How to create a Disease Prediction Application using React, Django and PostgreSQL

This application will be built using `ReactJs` & `TailwindCSS` on the frontend and `Django` `PostgreSQL` on the backend. We will using `SVM` Machine Learning Model with a Dataset to predict diseases

Setting up the Environment

The root directory will be the `server` directory which would contain the `frontend` directory.

To get started, open up your terminal and paste the following command

```
django-admin startproject my-project
```

This will setup your directory like the below tree

```
my-project/
├──my-project/
├──manage.py
```

Rename the inner `my-project` directory to `Backend`. This will serve as the root for our backend application.

Now we will build the Accounts Directory and Disease Predictor directory. Open your terminal and run the following commands

```
python3 manage.py startapp Accounts
```

```
python3 manage.py startapp DiseasePredictor
```

Create `.gitignore , mode.pkl and requirements.txt` file in the root like the below tree.

```
Application/
├── Accounts/
├── Backend/
├── DiseasePredictor/
├── frontend/ ( will create it using vite)
├── .gitignore
├── manage.py
├── model.pkl
├── readme.md
└── requirements.txt
```

`.gitingore`

```
# Byte-compiled / optimized / DLL files
__pycache__/
*.py[cod]
*$py.class

# C extensions
*.so

# Distribution / packaging
.Python
build/
develop-eggs/
dist/
downloads/
eggs/
.eggs/
lib/
lib64/
parts/
sdist/
var/
wheels/
share/python-wheels/
*.egg-info/
.installed.cfg
*.egg
MANIFEST

# PyInstaller
#  Usually these files are written by a python script from a ter
#  before PyInstaller builds the exe, so as to inject date/other
*.manifest
*.spec

# Installer logs
pip-log.txt
pip-delete-this-directory.txt

# Unit test / coverage reports
htmlcov/
.tox/
.nox/
.coverage
.coverage.*
.cache
nosetests.xml
coverage.xml
*.cover
*.py,cover
.hypothesis/
.pytest_cache/
cover/

# Translations
*.mo
*.pot

# Django stuff:
*.log
local_settings.py
db.sqlite3
db.sqlite3-journal

# Flask stuff:
instance/
```

```
.webassets-cache

# Scrapy stuff:
.scrapy

# Sphinx documentation
docs/_build/

# PyBuilder
.pybuilder/
target/

# Jupyter Notebook
.ipynb_checkpoints

# IPython
profile_default/
ipython_config.py

# pyenv
#   For a library or package, you might want to ignore these fil
#   intended to run in multiple environments; otherwise, check t
# .python-version

# pipenv
#   According to pypa/pipenv#598, it is recommended to include P
#   However, in case of collaboration, if having platform-specif
#   having no cross-platform support, pipenv may install depende
#   install all needed dependencies.
#Pipfile.lock

# poetry
#   Similar to Pipfile.lock, it is generally recommended to incl
#   This is especially recommended for binary packages to ensure
#   commonly ignored for libraries.
#   https://python-poetry.org/docs/basic-usage/#commit-your-poet
```

```
#poetry.lock

# pdm
#   Similar to Pipfile.lock, it is generally recommended to incl
#pdm.lock
#   pdm stores project-wide configurations in .pdm.toml, but it
#   in version control.
#   https://pdm.fming.dev/#use-with-ide
.pdm.toml

# PEP 582; used by e.g. github.com/David-OConnor/pyflow and gith
__pypackages__/

# Celery stuff
celerybeat-schedule
celerybeat.pid

# SageMath parsed files
*.sage.py

# Environments
.env
.venv
env/
venv/
ENV/
env.bak/
venv.bak/

# Spyder project settings
.spyderproject
.spyproject

# Rope project settings
.ropeproject
```

```
# mkdocs documentation
/site

# mypy
.mypy_cache/
.dmypy.json
dmypy.json

# Pyre type checker
.pyre/

# pytype static type analyzer
.pytype/

# Cython debug symbols
cython_debug/

# PyCharm
#  JetBrains specific template is maintained in a separate JetBr
#  be found at https://github.com/github/gitignore/blob/main/Glo
#  and can be added to the global gitignore or merged into this
#  option (not recommended) you can uncomment the following to i
#.idea/
# Ignore Django Migrations in Development if you are working on

# Only for Development only
**/migrations/**
!**/migrations
!**/migrations/__init__.py
```

`requirements.txt`

```
django
django-cors-headers
```

```
djangorestframework
numpy
scikit-learn
python-decouple
pandas
django-pandas
imbalanced-learn
scikit-learn
psycopg2-binary
python-dotenv
pickle5
```

`model.pkl` *file for Disease Prediction Model*

model.pkl

Setting up the Frontend of the App

Execute the command, open a new terminal and follow these instructions:

- Enter "frontend" when prompted for the Project name
- Select React as the framework and JavaScript as the variant

```
npm create vite@latest

✔ Project name: … frontend
✔ Select a framework: › React
✔ Select a variant: › JavaScript
```

This will create the frontend directory in your root `frontend` folder. Run the following commands after it.

```
cd application
npm install
npm run dev
```

This will install the initial dependencies and start the development server of Vite running on `localhost:5173`. Open your browser and write the `localhost` URL to see the development server.

Setting up the Frontend Directory

You can also setup the components directory outside the assets directory if you prefer that way.

```
frontend/
├── src/
│   ├── assets/
│     ├── components/
│     └── img/
│   ├── App.css
│   ├── App.jsx
│   ├── index.css
│   └── main.jsx
│
├── public/
├── .gitignore
├── index.html
├── package-lock.json
├── package.json
├── postcss.config.js
├── tailwind.config.js
└── vite.config.js
```

Installing the Dependencies

```
npm install @emotion/react @emotion/styled @mui/material @tansta

npm install --save-dev @types/react @types/react-dom @vitejs/plu
```

> If you get any versioning error make sure your version match to the below `package.json`

```
{
  "name": "getogether-frontend",
  "private": true,
  "version": "0.0.0",
  "type": "module",
  "scripts": {
    "dev": "vite --host",
    "build": "tsc && vite build",
    "lint": "eslint . --ext ts,tsx --report-unused-disable-direc
    "preview": "vite preview",
    "format": "prettier --write ."
  },
  "dependencies": {
    "@emailjs/browser": "^4.3.3",
    "@emotion/react": "^11.11.3",
    "@emotion/styled": "^11.11.0",
    "@mui/icons-material": "^5.15.4",
    "@mui/material": "^5.15.4",
    "@mui/styles": "^5.15.18",
    "@mui/x-data-grid": "^7.11.0",
    "@mui/x-date-pickers": "^7.5.0",
    "@radix-ui/react-accordion": "^1.1.2",
    "@radix-ui/react-avatar": "^1.0.4",
    "@radix-ui/react-checkbox": "^1.0.4",
    "@radix-ui/react-dialog": "^1.0.5",
    "@radix-ui/react-dropdown-menu": "^2.0.6",
    "@radix-ui/react-icons": "^1.3.0",
    "@radix-ui/react-label": "^2.0.2",
    "@radix-ui/react-popover": "^1.0.7",
    "@radix-ui/react-select": "^2.0.0",
    "@radix-ui/react-separator": "^1.0.3",
    "@radix-ui/react-slot": "^1.0.2",
    "@radix-ui/react-tabs": "^1.0.4",
    "@reduxjs/toolkit": "^2.0.1",
    "axios": "^1.6.7",
    "chart.js": "^4.4.3",
    "class-variance-authority": "^0.7.0",
    "clsx": "^2.1.1",
    "date-fns": "^3.6.0",
    "file-saver": "^2.0.5",
    "lodash": "^4.17.21",
    "lucide-react": "^0.379.0",
    "mui-one-time-password-input": "^2.0.2",
    "react": "^17.0.0 || ^18.0.0",
    "react-chartjs-2": "^5.2.0",
    "react-confirm": "^0.3.0-7",
    "react-date-range": "^2.0.1",
    "react-day-picker": "^8.10.1",
    "react-dom": "^17.0.0 || ^18.0.0",
    "react-hot-toast": "^2.4.1",
    "react-icons": "^5.0.1",
    "react-redux": "^9.1.0",
    "react-router-dom": "^6.21.2",
    "react-spreadsheet-import": "^4.6.1",
    "recharts": "^2.12.7",
    "socket.io-client": "^4.7.5",
    "tailwind-merge": "^2.3.0",
    "tailwindcss-animate": "^1.0.7",
    "vaul": "^0.9.1"
  },
  "devDependencies": {
    "@types/file-saver": "^2.0.7",
    "@types/node": "^20.12.12",
    "@types/react": "^18.2.43",
    "@types/react-dom": "^18.2.17",
    "@typescript-eslint/eslint-plugin": "^6.14.0",
    "@typescript-eslint/parser": "^6.14.0",
    "@vitejs/plugin-react": "^4.2.1",
    "autoprefixer": "^10.4.16",
    "eslint": "^8.55.0",
    "eslint-plugin-react-hooks": "^4.6.0",
    "eslint-plugin-react-refresh": "^0.4.5",
    "postcss": "^8.4.33",
    "prettier": "^3.2.5",
    "tailwindcss": "^3.4.1",
    "typescript": "^5.2.2",
    "vite": "^5.0.8"
  },
  "peerDependencies": {
    "react": "^17.0.0 || ^18.0.0",
    "react-dom": "^17.0.0 || ^18.0.0"
  },
  "prettier": {
    "semi": false,
    "singleQuote": false,
    "trailingComma": "all",
    "jsxSingleQuote": false,
    "tabWidth": 2
  }
}
```

Building the Server of the App

Installing Dependencies

We will now start with the server of the application, run the following command in your terminal to install the dependencies

```
pip install django django-cors-headers djangorestframework numpy
```

Setting up Backend Directory

This is the root of the server part. Setup the `Backend` directory according to the following component tree

```
Backend/
├── __init__.py
├── asgi.py
├── settings.py
├── urls.py
├── views.py
└── wsgi.py
```

`__init__.py, asgi.py` and `wsgi.py` are set by default and doesn’t needs to be changed. We will start with the other files.

Backend Components

Paste the following code inside the mentioned file

`settings.py` - Index file for server

```
"""
Django settings for Backend project.

Generated by 'django-admin startproject' using Django 4.1.1.
```

```
For more information on this file, see
https://docs.djangoproject.com/en/4.1/topics/settings/

For the full list of settings and their values, see
https://docs.djangoproject.com/en/4.1/ref/settings/
"""

from pathlib import Path
from decouple import config
from dotenv import load_dotenv
import os

# 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/4.1/howto/deployment/che

# SECURITY WARNING: keep the secret key used in production secre
SECRET_KEY = 'django-insecure-_7_br&7lg2-ndm@osr=x$0#gvs=j^29+i
# SECURITY WARNING: don't run with debug turned on in productio
DEBUG = True

ALLOWED_HOSTS = ['*']

CSRF_COOKIE_NAME = 'csrftoken'

CSRF_TRUSTED_ORIGINS = ['http://127.0.0.1:5173',]

CORS_ALLOW_ALL_ORIGINS = True

CORS_ALLOW_CREDENTIALS = True

script_dir = os.path.dirname(os.path.abspath(__file__))
```

```
# Get the absolute path of the root directory
root_dir = os.path.abspath(os.path.join(script_dir, '..'))

# Construct the absolute file path of .env
env_file_path = os.path.join(root_dir, '.env')

# Load environment variables from .env file
load_dotenv(env_file_path)

# Access environment variables
DATABASE_NAME = os.getenv('DATABASE_NAME')
USER = os.getenv('USER')
PASS = os.getenv('PASS')


SECURE_CROSS_ORIGIN_OPENER_POLICY = "same-origin-allow-popups"


# Application definition

INSTALLED_APPS = [
    'django.contrib.admin',
    'django.contrib.auth',
    'django.contrib.contenttypes',
    'django.contrib.sessions',
    'django.contrib.messages',
    'django.contrib.staticfiles',
    'rest_framework',
    'corsheaders',
    'Accounts',
    'DiseasePredictor',
]

MIDDLEWARE = [
    'corsheaders.middleware.CorsMiddleware',
    '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 = 'Backend.urls'

import os

TEMPLATES = [
    {
        'BACKEND': 'django.template.backends.django.DjangoTempla
        'DIRS': [os.path.join(BASE_DIR, 'frontend/dist')],
        '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.mess
            ],
        },
    },
]

WSGI_APPLICATION = 'Backend.wsgi.application'


# Database
# https://docs.djangoproject.com/en/4.1/ref/settings/#databases

DATABASES = {
    'default': {
        'ENGINE': 'django.db.backends.postgresql',
        'NAME': DATABASE_NAME,
        'USER': USER,
        'PASSWORD': PASS,
        'HOST': 'localhost',
        'PORT': 5432,
    }
}

AUTH_USER_MODEL = 'Accounts.AppUser'

REST_FRAMEWORK = {
    'DEFAULT_PERMISSION_CLASSES': (
        'rest_framework.permissions.IsAuthenticated',
    ),
    'DEFAULT_AUTHENTICATION_CLASSES': (
        'rest_framework.authentication.SessionAuthentication',
    ),
}


# Password validation
# https://docs.djangoproject.com/en/4.1/ref/settings/#auth-passw

AUTH_PASSWORD_VALIDATORS = [
    {
        'NAME': 'django.contrib.auth.password_validation.UserAtt
    },
    {
        'NAME': 'django.contrib.auth.password_validation.Minimum
    },
    {
        'NAME': 'django.contrib.auth.password_validation.CommonP
    },
    {
        'NAME': 'django.contrib.auth.password_validation.Numeric
    },
]


# Internationalization
# https://docs.djangoproject.com/en/4.1/topics/i18n/

LANGUAGE_CODE = 'en-us'

TIME_ZONE = 'UTC'

USE_I18N = True

USE_TZ = True


# Static files (CSS, JavaScript, Images)
# https://docs.djangoproject.com/en/4.1/howto/static-files/

STATIC_URL = 'assets/'
STATICFILES_DIRS = [
    os.path.join(BASE_DIR, 'frontend/dist/assets')
]

# Default primary key field type
# https://docs.djangoproject.com/en/4.1/ref/settings/#default-au

DEFAULT_AUTO_FIELD = 'django.db.models.BigAutoField'
```

`urls.py` - Routes for the server

```
from django.contrib import admin
from django.urls import path, include, re_path
from .views import index, load_icon
from django.views.generic.base import RedirectView
from django.contrib.staticfiles.storage import staticfiles_stor

urlpatterns = [
    path('admin/', admin.site.urls),
    path('', index),
    path("", include("Accounts.urls")),
    path("", include("DiseasePredictor.urls")),
    path("contactdoctor", index),
    path("dashboard", index),
    path('icon.svg', load_icon),
]
```

`views.py`

```
from django.shortcuts import render
from django.http import FileResponse
from django.conf import settings
import os

def index(request):
    return render(request, 'index.html')


def load_icon(request):
    icon_path = os.path.join(settings.BASE_DIR, 'frontend/dist/
    return FileResponse(open(icon_path, 'rb'), content_type='ima
```

Accounts Directory

Setup the accounts directory as the below ASCII tree and paste the code mentioned along the following file.

```
Accounts/
├── migrations/
├── __init__.py
├── admin.py
├── apps.py
├── models.py
├── serializers.py
├── tests.py
├── urls.py
├── validations.py
└── views.py
```

`admin.py`

```
from django.contrib import admin
from .models import AppUser, PatientProfile, DoctorProfile, symp

admin.site.register(AppUser)
admin.site.register(PatientProfile)
admin.site.register(DoctorProfile)
admin.site.register(symptoms_diseases)

# Register your models here.
```

`models.py`

```
from django.db import models
from django.contrib.auth.base_user import BaseUserManager
from django.contrib.auth.models import AbstractBaseUser, Permiss
from django.contrib.postgres.fields import ArrayField
from django.db.models import JSONField
```

```python
class AppUserManager(BaseUserManager):
    def create_user(self, email, password=None):
        if not email:
            raise ValueError('An email is required.')
        if not password:
            raise ValueError('A password is required.')
        email = self.normalize_email(email)
        user = self.model(email=email)
        user.set_password(password)
        user.save()
        # import PatientProfile here to avoid circular import
        from .models import PatientProfile
        profile = PatientProfile.objects.create(user=user)

        return user

    def create_superuser(self, email, password=None):
        if not email:
            raise ValueError('An email is required.')
        if not password:
            raise ValueError('A password is required.')
        user = self.create_user(email, password)
        user.is_staff = True
        user.is_superuser = True
        user.save()
        return user


class AppUser(AbstractBaseUser, PermissionsMixin):
    user_id = models.AutoField(primary_key=True)
    email = models.EmailField(max_length=50, unique=True)
    username = models.CharField(max_length=50)
    is_staff = models.BooleanField(default=False)
    is_superuser = models.BooleanField(default=False)
    USERNAME_FIELD = 'email'
    REQUIRED_FIELDS = []
    objects = AppUserManager()

    def __str__(self):
        return self.username



class PatientProfile(models.Model):
    user = models.OneToOneField(AppUser, on_delete=models.CASCAD
    age = models.IntegerField(default=0)
    sex = models.CharField(max_length=20, default='Not to say')
    first_name = models.CharField(max_length=20, default='a')
    last_name = models.CharField(max_length=20, default='a')
    medical_history = ArrayField(models.CharField(max_length=20
    dob_day = models.IntegerField(default=0)
    dob_month = models.IntegerField(default=0)
    dob_year = models.IntegerField(default=0)
    height = models.IntegerField(default=0)
    weight = models.IntegerField(default=0)
    current_med = ArrayField(models.CharField(max_length=200), 
    exercise = models.CharField(max_length=200, default='no exe
    diet = models.CharField(max_length=200, default='no diet')
    smoke_cons  = models.CharField(max_length=200, default='no 
    alcohol_cons = models.CharField(max_length=200, default='no
    bp_log = JSONField(default=dict)
    blood_glucose = JSONField(default=dict)
    new_patient = models.BooleanField(default=True)

class DoctorProfile(models.Model):
    name = models.CharField(max_length=200, default='NA')
    speciality = models.CharField(max_length=200, default='NA')
    sex = models.CharField(max_length=200, default='NA')
    experience = models.IntegerField(default=0)
    work_address = models.CharField(max_length=200, default='NA
    mobile_no = models.CharField(max_length=200, default='00000
    image_link = models.URLField(max_length=200)
    profile_link = models.URLField(max_length=200)

class symptoms_diseases(models.Model):
    itching = models.IntegerField()
    skin_rash = models.IntegerField()
    shivering = models.IntegerField()
    chills = models.IntegerField()
    joint_pain = models.IntegerField()
    stomach_pain = models.IntegerField()
    acidity = models.IntegerField()
    ulcers_on_tongue = models.IntegerField()
    muscle_wasting = models.IntegerField()
    vomiting = models.IntegerField()
    burning_micturition = models.IntegerField()
    spotting_urination = models.IntegerField()
    fatigue = models.IntegerField()
    weight_gain = models.IntegerField()
    anxiety = models.IntegerField()
    cold_hands_and_feets = models.IntegerField()
    mood_swings = models.IntegerField()
    weight_loss = models.IntegerField()
    restlessness = models.IntegerField()
    lethargy = models.IntegerField()
    patches_in_throat = models.IntegerField()
    irregular_sugar_level = models.IntegerField()
    cough = models.IntegerField()
    high_fever = models.IntegerField()
    sunken_eyes = models.IntegerField()
    breathlessness = models.IntegerField()
    sweating = models.IntegerField()
    dehydration = models.IntegerField()
    indigestion = models.IntegerField()
    headache = models.IntegerField()
    yellowish_skin = models.IntegerField()
    dark_urine = models.IntegerField()
    nausea = models.IntegerField()
    loss_of_appetite = models.IntegerField()
    pain_behind_the_eyes = models.IntegerField()
    back_pain = models.IntegerField()
    constipation = models.IntegerField()
    abdominal_pain = models.IntegerField()
    diarrhoea = models.IntegerField()
    mild_fever = models.IntegerField()
    yellow_urine = models.IntegerField()
    yellowing_of_eyes = models.IntegerField()
    acute_liver_failure = models.IntegerField()
    fluid_overload = models.IntegerField()
    swelling_of_stomach = models.IntegerField()
    swelled_lymph_nodes = models.IntegerField()
    malaise = models.IntegerField()
    blurred_and_distorted_vision = models.IntegerField()
    phlegm = models.IntegerField()
    throat_irritation = models.IntegerField()
    redness_of_eyes  = models.IntegerField()
    sinus_pressure = models.IntegerField()
    runny_nose = models.IntegerField()
    congestion = models.IntegerField()
    chest_pain = models.IntegerField()
    weakness_in_limbs = models.IntegerField()
    fast_heart_rate = models.IntegerField()
    pain_during_bowel_movements = models.IntegerField()
    pain_in_anal_region = models.IntegerField()
    bloody_stool = models.IntegerField()
    irritation_in_anus = models.IntegerField()
    neck_pain = models.IntegerField()
    dizziness = models.IntegerField()
    cramps = models.IntegerField()
    bruising = models.IntegerField()
    obesity = models.IntegerField()
    swollen_legs = models.IntegerField()
    swollen_blood_vessels = models.IntegerField()
    puffy_face_and_eyes = models.IntegerField()
    enlarged_thyroid = models.IntegerField()
    brittle_nails = models.IntegerField()
    swollen_extremeties = models.IntegerField()
    excessive_hunger = models.IntegerField()
    extra_marital_contacts = models.IntegerField()
    drying_and_tingling_lips = models.IntegerField()
    slurred_speech = models.IntegerField()
    knee_pain = models.IntegerField()
    hip_joint_pain = models.IntegerField()
    muscle_weakness = models.IntegerField()
    stiff_neck = models.IntegerField()
    swelling_joints = models.IntegerField()
    movement_stiffness = models.IntegerField()
    spinning_movements = models.IntegerField()
    loss_of_balance = models.IntegerField()
    unsteadiness = models.IntegerField()
    weakness_of_one_body_side = models.IntegerField()
    loss_of_smell = models.IntegerField()
    bladder_discomfort = models.IntegerField()
    foul_smell_of_urine = models.IntegerField()
    continuous_feel_of_urine = models.IntegerField()
    passage_of_gases = models.IntegerField()
    internal_itching = models.IntegerField()
    toxic_look = models.IntegerField()
    depression = models.IntegerField()
    irritability = models.IntegerField()
    muscle_pain = models.IntegerField()
    altered_sensorium = models.IntegerField()
    red_spots_over_body = models.IntegerField()
    belly_pain = models.IntegerField()
    abnormal_menstruation = models.IntegerField()
    dischromic_patches = models.IntegerField()
    watering_from_eyes = models.IntegerField()
    increased_appetite = models.IntegerField()
    polyuria = models.IntegerField()
    family_history = models.IntegerField()
    mucoid_sputum = models.IntegerField()
    rusty_sputum = models.IntegerField()
    lack_of_concentration = models.IntegerField()
    visual_disturbances = models.IntegerField()
    receiving_blood_transfusion = models.IntegerField()
    receiving_unsterile_injections = models.IntegerField()
    coma = models.IntegerField()
    stomach_bleeding = models.IntegerField()
    distention_of_abdomen = models.IntegerField()
    history_of_alcohol_consumption = models.IntegerField()
    blood_in_sputum = models.IntegerField()
    prominent_veins_on_calf = models.IntegerField()
    palpitations = models.IntegerField()
    painful_walking = models.IntegerField()
    pus_filled_pimples = models.IntegerField()
    blackheads = models.IntegerField()
    scurring = models.IntegerField()
    skin_peeling = models.IntegerField()
    silver_like_dusting = models.IntegerField()
    small_dents_in_nails = models.IntegerField()
    inflammatory_nails = models.IntegerField()
    blister = models.IntegerField()
    red_sore_around_nose = models.IntegerField()
    yellow_crust_ooze = models.IntegerField()
    prognosis = models.CharField(max_length=100)


    class Meta:
        db_table = 'symptoms_diseases'

class Predicted_Diseases(models.Model) :
    diseases = ArrayField(models.CharField(max_length=200), blan
    diseases_prob = ArrayField(models.FloatField(default=0), bla
    consult_doctor = models.CharField(max_length=100, default="
```

`serializers.py`

```
from django.core.exceptions import ValidationError
from rest_framework import serializers
from django.contrib.auth import get_user_model, authenticate
from .models import PatientProfile, Predicted_Diseases, DoctorPr

UserModel = get_user_model()

class UserRegisterSerializer(serializers.ModelSerializer):
    class Meta:
        model = UserModel
        fields = '__all__'
    def create(self, clean_data):
        user_obj = UserModel.objects.create_user(email=clean_dat
        user_obj.username = clean_data['username']
        user_obj.save()
        return user_obj

class UserLoginSerializer(serializers.Serializer):
    email = serializers.EmailField()
    password = serializers.CharField()
    ##
    def check_user(self, clean_data):
        user = authenticate(username=clean_data['email'], passwo
        if not user:
            raise ValidationError('user not found')
        return user

class UserSerializer(serializers.ModelSerializer):
    class Meta:
        model = UserModel
        fields = ('email', 'username')
```

```
class PatientSerializer(serializers.ModelSerializer):
    class Meta:
        model = PatientProfile
        fields = ('__all__')

    def create(self, validated_data):
        user = self.context['request'].user
        profile = PatientProfile.objects.create(user=user, **va
        return profile

    def update(self, instance, validated_data):
        for attr, value in validated_data.items():
            setattr(instance, attr, value)
        instance.save()
        return instance

class PredictionSerializer(serializers.ModelSerializer):
    class Meta:
        model = Predicted_Diseases
        fields = ('diseases', 'diseases_prob', 'consult_doctor'


class DoctorProfileSerializer(serializers.ModelSerializer):
    class Meta:
        model = DoctorProfile
        fields = '__all__'
```

`urls.py`

```
from django.urls import path
from . import views

urlpatterns = [
    path('register', views.UserRegister.as_view(), name='registe
    path('login', views.UserLogin.as_view(), name='login'),
    path('logout', views.UserLogout.as_view(), name='logout'),
    path('user', views.UserView.as_view(), name='user'),
    path('patient', views.PatientProfile.as_view(), name='patien
    path('doctor/<str:sp>/', views.DoctorProfileListAPIView.as_v
    path('insert', views.insert_data, name='data'),
    path('check_email', views.check_email, name='check_email'),
    path('check_admin', views.check_admin, name='check_admin'),
]
```

`validations.py`

```
from django.core.exceptions import ValidationError
from django.contrib.auth import get_user_model
UserModel = get_user_model()

def custom_validation(data):
    email = data['email'].strip()
    username = data['username'].strip()
    password = data['password'].strip()
    ##
    if not email or UserModel.objects.filter(email=email).exists
        print(UserModel.objects.filter(email=email).exists())
        raise ValidationError('choose another email')
    ##
    if not password or len(password) < 8:
        raise ValidationError('choose another password, min 8 ch
    ##
    if not username:
        raise ValidationError('choose another username')
    return data
```

```
def validate_email(data):
    email = data['email'].strip()
    if not email:
        raise ValidationError('an email is needed')
    return True

def validate_username(data):
    username = data['username'].strip()
    if not username:
        raise ValidationError('choose another username')
    return True

def validate_password(data):
    password = data['password'].strip()
    if not password:
        raise ValidationError('a password is needed')
    return True
```

`views.py`

```
from django.db import connection
from django.shortcuts import render

# Create your views here.
from django.contrib.auth import get_user_model, login, logout
from rest_framework.authentication import SessionAuthentication
from rest_framework.views import APIView
from rest_framework.response import Response
from .serializers import UserRegisterSerializer, UserLoginSerial
from rest_framework import permissions, status, generics
from .validations import custom_validation, validate_email, val
from .models import DoctorProfile, AppUser
from django.http import JsonResponse
```

```
class UserRegister(APIView):
    permission_classes = (permissions.AllowAny,)

    def post(self, request):
        clean_data = custom_validation(request.data)
        serializer = UserRegisterSerializer(data=clean_data)
        if serializer.is_valid(raise_exception=True):
            user = serializer.create(clean_data)
            if user:
                return Response(serializer.data, status=status.H
        return Response(status=status.HTTP_400_BAD_REQUEST)


class UserLogin(APIView):
    permission_classes = (permissions.AllowAny,)
    authentication_classes = (SessionAuthentication,)
    ##

    def post(self, request):
        data = request.data
        assert validate_email(data)
        assert validate_password(data)
        serializer = UserLoginSerializer(data=data)
        if serializer.is_valid(raise_exception=True):
            user = serializer.check_user(data)
            login(request, user)
            return Response(serializer.data, status=status.HTTP_


class UserLogout(APIView):
    permission_classes = (permissions.AllowAny,)
    authentication_classes = ()

    def post(self, request):
        logout(request)
        return Response(status=status.HTTP_200_OK)


class UserView(APIView):
    permission_classes = (permissions.IsAuthenticated,)
    authentication_classes = (SessionAuthentication,)
    ##

    def get(self, request):
        serializer = UserSerializer(request.user)
        return Response({'user': serializer.data}, status=status

def check_email(request):
    email = request.GET.get('email')
    if email:
        email_exists = AppUser.objects.filter(email=email).exist
        response_data = {'email_exists': email_exists}
        return JsonResponse(response_data)
    else:
        response_data = {'error': 'Email parameter is missing'}
        return JsonResponse(response_data, status=400)


class PatientProfile(APIView):
    permission_classes = (permissions.IsAuthenticated,)
    authentication_classes = (SessionAuthentication,)

    def get(self, request):
        profile = request.user.profile

        if not profile:
            return Response({'error': 'User does not have a prof

        serializer = PatientSerializer(profile)
        return Response(serializer.data, status=status.HTTP_200_
    def put(self, request):
        serializer = PatientSerializer(
            request.user.profile, data=request.data, partial=Tru
        if serializer.is_valid():
            serializer.save()
            return Response(serializer.data, status=status.HTTP_
        return Response(serializer.errors, status=status.HTTP_40


class DoctorProfileListAPIView(generics.ListAPIView):
    serializer_class = DoctorProfileSerializer

    def get_queryset(self):
        speciality = self.kwargs.get('sp', '')
        if speciality == 'All':
            queryset = DoctorProfile.objects.all()
        else:
            queryset = DoctorProfile.objects.filter(speciality_

        queryset = queryset.order_by('?')[:12]

        return queryset




def insert_data(request):
    query = """
        INSERT INTO "Accounts_doctorprofile" (name, speciality,
        VALUES (%s, %s, %s, %s, %s, %s, %s, %s);
    """
    values = [
    # Family Medicine
    ('Dr. John Smith', 'Family Medicine', 'male', 15, '123 Main
    ('Dr. Emma Thompson', 'Family Medicine', 'female', 12, '456
    ('Dr. David Wilson', 'Family Medicine', 'male', 10, '789 Oa
    ('Dr. Lily Rodriguez', 'Family Medicine', 'female', 8, '321


    # Internal Medicine
    ('Dr. Emily Johnson', 'Internal Medicine', 'female', 20, '12
    ('Dr. Benjamin Davis', 'Internal Medicine', 'male', 18, '456
    ('Dr. Sophia Thompson', 'Internal Medicine', 'female', 22,
    ('Dr. Daniel Wilson', 'Internal Medicine', 'male', 16, '321


    # Pediatrician
    ('Dr. Michael Johnson', 'Pediatrician', 'male', 18, '123 Ma
    ('Dr. Abigail Smith', 'Pediatrician', 'female', 15, '456 El
    ('Dr. Ethan Davis', 'Pediatrician', 'male', 10, '789 Oak St
    ('Dr. Isabella Wilson', 'Pediatrician', 'female', 12, '321 

    # Obstetricians/gynecologist (OBGYNs)
    ('Dr. Olivia Davis', 'Gynecologist', 'female', 25, '123 Mai
    ('Dr. Liam Johnson', 'Gynecologist', 'male', 22, '456 Elm S
    ('Dr. Ava Brown', 'Gynecologist', 'female', 20, '789 Oak Str
    ('Dr. Noah Thompson', 'Gynecologist', 'male', 18, '321 Mapl

    # Cardiologist
    ('Dr. Benjamin Smith', 'Cardiologist', 'male', 22, '123 Mai
    ('Dr. Charlotte Johnson', 'Cardiologist', 'female', 20, '45
    ('Dr. Samuel Brown', 'Cardiologist', 'male', 18, '789 Oak S
    ('Dr. Grace Thompson', 'Cardiologist', 'female', 16, '321 M

    # Oncologist
    ('Dr. William Wilson', 'Oncologist', 'male', 30, '123 Main 
    ('Dr. Sophia Johnson', 'Oncologist', 'female', 28, '456 Elm
    ('Dr. Alexander Brown', 'Oncologist', 'male', 25, '789 Oak 
    ('Dr. Mia Thompson', 'Oncologist', 'female', 22, '321 Maple

    # Gastroenterologist
    ('Dr. Olivia Smith', 'Gastroenterologist', 'female', 25, '1
    ('Dr. Liam Johnson', 'Gastroenterologist', 'male', 22, '456
    ('Dr. Ava Brown', 'Gastroenterologist', 'female', 20, '789 O
    ('Dr. Noah Thompson', 'Gastroenterologist', 'male', 18, '321

    # Pulmonologist
    ('Dr. Benjamin Wilson', 'Pulmonologist', 'male', 20, '123 Ma
    ('Dr. Charlotte Johnson', 'Pulmonologist', 'female', 18, '45
    ('Dr. Samuel Brown', 'Pulmonologist', 'male', 16, '789 Oak S
    ('Dr. Grace Thompson', 'Pulmonologist', 'female', 14, '321 M

    # Infectious Disease
    ('Dr. William Smith', 'Infectious Disease', 'male', 18, '123
    ('Dr. Sophia Johnson', 'Infectious Disease', 'female', 16,
    ('Dr. Alexander Brown', 'Infectious Disease', 'male', 14, '7
    ('Dr. Mia Thompson', 'Infectious Disease', 'female', 12, '32


    # Nephrologist
    ('Dr. Olivia Smith', 'Nephrologist', 'female', 25, '123 Main
    ('Dr. Liam Johnson', 'Nephrologist', 'male', 22, '456 Elm St
    ('Dr. Ava Brown', 'Nephrologist', 'female', 20, '789 Oak Str
    ('Dr. Noah Thompson', 'Nephrologist', 'male', 18, '321 Maple

    # Endocrinologist
    ('Dr. Benjamin Wilson', 'Endocrinologist', 'male', 20, '123
    ('Dr. Charlotte Johnson', 'Endocrinologist', 'female', 18,
    ('Dr. Samuel Brown', 'Endocrinologist', 'male', 16, '789 Oak
    ('Dr. Grace Thompson', 'Endocrinologist', 'female', 14, '321

    # Ophthalmologist
    ('Dr. William Smith', 'Ophthalmologist', 'male', 18, '123 Ma
    ('Dr. Sophia Johnson', 'Ophthalmologist', 'female', 16, '456
    ('Dr. Alexander Brown', 'Ophthalmologist', 'male', 14, '789
    ('Dr. Mia Thompson', 'Ophthalmologist', 'female', 12, '321 M

    # Otolaryngologist
    ('Dr. Olivia Smith', 'Otolaryngologist', 'female', 25, '123 
    ('Dr. Liam Johnson', 'Otolaryngologist', 'male', 22, '456 El
    ('Dr. Ava Brown', 'Otolaryngologist', 'female', 20, '789 Oak
    ('Dr. Noah Thompson', 'Otolaryngologist', 'male', 18, '321 M

    # Dermatologist
    ('Dr. Benjamin Wilson', 'Dermatologist', 'male', 20, '123 Ma
    ('Dr. Charlotte Johnson', 'Dermatologist', 'female', 18, '45
    ('Dr. Samuel Brown', 'Dermatologist', 'male', 16, '789 Oak S
    ('Dr. Grace Thompson', 'Dermatologist', 'female', 14, '321 M

    # Psychiatrist
    ('Dr. William Smith', 'Psychiatrist', 'male', 18, '123 Main 
    ('Dr. Sophia Johnson', 'Psychiatrist', 'female', 16, '456 El
    ('Dr. Alexander Brown', 'Psychiatrist', 'male', 14, '789 Oak
    ('Dr. Mia Thompson', 'Psychiatrist', 'female', 12, '321 Mapl

    # Neurologist
    ('Dr. Olivia Smith', 'Neurologist', 'female', 25, '123 Main 
    ('Dr. Liam Johnson', 'Neurologist', 'male', 22, '456 Elm Str
    ('Dr. Ava Brown', 'Neurologist', 'female', 20, '789 Oak Stre
    ('Dr. Noah Thompson', 'Neurologist', 'male', 18, '321 Maple 

    # Radiologist
    ('Dr. Benjamin Wilson', 'Radiologist', 'male', 20, '123 Main
    ('Dr. Charlotte Johnson', 'Radiologist', 'female', 18, '456 
    ('Dr. Samuel Brown', 'Radiologist', 'male', 16, '789 Oak Str
    ('Dr. Grace Thompson', 'Radiologist', 'female', 14, '321 Map

    # Anesthesiologist
    ('Dr. William Smith', 'Anesthesiologist', 'male', 18, '123 M
    ('Dr. Sophia Johnson', 'Anesthesiologist', 'female', 16, '45
    ('Dr. Alexander Brown', 'Anesthesiologist', 'male', 14, '789
    ('Dr. Mia Thompson', 'Anesthesiologist', 'female', 12, '321 

    # Surgeon
    ('Dr. Olivia Smith', 'Surgeon', 'female', 25, '123 Main Stre
    ('Dr. Liam Johnson', 'Surgeon', 'male', 22, '456 Elm Street,
    ('Dr. Ava Brown', 'Surgeon', 'female', 20, '789 Oak Street,
    ('Dr. Noah Thompson', 'Surgeon', 'male', 18, '321 Maple Stre

    # Physician Executive
    ('Dr. Benjamin Wilson', 'Physician Executive', 'male', 20,
    ('Dr. Charlotte Johnson', 'Physician Executive', 'female', 1
    ('Dr. Samuel Brown', 'Physician Executive', 'male', 16, '789
    ('Dr. Grace Thompson', 'Physician Executive', 'female', 14,
]

    with connection.cursor() as cursor:
        cursor.executemany(query, values)

        return render(request, 'index.html')

def check_admin(request):
    email = request.GET.get('email')
    if email:
        try:
            user = AppUser.objects.get(email=email)  # Retrieve
            is_superuser = user.is_superuser  # Access the is_su
            response_data = {'email_exists': True, 'is_superuser
            return JsonResponse(response_data)
        except AppUser.DoesNotExist:
            response_data = {'email_exists': False, 'is_superuse
            return JsonResponse(response_data)
    else:
        response_data = {'error': 'Email parameter is missing'}
        return JsonResponse(response_data, status=400)
```

`__init__.py`, `apps.py` and `tests.py` can remain with the default value they were initialized with. We don't need to edit it.

Disease Predictor Directory

This will the application directory we will use for the disease prediction model. Setup the directory according to the below component tree

```
DiseasePredictor/
├── Training.csv
├── __init__.py
├── admin.py
├── apps.py
├── models.py
├── tests.py
├── urls.py
└── views.py
```

Training Dataset for the Model

`Training.csv`

[Training.csv]

Create and Paste the following code in the components mentioned

`urls.py`

```
from django.urls import path
from .views import predict, insert_patient_data, train

urlpatterns = [
    path('prediction/<str:symptoms>/', predict),
    path('insertpd', insert_patient_data),
    path('train', train),
]
```

`views.py`

```
from Accounts.models import symptoms_diseases, Predicted_Disease
from Accounts.serializers import PredictionSerializer
from django.shortcuts import render
import pandas as pd
import numpy as np
from django_pandas.io import read_frame
from imblearn.over_sampling import RandomOverSampler
from sklearn.preprocessing import StandardScaler
from sklearn.svm import SVC
from rest_framework.decorators import api_view
from rest_framework.response import Response
import csv
from django.db import transaction
import os
import pickle


def insert_patient_data(request):
    data = os.path.join(os.path.dirname(
        os.path.realpath(__file__)), 'Training.csv')
    with open(data, 'r') as file:
        reader = csv.reader(file)
        next(reader)  # Skip the header row

        with transaction.atomic():
            for row in reader:
                # Map the values from the CSV row to the model
                # Exclude the last column
                symptom_values = [int(value) for value in row[:
```

```
                prognosis = row[-1]

                # Create a new instance of the model
                field_names = [field.name for field in symptoms_
                ) if field.name != 'id' and field.name != 'progr
                field_values = dict(zip(field_names, symptom_val
                instance = symptoms_diseases.objects.create(
                    prognosis=prognosis, **field_values)

                # Save the instance to the database
                instance.save()

            return render(request, 'index.html')


def scale_dataset(dataframe, oversample=False):
    X = dataframe[dataframe.columns[:-1]].values
    y = dataframe[dataframe.columns[-1]].values

    scaler = StandardScaler()
    X = scaler.fit_transform(X)

    if oversample:
        ros = RandomOverSampler()
        X, y = ros.fit_resample(X, y)

    data = np.hstack((X, np.reshape(y, (-1, 1))))

    return data, X, y


svm_model = None

def train(request):
    global svm_model
    data = pd.DataFrame.from_records(
        symptoms_diseases.objects.all().values()).drop('id', ax

    train, X, Y = scale_dataset(data, oversample=True)

    svm_model = SVC(probability=True)
    svm_model = svm_model.fit(X, Y)

    with open('model.pkl', 'wb') as f:
        pickle.dump(svm_model, f)

    return render(request, 'index.html')


@api_view()
def predict(request, symptoms=''):

    with open('model.pkl', 'rb') as f:
        svm_model = pickle.load(f)

    x = np.asarray(list(symptoms), dtype=np.int_)
    x = x[1:]
    x = x.reshape(-1, 1)

    scaler = StandardScaler()
    x = scaler.fit_transform(x)

    x_ = x.reshape(1, -1)
    Y_ = svm_model.predict(x_)

    probas = svm_model.predict_proba(x_)

    top5_indices = np.argsort(probas, axis=1)[:, -5:]
    top5_values = np.take_along_axis(probas, top5_indices, axis

    # Get the corresponding class labels
    top5_labels = svm_model.classes_[top5_indices]
```

```
# Print the top 5 class labels for the first sample in the t
pd = top5_labels[0][::-1].tolist()
predicted_disease = pd[0]

Rheumatologist = ['Osteoarthristis', 'Arthritis']

Cardiologist = ['Heart attack', 'Bronchial Asthma', 'Hyperte

ENT_specialist = [
    '(vertigo) Paroymsal  Positional Vertigo', 'Hypothyroidi

Neurologist = ['Varicose veins',
               'Paralysis (brain hemorrhage)', 'Migraine',

Allergist_Immunologist = ['Allergy', 'Pneumonia', 'AIDS',
                          'Common Cold', 'Tuberculosis', 'Ma

Urologist = ['Urinary tract infection', 'Dimorphic hemmorhoi

Dermatologist = ['Acne', 'Chicken pox',
                 'Fungal infection', 'Psoriasis', 'Impetigo

Gastroenterologist = ['Peptic ulcer diseae', 'GERD', 'Chroni
                      'Alcoholic hepatitis', 'Jaundice', 'he
                      'Hepatitis B', 'Hepatitis C', 'Hepatit

if predicted_disease in Rheumatologist:
    consultdoctor = "Rheumatologist"

if predicted_disease in Cardiologist:
    consultdoctor = "Cardiologist"

elif predicted_disease in ENT_specialist:
    consultdoctor = "ENT specialist"
elif predicted_disease in Neurologist:
    consultdoctor = "Neurologist"

elif predicted_disease in Allergist_Immunologist:
    consultdoctor = "Allergist/Immunologist"

elif predicted_disease in Urologist:
    consultdoctor = "Urologist"

elif predicted_disease in Dermatologist:
    consultdoctor = "Dermatologist"

elif predicted_disease in Gastroenterologist:
    consultdoctor = "Gastroenterologist"

else:
    consultdoctor = "other"

pd_prob = top5_values[0][::-1].astype(float).tolist()
Predicted_Diseases.objects.all().delete()
Predicted_Diseases(diseases=pd, diseases_prob=pd_prob, consu
data = Predicted_Diseases.objects.all()
serializer = PredictionSerializer(data, many=True)
return Response(serializer.data, template_name=None)
```

All the files left needs no changes and can remain by default

Database Migrations

Run the following command in your terminal to make database migrations

```
python manage.py makemigrations
python manage.py migrate
```

Environment Variables

Add the following environment variables to your `.env`

```
USER = <Postgres_Username>
DATABASE_NAME = <Database_Name>
PASS = <Password>
```

Building the Frontend

In the components directory, create the following components and pages. Copy and paste the provided code

Pages and Components

`About.jsx`

```
import patternImg from "../img/pattern.svg";
const About = () => {
  return (
    <div
      id="about"
      className="w-full flex justify-center mt-10
    "
    >
      <div className="about-container flex flex-col-reverse md:
        <div className="hero flex flex-col justify-center  md:w
          <div className="hero-text text-3xl text-center md:tex
            About Medware
          </div>
          <div className="hero-stanza  lg:text-lg flex items-cen
            Your one-stop healthcare provider. Our innovative me
            and disease predictor offer personalized insights in
            Convenient doctor consultations and a range of healt
            are just a click away. Experience the difference in
            and advanced technologies with Medware.
          </div>
        </div>
        <div className="img-wrapper w-80 mb-5 md:w-1/3 ">
          <img src={patternImg} alt="hero-image" className="bloc
        </div>
      </div>
    </div>
  );
};

export default About;
```

BP_Log.jsx

```
import React from "react";

const BP_Log = ({ responseData }) => {
  if (
    !responseData ||
    !responseData.bp_log ||
    !responseData.bp_log.date ||
    responseData.bp_log.date.length === 0
  ) {
    return (
      <p className="h-full w-full grid place-content-center ital
        Add your first value
      </p>
    );
  }

  return (
    <div className="p-2 mb-1 rounded-lg">
      {responseData.bp_log.date.map((date, index) => {
        const currentHigh = responseData.bp_log.high[index];
        const currentLow = responseData.bp_log.low[index];

        // Skip the log entry if high and low values are empty
        if (!currentHigh && !currentLow) {
          return null;
        }

        const isFirstValueOfDay =
          index === 0 || responseData.bp_log.date[index - 1] !==

        return (
          <div key={index} className="flex flex-col mb-1">
            {isFirstValueOfDay && (
              <div className="flex items-center mb-2 bg-slate-20
                <div className="h-2 w-2 bg-gray-700 rounded-full
                <h2 className="text-lg font-semibold text-gray-9
              </div>
            )}

            <div className="ml-1">
              <div
                className={`text-sm text-gray-700 border border-
                  currentHigh > 190 && currentLow > 90
                    ? "bg-purple-100"
                    : currentHigh > 190
                    ? "bg-red-100"
                    : currentLow > 90
                    ? "bg-orange-100"
                    : ""
                }`}
              >
                <div className="flex">
                  <p className="font-semibold">{currentHigh}</p
                  <span className="text-gray-500 mx-1">/</span>
                  <p>{currentLow}</p>
                </div>
                <div>High / Low</div>
              </div>
            </div>
          </div>
        );
      })}
    </div>
  );
};

export default BP_Log;
```

`BP_chart.jsx`

```
import React from "react";
import {
  Chart as ChartJS,
  CategoryScale,
  LinearScale,
  PointElement,
  BarElement,
  Title,
  Tooltip,
  Legend,
} from "chart.js";
import { Bar } from "react-chartjs-2";
```

```
export default function BP_chart({ chartData }) {
  ChartJS.register(
    CategoryScale,
    LinearScale,
    PointElement,
    BarElement,
    Title,
    Tooltip,
    Legend
  );

  const { low, date, high } = chartData;

  const options = {
    responsive: true,
    maintainAspectRatio: false,
    interaction: {
      mode: "index",
      intersect: false,
    },
    plugins: {
      title: {
        display: true,
        text: "Blood Pressure",
      },
    },
    scales: {
      y: {
        type: "linear",
        display: false,
      },
      y1: {
        type: "linear",
        display: true,
        position: "left",
        grid: {
          drawOnChartArea: false,
        },
      },
    },
  };

  const data = {
    labels: date,
    datasets: [
      {
        label: "Low",
        data: low,
        backgroundColor: "rgb(252, 99, 255, 0.7)",
        yAxisID: "y1",
        barPercentage: 0.6, // Adjust the bar width (0.6 means 6
        borderRadius: 10, // Adjust the border radius to make th
      },
      {
        label: "High",
        data: high,
        backgroundColor: "rgba(99, 99, 255, 0.7)",
        yAxisID: "y1",
        barPercentage: 0.6,
        borderRadius: 10,
      },
    ],
  };

  return <Bar options={options} data={data} />;
}
```

`Calendar.jsx`

```
import React, { useState, useRef, useEffect } from "react";

const Calendar = () => {
  const currentDate = new Date();
  const [selectedDate, setSelectedDate] = useState(currentDate)
  const [selectedMonth, setSelectedMonth] = useState(currentDat
  const [selectedYear, setSelectedYear] = useState(currentDate.
  const [isMonthDropdownOpen, setIsMonthDropdownOpen] = useStat
  const [isYearDropdownOpen, setIsYearDropdownOpen] = useState(

  const monthDropdownRef = useRef(null);
  const yearDropdownRef = useRef(null);

  const getDaysInMonth = (year, month) => {
    return new Date(year, month + 1, 0).getDate();
  };

  const getMonthName = (month) => {
    const options = { month: "short" };
    return new Intl.DateTimeFormat("en-US", options).format(
      new Date(2000, month, 1)
    );
  };

  const getWeekdayName = (day) => {
    const options = { weekday: "short" };
    return new Intl.DateTimeFormat("en-US", options).format(
      new Date(2000, 0, day + 1)
    );
  };

  const handleMonthChange = (month) => {
    setSelectedMonth(month);
    setIsMonthDropdownOpen(false);
  };
```

```
const handleYearChange = (year) => {
  setSelectedYear(year);
  setIsYearDropdownOpen(false);
};

const handleDateClick = (date) => {
  setSelectedDate(date);
};

const handleClickOutside = (event) => {
  if (
    monthDropdownRef.current &&
    !monthDropdownRef.current.contains(event.target)
  ) {
    setIsMonthDropdownOpen(false);
  }

  if (
    yearDropdownRef.current &&
    !yearDropdownRef.current.contains(event.target)
  ) {
    setIsYearDropdownOpen(false);
  }
};

useEffect(() => {
  document.addEventListener("click", handleClickOutside);
  return () => {
    document.removeEventListener("click", handleClickOutside)
  };
}, []);

const renderDaysList = () => {
  const daysList = [];
```

```
    for (let i = 0; i < 7; i++) {
      const dayName = getWeekdayName(i);
      daysList.push(
        <div
          key={`day-${i}`}
          className="flex-1 text-center w-8 text-gray-700 text-s
        >
          {dayName}
        </div>
      );
    }

    return daysList;
  };

  const renderMonthDropdown = () => {
    const monthOptions = Array.from({ length: 12 }, (_, i) => {
      const month = new Date(2000, i, 1).toLocaleString("defaul
        month: "short",
      });
      return {
        value: i,
        label: month,
      };
    });

    return (
      <div className="relative inline-block" ref={monthDropdownR
        <button
          className="bg-white text-teal-500  hover:scale-105 w-1
          onClick={() => setIsMonthDropdownOpen(!isMonthDropdown
        >
          {getMonthName(selectedMonth)}
        </button>
        {isMonthDropdownOpen && (
          <div className="absolute mt-2 py-1 w-20 overflow-y-aut
```

```
              {monthOptions.map((option) => (
                <div
                  key={option.value}
                  className="px-2 py-1 hover:bg-gray-200 cursor-pc
                  onClick={() => handleMonthChange(option.value)}
                >
                  {option.label}
                </div>
              ))}
            </div>
          )}
        </div>
      );
    };

    const renderYearDropdown = () => {
      const yearOptions = Array.from({ length: 50 }, (_, i) => {
        const year = currentDate.getFullYear() - 25 + i;
        return {
          value: year,
          label: year.toString(),
        };
      });

      return (
        <div className="relative inline-block" ref={yearDropdownRe
          <button
            className="bg-white text-gray-700 hover:scale-105 w-16
            onClick={() => setIsYearDropdownOpen(!isYearDropdownOp
          >
            {selectedYear}
          </button>
          {isYearDropdownOpen && (
            <div className="absolute mt-2 py-1 w-24 max-h-44 over
              {yearOptions.map((option) => (
                <div
                  key={option.value}
                  className="px-2 py-1 hover:bg-gray-200 cursor-po
                  onClick={() => handleYearChange(option.value)}
                >
                  {option.label}
                </div>
              ))}
            </div>
          )}
        </div>
      );
    };

    const renderCalendar = () => {
      const daysInMonth = getDaysInMonth(selectedYear, selectedMon
      const firstDay = new Date(selectedYear, selectedMonth, 1).ge

      const calendar = [];

      // Add empty cells for previous month
      for (let i = 0; i < firstDay; i++) {
        calendar.push(<div key={`empty-${i}`} className="w-8 h-8">
      }

      // Add cells for current month
      for (let i = 1; i <= daysInMonth; i++) {
        const date = new Date(selectedYear, selectedMonth, i);
        const isSelected = date.toDateString() === selectedDate.to
        const isCurrentDate = date.toDateString() === currentDate

        const cellClasses = `w-8 h-8 rounded-2xl text-sm text-cent
          isSelected
            ? "bg-teal-500 text-white transition-all duration-300
            : isCurrentDate
            ? "bg-sky-500 text-white"
            : "text-gray-600"
      }`;

      calendar.push(
        <div
          key={`date-${i}`}
          className={cellClasses}
          onClick={() => handleDateClick(date)}
        >
          <div className="flex items-center justify-center h-ful
        </div>
      );
    }

    return calendar;
  };

  return (
    <div className="w-full rounded-lg flex overflow-scroll flex-
      <div className="rounded-s-lg flex gap-1 items-center mb-2
        <div className="text-teal-500 font-bold text-xl">
          {renderMonthDropdown()}
        </div>
        <div className="text-2xl font-bold">{renderYearDropdown
      </div>
      <div className="flex flex-col rounded-e-md justify-center
        <div className="flex justify-center sm:mb-">
          <div className="grid-container mx-auto">
            <div className="gridd gap-1">{renderDaysList()}</di
          </div>
        </div>
        <div className="grid-container mx-auto">
          <div className="gridd gap-1">{renderCalendar()}</div>
        </div>
      </div>
    </div>
  );
```

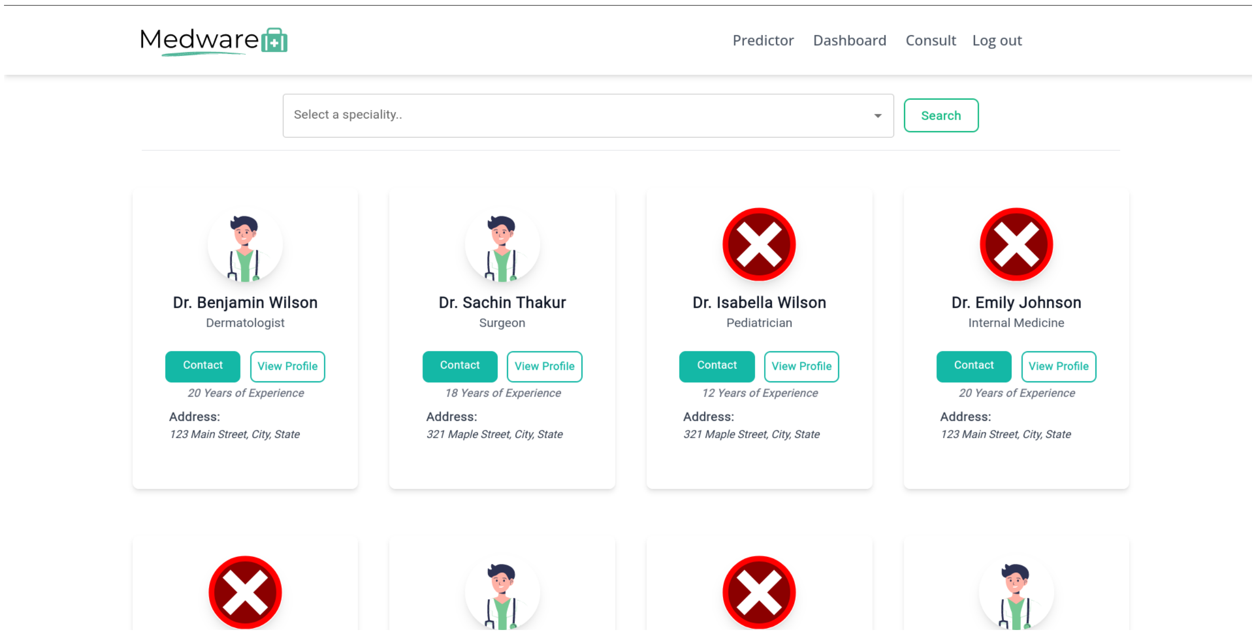
```
};

export default Calendar;
```

`ConsumptionModal.jsx`

```
import React, { useEffect } from "react";
import crossIcon from "../img/cross icon.svg";
import {
  TextField,
  Button,
  FormControl,
  Select,
  MenuItem,
  Grid,
  InputLabel,
} from "@mui/material";
import { useGlobalContext } from "./context";

const ConsumptionModal = ({ consumptionModal, setConsumptionModa
  const { handleDashboardChange, data, handleDashboardSubmit } =
    useGlobalContext();
  const closeConsumptionModal = () => {
    setConsumptionModal(false);
  };

  useEffect(() => {
    const handleClickOutside = (event) => {
      if (event.target.classList.contains("modal")) {
        closeConsumptionModal();
      }
    };

    if (consumptionModal) {
      document.addEventListener("click", handleClickOutside);
    }

    return () => {
      document.removeEventListener("click", handleClickOutside)
    };
  }, [consumptionModal]);

  if (!consumptionModal) {
    return null;
  }

  return (
    <div className="fixed top-0 left-0 w-screen h-screen flex ju
      <div className="flex flex-col justify-center items-center
        <div className="w-full flex justify-end">
          <button
            onClick={closeConsumptionModal}
            className="z-50 hover:scale-105"
          >
            <img src={crossIcon} alt="cross-icon" loading="lazy"
          </button>
        </div>
        <form
          className="w-full flex flex-col gap-6"
          onSubmit={(e) => {
            e.preventDefault();
            closeConsumptionModal();
            handleDashboardSubmit(e);
          }}
        >
          <Grid container spacing={4}>
            <Grid item xs={12}>
              <h1 className="text-2xl p-1 font-semibold text-gra
                Consumption Data
              </h1>
            </Grid>
            <Grid item xs={12}>
              <FormControl variant="outlined" fullWidth>
                <InputLabel id="demo-simple-select-label">
                  Smoking Consumption
                </InputLabel>
                <Select
                  labelId="dropdown-label"
                  value={data.smoke_cons}
                  onChange={handleDashboardChange}
                  name="smoke_cons"
                  label="Smoke Consumption"
                >
                  <MenuItem value={"No Consumption"}>Non Smoker
                  <MenuItem value={"Mild Smoking"}>Mild Smoking
                  <MenuItem value={"Oftenly Smokes/ Addiction"}
                    Addiction
                  </MenuItem>
                </Select>
              </FormControl>
            </Grid>
            <Grid item xs={12}>
              <FormControl variant="outlined" fullWidth>
                <InputLabel> Alcohol Consumption</InputLabel>
                <Select
                  labelId="dropdown-label"
                  value={data.alcohol_cons}
                  onChange={handleDashboardChange}
                  name="alcohol_cons"
                  label="Alcohol Consumption"
                >
                  <MenuItem value={"No Consumption"}>No Consump
                  <MenuItem value={"Mild Consumption"}>Mild</Me
                  <MenuItem value={"High Consumption"}>
                    High Consumption
                  </MenuItem>
                </Select>
              </FormControl>
            </Grid>
          </Grid>
          <Button variant="outlined" color="success" type="subm
            Submit
          </Button>
        </form>
      </div>
    </div>
  );
};

export default ConsumptionModal;
```

ContactDoctor.jsx

```
import React, { useEffect, useRef, useState } from "react";
import axios from "axios";
import DoctorProfile from "./DoctorProfile";
import SkeletonLoader from "./SkeletonLoader";
import { Autocomplete, TextField } from "@mui/material";

const docOptions = [
  "Family Medicine",
  "Internal Medicine",
  "Pediatrician",
  "Gynecologist",
  "Cardiologist",
  "Oncologist",
  "Gastroenterologist",
  "Pulmonologist",
  "Infectious disease",
  "Nephrologist",
  "Endocrinologist",
  "Ophthalmologist",
  "Otolaryngologist",
  "Dermatologist",
  "Psychiatrist",
  "Neurologist",
  "Radiologist",
  "Anesthesiologist",
  "Surgeon",
  "Physician executive",
];

const ContactDoctor = () => {
  const [doctors, setDoctors] = useState([]);
  const [speciality, setSpeciality] = useState(null);
  const doctorType = useRef("All");

  const fetchData = async () => {
    try {
      const response = await axios.get(
        `http://127.0.0.1:8000/doctor/${doctorType.current}`
      );
      console.log(response.data);
      setDoctors(response.data);
    } catch (error) {
      console.error(error);
    }
  };

  useEffect(() => {
    const fetchData = async () => {
      try {
        const response = await axios.get(
          `http://127.0.0.1:8000/doctor/${doctorType.current}`
        );
        console.log(response.data);
        setDoctors(response.data);
      } catch (error) {
        console.error(error);
      }
    };

    fetchData();
  }, []);

  const handleDocChange = () => {
    if (speciality === "" || !docOptions.includes(speciality)) 
      alert("Please select a valid option");
    } else {
      doctorType.current = speciality;
      fetchData();
    }
  };

  return (
    <section className="w-screen flex flex-col items-center p-5
      <div className="flex w-80 md:w-3/5   justify-center gap-3
        <Autocomplete
          options={docOptions}
          value={speciality}
          onChange={(e, newValue) => {
            setSpeciality(newValue);
          }}
          className="searchbox w-5/6  bg-white"
          renderInput={(params) => (
            <TextField
              variant="outlined"
              color="primary"
              {...params}
              label="Select a speciality.."
            />
          )}
          />
          <button
            onClick={handleDocChange}
            className="w-24 text-base font-semibold text-center ho
          >
            Search
          </button>
        </div>
        <div className="border-t border-gray-200 mb-8 w-4/5"></di

        <div className="flex justify-center w-full mt-4">
          {doctors.length ? (
            <DoctorProfile doctors={doctors} />
          ) : (
            <SkeletonLoader />
          )}
        </div>
        <article id="info-contact doctor"></article>
      </section>
    );
  };

export default ContactDoctor;
```


Medware

Predictor Dashboard Consult Log out

Select a speciality.. Search

Dr. Benjamin Wilson
Dermatologist
Contact View Profile
20 Years of Experience
Address:
123 Main Street, City, State

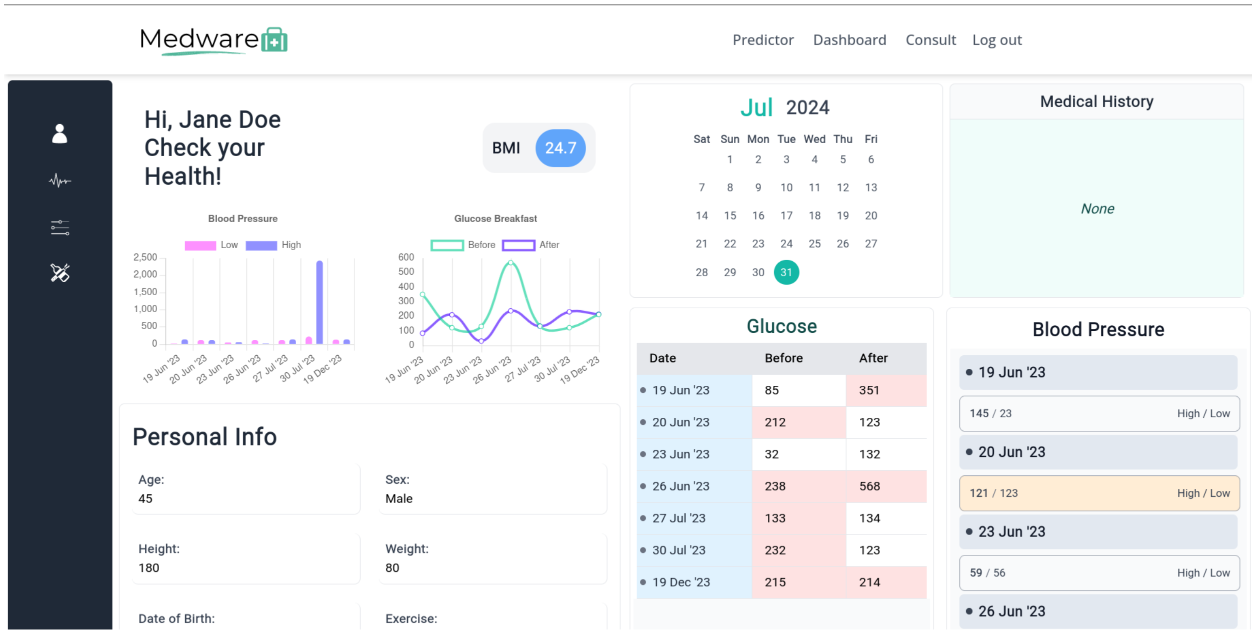
Dr. Sachin Thakur
Surgeon
Contact View Profile
18 Years of Experience
Address:
321 Maple Street, City, State

Dr. Isabella Wilson
Pediatrician
Contact View Profile
12 Years of Experience
Address:
321 Maple Street, City, State

Dr. Emily Johnson
Internal Medicine
Contact View Profile
20 Years of Experience
Address:
123 Main Street, City, State

`Dashboard.jsx`

```
import React, { useState, useEffect } from "react";
import axios from "axios";
import PatientForm from "./PatientForm";
import PatientProfile from "./PatientProfile";
import { useGlobalContext } from "./context";

axios.defaults.xsrfCookieName = "csrftoken";
axios.defaults.xsrfHeaderName = "X-CSRFToken";

const Dashboard = () => {
  const { handleInputChange, formData, handleFormSubmit, data,
    useGlobalContext();

  useEffect(() => {
    fetchData();
  }, []);
  return (
    <div className="pt-24">
      <PatientForm
        profileData={formData}
        handleInputChange={handleInputChange}
        handleFormSubmit={handleFormSubmit}
        patientData={data}
      />
      <PatientProfile responseData={data} />
    </div>
  );
};

export default Dashboard;
```



Hi, Jane Doe
Check your
Health!

BMI 24.7

Personal Info

Age:
45

Sex:
Male

Height:
180

Weight:
80

Date of Birth:

Exercise:

Jul 2024

Sat	Sun	Mon	Tue	Wed	Thu	Fri
	1	2	3	4	5	6
7	8	9	10	11	12	13
14	15	16	17	18	19	20
21	22	23	24	25	26	27
28	29	30	31			

Medical History

None

Glucose

Date	Before	After
19 Jun '23	85	351
20 Jun '23	212	123
23 Jun '23	32	132
26 Jun '23	238	568
27 Jul '23	133	134
30 Jul '23	232	123
19 Dec '23	215	214

Blood Pressure

- 19 Jun '23

  145 / 23 High / Low
- 20 Jun '23

  121 / 123 High / Low
- 23 Jun '23

  59 / 56 High / Low
- 26 Jun '23

`DoctorProfile.jsx`

```
const DoctorProfile = ({ doctors }) => {
  return (
    <article className="flex justify-center w-screen gap-10 fle
```

```
        {doctors.map((doctor) => (
          <div
            key={doctor.id}
            className="w-72 h-96 max-w-sm bg-white rounded-lg shad
          >
            <img
              className="w-24 h-24 mb-3 rounded-full shadow-lg"
              src={doctor.image_link}
              alt="Image"
            />
            <h5 className=" text-xl text-gray-900 font-semibold">
              {doctor.name}
            </h5>
            <span
              className=" font-normal
                text-gray-600"
            >
              {doctor.speciality}
            </span>
            <div className="flex mt-4 space-x-3 md:mt-6 ">
              <a
                href="tel:${doctor.mobile_no}"
                className="w-24 h-10 py-2 text-sm font-semibold te
              >
                Contact
              </a>
              <a
                href="#"
                className="w-24 h-10 py-2 text-sm font-semibold te
              >
                View Profile
              </a>
            </div>
            <div className="mt-1 italic text-gray-500 text-sm font
              {doctor.experience} Years of Experience
            </div>
            <div className="px-1 mt-2 text-gray-700 pl-4 w-5/6">
              <h2 className=" text-base font-semibold font  w-full
              <p className="italic text-sm h-14 overflow-scroll">
                {doctor.work_address}
              </p>
            </div>
          </div>
        ))}
      </article>
    );
  };

  export default DoctorProfile;
```

`Footer.jsx`

```
import footer_logo from "../img/footerimg.svg";

export default function Footer() {
  return (
    <footer className="bg-white dark:bg-gray-900">
      <div className="mx-auto w-full max-w-screen-xl p-4 py-6 lg
        <div className="md:flex md:justify-between">
          <div className="mb-6 md:mb-0">
            <a href="" className="flex items-center">
              <img src={footer_logo} className="h-8 mr-3" alt="
              <span className="self-center text-2xl font-semibol
                Medware
              </span>
            </a>
          </div>
          <div className="grid grid-cols-2 gap-12 sm:gap-8 sm:gr
            <div>
              <h2 className="mb-6 text-sm font-semibold text-gra
```

```
                Follow us
              </h2>
              <ul className="text-gray-600 dark:text-gray-400 f
                <li className="mb-4">
                  <a href="" className="hover:underline ">
                    Github
                  </a>
                </li>
                <li>
                  <a href="" className="hover:underline">
                    Discord
                  </a>
                </li>
              </ul>
            </div>
            <div>
              <h2 className="mb-6 text-sm font-semibold text-gra
                Legal
              </h2>
              <ul className="text-gray-600 dark:text-gray-400 f
                <li className="mb-4">
                  <a href="#" className="hover:underline">
                    Privacy Policy
                  </a>
                </li>
                <li>
                  <a href="#" className="hover:underline">
                    Terms &amp; Conditions
                  </a>
                </li>
              </ul>
            </div>
          </div>
        </div>
        <hr className="my-6 border-gray-200 sm:mx-auto dark:bor
        <div className="sm:flex sm:items-center sm:justify-betw
```

```
            <span className="text-sm text-gray-500 sm:text-center
              © 2023{" "}
              <a href="" className="hover:underline">
                Medware™
              </a>
              . All Rights Reserved.
            </span>
            <div className="flex mt-4 space-x-6 sm:justify-center
              <a
                href="#"
                className="text-gray-500 hover:text-gray-900 dark
              >
                <svg
                  className="w-5 h-5"
                  fill="currentColor"
                  viewBox="0 0 24 24"
                  aria-hidden="true"
                >
                  <path
                    fillRule="evenodd"
                    d="M22 12c0-5.523-4.477-10-10-10S2 6.477 2 12c
                    clipRule="evenodd"
                  />
                </svg>
                <span className="sr-only">Facebook page</span>
              </a>
              <a
                href="#"
                className="text-gray-500 hover:text-gray-900 dark
              >
                <svg
                  className="w-5 h-5"
                  fill="currentColor"
                  viewBox="0 0 24 24"
                  aria-hidden="true"
                >
                  <path
                    fillRule="evenodd"
                    d="M12.315 2c2.43 0 2.784.013 3.808.06 1.064.(
                    clipRule="evenodd"
                  />
                </svg>
                <span className="sr-only">Instagram page</span>
              </a>
              <a
                href="#"
                className="text-gray-500 hover:text-gray-900 dark
              >
                <svg
                  className="w-5 h-5"
                  fill="currentColor"
                  viewBox="0 0 24 24"
                  aria-hidden="true"
                >
                  <path d="M8.29 20.251c7.547 0 11.675-6.253 11.67
                </svg>
                <span className="sr-only">Twitter page</span>
              </a>
              <a
                href="#"
                className="text-gray-500 hover:text-gray-900 dark
              >
                <svg
                  className="w-5 h-5"
                  fill="currentColor"
                  viewBox="0 0 24 24"
                  aria-hidden="true"
                >
                  <path
                    fillRule="evenodd"
                    d="M12 2C6.477 2 2 6.484 2 12.017c0 4.425 2.86
                    clipRule="evenodd"
```

```
                  />
                </svg>
                <span className="sr-only">GitHub account</span>
              </a>
              <a
                href="#"
                className="text-gray-500 hover:text-gray-900 dark
              >
                <svg
                  className="w-5 h-5"
                  fill="currentColor"
                  viewBox="0 0 24 24"
                  aria-hidden="true"
                >
                  <path
                    fillRule="evenodd"
                    d="M12 2C6.48 2 2 6.48 2 12s4.48 10 10 10c5.5
                    clipRule="evenodd"
                  />
                </svg>
                <span className="sr-only">Dribbble account</span>
              </a>
            </div>
          </div>
        </div>
      </footer>
    );
  }
```

`GlucoseLevel.jsx`

```
import React from "react";

const GlucoseLevel = ({ responseData }) => {
  if (
    !responseData ||
    !responseData.blood_glucose ||
    !responseData.blood_glucose.date ||
    responseData.blood_glucose.date.length === 0
  ) {
    return (
      <p className="w-full h-full grid place-content-center ital
        No values in log
      </p>
    );
  }

  const getCellStyles = (value, isAfterColumn) => {
    if (isAfterColumn) {
      if (value > 180) {
        return "bg-red-100";
      }
    } else {
      if (value > 120) {
        return "bg-red-100";
      }
    }
    return "";
  };

  let prevDate = null; // Track previous date

  return (
    <div className="px-1 bg-white">
      <table className="w-full border-collapse">
        <thead>
          <tr className="bg-gray-200">
            <th className="py-2 px-4 border-b font-semibold text
            <th className="py-2 px-4 border-b font-semibold text
              Before
            </th>
            <th className="py-2 px-4 border-b font-semibold text
              After
            </th>
          </tr>
        </thead>
        <tbody>
          {responseData.blood_glucose.date.map((date, index) =>
            const currentBefore = responseData.blood_glucose.be
            const currentAfter = responseData.blood_glucose.afte

            // Skip the date if both before and after values are
            if (!currentBefore && !currentAfter) {
              return null;
            }

            const isFirstDate = prevDate !== date;
            prevDate = date;

            return (
              <tr key={index} className="border-b">
                <td
                  className={`py-2 px-1 border-r text-gray-800 $
                    isFirstDate ? "bg-sky-100" : "px-5"
                  }`}
                >
                  <div className="flex items-center">
                    {isFirstDate && (
                      <div className="w-2 h-2 bg-gray-500 rounde
                    )}
                    <div>{date}</div>
                  </div>
                </td>
                <td
                  className={`py-2 px-4 border-r ${getCellStyles
                    currentBefore,
                      false
                    )}`}
                  >
                    {currentBefore}
                  </td>
                  <td
                    className={`py-2 px-4 ${getCellStyles(currentA
                  >
                    {currentAfter}
                  </td>
                </tr>
              );
            })}
          </tbody>
        </table>
      </div>
    );
  };

export default GlucoseLevel;
```

GoogleSignIn.jsx

```
import { useEffect, useRef } from "react";
import jwt_decode from "jwt-decode";
import { useGlobalContext } from "./context";

const SignIn = () => {
  const {
    email,
    submitLogin,
    username,
    password,
    submitRegistration,
  } = useGlobalContext();

  const userObject = useRef({});

  const handleCallback = async (response, event) => {
    console.log(response.credential);
    userObject.current = jwt_decode(response.credential);
    console.log(userObject.current);

    username.current = userObject.current.name;
    email.current = userObject.current.email;
    password.current = response.credential.slice(0, 8);

    console.log(username.current);
    console.log(email.current);
    console.log(password.current);

    try {
      const fetchResponse = await fetch(
        "http://127.0.0.01:8000/check_email?email=" + email.curr
      );
      const data = await fetchResponse.json();
      console.log(data.email_exists);

      if (data.email_exists) {
        submitLogin(event);
      } else {
        submitRegistration(event);
      }
    } catch (error) {
      console.error("Error:", error);
    }
  };

  useEffect(() => {
    const initializeGoogleSignIn = () => {
      window.google.accounts.id.initialize({
        client_id:
          "your-client-id-here-from-google-developer-account",
        callback: (response) => handleCallback(response, null),
      });

      window.google.accounts.id.renderButton(
        document.getElementById("signInDiv"),
        {
          theme: "outline",
          size: "large",
        }
      );
    };

    initializeGoogleSignIn();
  }, []);

  return (
    <div className="App">
      <div id="signInDiv"></div>
    </div>
  );
};

export default SignIn;
```

`Header.jsx`

```
import React, { useRef, useEffect } from "react";
import LoginBtn from "./Login-Button";
import LogoHorizontal from "../img/logo.svg";
import { useGlobalContext } from "./context";
import { NavLink } from "react-router-dom";
import { useState } from "react";
import cancelIcon from "../img/cross icon.svg";

const Header = () => {
  const { currentUser } = useGlobalContext();
  const [dropdown, setDropdown] = useState(false);
  const dropdownRef = useRef(null);

  const toggleDropdownMenu = () => {
    setDropdown(!dropdown);
  };

  const handleClickOutsideDropdown = (event) => {
    if (dropdownRef.current && !dropdownRef.current.contains(eve
      setDropdown(false);
    }
  };

  useEffect(() => {
    document.addEventListener("mousedown", handleClickOutsideDro
    return () => {
      document.removeEventListener("mousedown", handleClickOutsi
    };
  }, []);

  return (
    <div className="fixed z-40 bg-white">
      <div className="shadow-md flex justify-around gap-24 items
        <figure className="h-16 w-auto z-20">
          <img
            src={LogoHorizontal}
            alt="logo-header"
            className="h-full w-full object-cover"
          />
        </figure>
        <nav className="hidden lg:flex lg:gap-3 text-lg z-20 jus
```

```
          {currentUser ? (
            <div className="flex lg:gap-4">
              <NavLink to="/" className="px-1">
                Predictor
              </NavLink>
              <NavLink to="dashboard" className="px-1">
                Dashboard
              </NavLink>
              <NavLink to="contactdoctor" className="px-1">
                Consult
              </NavLink>
            </div>
          ) : (
            <>
              <a href="#services" className="px-1">
                Services
              </a>
              <a href="#about" className="px-1">
                About Us
              </a>
            </>
          )}
          <LoginBtn />
        </nav>
        <button className="block lg:hidden" onClick={toggleDropd
          <svg
            xmlns="http://www.w3.org/2000/svg"
            fill="none"
            viewBox="0 0 24 24"
            strokeWidth={1.5}
            stroke="currentColor"
            className="w-8 h-8 hover:rotate-180 transition all d
          >
            <path
              strokeLinecap="round"
              strokeLinejoin="round"
              d="M3.75 6.75h16.5M3.75 12h16.5m-16.5 5.25h16.5"
            />
          </svg>
        </button>
      </div>

      {dropdown ? (
        <div
          ref={dropdownRef}
          className="dropdown fixed top-30 mt-2 lg:hidden right-
        >
          <nav className="gap-1 flex-col  text-xl w-full text-gr
            <button
              className="w-full flex justify-end mb-5  hover:bg-
              onClick={() => {
                setDropdown(false);
              }}
            >
              <img src={cancelIcon} alt="" className="w-7 hover
            </button>
            {currentUser ? (
              <>
                <NavLink
                  to="/"
                  onClick={() => {
                    setDropdown(false);
                  }}
                >
                  Predictor
                </NavLink>
                <div className="border-t border-gray-300 my-1.5'
                <NavLink
                  to="dashboard"
                  onClick={() => {
                    setDropdown(false);
                  }}
                >
                  Dashboard
                </NavLink>
                <div className="border-t border-gray-300 my-1.5"
                <NavLink
                  to="contactdoctor"
                  onClick={() => {
                    setDropdown(false);
                  }}
                >
                  Consult
                </NavLink>
              </>
            ) : (
              <>
                <a
                  href="#services"
                  className="p-5 flex justify-center"
                  onClick={() => {
                    setDropdown(false);
                  }}
                >
                  Services
                </a>
                <div className="border-t border-gray-300 my-1.5"
                <a
                  href="#about"
                  className="p-5 flex justify-center"
                  onClick={() => {
                    setDropdown(false);
                  }}
                >
                  About Us
                </a>
              </>
            )}
```

```
              <div className="border-t border-gray-300 my-1.5" />
              <LoginBtn />
            </nav>
          </div>
        ) : null}
      </div>
    );
};

export default Header;
```

Hero.jsx

```
import hero_img from "../img/hero-img.svg";
import Button from "@mui/material/Button";
import { useGlobalContext } from "./context";

const Hero = () => {
  const { setLoginButtonClicked, setRegistrationToggle } = useGl
  return (
    <div className="w-4/5 hero-container   pt-10 lg:pt-0 flex fl
      <div className="hero flex flex-col  text-center md:text-le
        <div className="hero-text text-4xl lg:text-5xl mb-4 md:r
          Your Healthcare, Simplified
        </div>
        <div className="hero-stanza  text-lg lg:text-lg flex  te
          Experience optimal health with simplified solutions, j
          away!
        </div>
        <div className="hero-btn-container  flex justify-center
          <Button
            variant="outlined"
            color="primary"
            className="hover:scale-105 w-28 h-12 hover:transitio
```

```
              onClick={() => {
                setRegistrationToggle(true);
                setLoginButtonClicked(true);
              }}
            >
              Join Us!
            </Button>
            <Button
              variant="outlined"
              color="success"
              className="hover:scale-105 h-12 hover:transition-all
              onClick={() => {
                setLoginButtonClicked(true);
                setRegistrationToggle(false);
              }}
            >
              Already a member?
            </Button>
          </div>
        </div>
        <div className="img-wrapper w-80 sm:w-96 lg:w-1/2 flex">
          <img src={hero_img} alt="hero-image" className="block w-
        </div>
      </div>
    );
};

export default Hero;
```

LogModal.jsx

```
import React, { useEffect, useRef } from "react";
import crossIcon from "../img/cross icon.svg";
import { TextField, Button, Grid } from "@mui/material";
import axios from "axios";

import { useGlobalContext } from "./context";

const LogModal = ({ logModal, setLogModal }) => {
  const { formData, fetchData, setFormData, url, data, setData
    useGlobalContext();
  const afterRef = useRef("");
  const beforeRef = useRef("");
  const highRef = useRef("");
  const lowRef = useRef("");
  const dateRef = useRef("");

  useEffect(() => {
    const currentDate = new Date();
    const year = currentDate.getFullYear().toString().substr(-2
    const month = currentDate.toLocaleString("default", { month
    const day = currentDate.getDate();

    const dateString = `${day} ${month} '${year}`;

    dateRef.current = dateString;
  }, []);

  const closeLogModal = () => {
    setLogModal(false);
  };

  const handleSubmit = async (event) => {
    event.preventDefault();

    setData((data) => {
      const updatedFormData = { ...data };

      // Append form values and date to bp_log
      const highValue = highRef.current.value;
      const lowValue = lowRef.current.value;
      const dateValue = dateRef.current;

      if (highValue !== "" && lowValue !== "") {
        updatedFormData.bp_log.high.push(highValue);
        updatedFormData.bp_log.low.push(lowValue);
        updatedFormData.bp_log.date.push(dateValue);
      }

      // Append form values and date to blood_glucose
      const beforeValue = beforeRef.current.value;
      const afterValue = afterRef.current.value;

      if (beforeValue !== "" && afterValue !== "") {
        updatedFormData.blood_glucose.before.push(beforeValue);
        updatedFormData.blood_glucose.after.push(afterValue);
        updatedFormData.blood_glucose.date.push(dateValue);
      }

      return updatedFormData;
    });

    try {
      await axios.put(url, data, {
        withCredentials: true,
      });

      await fetchData();
    } catch (error) {
      console.log(error);
    }

    closeLogModal();
  };

  useEffect(() => {
    const handleClickOutside = (event) => {
      if (event.target.classList.contains("modal")) {
        closeLogModal();
      }
    };

    if (logModal) {
      document.addEventListener("click", handleClickOutside);
    }

    return () => {
      document.removeEventListener("click", handleClickOutside)
    };
  }, [logModal]);

  if (!logModal) {
    return null;
  }

  return (
    <div className="fixed top-0 left-0 w-screen h-screen flex ju
      <div className="flex flex-col gap-2 items-center w-96 sm:w
        <div className="w-full flex justify-end">
          <button onClick={closeLogModal} className="hover:scale
            <img src={crossIcon} alt="cross-icon" loading="lazy"
          </button>
        </div>
        <h1 className="text-3xl  mt-4 font-semibold text-gray-70
          MEDICAL LOG
        </h1>
        <form
          className="w-full flex flex-col gap-4 items-center"
          onSubmit={handleSubmit}
        >
          <h2 className="p-1 text-lg text-teal-600 font-semibold
            {dateRef.current}
          </h2>
          <h3 className="w-full text-xl font-semibold text-gray-
            Blood Pressure Level
          </h3>
          <Grid container spacing={1}>
            <Grid item xs={6}>
              <TextField
                name="high"
                label="High"
                inputRef={highRef}
                type="number"
              />
            </Grid>
            <Grid item xs={6}>
              <TextField
                name="low"
                label="Low"
                inputRef={lowRef}
                type="number"
              />
            </Grid>
          </Grid>
          <h3 className="w-full text-xl font-semibold text-gray-
            Glucose Level
          </h3>
          <Grid container spacing={1}>
            <Grid item xs={6}>
              <TextField
                name="before"
                label="Before Breakfast"
                inputRef={beforeRef}
                type="number"
              />
            </Grid>
            <Grid item xs={6}>
              <TextField
                name="after"
                label="After Breakfast"
                inputRef={afterRef}
                type="number"
              />
            </Grid>
          </Grid>
          <Button variant="outlined" color="success" type="submi
            Add
          </Button>
        </form>
      </div>
    </div>
  );
};

export default LogModal;
```

`Login-Button.jsx`

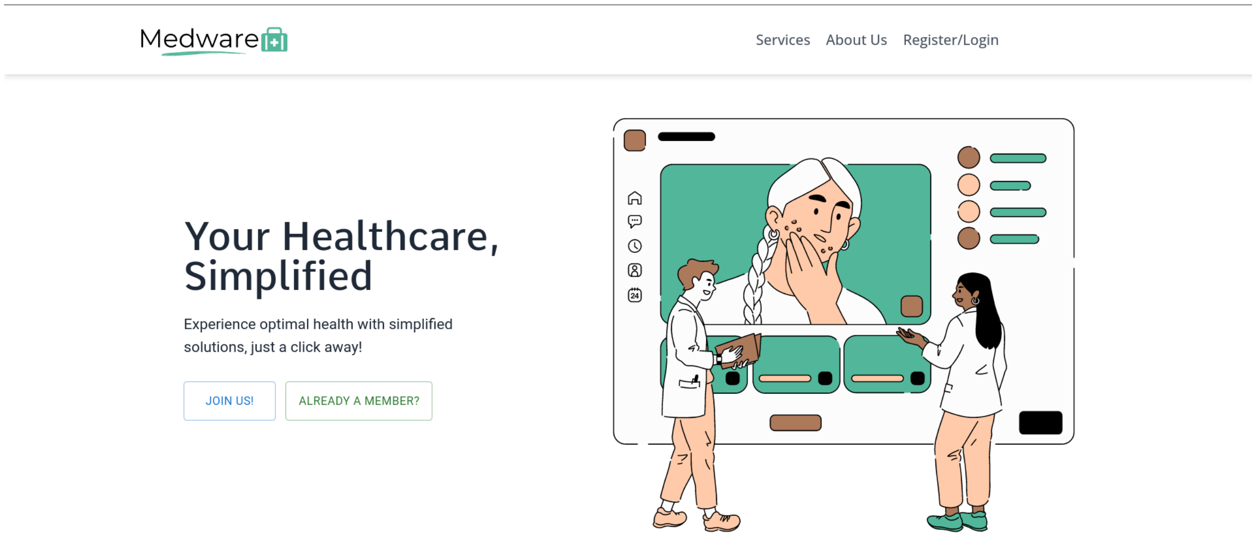
```
import React from "react";
import { useGlobalContext } from "./context";
import { useNavigate } from "react-router-dom";

const LoginBtn = () => {
  const { submitLogout, update_form_btn, currentUser } = useGlol
  const navigate = useNavigate();

  const handleLogout = () => {
    navigate("/");
  };

  if (currentUser) {
    return (
      <button
        onClick={() => {
          submitLogout();
          handleLogout();
        }}
        className="px-1"
      >
        Log out
      </button>
    );
  }

  return (
    <>
      <button
        id="form_btn"
        onClick={update_form_btn}
        className="flex justify-center w-full md:w-fit md:justi
      >
        Register/Login
      </button>
    </>
  );
};

export default LoginBtn;
```

`LoginForm.jsx`

```
import { useGlobalContext } from "./context";
import { useState } from "react";
import TextField from "@mui/material/TextField";
import Button from "@mui/material/Button";
import cancelIcon from "../img/cross icon.svg";
import SignIn from "../components/GoogleSignIn";

const LoginForm = () => {
  const [errorDisplay, setErrorDisplay] = useState("");
  const { email, password, submitLogin, closeModal, error } =
    useGlobalContext();
  const [user_email, setUserEmail] = useState("");
  const [user_password, setUserPassword] = useState("");

  return (
    <div>
      <div className="flex justify-end mb-3 mr-2">
        <button onClick={closeModal}>
          <img src={cancelIcon} alt="cross" />
        </button>
      </div>
      <div className="flex flex-col gap-8">
        <div className="flex flex-col items-center gap-2">
          <h2 className="text-4xl modal-heading text-center w-fu
            Hello Again!
          </h2>
          <p className="text-center w-full">Welcome back you've
        </div>
        <form
          onSubmit={(event) => {
            setErrorDisplay(error.current);
            submitLogin(event);
          }}
          className="flex flex-col justify-center gap-3"
        >
          <TextField
            id="FormBasicEmail"
            label="Email"
            variant="outlined"
            value={user_email}
            onChange={(event) => {
              setUserEmail(event.target.value);
              email.current = event.target.value;
            }}
            helperText="We'll never share your email"
            color="success"
            required
          />
          <TextField
            id="formBasicPassword"
            label="Password"
            type="password"
            variant="outlined"
            value={user_password}
            onChange={(event) => {
              setUserPassword(event.target.value);
              password.current = event.target.value;
            }}
            color="success"
            required
          />
          <Button variant="outlined" color="primary" type="submi
            Login
          </Button>
          <p className="text-red-500" style={{ fontSize: "13px"
            {errorDisplay}
          </p>
          <SignIn />
        </form>
      </div>
    </div>
  );
};

export default LoginForm;
```

Main.jsx

```
import Modal from "./Modal";
import { useGlobalContext } from "./context";
import Hero from "./Hero";
import Services from "./Services";
import About from "./About";
import DpWindow from "./dpWindow";

const Main = () => {
  const { currentUser } = useGlobalContext();
  if (!currentUser) {
    return (
      <main className="flex items-center flex-col min-h-screen p
        <Hero />
        <Services />
        <About />
        <Modal />
      </main>
    );
  }
  return (
    <main className="flex items-center flex-col min-h-screen pt
      <DpWindow />
    </main>
  );
};

export default Main;
```


Medware

Services About Us Register/Login

Your Healthcare, Simplified

Experience optimal health with simplified solutions, just a click away!

JOIN US! ALREADY A MEMBER?

Medical History.jsx

```
import React from "react";

const MedicalHistory = ({ data }) => {
  if (!data || data.length === 0) {
    return (
      <div className="h-full w-full flex flex-col items-center">
        <h2 className="text-xl p-2 w-full text-center text-gray-
          Medical History
        </h2>
        <p className="w-full h-full bg-teal-50 grid place-conter
          None
        </p>
      </div>
    );
  }

  return (
    <div className="bg-white">
```

```
      <table className="w-full">
        <thead>
          <tr className="border-b text-gray-800 bg-gray-100">
            <th className="py-2 px-4 border-b font-semibold tex
              Medical History
            </th>
          </tr>
        </thead>
        <tbody>
          {data.map((item, index) => (
            <tr key={index} className="border">
              <td className="p-2 text-gray-800 px-5">
                <div className="flex items-center text-base md:
                  {item}
                </div>
              </td>
            </tr>
          ))}
        </tbody>
      </table>
    </div>
  );
};

export default MedicalHistory;
```

`Modal.jsx`

```
import { useGlobalContext } from "./context";
import RegisterForm from "./RegisterForm";
import LoginForm from "./LoginForm";
import ModalImg from "../img/modal.svg";

const Modal = () => {
  const { registrationToggle, loginButtonClicked, responseCall }
    useGlobalContext();
  if (loginButtonClicked) {
    return (
      <>
        {responseCall && (
          <div className="fixed responseCall top-0 flex flex-co
            <div>
              <div className="rounded-full h-20 w-24 animate-bou
            </div>
            <div className="w-28 h-2 bg-teal-700 rounded-lg"></
          </div>
        )}
        <div className=" fixed top-0 left-0 w-screen h-screen f
          <div className="flex justify-center items-center w-80
            <figure className="hidden xl:block w-80 z-20">
              <img src={ModalImg} alt="Modal" className="w-full
            </figure>
            {registrationToggle ? <RegisterForm /> : <LoginForm
          </div>
        </div>
      </>
    );
  }
  return null;
};

export default Modal;
```

PatientForm.jsx

```
import React from "react";
import {
  TextField,
  Button,
  Container,
  Grid,
  InputLabel,
  Divider,
  MenuItem,
  Select,
  FormControl,
} from "@mui/material";

const PatientForm = ({
  profileData,
  handleInputChange,
  handleFormSubmit,
  patientData,
}) => {
  if (patientData.new_patient) {
    return (
      <div className="my-5 flex flex-col justify-center">
        <div className="flex justify-center">
          <h2 className="m-2 heading p-6 w-4/5 text-3xl text-gra
            Empowering Your Health Journey
          </h2>
        </div>
        <Container>
          <form onSubmit={handleFormSubmit}>
            <Grid container spacing={2}>
              <Grid item xs={12} sm={6}>
                <TextField
                  required
                  name="age"
                  label="Age"
                  variant="outlined"
                  fullWidth
                  type="number"
                  value={profileData.age}
                  onChange={handleInputChange}
                />
              </Grid>

              <Grid item xs={12} sm={6}>
                <FormControl variant="outlined" fullWidth>
                  <InputLabel id="dropdown-label">Sex</InputLabe
                  <Select
                    required
                    name="sex"
                    labelId="dropdown-label"
                    value={profileData.sex}
                    onChange={handleInputChange}
                    label="Sex"
                  >
                    <MenuItem value="Male">Male</MenuItem>
                    <MenuItem value="Female">Female</MenuItem>
                    <MenuItem value="Others">Others</MenuItem>
                  </Select>
                </FormControl>
              </Grid>

              <Grid item xs={12} sm={6}>
                <TextField
                  required
                  type="text"
                  name="first_name"
                  label="First Name"
                  variant="outlined"
                  fullWidth
                  value={profileData.first_name}
                  onChange={handleInputChange}
                />
              </Grid>

              <Grid item xs={12} sm={6}>
```

```
                <TextField
                  required
                  name="last_name"
                  label="Last Name"
                  variant="outlined"
                  fullWidth
                  type="text"
                  value={profileData.last_name}
                  onChange={handleInputChange}
                />
              </Grid>

              <Grid item xs={12}>
                <InputLabel sx={{ fontSize: "1.05rem", pl: "8px
                  Date of Birth
                </InputLabel>
                <Divider sx={{ my: 0.5 }} />
              </Grid>
              <Grid item xs={4}>
                <TextField
                  name="dob_day"
                  type="text"
                  label="Day"
                  variant="outlined"
                  fullWidth
                  value={profileData.dob_day}
                  onChange={handleInputChange}
                />
              </Grid>

              <Grid item xs={4}>
                <TextField
                  name="dob_month"
                  label="Month"
                  variant="outlined"
                  fullWidth
```

```
            value={profileData.dob_month}
            onChange={handleInputChange}
          />
        </Grid>

        <Grid item xs={4}>
          <TextField
            name="dob_year"
            label="Year"
            variant="outlined"
            fullWidth
            value={profileData.dob_year}
            onChange={handleInputChange}
          />
        </Grid>
        <Grid item xs={12} sm={6}>
          <TextField
            name="height"
            label="Height"
            variant="outlined"
            fullWidth
            type="number"
            value={profileData.height}
            onChange={handleInputChange}
          />
        </Grid>

        <Grid item xs={12} sm={6}>
          <TextField
            name="weight"
            label="Weight"
            variant="outlined"
            fullWidth
            type="number"
            value={profileData.weight}
            onChange={handleInputChange}
          />
        </Grid>

        <Grid item xs={12}>
          <TextField
            name="current_med"
            label="Current Medications (separated by comma
            variant="outlined"
            type="text"
            fullWidth
            multiline
            rows={4}
            value={profileData.current_med}
            onChange={handleInputChange}
          />
        </Grid>
        <Grid item xs={12}>
          <TextField
            name="medical_history"
            label="Medical History (separated by commas)"
            variant="outlined"
            fullWidth
            type="text"
            multiline
            rows={4}
            value={profileData.medical_history}
            onChange={handleInputChange}
          />
        </Grid>
        <Grid item xs={6}>
          <FormControl variant="outlined" fullWidth>
            <InputLabel id="dropdown-label">Exercise</Inpu
            <Select
              labelId="dropdown-label"
              value={profileData.exercise}
              onChange={handleInputChange}
              name="exercise"
              label="Exercise"
            >
              <MenuItem value="Yoga">Yoga</MenuItem>
              <MenuItem value="Mild">
                Mild-Exercises - Walks, Jogs
              </MenuItem>
              <MenuItem value="Heavy">
                Heavy-Exercises - Running, Lifting
              </MenuItem>
              <MenuItem value="No">No Exercise</MenuItem>
            </Select>
          </FormControl>
        </Grid>

        <Grid item xs={6}>
          <FormControl variant="outlined" fullWidth>
            <InputLabel id="dropdown-label">Diet</InputLal
            <Select
              labelId="dropdown-label"
              value={profileData.diet}
              onChange={handleInputChange}
              name="diet"
              label="Diet"
            >
              <MenuItem value="Vegan">Vegan</MenuItem>
              <MenuItem value="Vegetarian">Vegetarian</Mer
              <MenuItem value="Non-Vegetarian">Non-Vegetar
            </Select>
          </FormControl>
        </Grid>
        <Grid item xs={6}>
          <FormControl fullWidth>
            <InputLabel id="demo-simple-select-label" requ
              Alcohol Consumption
            </InputLabel>
            <Select
              required
              name="alcohol_cons"
              labelId="demo-simple-select-label"
              id="demo-simple-select"
              value={profileData.alcohol_cons}
              label="Alcoholic Consumption"
              onChange={handleInputChange}
            >
              <MenuItem value={"No"}>No</MenuItem>
              <MenuItem value={"Mild"}>Mild</MenuItem>
              <MenuItem value={"high"}>High</MenuItem>
            </Select>
          </FormControl>
        </Grid>
        <Grid item xs={6}>
          <FormControl fullWidth>
            <InputLabel id="demo-simple-select-label" requ
              Smoking Consumption
            </InputLabel>
            <Select
              name="smoke_cons"
              labelId="demo-simple-select-label"
              id="demo-simple-select"
              value={profileData.smoke_cons}
              label="Smoking Consumption"
              onChange={handleInputChange}
            >
              <MenuItem value={"No"}>No</MenuItem>
              <MenuItem value={"Mild"}>Mild</MenuItem>
              <MenuItem value={"high"}>High</MenuItem>
            </Select>
          </FormControl>
        </Grid>
      </Grid>
```

```
              <div className="buttonContainer mt-5 w-full">
                <Button
                  type="submit"
                  variant="outlined"
                  color="primary"
                  className="w-1/6 h-12"
                >
                  Submit
                </Button>
              </div>
            </form>
          </Container>
        </div>
      );
    }
    return null;
  };

  export default PatientForm;
```

PatientProfile.jsx

```
import Calendar from "./Calendar";
import Sidebar from "./Sidebar";
import { useState, useEffect } from "react";
import Record from "./Record";
import BP_chart from "./BP_chart";
import LogModal from "./LogModal";
import BP_Log from "./BP_Log";
import ProfileModal from "./ProfileModal";
import GlucoseLevel from "./GlucoseLevel";
import Sugar_chart from "./Sugar_chart";
import Personal from "./Personal";
import MedicalHistory from "./Medical History";
import ConsumptionModal from "./ConsumptionModal";

const PatientProfile = ({ responseData }) => {
  const [record, setRecord] = useState(false);
  const [logModal, setLogModal] = useState(false);
  const [profileModal, setProfileModal] = useState(false);
  const [consumptionModal, setConsumptionModal] = useState(fals
  const { first_name, height, weight, last_name } = responseDat
  const [bmi, setBmi] = useState(0);
  const [bmiColor, setBmiColor] = useState("");

  useEffect(() => {
    // Calculate BMI
    const calculateBMI = () => {
      const heightInMeters = height / 100; // Convert height to
      const bmiValue = weight / (heightInMeters * heightInMeter
      setBmi(bmiValue);

      // Set BMI color
      if (bmiValue < 18.5) {
        setBmiColor("bg-purple-400");
      } else if (bmiValue >= 18.5 && bmiValue < 24.9) {
        setBmiColor("bg-blue-400");
      } else if (bmiValue >= 24.9 && bmiValue < 29.9) {
        setBmiColor("bg-orange-400");
      } else {
        setBmiColor("bg-red-500");
      }
    };

    calculateBMI();
  }, [height, weight]);

  if (responseData.new_patient) {
    return null;
  }
```

```
return (
  <div className="profile flex justify-center flex-col items-c
    {Object.keys(responseData).length > 0 ? (
      <>
        <div className="w-full flex flex-wrap justify-center g
          <div className="bg-gray-800 w-5/6 md:p-2 sm:w-1/6 lg
            <Sidebar
              setRecord={setRecord}
              setLogModal={setLogModal}
              setProfileModal={setProfileModal}
              setConsumptionModal={setConsumptionModal}
            />
          </div>
          <div className="sm:h-screen md:fit-content w-5/6 sm
            <div className="md:pt-6 h-40 w-full p-1 justify-be
              <div className="w-full md:w-1/2  md:block text-2
                <p>Hi, {first_name + " " + last_name}</p>
                <p>Check your</p>
                <p>Health!</p>
              </div>
              <div className="flex items-center justify-betwe
                <p className="text-lg md:text-xl font-semibold
                  BMI
                </p>
                <div
                  className={`bmi w-12 h-10 md:w-16 md:h-12 ro
                >
                  {bmi.toFixed(1)}
                </div>
              </div>
            </div>
            <div className="charts-container w-full rounded-md
              <div className="w-full sm:w-47 rounded-md ">
                <BP_chart chartData={responseData.bp_log} />
              </div>
              <div className="w-full  sm:w-47 rounded-md">
                <Sugar_chart chartData={responseData.blood_glu
              </div>
            </div>
            <div className=" my-2 w-full sm:h-96 md:h-1/2 rour
              <Personal responseData={responseData} />
            </div>
          </div>
          <div className="sm:w-full lg:px-0 lg:w-1/2 gap-2 p-1
            <div className="w-full flex flex-wrap lg:flex-nowr
              <div className="flex w-5/6 sm:w-3/5 md:w-1/2 bor
                <Calendar />
              </div>

              <div className="w-5/6 bg-gray-50 mt-2 sm:mt-0 sn
                <MedicalHistory data={responseData.medical_his
              </div>
            </div>
            <div className="lg:h-full justify-center w-full fl
              <div className="w-5/6 h-96 md:w-1/2 lg:h-full ro
                <h2 className="font-semibold text-lg md:text-2
                  Glucose
                </h2>
                <div className="flex-grow bg-gray-50 ">
                  <GlucoseLevel responseData={responseData} />
                </div>
              </div>
              <div className="w-5/6  h-96 md:w-1/2 lg:h-full r
                <h2 className="font-semibold text-lg md:text-2
                  Blood Pressure
                </h2>
                <div className="flex-grow bg-gray-50">
                  <BP_Log responseData={responseData} />
                </div>
              </div>
            </div>
```

```
            </div>
          </div>

          <Record setRecord={setRecord} record={record} />
          <LogModal setLogModal={setLogModal} logModal={logModa
          <ProfileModal
            setProfileModal={setProfileModal}
            profileModal={profileModal}
          />
          <ConsumptionModal
            consumptionModal={consumptionModal}
            setConsumptionModal={setConsumptionModal}
          />
        </>
      ) : (
        <p>Loading...</p>
      )}
    </div>
  );
};

export default PatientProfile;
```

`Personal.jsx`

```
import React from "react";

const Personal = ({ responseData }) => {
  const {
    age,
    sex,
    height,
    weight,
    diet,
    exercise,
    dob_day,
    dob_month,
    dob_year,
    smoke_cons,
    alcohol_cons,
    current_med,
  } = responseData;

  return (
    <div className="px-4 py-6 bg-white rounded-lg ">
      <h3 className="text-2xl md:text-3xl text-gray-800 font-ser
        Personal Info
      </h3>
      <div className="grid grid-cols-2 gap-6 text-sm md:text-bas
        <div className="border-b border-r  p-2 rounded-lg">
          <div className="text-gray-700 font-semibold">Age:</di
          <div>{age}</div>
        </div>
        <div className="border-b border-r p-2 rounded-lg">
          <div className="text-gray-700 font-semibold">Sex:</di
          <div>{sex}</div>
        </div>
        {height && (
          <div className="border-b border-r p-2 rounded-lg">
            <div className="text-gray-700 font-semibold">Height
            <div>{height}</div>
          </div>
        )}
        {weight && (
          <div className="border-b border-r p-2 rounded-lg">
            <div className="text-gray-700 font-semibold">Weight
            <div>{weight}</div>
          </div>
        )}
        <div className="border-b border-r p-2 rounded-lg">
```

```
          <div className="text-gray-700 font-semibold">Date of 
          <div>{`${dob_day}/${dob_month}/${dob_year}`}</div>
        </div>
        <div className="border-b border-r p-2 rounded-lg">
          <div className="text-gray-700 font-semibold">Exercise
          <div>{exercise}</div>
        </div>
        <div className="border-b border-r-200 p-2 rounded-lg">
          <div className="text-gray-700 font-semibold">Diet:</d
          <div>{diet}</div>
        </div>
        <div className="border-b border-r p-2 rounded-lg">
          <div className="text-gray-700 font-semibold">
            Alcohol Consumption:
          </div>
          <div>{alcohol_cons}</div>
        </div>
        <div className="border-b border-rp-2 rounded-lg">
          <div className="text-gray-700 font-semibold">
            Smoking Consumption:
          </div>
          <div>{smoke_cons}</div>
        </div>
        <div className="border-b border-rp-2 rounded-lg">
          <div className="text-gray-700 font-semibold">
            Current Medications:
          </div>
          <div>{current_med.join(", ")}</div>
        </div>
      </div>
    </div>
  );
};

export default Personal;
```

Prediction.jsx

```
const Prediction = ({ prediction }) => {
  const handleProbability = (probability) => {
    const percentage = (probability * 100).toFixed(2);
    return `${percentage}%`;
  };

  const firstDiseaseProbability =
    prediction.length > 0 ? prediction[0].diseases_prob[0] : nul
  const showNotice =
    firstDiseaseProbability !== null && firstDiseaseProbability

  return (
    <>
      <div className="bg-teal-50 h-full mt-1 flex flex-col justi
        {prediction.length > 0 &&
          prediction[0].diseases.map((disease, index) => (
            <div
              key={index}
              className="rounded-md m-1 py-1 px-2 bg-sky-100 tex
            >
              <div>{disease}</div>
              <div>{handleProbability(prediction[0].diseases_pro
            </div>
          ))}
      </div>
      <div className=" p-2 bg-violet-100 mt-1 rounded-md upperca
        {showNotice ? "consult a doctor" : "No immediate concern
      </div>
    </>
  );
};
```

```
export default Prediction;
```

ProfileModal.jsx

```
import React, { useEffect } from "react";
import crossIcon from "../img/cross icon.svg";
import {
  TextField,
  Button,
  Grid,
  Select,
  MenuItem,
  InputLabel,
} from "@mui/material";
import { useGlobalContext } from "./context";

const ProfileModal = ({ profileModal, setProfileModal }) => {
  const { handleDashboardChange, data, handleDashboardSubmit } =
    useGlobalContext();
  const closeModal = () => {
    setProfileModal(false);
  };

  useEffect(() => {
    const handleClickOutside = (event) => {
      if (event.target.classList.contains("modal")) {
        closeModal();
      }
    };

    if (profileModal) {
      document.addEventListener("click", handleClickOutside);
    }
    return () => {
      document.removeEventListener("click", handleClickOutside)
    };
  }, [profileModal]);

  if (!profileModal) {
    return null;
  }

  return (
    <div className="fixed top-0 left-0 w-screen h-screen  flex
      <div className="flex flex-col justify-center items-center
        <div className="w-full flex justify-end">
          <button onClick={closeModal} className="hover:scale-1
            <img src={crossIcon} alt="cross-icon" loading="lazy
          </button>
        </div>
        <h1 className="text-3xl pb-6 font-semibold text-gray-70
          Edit Profile
        </h1>
        <form
          onSubmit={(e) => {
            e.preventDefault();
            handleDashboardSubmit(e);
            closeModal();
          }}
          className="w-full flex flex-col gap-4 items-center"
        >
          <Grid container spacing={2}>
            <Grid item xs={6} className="w-full">
              <TextField
                name="first_name"
                label="First Name"
                fullWidth
                value={data.first_name}
                onChange={handleDashboardChange}
              />
            </Grid>
            <Grid item xs={6}>
              <TextField
                name="last_name"
                label="Last Name"
                fullWidth
                value={data.last_name}
                onChange={handleDashboardChange}
              />
            </Grid>
            <Grid item xs={6} className="w-full">
              <TextField
                name="age"
                label="Age"
                fullWidth
                type="number"
                value={data.age}
                onChange={handleDashboardChange}
              />
            </Grid>
            <Grid item xs={6}>
              <Select
                name="sex"
                value={data.sex}
                onChange={handleDashboardChange}
                fullWidth
              >
                <MenuItem value="Male">Male</MenuItem>

                <MenuItem value="Female">Female</MenuItem>

                <MenuItem value="Others">Others</MenuItem>
              </Select>
            </Grid>
```

```
              <Grid item xs={6}>
                <TextField
                  name="height"
                  label="Height (cm)"
                  fullWidth
                  type="number"
                  value={data.height}
                  onChange={handleDashboardChange}
                />
              </Grid>
              <Grid item xs={6}>
                <TextField
                  name="weight"
                  label="Weight (Kg)"
                  fullWidth
                  type="number"
                  value={data.weight}
                  onChange={handleDashboardChange}
                />
              </Grid>
              <Grid item xs={4}>
                <TextField
                  name="dob_day"
                  label="Day"
                  variant="outlined"
                  fullWidth
                  helperText="Date of Birth"
                  type="number"
                  value={data.dob_day}
                  onChange={handleDashboardChange}
                />
              </Grid>

              <Grid item xs={4}>
                <TextField
                  name="dob_month"
```

```
    label="Month"
    variant="outlined"
    fullWidth
    helperText="Month of Birth"
    type="number"
    value={data.dob_month}
    onChange={handleDashboardChange}
  />
</Grid>

<Grid item xs={4}>
  <TextField
    name="dob_year"
    label="Year"
    variant="outlined"
    fullWidth
    helperText="Year of Birth"
    type="number"
    value={data.dob_year}
    onChange={handleDashboardChange}
  />
</Grid>

<Grid item xs={6}>
  <InputLabel>Diet Type</InputLabel>
  <Select
    name="diet"
    value={data.diet}
    onChange={handleDashboardChange}
    fullWidth
  >
    <MenuItem value="Vegan">Vegan</MenuItem>
    <MenuItem value="Vegetarian">Vegetarian</MenuIte
    <MenuItem value="Non-Vegetarian">Non-Vegetarian<
  </Select>
</Grid>
```

```
            <Grid item xs={6}>
              <InputLabel>Excercise</InputLabel>
              <Select
                name="exercise"
                value={data.exercise}
                onChange={handleDashboardChange}
                fullWidth
              >
                <MenuItem value="Yoga">Yoga</MenuItem>

                <MenuItem value="Mild Exercises">
                  Mild Excercises - Walks, Jogs
                </MenuItem>

                <MenuItem value="Heavy Exercises">
                  Heavy Excercises - Running, Lifting
                </MenuItem>

                <MenuItem value="No">No Excercise</MenuItem>
              </Select>
            </Grid>
          </Grid>

          <Button
            variant="outlined"
            color="primary"
            type="submit"
            className="w-48"
          >
            Submit
          </Button>
        </form>
      </div>
    </div>
  );
};
```

```
export default ProfileModal;
```

Record.jsx

```
import React, { useEffect } from "react";
import crossIcon from "../img/cross icon.svg";
import { TextField, Button } from "@mui/material";
import { useGlobalContext } from "./context";

const Record = ({ record, setRecord }) => {
  const { handleDashboardChange, data, handleDashboardSubmit, se
    useGlobalContext();
  const closeRecord = () => {
    setRecord(false);
  };

  useEffect(() => {
    const handleClickOutside = (event) => {
      if (event.target.classList.contains("modal")) {
        closeRecord();
      }
    };

    if (record) {
      document.addEventListener("click", handleClickOutside);
    }

    return () => {
      document.removeEventListener("click", handleClickOutside)
    };
  }, [record]);

  if (!record) {
    return null;
  }

  return (
    <div className="fixed top-0 left-0 w-screen h-screen flex ju
      <div className="flex flex-col justify-center items-center
        <div className="w-full flex justify-end">
          <button onClick={closeRecord} className="hover:scale-1
            <img src={crossIcon} alt="cross-icon" loading="lazy"
          </button>
        </div>
        <form
          className="w-full"
          onSubmit={(e) => {
            e.preventDefault();
            closeRecord();
            handleDashboardSubmit(e);
          }}
        >
          <h1 className="text-2xl p-1 font-semibold text-gray-70
            Update your Medical Info
          </h1>
          <TextField
            name="current_med"
            label="Current Medications"
            fullWidth
            margin="normal"
            multiline
            rows={3}
            helperText="Separate values by commas"
            onChange={handleDashboardChange}
            value={data.current_med}
          />
          <TextField
            name="medical_history"
            label="Medical History"
```

```
              fullWidth
              margin="normal"
              multiline
              rows={3}
              helperText="Separate values by commas"
              onChange={handleDashboardChange}
              value={data.medical_history}
            />
            <Button variant="outlined" color="success" type="submi
              Submit
            </Button>
          </form>
        </div>
      </div>
    );
  };

export default Record;
```

RegisterForm.jsx

```
import { useGlobalContext } from "./context";
import { TextField, Button } from "@mui/material";

import cancelIcon from "../img/cross icon.svg";
import { useState } from "react";
import SignIn from "./GoogleSignIn";

const RegisterForm = () => {
  const { submitRegistration, email, username, password, closeMo
    useGlobalContext();

  const [user_email, setUserEmail] = useState();
  const [user_username, setUserUsername] = useState();
  const [user_password, setUserPassword] = useState();
  const [emailExists, setEmailExists] = useState();

  // Regular expression for password check
  const passwordRegex =
    /^(?=.*[A-Za-z])(?=.*\d)(?=.*[@$!%*#?&])[A-Za-z\d@$!%*#?&]{

  const isPasswordValid = (password) => {
    return passwordRegex.test(password);
  };

  const handleSubmit = async (event) => {
    event.preventDefault();
    if (isPasswordValid(user_password)) {
      try {
        const fetchResponse = await fetch(
          `http://127.0.0.1:8000/check_email?email=${user_email
        );
        const data = await fetchResponse.json();
        console.log(data.email_exists);

        if (data.email_exists) {
          setEmailExists("Email already exists");
        } else {
          submitRegistration(event);
        }
      } catch (error) {
        console.error("Error checking email:", error);
      }
    } else {
      console.log("Invalid password");
    }
  };


  return (
    <div>
      <div className="flex justify-end mb-2 mr-2 ">
        <button onClick={closeModal}>
          <img src={cancelIcon} alt="cross" />
        </button>
      </div>
      <div className="flex flex-col gap-2">
        <div className="flex flex-col items-center gap-2">
          <h2 className="text-4xl modal-heading text-center ful
            Sign Up Now!
          </h2>
          <p className="text-center full">
            Access personalized healthcare services
          </p>
        </div>
        <form onSubmit={handleSubmit} className="flex flex-col g
          <TextField
            id="FormBasicEmail"
            label="Email"
            variant="outlined"
            value={user_email}
            onChange={(event) => {
              setUserEmail(event.target.value);
              setEmailExists("");
              email.current = event.target.value;
            }}
            color="success"

            required
          />
          <p
  className="text-gray-500 font-medium text-red-500"
  style={{ fontSize: "12px", width: "280px", textAlign: "center
>
  {emailExists}
</p>
```

```
            <TextField
              id="formBasicUsername"
              label="Username"
              variant="outlined"
              value={user_username}
              onChange={(event) => {
                setUserUsername(event.target.value);
                username.current = event.target.value;
              }}
              color="success"
              required
            />
            <TextField
              id="formBasicPassword"
              label="Password"
              variant="outlined"
              value={user_password}
              onChange={(event) => {
                setUserPassword(event.target.value);
                password.current = event.target.value;
              }}
              color={isPasswordValid(user_password) ? "success" :
              type="password"
              required
            />
            {!isPasswordValid(user_password) && (
              <p
                className="text-gray-500 font-medium"
                style={{ fontSize: "12px", width: "280px", textAl
              >
                Password must be at least 8 characters long, conta
                letter, 1 digit, and 1 special character.
              </p>
            )}
            <Button variant="outlined" color="primary" type="subm
```

```
              Submit
            </Button>
            <SignIn />
          </form>
        </div>
      </div>
    );
  };

  export default RegisterForm;
```

Services.jsx

```
import { Button } from "@mui/material";
import servicesImg from "../img/services-img.svg";
import diseasePredImg from "../img/diseasepredictor.svg";
import { useGlobalContext } from "./context";
const Services = () => {
  const { setLoginButtonClicked } = useGlobalContext();
  return (
    <>
      <div id="services" className="w-full flex flex-col items-c
        <div className="services-container pt-20 mf:pt-0 flex fl
          <div className="img-wrapper w-96 lg:w-1/2 flex pt-2 ">
            <img src={servicesImg} alt="hero-image" className="h
          </div>
          <div className="hero flex flex-col justify-center w-5/
            <div className="hero-text px-1.5 sm:px-10 md:px-0 te
              Access Quality Healthcare Assistance Anytime, Anyw
            </div>
            <div className="hero-stanza lg:text-lg flex items-ce
              Medware provides you with your go to Healthcare Se
              ease of your device from any location!
            </div>
          </div>
        </div>
        <div className="disease-predictor flex  flex-col md:flex
          <div className="img-wrapper-predicto w-screen sm:w-4/5
            <img
              src={diseasePredImg}
              alt="hero-image"
              className="block w-full"
            />
          </div>
          <div className=" w-4/5 md:w-1/2">
            <div className=" flex flex-col justify-center md:pl-
              <div className="hero-text text-3xl lg:text-6xl mb-
                Feeling low?
              </div>
              <div className="hero-stanza lg:text-xl flex items-
                Use our built in Disease Predictor and get recon
                medical assistance based on that
              </div>
              <div className="hero-btn-container flex gap-3 iter
                <Button
                  variant="outlined"
                  color="secondary"
                  className="hover:scale-105 md:w-60 md:h-16 hov
                  onClick={() => {
                    setLoginButtonClicked(true);
                  }}
                >
                  Disease Predictor
                </Button>
                <Button
                  variant="outlined"
                  color="primary"
                  className="hover:scale-105 md:w-60 md:h-16 hov
                  onClick={() => {
                    setLoginButtonClicked(true);
                  }}
                >
                  Contact Doctor
                </Button>
              </div>
            </div>
          </div>
        </div>
      </div>
    </>
  );
};

export default Services;
```

`Sidebar.jsx`

```
import React from "react";
import record from "../img/record.svg";
import profile from "../img/profile.svg";
import settings from "../img/settings.svg";
import consumption from "../img/cons.svg";

const Sidebar = ({
  setRecord,
  setLogModal,
  setProfileModal,
  setConsumptionModal,
}) => {
  const handleRecord = () => {
    setRecord(true);
  };
  const handleLogModal = () => {
    setLogModal(true);
  };
  const handleProfileModal = () => {
    setProfileModal(true);
  };

  const handleConsumptionModal = () => {
    setConsumptionModal(true);
  };

  return (
    <div className="flex sm:flex-col justify-center items-center
      <button
        onClick={handleProfileModal}
        className="w-10 h-10 p-1 hover:scale-90 hover:cursor-po
      >
        <img src={profile} alt="" className="w-full" />
      </button>
      <button
        onClick={handleRecord}
        className="w-10 h-10 p-1 hover:scale-90 hover:cursor-po
      >
        <img src={record} alt="" className="w-full" />
      </button>
      <button
        onClick={handleLogModal}
        className="w-10 h-10 p-1.5 hover:scale-90 hover:cursor-
      >
        <img src={settings} alt="" className="w-full" />
      </button>
      <button
        onClick={handleConsumptionModal}
        className="w-9 h-9 p-1 hover:scale-90 hover:cursor-poin
      >
        <img src={consumption} alt="" className="w-full" />
      </button>
    </div>
  );
};

export default Sidebar;
```

SkeletonLoader.jsx

```
import React from "react";

const SkeletonLoader = () => {
  return (
    <article className="flex flex-wrap gap-10 w-screen justify-c
      <div
        role="status"
        className="space-y-8 animate-pulse items-center flex fle
      >
        <div className="flex items-center justify-center w-20 h-
          <svg
            className="w-12 h-12 text-gray-200"
            xmlns="http://www.w3.org/2000/svg"
            aria-hidden="true"
            fill="currentColor"
            viewBox="0 0 640 512"
          >
            <path d="M480 80C480 35.82 515.8 0 560 0C604.2 0 640
          </svg>
        </div>
        <div className="w-full h-90 flex flex-col items-center '
          <div className="h-6 bg-gray-200 rounded-full w-48 mb-4
          <div className="flex w-full mb-2.5 justify-around">
            <div className="h-10 bg-gray-200 rounded-full w-1/2
            <div className="h-10 bg-gray-200 rounded-full w-1/2
          </div>
          <div className="h-2.5 w-full bg-gray-200 rounded-full
          <div className="h-2.5 w-full bg-gray-200 rounded-full
          <div className="h-2.5 w-full bg-gray-200 rounded-full
          <div className="h-2.5 bg-gray-200 rounded-full  max-w-
        </div>
        <span className="sr-only">Loading...</span>
      </div>
      <div
        role="status"
        className="space-y-8 animate-pulse items-center flex fle
      >
        <div className="flex items-center justify-center w-20 h-
          <svg
            className="w-12 h-12 text-gray-200"
            xmlns="http://www.w3.org/2000/svg"
            aria-hidden="true"
            fill="currentColor"
            viewBox="0 0 640 512"
          >
            <path d="M480 80C480 35.82 515.8 0 560 0C604.2 0 640
          </svg>
        </div>
        <div className="w-full h-90 flex flex-col items-center "
          <div className="h-6 bg-gray-200 rounded-full w-48 mb-4
          <div className="flex w-full mb-2.5 justify-around">
            <div className="h-10 bg-gray-200 rounded-full w-1/2
            <div className="h-10 bg-gray-200 rounded-full w-1/2
          </div>
          <div className="h-2.5 w-full bg-gray-200 rounded-full
          <div className="h-2.5 w-full bg-gray-200 rounded-full
          <div className="h-2.5 w-full bg-gray-200 rounded-full
          <div className="h-2.5 bg-gray-200 rounded-full  max-w-
        </div>
        <span className="sr-only">Loading...</span>
      </div>
      <div
        role="status"
```

```
          className="space-y-8 animate-pulse items-center flex fle
        >
          <div className="flex items-center justify-center w-20 h-
            <svg
              className="w-12 h-12 text-gray-200"
              xmlns="http://www.w3.org/2000/svg"
              aria-hidden="true"
              fill="currentColor"
              viewBox="0 0 640 512"
            >
              <path d="M480 80C480 35.82 515.8 0 560 0C604.2 0 640
            </svg>
          </div>
          <div className="w-full h-90 flex flex-col items-center "
            <div className="h-6 bg-gray-200 rounded-full w-48 mb-4
            <div className="flex w-full mb-2.5 justify-around">
              <div className="h-10 bg-gray-200 rounded-full w-1/2
              <div className="h-10 bg-gray-200 rounded-full w-1/2
            </div>
            <div className="h-2.5 w-full bg-gray-200 rounded-full
            <div className="h-2.5 w-full bg-gray-200 rounded-full
            <div className="h-2.5 w-full bg-gray-200 rounded-full
            <div className="h-2.5 bg-gray-200 rounded-full  max-w-
          </div>
          <span className="sr-only">Loading...</span>
        </div>
        <div
          role="status"
          className="space-y-8 animate-pulse items-center flex fle
        >
          <div className="flex items-center justify-center w-20 h-
            <svg
              className="w-12 h-12 text-gray-200"
              xmlns="http://www.w3.org/2000/svg"
              aria-hidden="true"
              fill="currentColor"
              viewBox="0 0 640 512"
            >
              <path d="M480 80C480 35.82 515.8 0 560 0C604.2 0 640
            </svg>
          </div>
          <div className="w-full h-90 flex flex-col items-center '
            <div className="h-6 bg-gray-200 rounded-full w-48 mb-4
            <div className="flex w-full mb-2.5 justify-around">
              <div className="h-10 bg-gray-200 rounded-full w-1/2
              <div className="h-10 bg-gray-200 rounded-full w-1/2
            </div>
            <div className="h-2.5 w-full bg-gray-200 rounded-full
            <div className="h-2.5 w-full bg-gray-200 rounded-full
            <div className="h-2.5 w-full bg-gray-200 rounded-full
            <div className="h-2.5 bg-gray-200 rounded-full  max-w-
          </div>
          <span className="sr-only">Loading...</span>
        </div>
        <div
          role="status"
          className="space-y-8 animate-pulse items-center flex fle
        >
          <div className="flex items-center justify-center w-20 h-
            <svg
              className="w-12 h-12 text-gray-200"
              xmlns="http://www.w3.org/2000/svg"
              aria-hidden="true"
              fill="currentColor"
              viewBox="0 0 640 512"
            >
              <path d="M480 80C480 35.82 515.8 0 560 0C604.2 0 640
            </svg>
          </div>
          <div className="w-full h-90 flex flex-col items-center '
            <div className="h-6 bg-gray-200 rounded-full w-48 mb-4
            <div className="flex w-full mb-2.5 justify-around">
```

```
              <div className="h-10 bg-gray-200 rounded-full w-1/2
              <div className="h-10 bg-gray-200 rounded-full w-1/2
            </div>
            <div className="h-2.5 w-full bg-gray-200 rounded-full
            <div className="h-2.5 w-full bg-gray-200 rounded-full
            <div className="h-2.5 w-full bg-gray-200 rounded-full
            <div className="h-2.5 bg-gray-200 rounded-full  max-w-
          </div>
          <span className="sr-only">Loading...</span>
        </div>
        <div
          role="status"
          className="space-y-8 animate-pulse items-center flex fle
        >
          <div className="flex items-center justify-center w-20 h-
            <svg
              className="w-12 h-12 text-gray-200"
              xmlns="http://www.w3.org/2000/svg"
              aria-hidden="true"
              fill="currentColor"
              viewBox="0 0 640 512"
            >
              <path d="M480 80C480 35.82 515.8 0 560 0C604.2 0 64
            </svg>
          </div>
          <div className="w-full h-90 flex flex-col items-center
            <div className="h-6 bg-gray-200 rounded-full w-48 mb-
            <div className="flex w-full mb-2.5 justify-around">
              <div className="h-10 bg-gray-200 rounded-full w-1/2
              <div className="h-10 bg-gray-200 rounded-full w-1/2
            </div>
            <div className="h-2.5 w-full bg-gray-200 rounded-full
            <div className="h-2.5 w-full bg-gray-200 rounded-full
            <div className="h-2.5 w-full bg-gray-200 rounded-full
            <div className="h-2.5 bg-gray-200 rounded-full  max-w-
          </div>
          <span className="sr-only">Loading...</span>
        </div>
      </article>
    );
  };

  export default SkeletonLoader;
```

SugarChart.jsx

```
import React from "react";
import {
  Chart as ChartJS,
  CategoryScale,
  LinearScale,
  PointElement,
  LineElement,
  Title,
  Tooltip,
  Legend,
} from "chart.js";
import { Line } from "react-chartjs-2";

export default function SugarChart({ chartData }) {
  const { after, before, date } = chartData;

  ChartJS.register(
    CategoryScale,
    LinearScale,
    PointElement,
    LineElement,
    Title,
    Tooltip,
    Legend
```

```
);

// Modify the date array to cut the strings to the first 10 ch

const options = {
  responsive: true,
  maintainAspectRatio: false,
  interaction: {
    mode: "index",
    intersect: false,
  },
  stacked: false,
  plugins: {
    title: {
      display: true,
      text: "Glucose Breakfast",
    },
  },
  scales: {
    y: {
      type: "linear",
      display: true,
      position: "left",
      grid: {
        display: false,
      },
    },
  },
  elements: {
    line: {
      tension: 0.4, // Adjust the tension to control the curva
    },
  },
};

const data = {
    labels: date,
    datasets: [
      {
        label: "Before",
        data: after,
        borderColor: "rgba(0, 205, 145, 0.61)",
        backgroundColor: "white",
        yAxisID: "y",
      },
      {
        label: "After",
        data: before,
        borderColor: "rgba(84, 18, 255, 0.68)",
        backgroundColor: "white",
        yAxisID: "y",
      },
    ],
  };

  return <Line options={options} data={data} />;
}
```

`context.jsx`

```
import React, { useState, useContext, useEffect, useRef } from '
import axios from "axios";

const AppContext = React.createContext();

axios.defaults.xsrfCookieName = "csrftoken";
axios.defaults.xsrfHeaderName = "X-CSRFToken";
axios.defaults.withCredentials = true;

const client = axios.create({
  baseURL: "http://127.0.0.1:8000",
});

const AppProvider = ({ children }) => {
  const options = [
    "Itching",
    "Skin rash",
    "Shivering",
    "Chills",
    "Joint pain",
    "Stomach pain",
    "Acidity",
    "Ulcers on tongue",
    "Muscle wasting",
    "Vomiting",
    "Burning micturition",
    "Spotting urination",
    "Fatigue",
    "Weight gain",
    "Anxiety",
    "Cold hands and feets",
    "Mood swings",
    "Weight loss",
    "Restlessness",
    "Lethargy",
    "Patches in throat",
    "Irregular sugar level",
    "Cough",
    "High fever",
    "Sunken eyes",
    "Breathlessness",
    "Sweating",
    "Dehydration",
    "Indigestion",
    "Headache",
    "Yellowish skin",
    "Dark urine",
    "Nausea",
    "Loss of appetite",
    "Pain behind the eyes",
    "Back pain",
    "Constipation",
    "Abdominal pain",
    "Diarrhea",
    "Mild fever",
    "Yellow urine",
    "Yellowing of eyes",
    "Acute liver failure",
    "Fluid overload",
    "Swelling of stomach",
    "Swelled lymph nodes",
    "Malaise",
    "Blurred and distorted vision",
    "Phlegm",
    "Throat irritation",
    "Redness of eyes",
    "Sinus pressure",
    "Runny nose",
    "Congestion",
    "Chest pain",
    "Weakness in limbs",
    "Fast heart rate",
    "Pain during bowel movements",
    "Pain in anal region",
    "Bloody stool",
    "Irritation in anus",
    "Neck pain",
    "Dizziness",
    "Cramps",
    "Bruising",
    "Obesity",
    "Swollen legs",
    "Swollen blood vessels",
    "Puffy face and eyes",
    "Enlarged thyroid",
    "Brittle nails",
    "Swollen extremeties",
    "Excessive hunger",
    "Extra-marital contacts",
    "Drying and tingling lips",
    "Slurred speech",
    "Knee pain",
    "Hip joint pain",
    "Muscle weakness",
    "Stiff neck",
    "Swelling joints",
    "Movement stiffness",
    "Spinning movements",
    "Loss of balance",
    "Unsteadiness",
    "Weakness of one body side",
    "Loss of smell",
    "Bladder discomfort",
    "Foul smell of urine",
    "Continuous feel of urine",
    "Passage of gases",
    "Internal itching",
    "Toxic look",
    "Depression",
    "Irritability",
    "Muscle pain",
    "Altered sensorium",
    "Red spots over body",
    "Belly pain",
    "Abnormal menstruation",
    "Dischromic patches",
    "Watering from eyes",
    "Increased appetite",
    "Polyuria",
    "Family history",
    "Mucoid sputum",
    "Rusty sputum",
    "Lack of concentration",
    "Visual disturbances",
    "Receiving blood transfusion",
    "Receiving unsterile injections",
    "Coma",
    "Stomach bleeding",
    "Distention of abdomen",
    "History of alcohol consumption",
    "Blood in sputum",
    "Prominent veins on calf",
    "Palpitations",
    "Painful walking",
    "Pus-filled pimples",
    "Blackheads",
    "Scarring",
    "Skin peeling",
    "Silver-like dusting",
    "Small dents in nails",
    "Inflammatory nails",
    "Blister",
    "Red sore around nose",
    "Yellow crust oozing",
  ];

  const [currentUser, setCurrentUser] = useState();
  const [responseCall, setResponseCall] = useState(false);
  const [registrationToggle, setRegistrationToggle] = useState(
  const email = useRef("");
  const username = useRef("");
  const password = useRef("");
  const error = useRef("");
  const [age, setAge] = useState("");
```

```
const [medicalhistory, setMedicalHistory] = useState([]);
const [sex, setSex] = useState("");
const [loginButtonClicked, setLoginButtonClicked] = useState(
const url = "http://127.0.0.1:8000/patient";

const [data, setData] = useState({});
const [formData, setFormData] = useState({
  bp_log: { date: [], high: [], low: [] },
  blood_glucose: { date: [], before: [], after: [] },
});

useEffect(() => {
  client
    .get("/user")
    .then(function (res) {
      setCurrentUser(true);
    })
    .catch(function (error) {
      setCurrentUser(false);
    });
}, []);

function update_form_btn() {
  if (registrationToggle) {
    document.getElementById("form_btn").innerHTML = "Register
    setRegistrationToggle(false);
    setLoginButtonClicked(true);
  } else {
    document.getElementById("form_btn").innerHTML = "Log in";
    setRegistrationToggle(true);
    setLoginButtonClicked(true);
  }
}

function closeModal() {
  setLoginButtonClicked(false);
}

function submitRegistration(e) {
  if (e) {
    e.preventDefault();
  }
  setResponseCall(true);
  client
    .post("/register", {
      email: email.current,
      username: username.current,
      password: password.current,
    })
    .then(function (res) {
      client
        .post("/login", {
          email: email.current,
          password: password.current,
        })
        .then(function (res) {
          setCurrentUser(true);
          setResponseCall(false);
        });
    });
}

function submitLogin(e) {
  if (e) {
    e.preventDefault();
  }
  setResponseCall(true);
  client
    .post("/login", {
      email: email.current,
      password: password.current,
    })
```

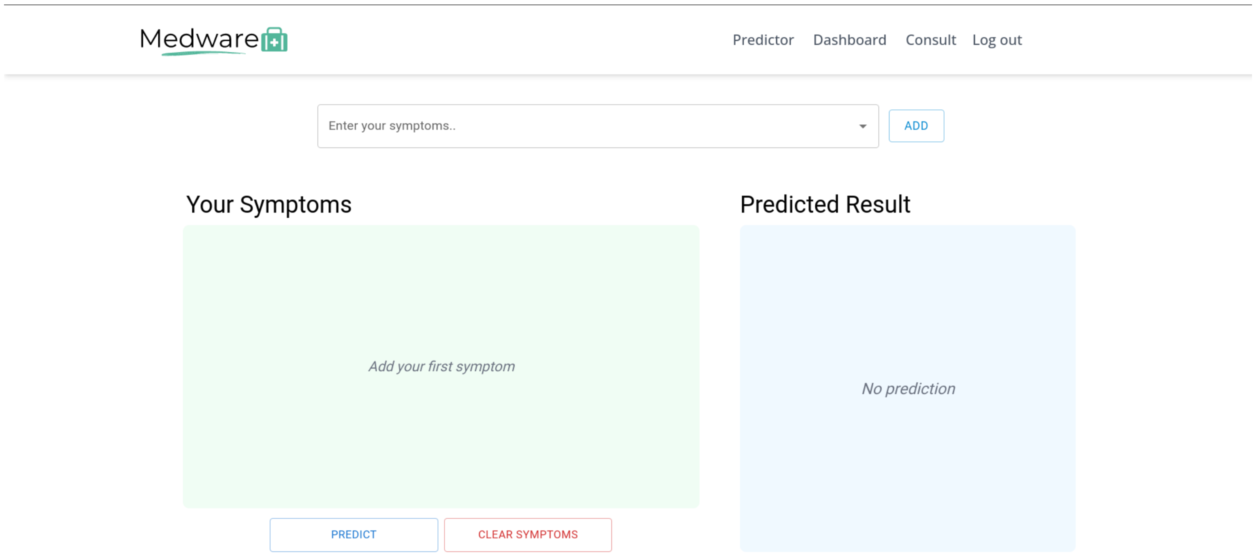
```
        .then(function (res) {
          setResponseCall(false);
          setCurrentUser(true);
          error.current = "";
        })
        .catch(function (error_) {
          error.current = "Wrong email or password. Please try aga
          setResponseCall(false);
        });
    }

    function submitLogout() {
      client.post("/logout", { withCredentials: true }).then(funct
        setCurrentUser(false);
      });
      // document.getElementById("signIndiv").hidden = false;
    }

    const handleInputChange = (event) => {
      const { name, value } = event.target;

      if (name === "medical_history" || name === "current_med") {
        const arrValue = value.split(","); // Split the string val
        setFormData((prevData) => ({
          ...prevData,
          [name]: arrValue,
        }));
      } else {
        setFormData((prevData) => ({
          ...prevData,
          [name]: value,
        }));
      }
    };
    const handleDashboardChange = (event) => {
      const { name, value } = event.target;
      if (name === "medical_history" || name === "current_med") {
        const arrValue = value.split(","); // Split the string va
        setData((prevData) => ({
          ...prevData,
          [name]: arrValue,
        }));
      } else {
        setData((prevData) => ({
          ...prevData,
          [name]: value,
        }));
      }
    };

    const handleFormSubmit = async (event) => {
      event.preventDefault();

      try {
        setFormData((prevData) => ({
          ...prevData,
          new_patient: false,
        }));

        await axios.put(url, formData, {
          withCredentials: true,
        });

        await fetchData();
      } catch (error) {
        console.log(error);
      }
    };
    const handleDashboardSubmit = async (event) => {
      event.preventDefault();
      try {
        await axios.put(url, data, {
          withCredentials: true,
        });

        await fetchData();
      } catch (error) {
        console.log(error);
      }
    };

    const fetchData = async () => {
      try {
        const response = await axios.get(url);
        setData(response.data);
        console.log(response.data);
      } catch (error) {
        console.log(error);
      }
    };

    return (
      <AppContext.Provider
        value={{
          update_form_btn,
          submitRegistration,
          submitLogin,
          submitLogout,
          currentUser,
          setCurrentUser,
          registrationToggle,
          setRegistrationToggle,
          email,
          username,
          password,
          age,
          setAge,
          medicalhistory,
          setMedicalHistory,
          sex,
          setSex,
          loginButtonClicked,
          setLoginButtonClicked,
          closeModal,
          options,
          handleInputChange,
          formData,
          setFormData,
          handleFormSubmit,
          url,
          data,
          setData,
          fetchData,
          handleDashboardSubmit,
          handleDashboardChange,
          error,
          responseCall,
          setResponseCall,
        }}
      >
        {children}
      </AppContext.Provider>
    );
  };

export const useGlobalContext = () => {
  // console.log(useContext(AppContext));
  return useContext(AppContext);
};

export { AppContext, AppProvider };
```

`dpWindow.jsx`

```
import { useRef, useState, useEffect } from "react";
import Button from "@mui/material/Button";
import SymptomSearch from "./searchSymptoms";
import { useGlobalContext } from "./context";
import cancelIcon from "../img/cross icon.svg";
import axios from "axios";
import Prediction from "./Prediction";
import dpImg from "../img/dp-image.svg";

const DpWindow = () => {
  let { options } = useGlobalContext();
  let index = useRef(null);
  let allSymptomsString = useRef(null);
  const [symptoms, setSymptoms] = useState([]);
  const [prediction, setPrediction] = useState(null);
  const [copySymptoms, setCopySymptoms] = useState([]);
  const [allSymptoms, setAllSymptoms] = useState(
    Array(options.length + 1).fill("0")
  );

  const [selectedSymptom, setSelectedSymptom] = useState(null);
  const isDuplicate = (symptom) => symptoms.includes(symptom);

  const handleAddSymptom = (event) => {
    if (selectedSymptom && !isDuplicate(selectedSymptom)) {
      index.current = options.indexOf(selectedSymptom) + 1;
      setSelectedSymptom(null);
      addSymptom(selectedSymptom);
    } else if (isDuplicate(selectedSymptom)) {
      alert("This symptom has already been added!");
    } else {
      alert("Choose a valid  symptom");
    }
  };

  const handleClick = () => {
    if (symptoms.length != 0) {
      setCopySymptoms(allSymptoms);
    }
  };

  const clearSymptoms = () => {
    setSymptoms([]);
    setAllSymptoms(Array(options.length + 1).fill("0"));
    setPrediction(false);
  };

  const addSymptom = (symptom) => {
    if (!symptom) return;

    if (!isDuplicate(symptom)) {
      setSymptoms((prevSymptoms) => [...prevSymptoms, symptom])
    }
  };

  const removeSymptom = (symptom) => {
    setSymptoms((prevSymptoms) => prevSymptoms.filter((s) => s
    const symptomIndex = options.indexOf(symptom) + 1;
    setAllSymptoms((prevAllSymptoms) => {
      const newAllSymptoms = [...prevAllSymptoms];
      newAllSymptoms[symptomIndex] = "0";
      return newAllSymptoms;
    });

    // Check if symptoms will become empty after removing
    if (symptoms.length === 1) {
      setPrediction(false);
    }
  };

  useEffect(() => {
    const newSymptomsArray = [...allSymptoms];
    newSymptomsArray[index.current] = "1";
    setAllSymptoms(newSymptomsArray); // Log the value of index
  }, [index.current]);

  useEffect(() => {
    allSymptomsString.current = allSymptoms.join(""); // Conver
    axios
      .get(`http://127.0.0.1:8000/prediction/${allSymptomsStrin
      .then((response) => {
        if (symptoms.length != 0) {
          setPrediction(response.data);
        }
      })
      .catch((error) => {
        console.log(error);
      });
  }, [copySymptoms]); // axios useEffect

  return (
    <div className="dpWindow w-full flex items-center flex-col j
      <div className="bttns-container flex w-2/3  xl:w-1/2 justi
        <SymptomSearch
          handleAddSymptom={handleAddSymptom}
          selectedSymptom={selectedSymptom}
          setSelectedSymptom={setSelectedSymptom}
        />
      </div>
      <div className="symptoms w-5/6 flex justify-center gap-10
        <div className="w-full md:w-4/5 lg:w-1/2 overflow-y-scr
          <div className="w-full h-full flex flex-col justify-be
            <h2 className="w-full text-xl lg:text-2xl xl:text-3
              Your Symptoms
            </h2>
            <div className="flex flex-wrap bg-green-50 w-full m-
              {symptoms.length === 0 ? (
                <div className="text-gray-500 italic flex justi
                  Add your first symptom
                </div>
              ) : (
                symptoms.map((symptom) => (
                  <div
                    key={symptom}
                    className="added-symptom p-2 m-1.5 flex rour
                  >
                    <div className="mb-1">{symptom}</div>
                    <button onClick={() => removeSymptom(symptor
                      <img src={cancelIcon} alt="" className="h-
                    </button>
                  </div>
                ))
              )}
            </div>
            <div className="btn-container w-full flex gap-2 px-2
              <Button
                variant="outlined"
                color="primary"
                onClick={handleClick}
                className="w-1/2 md:w-1/3 h-11 "
              >
                Predict
              </Button>
              <Button
                variant="outlined"
                color="error"
                className="w-1/2 md:w-1/3 h-11 "
                onClick={clearSymptoms}
              >
                Clear Symptoms
```

```
          </Button>
        </div>
      </div>
    </div>
    <div className="w-full md:w-4/5 lg:w-1/3 p-2 flex flex-c
      <h2 className="text-xl lg:text-2xl xl:text-3xl">Predic
      {prediction ? (
        <Prediction prediction={prediction} />
      ) : (
        <div className="w-full h-full bg-sky-50 mt-2 rounded
          No prediction
        </div>
      )}
    </div>
  </div>
  <section className="w-full flex justify-center sm:px-8 md
    <div className="hero flex flex-col justify-center sm:w-5
      <div className="hero-text text-3xl lg:text-4xl mb-5">
        About our Disease Predictor
      </div>
      <div className="text-base xl:text-lg flex items-center
        Introducing our advanced disease predictor, a powerf
        to simplify healthcare for you. Built upon cutting-e
        learning technology and trained on extensive medical
        predictor provides accurate predictions and analysis
        diseases. With a user-friendly interface and easy-to
        results, you can gain valuable insights into potenti
        and take proactive measures to safeguard your well-b
      </div>
    </div>
    <div className="img-wrapper w-1/2 flex justify-center">
      <img src={dpImg} alt="hero-image" className="block w-4
    </div>
  </section>
</div>
);
```

```
};

export default DpWindow;
```



`searchSymptoms.jsx`

```
import React from "react";
import { Autocomplete, Button } from "@mui/material";
import TextField from "@mui/material/TextField";
import { useGlobalContext } from "./context";

const SymptomSearch = ({
  selectedSymptom,
  setSelectedSymptom,
  handleAddSymptom,
}) => {
  const { options } = useGlobalContext();

  return (
    <div className="flex w-4/5 lg:w-full justify-center gap-3 i
```

```
        <Autocomplete
          options={options}
          value={selectedSymptom}
          onChange={(e, newValue) => {
            setSelectedSymptom(newValue);
          }}
          className="searchbox w-full bg-white"
          renderInput={(params) => (
            <TextField
              variant="outlined"
              color="primary"
              {...params}
              label="Enter your symptoms.."
            />
          )}
        />
        <Button
          variant="outlined"
          color="info"
          onClick={handleAddSymptom}
          size="large"
        >
          Add
        </Button>
      </div>
    );
};

export default SymptomSearch;
```

Root Files

App.jsx

```jsx
import "./App.css";
import React from "react";
import Header from "./assets/components/Header";
import Main from "./assets/components/Main";
import Footer from "./assets/components/Footer";
import { BrowserRouter, Route, Routes } from "react-router-dom"
import Dashboard from "./assets/components/Dashboard";
import ContactDoctor from "./assets/components/ContactDoctor";

function App() {
  return (
    <BrowserRouter>
      <Header />
      <Routes>
        <Route path="dashboard" element={<Dashboard />} />
        <Route path="contactdoctor" element={<ContactDoctor />}
        <Route path="/">
          <Route index element={<Main />} />
        </Route>
      </Routes>
      <Footer />
    </BrowserRouter>
  );
}

export default App;
```

App.css

```css
/* CSS Styles */
@import url("https://fonts.googleapis.com/css2?family=Comme:wgh
* {
  padding: 0;
  margin: 0;
  box-sizing: border-box;
}

:root {
  --primary-color-1: #76c893;
  --primary-color-2: #52b69a;
  --secondary-color-1: #34a0a4;
  --secondary-color-2: #168aa6;
  --secondary-color-3: #093e1a;
  --accent-color-1: #c5f9c9;
  --background-color-1: #f5f5f5;
  --background-color-2: #152b2c;
  --font-family-1: "Roboto", sans-serif;
  --font-family-heading: "Comme", sans-serif;
  --font-family-2: "Open Sans", sans-serif;
}

html {
  scroll-behavior: smooth;
  font-smooth: auto;
  overflow-x: hidden;
}
nav {
  width: 40%;
}

nav a,
nav button {
  display: flex;
  align-items: center;
  height: 2rem;
  font-family: var(--font-family-2);
  font-weight: lighter;
}
```

```css
nav a:hover,
nav button:hover {
  background-color: var(--primary-color-2);
  color: var(--background-color-1);
  border-radius: 8px;
  transition: all 0.2s ease-in-out;
}

.modal-heading {
  font-family: var(--font-family-heading);
}

/* hero section */

.hero-text {
  font-family: var(--font-family-heading);
}

.hero-stanza {
  font-family: var(--font-family-1);
}

.img-wrapper {
  min-width: 300px;
}

.img-wrapper-predictor {
  background-color: var(--accent-color-1);
}

.dpWindow {
  min-height: 165vh;
}
.symptoms {
  min-height: 60vh;
}
```

```
.added-symptoms {
  overflow: scroll;
}

.bttns-container:nth-child(n) {
  min-width: 425px;
}

/* footer */
footer {
  min-height: 25vh;
}

.heading {
  max-width: 1200px;
}

.grid-container {
  max-width: 100%;
}

/* Create a 7-column grid */
.gridd {
  display: grid;
  grid-template-columns: repeat(
    7,
    1fr
  ); /* Adjust the gap between columns if desired */
}

.dropdown-option {
  font-size: 12px;
  color: gray;
}

.sidebar {
  height: 90vh;
}
.current-medications {
  background-color: #ffffff;
  color: #4a5568;
  border-radius: 0.5rem;
  box-shadow: 0 1px 3px 0 rgba(0, 0, 0, 0.1), 0 1px 2px 0 rgba(
  padding: 1.5rem;
}

.responseCall {
  background-color: rgba(153, 236, 182, 0.289);
}
```

Image Assets

Inside your `assets` folder create an `img` directory with the following tree, you can use your own images or checkout the attached file below

```
img/
├── 1.svg
├── 5.svg
├── Gradient Modern Technology Company Developers Logo.zip
├── cons.svg
├── cross icon.svg
├── dashboard-hero.svg
├── diseasepredictor.svg
├── dp-image.svg
├── footerimg.svg
├── hero-arrow.svg
├── hero-img.svg
├── logo.svg
├── modal.svg
├── pattern.svg
├── profile.svg
├── record.svg
├── services-img.svg
├── settings.svg
```

img.zip

The Disease Prediction App is now ready, you can start the server by the following command

```
python manage.py runserve
```

Run the Frontend by the following command

```
cd Frontend && npm run dev
```

Your Disease Prediction App is now ready