MINI HOSPITAL MANAGEMENT SYSTEM

1. DJANGO BACKEND SETUP

python -m venv venv

source venv/bin/activate

pip install django psycopg2-binary google-auth google-auth-oauthlib google-auth-httplib2 google-api-python-client requests python-decouple

django-admin startproject hms .

python manage.py startapp accounts

python manage.py startapp appointments

settings.py

```
import os
from pathlib import Path
from decouple import config

BASE_DIR = Path(__file__).resolve().parent.parent

SECRET_KEY = config('SECRET_KEY', default='your-secret-key-here')
DEBUG = True
ALLOWED_HOSTS = ['localhost', '127.0.0.1']

INSTALLED_APPS = [
    'django.contrib.admin',
```

```
    'django.contrib.auth',
    'django.contrib.contenttypes',
    'django.contrib.sessions',
    'django.contrib.messages',
    'django.contrib.staticfiles',
    'accounts',
    'appointments',
]

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 = 'hms.urls'

TEMPLATES = [
    {
        'BACKEND': 'django.template.backends.django.DjangoTemplates',
        'DIRS': [BASE_DIR / 'templates'],
        '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',
            ],
        },
    },
]

DATABASES = {
    'default': {
        'ENGINE': 'django.db.backends.postgresql',
        'NAME': config('DB_NAME', default='hms_db'),
        'USER': config('DB_USER', default='postgres'),
        'PASSWORD': config('DB_PASSWORD', default='password'),
        'HOST': config('DB_HOST', default='localhost'),
        'PORT': config('DB_PORT', default='5432'),
    }
}

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'},
]

LANGUAGE_CODE = 'en-us'
```

```
TIME_ZONE = 'UTC'

USE_I18N = True
USE_TZ = True

STATIC_URL = 'static/'
STATICFILES_DIRS = [BASE_DIR / 'static']
DEFAULT_AUTO_FIELD = 'django.db.models.BigAutoField'

LOGIN_URL = '/accounts/login/'
LOGIN_REDIRECT_URL = '/accounts/dashboard/'

EMAIL_SERVICE_URL = config('EMAIL_SERVICE_URL',
default='http://localhost:3000/dev/send-email')
GOOGLE_CALENDAR_CREDENTIALS_PATH = BASE_DIR / 'google_credentials.json'
```

MODELS

```
from django.db import models
from django.contrib.auth.models import User
from django.utils import timezone

class UserProfile(models.Model):
    ROLE_CHOICES = [
        ('doctor', 'Doctor'),
        ('patient', 'Patient'),
    ]
```

```
    user = models.OneToOneField(User, on_delete=models.CASCADE,
related_name='profile')
    role = models.CharField(max_length=10, choices=ROLE_CHOICES)
    phone = models.CharField(max_length=15, blank=True)
    specialization = models.CharField(max_length=100, blank=True)
    google_calendar_token = models.TextField(blank=True)

    def __str__(self):
        return f"{self.user.username} - {self.role}"

    class Meta:
        db_table = 'user_profiles'


class DoctorAvailability(models.Model):
    doctor = models.ForeignKey(User, on_delete=models.CASCADE,
related_name='availabilities')
    date = models.DateField()
    start_time = models.TimeField()
    end_time = models.TimeField()
    is_booked = models.BooleanField(default=False)
    created_at = models.DateTimeField(auto_now_add=True)

    class Meta:
        db_table = 'doctor_availabilities'
        ordering = ['date', 'start_time']
        unique_together = ['doctor', 'date', 'start_time']

    def __str__(self):
```

```
        return f"Dr. {self.doctor.username} - {self.date} {self.start_time}-{self.end_time}"

    def is_available(self):
        now = timezone.now()
        slot_datetime = timezone.make_aware(
            timezone.datetime.combine(self.date, self.start_time)
        )
        return not self.is_booked and slot_datetime > now



class Appointment(models.Model):
    STATUS_CHOICES = [
        ('pending', 'Pending'),
        ('confirmed', 'Confirmed'),
        ('cancelled', 'Cancelled'),
        ('completed', 'Completed'),
    ]

    patient = models.ForeignKey(User, on_delete=models.CASCADE,
related_name='patient_appointments')
    doctor = models.ForeignKey(User, on_delete=models.CASCADE,
related_name='doctor_appointments')
    availability = models.OneToOneField(DoctorAvailability,
on_delete=models.CASCADE, related_name='appointment')
    status = models.CharField(max_length=20, choices=STATUS_CHOICES,
default='pending')
    notes = models.TextField(blank=True)
    google_calendar_event_id = models.CharField(max_length=255, blank=True)
    created_at = models.DateTimeField(auto_now_add=True)
```

```
    updated_at = models.DateTimeField(auto_now=True)

    class Meta:
        db_table = 'appointments'
        ordering = ['-created_at']

    def __str__(self):
        return f"{self.patient.username} with Dr. {self.doctor.username} on
{self.availability.date}"
```

FORMS

```
from django import forms
from django.contrib.auth.models import User
from django.contrib.auth.forms import UserCreationForm
from .models import UserProfile, DoctorAvailability, Appointment
from django.utils import timezone

class SignUpForm(UserCreationForm):
    email = forms.EmailField(required=True)
    role = forms.ChoiceField(choices=UserProfile.ROLE_CHOICES, required=True)
    phone = forms.CharField(max_length=15, required=False)
    specialization = forms.CharField(max_length=100, required=False)

    class Meta:
        model = User
        fields = ('username', 'email', 'password1', 'password2', 'role', 'phone', 'specialization')

    def save(self, commit=True):
```

```
        user = super().save(commit=False)
        user.email = self.cleaned_data['email']
        if commit:
            user.save()
            UserProfile.objects.create(
                user=user,
                role=self.cleaned_data['role'],
                phone=self.cleaned_data.get('phone', ''),
                specialization=self.cleaned_data.get('specialization', '')
            )
        return user


class AvailabilityForm(forms.ModelForm):
    class Meta:
        model = DoctorAvailability
        fields = ['date', 'start_time', 'end_time']
        widgets = {
            'date': forms.DateInput(attrs={'type': 'date', 'class': 'form-control'}),
            'start_time': forms.TimeInput(attrs={'type': 'time', 'class': 'form-control'}),
            'end_time': forms.TimeInput(attrs={'type': 'time', 'class': 'form-control'}),
        }

    def clean(self):
        cleaned_data = super().clean()
        date = cleaned_data.get('date')
        start_time = cleaned_data.get('start_time')
        end_time = cleaned_data.get('end_time')
```

```
        if date and date < timezone.now().date():
            raise forms.ValidationError("Cannot create availability for past dates")
        if start_time and end_time and start_time >= end_time:
            raise forms.ValidationError("End time must be after start time")
        return cleaned_data


class BookingForm(forms.ModelForm):
    class Meta:
        model = Appointment
        fields = ['notes']
        widgets = {
            'notes': forms.Textarea(attrs={'rows': 3, 'class': 'form-control'})
        }
```

VIEWS

[VIEWS BLOCK]

(doctor dashboard, patient dashboard, booking logic, delete availability etc.)

GOOGLE CALENDAR (STUB)

[calendar integration stub]

URLS

[urls.py content]

SERVERLESS EMAIL SERVICE

[YAML + email handlers + SMTP]

TEMPLATES (HTML)

[base.html, login.html, signup.html, dashboards, forms, etc.]

CSS

```
body { background-color: #f8f9fa; }
h2 { margin-bottom: 20px; }
ul { list-style-type: none; padding: 0; }
li { margin-bottom: 10px; }
.btn { min-width: 120px; }
```