Revitalize

About

This is one half of the Revitalize Virtual Wellness System. This repository contains the backend, written in Django, dealing with the API, authentication, database and the admin/labtech backend interface. The Revitalize Frontend can be found here

Contributors

- Allan Stewart
- Matthew Bowlan

Running the App

Initial Setup

It is recommended that you use pycharm for working with the backend.

First clone this repo, install python 3.7 and the Python package manager pip

If you have pycharm locate the terminal and type in the following commands

`./setup_environment.sh` This will take you through the initial setup including setting up the virtual environment, installing the required packages and will instruct you to run the following commands.

1. `./Scripts/reinitialize_database.sh` This will setup the database
2. `./manage.py createsuperuser` This will ask you for login information for the superuser
3. `./Scripts/do_migration.sh` This finishes the setup

To load surveys the following commands must be executed

`python ./manage.py shell` to enter a python command mode

`exec(open("./load_surveys.py").read())` will load the surveys into the database

Starting the Project

In pycharm there is a `Run` button in the top right corner, there will be a configuration called `start_server`, with the parameter `runserver`, that is selected. Clicking the Run button should start the server. Clicking the url that is shown in the run console will open your web browser where you can navigate to either `.../admin/` or `.../labtech/`.

Third Party Dependencies

Name	Version	Description

Name	Version	Description
asgiref	3.2.3	ASGI is a standard for Python asynchronous web apps and servers to communicate with each other, and positioned as an asynchronous successor to WSGI.
Django	3.0.3	Django is a high-level Python Web framework that encourages rapid development and clean, pragmatic design.
django-cors-headers	3.2.1	A Django App that adds Cross-Origin Resource Sharing (CORS) headers to responses. This allows in-browser requests to your Django application from other origins.
djangorestframework	3.11.0	Django REST framework is a powerful and flexible toolkit for building Web APIs.
djangorestframework-simplejwt	4.4.0	A JSON Web Token authentication plugin for the Django REST Framework.
Pillow	7.0.0	Python Imaging Library
PyJWT	1.7.1	Python library which allows you to encode and decode JSON Web Tokens (JWT)
sqlparse	0.3.1	a non-validating SQL parser module.

Features

API

The API is created using the Django REST Framework and it is how the frontend and backend communicate. 'serializers.py' allows for complex data types to be converted to native python

Authentication

Authentication is handled by simple jwt that works on top of the Django REST Framework This package gives a version of a JSON Web Token that is more secure than other common authentication methods that are included in django.

Database

The database is represented by the `models.py` file and can be described generally by looking at the `user` and `form` models.

User

`User` contains basic information including the username, password, and boolean flags such as `is_staff` Users are linked to a `profile` that contains much more detailed information about the person. The user model is also used to link a person to anything that involves them. This includes a `submissions` or a form of data point.

Form

`Form` contains the information of anything that can be described in the format of a form. This can be a `Survey` or a `MedicalLab` and will contain all the information needed for the instructions, questions, answers, and format of the form. The form model is also used to link a form to a `submission` so that the data saved in the submission can reference the form.

Submission

`Submission` contains a combination of data points and responses that can be referenced by what user the submission is for and the form that the submission represents.

Backend Interface

Django includes a very flexible admin interface that provides a user friendly way of interacting with database values. In our application there are two different admin interfaces divided into high level admin users and labtech users who have fewer permissions. These users are identified by the boolean fields `is_lab_tech`, `is_staff` and `is_superuser`.