**Willhelm International**

**# Wi\_international**

 Willhelm International Care

**## Description**

An appointment website designed for Willhelm International.

**## Tech stack**

Django(full stack yay!), Sql, Bootstrap, jQuery.

<!-- \*\*Webiste:\*\* [Deployed on Pythonanywhere](http://jasonchan.pythonanywhere.com) -->

**## Admin view:**

Administrator can be created only through terminal of server by using "./manage.py createsuperuser" command. An [administrator](username: test, password: test) has been created, please feel free to test the data intergrity and consistency.

1. Administrator has Create, Read, Update, and Delete control of the user.

2. Administrator has Create, Read, Update, and Delete control of the appointment.

3. Administrator has Create, Read, Update, and Delete control of the prescription.

**## Doctor view**

1. Prescribe medication (medication/dosage/duration).

2. View scheduled appointments

3. Schedule Follow up appointment in the future

4. See patient appointment and medication history

**## Patient View**

1. book appointments with specific doctors - view only current medication and dosage.

2. Book appointment by types(primary care, flu, mental care, and etc)

**# Django hooks**

**# Deploying options**

**## Deploying on AWS**

1. Deploying a Django application to Elastic Beanstalk

https://docs.aws.amazon.com/elasticbeanstalk/latest/dg/create-deploy-python-django.html

**### pros:**

Easy to see the backend interface

**### cons:**

Will need time to manage

Need a domain name

**## Deploying on Heroku**

**### pros:**

No need domain or dbms

Easy to setup and free usage

**### cons:**

https://devcenter.heroku.com/articles/django-app-configuration

'''

You're using the staticfiles app without having set the STATIC\_ROOT setting to a filesystem path.

'''

1. **Setting up for local deployment**

Assuming that Python 3.9 or up and Dip has been installed

**2. Cloning the repository**

--> Clone the repository using the command below :

git clone https://github.com/killerfrost22/Wi\_international

--> Move into the directory where we have the project files :

cd Wi\_international

--> Create a virtual environment :

# Let's install virtualenv first

pip install virtualenv

# Then we create our virtual environment

virtualenv envname

--> Activate the virtual environment :

envname\scripts\activate

--> Install the requirements :

pip install -r requirements.txt

**3.Setup Virtualenvironment**

py -m venv venv # Create virtual environment

source venv/bin/activate # Activate virtual environment

pip install -r requirements.txt # Install requirements

python server.py # Run server

deactivate # Deactivate virtual environment

**4.Migrate the database**

python manage.py makemigrations

python manage.py migrate --run-syncdb

5. **Running the App**

--> To run the App, we use :

python manage.py runserver

⚠ Then, the development server will be started at <http://127.0.0.1:8000/>

**Django Admin mode**

<http://127.0.0.1:8000/admin/login/?next=/admin/>

Admin: [ted@gmail.com](mailto:ted@gmail.com)

Pswd: josephao

Admin links admin functions and it links admin profile, it can also display change log

After running, the server and accessing the homepage, go to admin panel to add all the medicines,

their preparations and departments as they are all managed by the admin.

http://127.0.0.1:8000/admin

**Django patient account**

Username: [bucky@gmail.com](mailto:bucky@gmail.com)

Pswd: josephao

**Django doctor account**

Username: [da@gmail.com](mailto:da@gmail.com)

Pswd: josephao

Links doctor functions links to doctors profile Displays upcoming appts

**Change password**

In the command line python .\manage.py changepassword [bucky@gmail.com](mailto:bucky@gmail.com)

**DataBase Design:**

Diagram

Description automatically generated

Graphical user interface, application

Description automatically generated

**Overview of project:**

**Homepage**

Graphical user interface, application, website, Teams

Description automatically generated

Doctor viewGraphical user interface, application, website

Description automatically generated

Patients View:

Graphical user interface, application

Description automatically generated

Medicine ViewGraphical user interface, application

Description automatically generated

**What could be done better?**

1. **Google Calendar API**

**pip install google-api-python-client (from the requirements already)**

**I can avoid running the server and storing all the appointment data which in the end messed up my code and I have to remove all of the calendar functions to be able to run the code successfully. If I got more time I would change the original schema to be hosted on google calendar and make the patient to send request and the doctor will receive emails and doctors can either choose approve or choose a different time.**

1. **AWS Beanstalk**

**This part is the pain in my project. My original deployment was on Heroku however, my code of calendar was malfunction and I have to go back to ground zero. And Heroku refuses to work with me again. So I have to redirect to the AWS beanstalk which works but partially.**