DAS 2020 - TECHNICAL DOCUMENTATION

CONTENTS

1. INTRODUCTION

This documentation will guide you through setting up and maintaining your instance of DAS 2020.

It is important you go through this entire documentation. We will try to keep it as brief as possible. If you are having issues or are finding this documentation lacking detail, feel free to contact us - we will try our best to help (see section 4, Contact Us).

For reference, here are the technologies we have used:

- Django 2.2 with Python 3.7.4 - for the backend REST API
  https://docs.djangoproject.com/en/2.2/
- React 16.11.0 with TypeScript 3.6.4 - for the frontend user interface
  https://reactjs.org/versions

We used a simple local database for testing purposes, you may wish to use MySQL or PostgreSQL - we discuss this in more detail later.

2. SETUP

Tech Overview

Before continuing, explore the project high level directory structure, notice how the *Frontend* and *Backend* are in separate directories, their dependencies are in *frontend/package.json* and *backend/requirements.txt*. This will allow us to set them up indepedently and then combine them.

Note, you do NOT need to know how to use these technologies to setup DAS 2020.

For reference, React is a JavaScript library for building user interfaces. You can learn more here: https://reactjs.org. Node package manager (npm) is a package manager for maintaining our React Frontend dependencies. You can learn more here: https://www.w3schools.com/whatis/whatis_npm.asp.

Django is a high-level Python Web Framework for fast, secure, and scalable web applications. You can learn more here: https://www.djangoproject.com/.

Let's begin with the setup.

Step 1 - Ensure you have Node, NPM, Python, and PIP installed

DAS 2020 requires the four main technologies to be installed. Details on how to install these for your operating system can be found online.

We recommend you install the following versions as these were used for development:

- node v10.16.0
- npm v6.9.0
- python 3.7.4
- pip v19.3.1

Step 2 - Configuration

1. Open the hidden configuration file: **backend/.env**
2. This file contains various important settings for a proper set-up of the application. We need to setup three key areas - the database used to store data, the email account used to send emails and other crucial, more general settings.
    1. General
        1. Ensure **ENV=production** is set, this is the deployment scope, during development this is set to dev. If this is not set correctly, it may not connect to the database.
        2. Create a new secret key here https://djecrety.ir/. Ensure no one sees this key. Set **SECRET_KEY=your_newly_generated_secret_key**. If this is not set correctly, it allows for certain security vulnerabilities.
        3. Ensure **DEBUG=False** is set, this is very important, it ensures the Backend returns the appropriate sensitive data free error messages.
        4. Ensure **PRODUCTION_HOST_URL** is set to the URL from which you wish to

access the application, for example, if we wish to access the site homepage via chem.gla.ac.uk/das-project then we will set **PRODUCTION_HOST_URL=chem.gla.ac.uk/das-project**. This is what Django calls an Allowed Host, for further information on how Django handles Allowed Hosts, see https://docs.djangoproject.com/en/3.0/ref/settings/#std:setting-ALLOWED_HOSTS. If this is not set correctly, you will not be able to access the application.

2. Database
    1. Ensure you set the following five variables correctly:

       DB_ENGINE - this is the type of database, there are ONLY two options, these are:

        1. **DB_ENGINE=mysql**, or
        2. **DB_ENGINE=postgres**

       Depending on which you use, you must install **mysqlclient** or **psycopg2**. You can find instructions online. We recommend using PostgreSQL over MySQL.

       DB_NAME - this is the name of the database
       DB_USER - this is the username that can access the database
       DB_PASSWORD - this is the password associated with the username
       DB_HOST - this is the host address of the database
       DB_PORT - this is the port through which the database server will accept the connection

3. Emails
    1. Ensure you set the following five variables correctly:
       EMAIL_HOST - this is the name of the email server, e.g. smtp.gmail.com
       EMAIL_USE_TLS - whether email should use TLS, e.g. True
       EMAIL_PORT - this is the port number of the email server, e.g. 587
       EMAIL_ADDRESS - this is the email address from which emails will be sent
       EMAIL_PASSWORD - this is the password required to sign into EMAIL_ADDRESS

Once the environment file has been setup correctly, run the following commands in the **backend** directory:

1. **python manage.py makemigrations api** - This will create migrations based on our models
2. **python manage.py migrate** - This will apply the newly created migrations to your database - if an error is thrown here, it may be the Django server is failing to connect to the database. See https://docs.djangoproject.com/en/3.0/topics/migrations/ for more on migrations. You must fix these errors before continuing.
3. **python manage.py runserver** - This will run the server locally. Ensure this command returns no errors.```

If you see the following error `ValueError: invalid truth value 'update_this'` , you need to complete the setup of the environment file.

Step 4 - Headers and HTTPS

We recommend the following setup for Headers and HTTPS. These are not included in the code.

It is very important you understand the importance of correct headers to prevent certain attacks such as CSRF and XSS. It is very important to run this site over HTTPS as it contains sensitive data and user login credentials.

Please read Django documentation to learn what each of these does and what other options there are for Headers and HTTPS. This can be found here, https://docs.djangoproject.com/en/3.0/topics/security/.

You will need to add the flags (for example, the ones below) to the settings.py file which can be found in **backend/backend/settings.py**. Around **line 7** of the file you will see `# INSERT HEADERS AND HTTPS FLAGS HERE`, directly underneath this line create a few blank lines and insert your header flags there. One per line.

Example Header Flags in backend/backend/settings.py

```
# INSERT HEADERS AND HTTPS FLAGS HERE
SECURE_PROXY_SSL_HEADER = ('HTTP_X_FORWARDED_PROTO', 'https')
SESSION_COOKIE_SECURE = True
CSRF_COOKIE_SECURE = True
SECURE_SSL_REDIRECT = True
SECURE_HSTS_SECONDS = 31536000
SECURE_HSTS_INCLUDE_SUBDOMAINS = True
SECURE_CONTENT_TYPE_NOSNIFF = True
SECURE_BROWSER_XSS_FILTER = True
SECURE_HSTS_PRELOAD = True
```

Step 5 - Configure API URLs

This will ensure the Frontend will send requests to the correct Backend.

1. Open frontend/index.tsx
2. Search for `const STAGING_URL = "http://teamdas123.pythonanywhere.com";`. This line tells the Frontend that all REST API requests need to be sent to http://teamdas123.pythonanywhere.com/api/. Notice the **/api/**. This is the endpoint at which the REST API will be accessible when the Backend server is running.
3. Replace this URL with the URL the server will be accessible at, e.g. `const STAGING_URL = "http://www.chem.gla.ac.uk/das-project";`

Step 6 - Build React for Staging

1. Navigate to the **backend** directory, if there exists the a folder called **react-frontend**, delete it.
2. Ensure the environment file is setup correctly and **DEBUG=FALSE** is in **backend/.env** (as discussed in step 3)
3. Navigate to the **frontend** directory, if there exists the a folder called **build**, delete it.
4. Run the following commands within the **frontend** directory:
   `npm install` This will ensure the node_modules for React are up to date
   `npm run-script build-staging` This will build the frontend for Production and create a new folder called **build** within the **frontend** directory.
5. Move the **build** folder into the **backend** directory such that the following path exists: **backend/build/**.
6. Create a new folder in the **backend** directory called **react-frontend**
7. Move the build folder into the **react-frontend** directory.

We have built the frontend and stored the build files in the **backend/react-frontend/** directory.

Step 7 - Deployment

All the files within the backend directory can be deployed to your server.

The server will require static files to be in a certain location. It retrieves them from a URL. We must map the URL to the Directory.

Below is a table mapping URLs to Directories, this is an example of how our URLs were mapped during development.

URL	Directory
/react-frontend/build/static/admin/	/home/.virtualenvs/das_virtual_env/lib/python3.6/site-packages/django/contrib/admin/static/admin)
/static/	/home/backend/react-frontend/build/static

When Django requests for static files, e.g. from **/static/build/index.html**, these files are actually stored in **/home/backend/react-frontend/build/static/index.html**.

Step 8 - Super user

Once the application has been deployed, a superuser must be created via the command line on the server. This superuser will be the first user in the system and is responsible for creating accounts for other users and administrators. Once an administrator account has been created, that account has the privileges to create new accounts and access all the

administrator settings via the admin page, accessible from **/admin/**.

That is to say, this super user account must not be used on a regular basis but only for site maintenance.

We recommend creating the super user account and an admin account. Once confirmed the admin account has access to the admin page, you can leave the super user account and only come back to it for site maintenance.

To create a super user, run the following command:

`python manage.py createsuperuser` This will ask a few questions, such as username, email, password, confirm password, etc. Make sure the password you provide is secure.

3. MAINTENANCE

DB Backups

You can never be too cautious, database backups are always good to have.

Shell queries

You can execute shell queries on the server command line to retrieve certain information, such as how many users are in the system, or custom database queries.

Learn more about Django Shell online.

4. CONTACT US

We currently have an instance of DAS 2020 running and plan to keep it running, it is accessible at http://teamdas123.pythonanywhere.com/. If you wish to have access to this site to ensure it is suitable for your needs, feel free to contact our development team. Furthermore, if you have any issues, questions, comments or feedback, feel free to contact our development team.

Mohammad Majid @ mohammadmajid58@yahoo.com