
```

- **Dynamic Button**: The button is created dynamically using JavaScript and styled to match the project's design.

- **Scroll Behavior**: The button appears when the user scrolls down more than 300 pixels and disappears when they return to the top.
- **Smooth Scroll**: Clicking the button smoothly scrolls the page back to the top.

6. Linking Pages in Django

6.1. URL Configuration

The URLs for the pages are defined in hospital_management/urls.py and hospital_app/urls.py. Each page is mapped to a corresponding view.

urls.py file

```
from django.contrib import admin
from django.urls import path
from hospital_app.views import home, book_appointment
from hospital_app.views import (
    home,
    advanced_bed_facilities,
    dental_care,
    massage_therapy,
    cardiology,
    diagnosis,
    ambulance_service,
    physiotherapy,
    free_checkup,
    experts_consultancy,
    pharmacy_medicine,
    comprehensive_care,
)

urlpatterns = [
    path('admin/', admin.site.urls),
    path('home/', home, name='home'),
    path('home/advanced_bed_facilities/', advanced_bed_facilities,
name='advanced_bed_facilities'),
    path('home/dental_care/', dental_care, name='dental_care'),
    path('home/massage_therapy/', massage_therapy,
```

```
name='massage_therapy'),
    path('home/cardiology/', cardiology, name='cardiology'),
    path('home/diagnosis/', diagnosis, name='diagnosis'),
    path('home/ambulance_service/', ambulance_service,
name='ambulance_service'),
    path('home/physiotherapy/', physiotherapy, name='physiotherapy'),
    path('home/free_checkup/', free_checkup, name='free_checkup'),
    path('home/experts_consultancy/', experts_consultancy,
name='experts_consultancy'),
    path('home/pharmacy_medicine/', pharmacy_medicine,
name='pharmacy_medicine'),
    path('home/comprehensive_care/', comprehensive_care,
name='comprehensive_care'),
    path('home/book-appointment/', book_appointment,
name='book_appointment'),
]
```

6.2. Views

Each page has a corresponding view in hospital_app/views.py that renders the appropriate template.

views.py file

```
from django.shortcuts import render
from .models import Appointment

def home(request):
    return render(request, 'hospital_app/home.html')

def advanced_bed_facilities(request):
    return render(request,
'hospital_app/advanced_bed_facilities.html')

def dental_care(request):
    return render(request, 'hospital_app/dental_care.html')

def massage_therapy(request):
    return render(request, 'hospital_app/massage_therapy.html')
```

```python
def cardiology(request):
    return render(request, 'hospital_app/cardiology.html')

def diagnosis(request):
    return render(request, 'hospital_app/diagnosis.html')

def ambulance_service(request):
    return render(request, 'hospital_app/ambulance_service.html')

def physiotherapy(request):
    return render(request, 'hospital_app/physiotherapy.html')

def free_checkup(request):
    return render(request, 'hospital_app/free_checkup.html')

def experts_consultancy(request):
    return render(request, 'hospital_app/experts_consultancy.html')

def pharmacy_medicine(request):
    return render(request, 'hospital_app/pharmacy_medicine.html')

def comprehensive_care(request):
    return render(request, 'hospital_app/comprehensive_care.html')

def book_appointment(request):
    if request.method == 'POST':
        name = request.POST.get('name')
        phone_number = request.POST.get('phone_number')
        email = request.POST.get('email')
        appointment_date = request.POST.get('appointment_date')

        # Save the data to the database
        Appointment.objects.create(
            name=name,
            phone_number=phone_number,
            email=email,
            appointment_date=appointment_date,
        )
        return redirect('home')  # Redirect to the homepage after
```

```
submission
    return render(request, 'hospital_app/home.html')
```

6.3. Template Links

The pages are linked together using Django's {% url %} template tag. For example, in home.html:

```
<a href="{% url 'advanced_bed_facilities' %}">Advanced Bed
Facilities</a>
<a href="{% url 'ambulance_service' %}">Ambulance Service</a>
<a href="{% url 'cardiology' %}">Cardiology</a>
```

7. Database Integration

The Hospital Management Django Project uses MySQL as the database backend to store data, such as patient appointments. This section explains how the database is integrated into the project, how the Appointment model is defined, and how data is added to the SQL table through form submissions.

7.1. Database Configuration

The database configuration is defined in the **settings.py** file of the Django project. MySQL is used as the database backend, and the connection details are specified as follows:

```
DATABASES = {
    'default': {
        'ENGINE': 'django.db.backends.mysql',
        'NAME': 'hospital_management',  # Database name
        'USER': 'root',                 # Database user
        'PASSWORD': 'password',         # Database password
        'HOST': 'localhost',            # Database host
        'PORT': '3306',                 # Database port
    }
}
```

- **ENGINE**: Specifies the database backend (MySQL in this case).
- **NAME**: The name of the database.
- **USER**: The username for accessing the database.
- **PASSWORD**: The password for the database user.
- **HOST**: The host where the database is running (e.g., localhost).
- **PORT**: The port on which the database server is listening (default is 3306 for MySQL).

7.2 Model Definition

The **Appointment** model is defined in **hospital_app/models.py**. This model represents the structure of the table in the MySQL database where appointment data is stored.

```
from django.db import models

class Appointment(models.Model):
    name = models.CharField(max_length=100)
    phone_number = models.CharField(max_length=15)
    email = models.EmailField()
    appointment_date = models.DateField()

    def __str__(self):
        return f"{self.name} - {self.appointment_date}"
```

- **Fields**:
    - name: Stores the patient's name (CharField with a maximum length of 100 characters).
    - phone_number: Stores the patient's phone number (CharField with a maximum length of 15 characters).
    - email: Stores the patient's email address (EmailField).
    - appointment_date: Stores the date of the appointment (DateField).
- **__str__ Method**: Provides a human-readable representation of the model instance, which is useful in the Django admin interface.

7.3. Form Submission and Data Insertion

The book_appointment view in hospital_app/views.py handles the form submission and inserts the data into the SQL table.

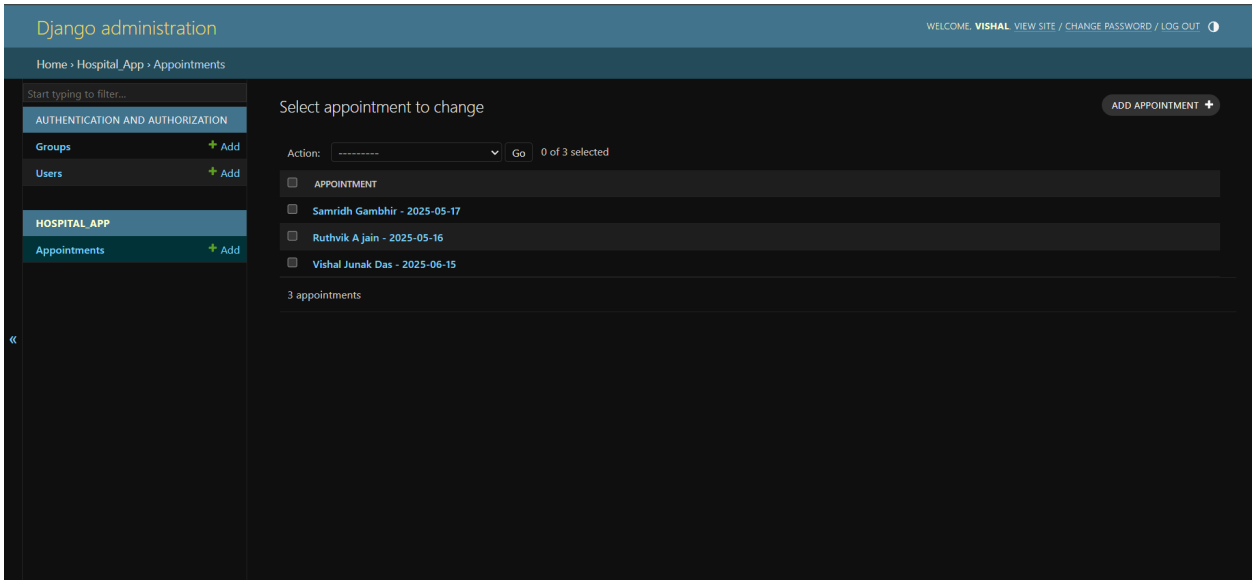
```
from django.shortcuts import render, redirect
from .models import Appointment

def book_appointment(request):
    if request.method == 'POST':
        # Retrieve form data from the POST request
        name = request.POST.get('name')
        phone_number = request.POST.get('phone_number')
        email = request.POST.get('email')
        appointment_date = request.POST.get('appointment_date')

        # Create a new Appointment instance and save it to the
database
        Appointment.objects.create(
            name=name,
            phone_number=phone_number,
            email=email,
            appointment_date=appointment_date,
        )
        return redirect('home')  # Redirect to the homepage after
submission
    return render(request, 'hospital_app/home.html')
```

- **Process**:
    1. When the user submits the appointment form on the homepage (home.html), the form data is sent to the book_appointment view via a POST request.
    2. The view retrieves the form data using request.POST.get().
    3. A new instance of the Appointment model is created using the Appointment.objects.create() method, and the data is saved to the database.
    4. After saving the data, the user is redirected to the homepage using redirect('home').

7.4. Verifying Data Insertion

To verify that the data has been successfully added to the SQL table, we use:

- Access the Django admin interface at `http://127.0.0.1:8000/admin/`.
- Log in with your superuser credentials.
- Navigate to the `Appointment` model to view the newly added data.



8. Conclusion

The Hospital Management Django Project is a fully functional web application that provides information about hospital services and allows patients to book appointments. The project demonstrates the use of Django's MVC architecture, MySQL for data storage, and Django templates for rendering dynamic content. All pages are seamlessly linked using Django's URL routing system, ensuring a smooth user experience.

9. Future Enhancements

- Add user authentication for patients and staff.
- Implement a dashboard for managing appointments and patient records.
- Integrate payment gateways for online payments.
- Add more interactive features, such as live chat support.