

# Proj 2 Web Service 'Plantronix' Plant Information System

Thomas Rhodes  
459274825

Last Updated: May 30, 2022

## Contents

	Page
<b>A Functionality</b>	<b>3</b>
1. Login/Logout functionality and security applied to admin page(s)	3
1.1 Additional Information . . . . .	3
2. At least two (2) examples of CRUD implemented in UI	5
3. Errors/Notices/Warnings displayed when incorrect action taken	11
4. Data in database manipulated as a result of interactions with admin panel	14
<b>B Technology</b>	<b>16</b>
5. Use of an architectural framework	16
6. Use a framework for form validation	18
7. Use a layout framework	18
<b>C Security</b>	<b>20</b>
8. admin Passwords use one way encryption	20
9. Source IP whitelist to restrict access to admin panel	20
10.CSRF mitigation and server-side validation	21
<b>D Meta-cognition</b>	<b>23</b>



---

## Part A

# Functionality

## 1. Login/Logout functionality and security applied to admin page(s)

Logging into the admin panel through the web browser is done with this link: <https://plantronics-backend.herokuapp.com/backend>, and inputting there Username, Password and OTP code, provided they have the correct permissions. This is the only way to be able to access the admin panel.

There are only two permission groups able to login to the admin panel admins, and staff, without having either of these groups and no OTP code. Once logged in they may change their username and password to there preferences.

When an admin or staff successfully login they are given a HTTP\_ONLY\_COOKIE and a CSRF Token which both live in the headers of the http client. While a standard Django application allows anyone with the paramers of is\_staff and is\_superuser to access the admin panel, this is expanded upon by the application of the required otp code which can only be setup by an already authenticated user.

Upon login out of the admin site django automatically flushes the session from the users browser.

### 1..1 Additional Information

If whomever is marking this assessment would like to login to the admin panel an temp account has been made for this purpose.

- Username: TestUser
- Password: DJJK9vNVidYSFSd
- QR Code, to be scanned with any 2fa app such as Authy, Google Authenticator, LastPass, and there are many others out there.



---

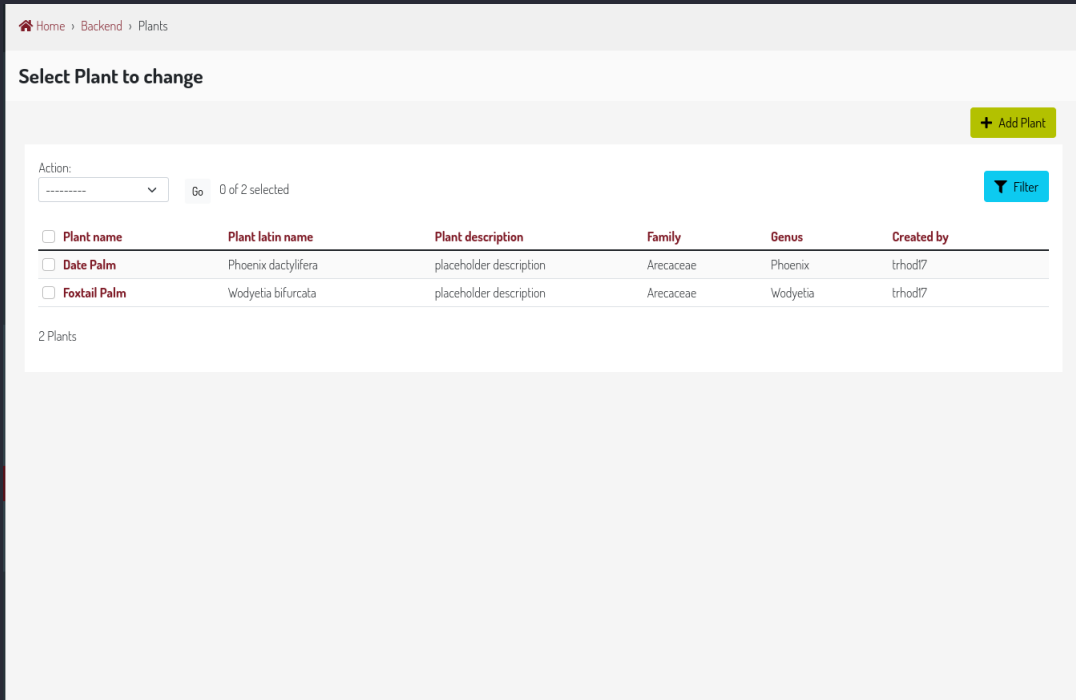
Without your ip i cant add you to the whitelist to allow access to any pages in the backend route including login.

The following account and otp will not work on the local instance of the project as it uses a sqlite database whereas the hosted backend uses a heroku postgresql database. This information is provided only for those marking and will be deleted upon completion of this course

## 2. At least two (2) examples of CRUD implemented in UI

As django handles the creation of crud forms in the panel there are numerous examples of CRUD available to us but i will focus on one entity in particular, Upon successfully logging in and authenticating oneself, then navigating to the plants object, below is what is presented to the user.

*Fig. 1: Plant Object list*



The screenshot displays a web interface for managing plants. At the top, a breadcrumb trail shows 'Home > Backend > Plants'. Below this, a section titled 'Select Plant to change' contains a table of plant objects. The table has columns for 'Plant name', 'Plant latin name', 'Plant description', 'Family', 'Genus', and 'Created by'. Two plants are listed: 'Date Palm' and 'Foxtail Palm'. Each row has a checkbox in the first column. Above the table, there is an 'Action:' dropdown menu, a 'Go' button, and a text indicator '0 of 2 selected'. To the right of the table, there is a '+ Add Plant' button and a 'Filter' button. Below the table, it says '2 Plants'.

<input type="checkbox"/>	Plant name	Plant latin name	Plant description	Family	Genus	Created by
<input type="checkbox"/>	Date Palm	Phoenix dactylifera	placeholder description	Arecaceae	Phoenix	trhod17
<input type="checkbox"/>	Foxtail Palm	Wodyetia bifurcata	placeholder description	Arecaceae	Wodyetia	trhod17

2 Plants

Fig. 2: Create Plant Object

[Home](#) > [Backend](#) > [Plants](#) > Add Plant

### Add Plant

Plant name:

Plant latin name:

Plant image:

Plant description:

Family:

Genus:

Created by:

Info

Soil Preferences

Fig. 3: Plant Object Successfully Created

[Home](#) > [Backend](#) > [Plants](#)

Select Plant to change

✓ The Plant "Black Mulberry" was added successfully.

+ Add Plant

Action:  
----- ▾

Go 0 of 3 selected

Filter

<input type="checkbox"/> Plant name	Plant latin name	Plant description	Family	Genus	Created by
<input type="checkbox"/> <b>Black Mulberry</b>	Morus Nigra	The black mulberry grows delicious fruit.	Moraceae	Morus	Trhod17
<input type="checkbox"/> <b>Date Palm</b>	Phoenix dactylifera	placeholder description	Areaceae	Phoenix	trhod17
<input type="checkbox"/> <b>Foxtail Palm</b>	Wodyetia bifurcata	placeholder description	Areaceae	Wodyetia	trhod17

3 Plants

Support

copyright © 2022 Thomas Rhodes

PlantTronix - Developed by Thomas Rhodes

Fig. 4: Plant Object Update

[Home](#) > [Backend](#) > [Plants](#) > Black Mulberry

### Change Plant

Black Mulberry

[History](#)
[View on site](#)

Plant name:

Plant latin name:

Plant image:  No file chosen

Plant description:

Family:  [✎](#) [✖](#) [+](#)

Genus:  [✎](#) [✖](#) [+](#)

Created by:  [✎](#) [✖](#) [+](#)

Info

Fig. 5: Plant Object Update Successfull

[Home](#) > [Backend](#) > [Plants](#)

### Select Plant to change

[+ Add Plant](#)

[Action:](#)  [Go](#) 0 of 3 selected [Filter](#)

<input type="checkbox"/>	Plant name	Plant latin name	Plant description	Family	Genus	Created by
<input type="checkbox"/>	<b>Black Mulberry</b>	Morus Nigra	The black mulberry grows delicious fruit which are not technically fruit.	Moraceae	Morus	Trhod17
<input type="checkbox"/>	<b>Date Palm</b>	Phoenix dactylifera	placeholder description	Areaceae	Phoenix	trhod17
<input type="checkbox"/>	<b>Foxtail Palm</b>	Wodyetia bifurcata	placeholder description	Areaceae	Wodyetia	trhod17

3 Plants



Fig. 6: Plant Object Delete

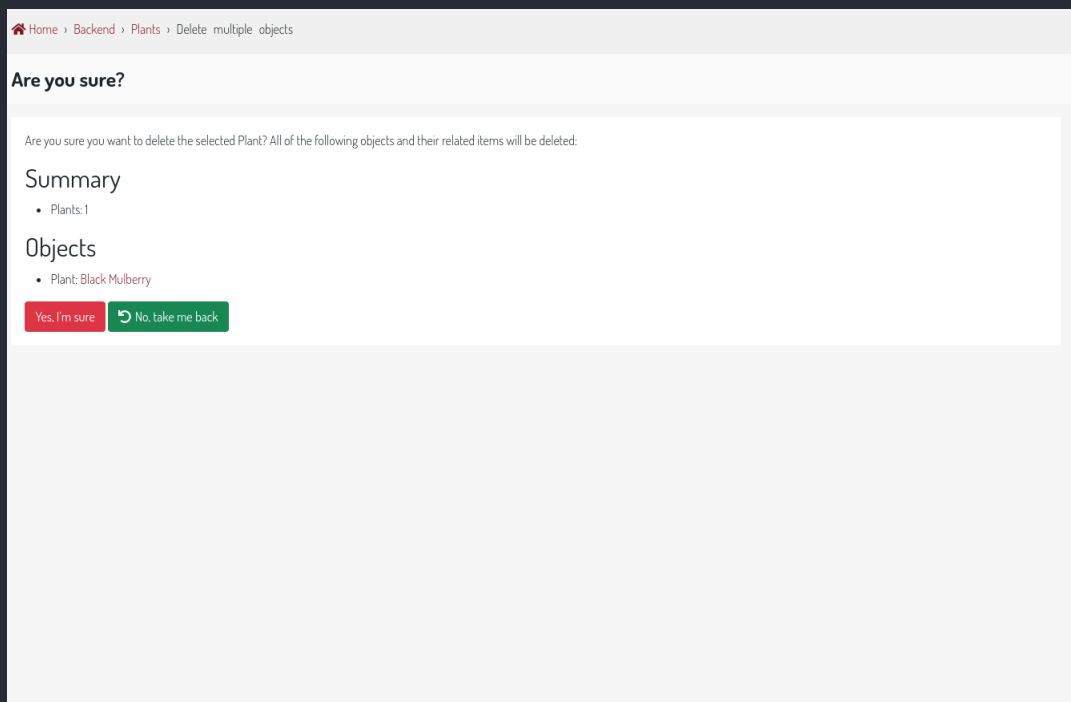


Fig. 7: Plant Object Delete Successful

Home > Backend > Plants

Select Plant to change

✓ Successfully deleted 1 Plant.

+ Add Plant

Action:  
----- ▾

Go 0 of 2 selected

Filter

<input type="checkbox"/> Plant name	Plant latin name	Plant description	Family	Genus	Created by
<input type="checkbox"/> Date Palm	Phoenix dactylifera	placeholder description	Arecaceae	Phoenix	trhod17
<input type="checkbox"/> Foxtail Palm	Wodyetia bifurcata	placeholder description	Arecaceae	Wodyetia	trhod17

2 Plants

Support

copyright © 2022 Thomas Rhodes

PlanTronix - Developed by Thomas Rhodes

---

### 3. Errors/Notices/Warnings displayed when incorrect action taken

There are certain constraints set in models.py, that determine what fields are required for an entity and what data these fields will contain.

*Fig. 8:* Lets see what happens when we attempt to change a plant, but removing its family

Foxtail Palm

[History](#) [View on site](#)

Please correct the errors below.

Plant name:

Plant latin name:

! File extension "" is not allowed. Allowed extensions are: jpg, png, jpeg, webp.

Plant image: Currently: [media/images/istockphoto-1174829700-1706667a\\_nroall](#) ☐ Clear

Change:

Plant description:

! This field is required.

Family:

Genus:

Created by:

From here we can see that an error has been thrown and the form cannot be submitted.

*Fig. 9:* The result of trying to create a plant with empty fields

Please correct the errors below.

! This field is required.

Plant name:




! This field is required.

Plant latin name:




Plant image:

Plant description:




! This field is required.

Family:    

! This field is required.

Genus:    

! This field is required.

Created by:    

Info

And as we can see above creating a plant with no values returns numerous errors.

---

Below is a couple of code snippets should the code for validating the models, in the first image we have the plant model which contains MinLength, MaxLength and a FileExtensionValidator. The Min and Max length validators will stop data being entering into the database if it is not within those parameters, the FileExtensionValidator checks to make sure that uploaded images are of the correct filetype.

*Fig. 10: Validators in the Plant Model in models.py*



```
models.py

from django.db import models
from django.contrib.auth.models import User
from django.urls import reverse
from django.core.validators import MinLengthValidator, MaxLengthValidator, FileExtensionValidator

class Plant(models.Model):
    plant_name = models.CharField(max_length=255, validators=[MaxLengthValidator(255),
MinLengthValidator(3)])
    plant_latin_name = models.CharField(max_length=255, validators=[MaxLengthValidator(255),
MinLengthValidator(3)])
    plant_image = models.ImageField(upload_to='images/', blank=True, validators=[
FileExtensionValidator(['jpg', 'png', 'jpeg', 'webp'])])
    plant_description = models.CharField(max_length=1000, null=True,
blank=True)
    family = models.ForeignKey('backend.Family', on_delete=models.RESTRICT)
    genus = models.ForeignKey(
'backend.genus', on_delete=models.RESTRICT)
    created_by = models.ForeignKey(User, on_delete=models.SET_NULL, null=True)

    class Meta:
        verbose_name = ('Plant')

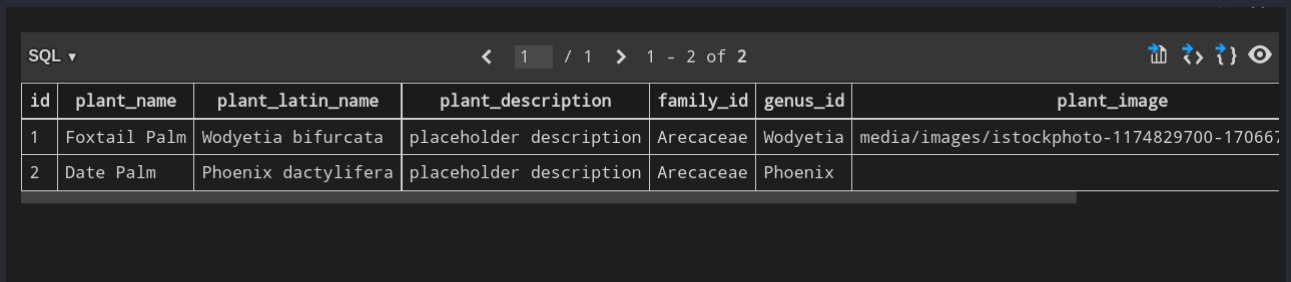
    def __str__(self):
        return self.plant_name

    def get_absolute_url(self):
        return reverse('', kwargs={'pk': self.pk})
```

## 4. Data in database manipulated as a result of interactions with admin panel

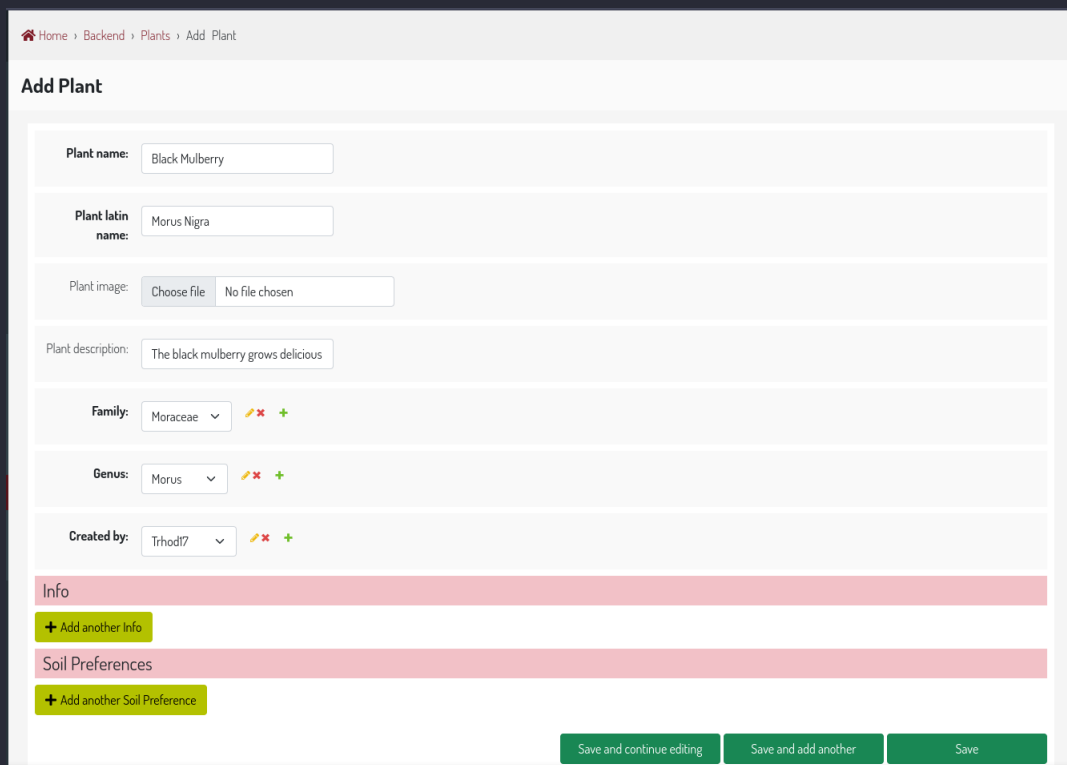
To show the data in the database is being changed, lets create a new plant entry for the black mulberry, but first lets look at the plants currently in the table.

Fig. 11: Plants table in database



id	plant_name	plant_latin_name	plant_description	family_id	genus_id	plant_image
1	Foxtail Palm	Wodyetia bifurcata	placeholder description	Areaceae	Wodyetia	media/images/istockphoto-1174829700-170667
2	Date Palm	Phoenix dactylifera	placeholder description	Areaceae	Phoenix	

Here we can see a total of 2 plants currently in the table and the black mulberry isnt one of them.



Home » Backend » Plants » Add Plant

### Add Plant

Plant name:

Plant latin name:

Plant image:

Plant description:

Family:

Genus:

Created by:

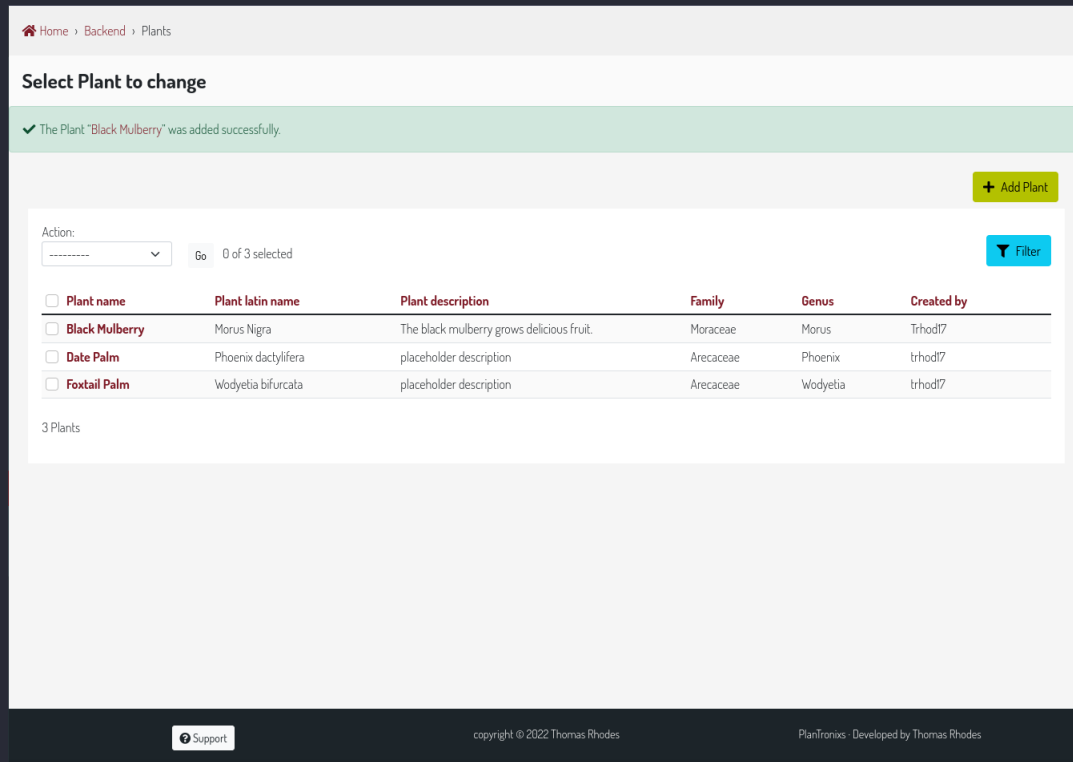
Info

Soil Preferences

Fig. 12: Create plant page filled in

In the screenshot of the Create Plant Page above , here we can see the filled out form and are about to hit save to see what happens next.

Fig. 13: Plants created successfully



Here we can see our new plant was successfully created, by the success message in the screen to the left, now lets check the database to see if the table has been updated.

Fig. 14: Plant successfully created in database

SQL ▾						
< 1 / 1 > 1 - 3 of 3 <span style="float: right;">             📄 ↺ ↻ ⌕           </span>						
id	plant_name	plant_latin_name	plant_description	family_id	genus_id	pl
1	Foxtail Palm	Wodyetia bifurcata	placeholder description	Arecaceae	Wodyetia	media/images/istockphc
2	Date Palm	Phoenix dactylifera	placeholder description	Arecaceae	Phoenix	
6	Black Mulberry	Morus Nigra	The black mulberry grows delicious fruit.	Moraceae	Morus	

---


## Part B

# Technology

### 5. Use of an architectural framework

The admin panel uses Django as its architectural framework, Django is a full stack framework built in Python, That is utilised by many companies such as Mozilla, Google, Instagram, Bitbucket. In Django the admin panel comes prebuilt and requires small amount of configuration to register and configure our models to be accessible to users using the panel.

*Fig. 15: requirements.txt*



```
○○○  
autopep8  
colored-logs  
coloredlogs  
cryptography  
dj-rest-auth  
django-allauth  
django-audit-log  
django-audit-log-middleware  
django-baton  
django-cloudinary-storage  
django-cors-headers  
django-enviro  
django-filter  
django-heroku  
django-ip-logger  
django-otp  
django-request-logger  
django-request-logging  
django-requestlogs  
django-rest-framework  
django-simple-ip-restrict  
djangorestframework-jsonapi  
environ  
gunicorn  
openapi-codec  
pillow  
pip-chill  
psycpg2-binary  
qrcode  
redgreenunittest  
simplejson
```

Above is the requirements.txt which is python's system for transferring requirements between all the users working on the project, it contains a list of all the required packages to create the Admin Panel for this implementation, which proves an architectural framework was used.

For further reading relating to Django and its Admin Panel, please refer to the provided link



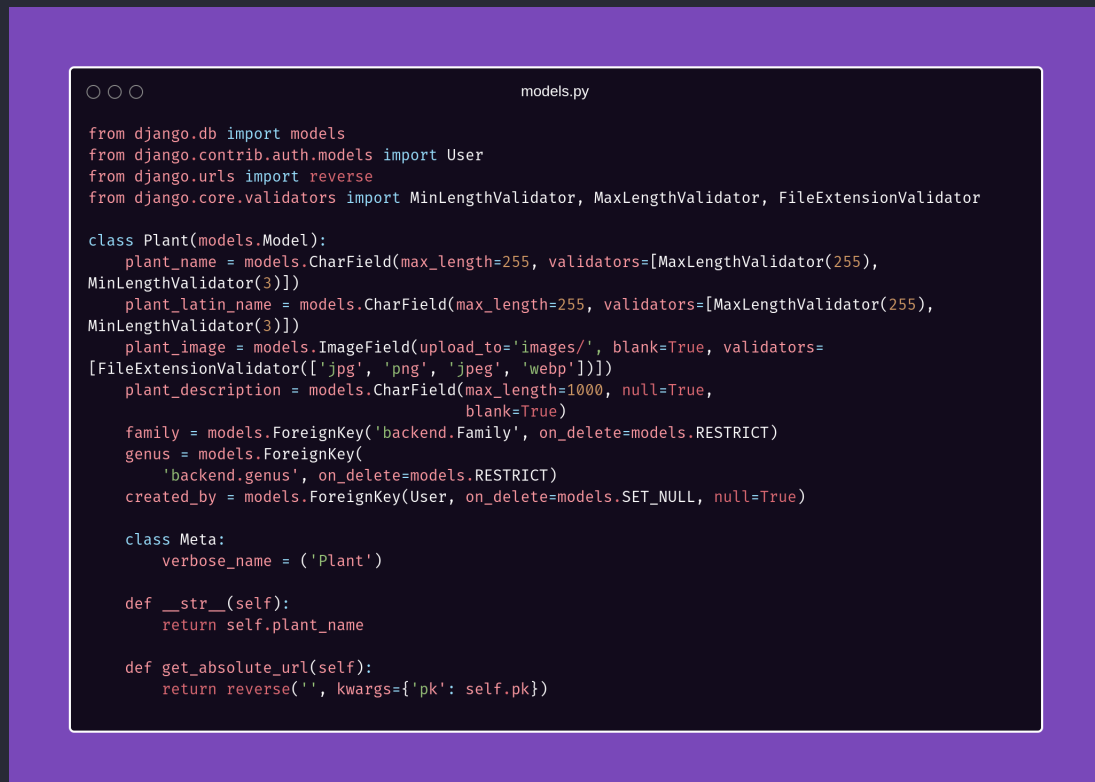


---

## 6. Use a framework for form validation

Django was chosen for form validation. Django supplies its own validators that can be applied to each field in the 'models.py' file where we instantiated our database models/entities, Below is a screenshot taken from 'models.py', on line 4 we are importing the 3 validators which are being used from 'django.core.validators'. These validators are effective on the admin panels forms and also on the server side validation level.

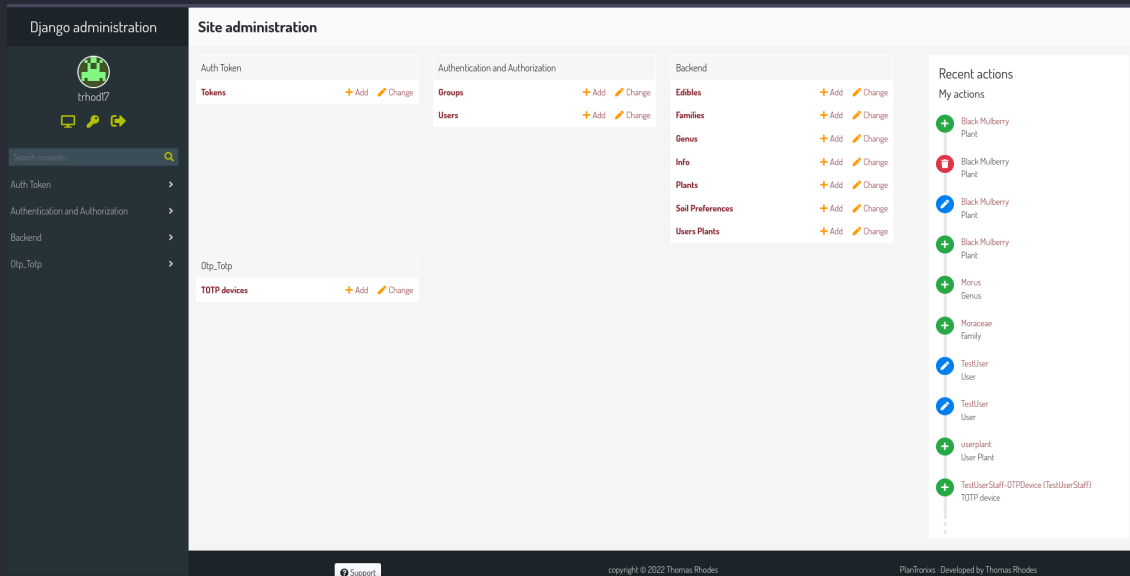
*Fig. 16: models.py*



## 7. Use a layout framework

The admin panel layout relies on an extension for Django called Django-Baton which overrides and redesigns nearly all of the default admin panel, the styling itself is done by bootstrap .

Fig. 17: The admin panel with Django-Baton and bootstrap



## Part C

# Security

### 8. admin Passwords use one way encryption

In this application all passwords regardless of account use one way encryption, upon a valid form submission, Django uses the PBKDF2 algorithm with SHA256 hash for key stretching purposes, which convert the plain text password into a random string of hashed characters with a salt value added, which is done several thousand times for each password which creates a derived key, this derived key is saved to the database.

Below is a screenshot of the user table in the database showing the hashed passwords.

Fig. 18: Hashed Passwords

id	password	last_login	is_superuser	username	last_name	email	is_staff	is_active	date_joined	first_name
1	pbkdf2_sha256\$320000\$80a6sJaf4kYsXNu19w1zwY\$S62Lq1tGH0q0IhKL79mRKJwgcQaba51RNARbhzQBE=	2022-05-29 23:24:09.549100	1	trhod17			1	1	2022-04-07 06:58:20.941575	
2	pbkdf2_sha256\$320000\$J2JxzRPR140Z9B1cYQtkcz\$JvLztwZn9eOJYacqR08t+VfA+dtvMRauvYGubBJk7ro=	NULL	0	TestUser			0	1	2022-04-12 23:35:17	
3	pbkdf2_sha256\$320000\$WYScnxj1XIwUAT5TgSHRskDKEZpjn77/tH7kyzCelbdJpttJJ8guEXQsOwgr1R6c=	2022-05-02 08:23:16	0	Trhod17	Rhodes	hedghegog.thomas@gmail.com	0	1	2022-04-30 05:22:28	Tho
4	pbkdf2_sha256\$320000\$1N7A631k7bDXKrDN2p91S4s1VEZnonFs3QLv3n3g48XnglM+J81FZQM25qgkTp421=	NULL	0	frontend			0	1	2022-05-01 01:50:27	
5	pbkdf2_sha256\$320000\$Hf0TTkaJQ2MHkNmfrnKSmF\$H2wyu3ZENMFKicINffAD93a7xmbn2xgg8Hd1luV00E=	NULL	0	TestUser2	Rhodes	hedghegog.thomas@gmail.com	0	1	2022-05-04 23:05:35.624002	Tho

### 9. Source IP whitelist to restrict access to admin panel

While the application is setup to use a middleware to only allow certain whitelisted ips (e.g the site) to be able to connect and make requests, which is achieved by installing a package called django-simple-ip-restrict found at <https://pypi.org/project/django-simple-ip-restrict/>. this allows us in addition to our global whitelist to create a whitelist specifically for the admin panel.

Fig. 19: Middleware installed

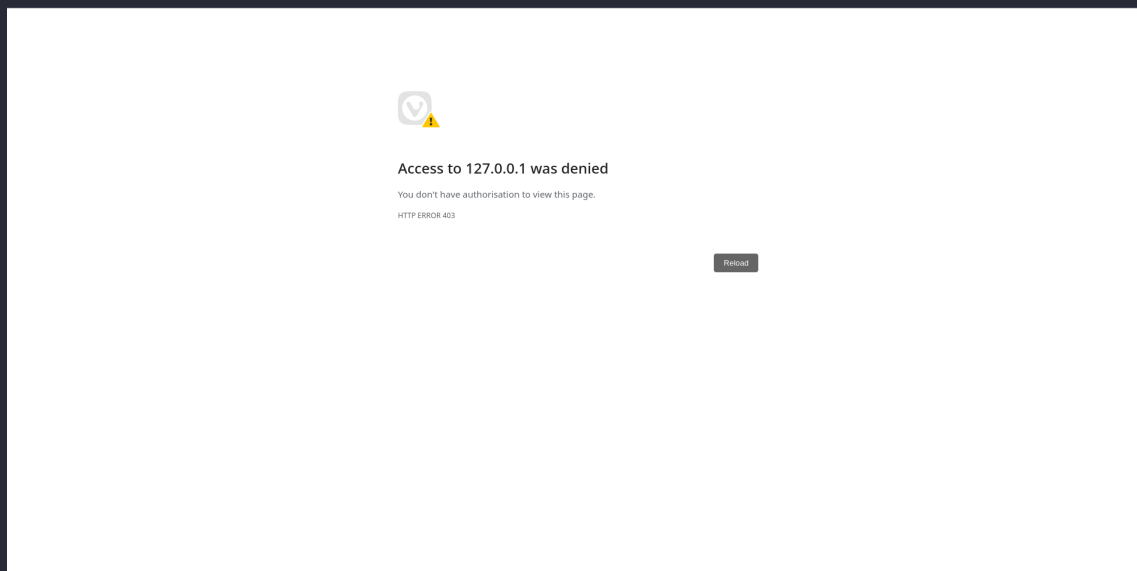
```
MIDDLEWARE = [
    'django.middleware.security.SecurityMiddleware',
    'corsheaders.middleware.CorsMiddleware',
    'django.contrib.sessions.middleware.SessionMiddleware',
    'django_simple_ip_restrict.middleware.ip_filter', #whitelist middleware
    'django.middleware.common.CommonMiddleware',
    'django.middleware.csrf.CsrfViewMiddleware',
    'django.contrib.auth.middleware.AuthenticationMiddleware',
    'django.contrib.messages.middleware.MessageMiddleware',
    'django.contrib.auth.middleware.AuthenticationMiddleware',
    'django_otp.middleware.OTPMiddleware',
    'django_audit_log_middleware.AuditLogMiddleware',
]
```

*Fig. 20: Whitelist*

```
IP_PROTECTED_NAMESPACES = ["admin"]
IP_NETWORKS_WHITELIST = [
    '127.0.0.1',
    '192.168.0',
    '127.0.0.1',
    '127.0.0.1',
    '138.44.128.242',
]
```

This can be easily tested by whomever is reading this by simply trying to connect to the backend at which point you will get an access denied page.

*Fig. 21: Access Denied Page*



## 10. CSRF mitigation and server-side validation

CSRF mitigation is handled by Django's default authentication system, this includes admin panel logins and usage. The CSRF middleware is located at “/starter-upp/env/lib/python3.10/site-packages/django/middleware/csrf.py”, here on line 161 you will see a class called `CsrfMiddleware` with about a dozen or so methods dictating how requests are handled, based on whether they have a valid csrf token or not.

Fig. 22: CSRF Token in browser storage

Name	Value	Domain	Path	Expire...	Size	HttpO...	Secure	Same...	Same...	Partiti...	Prio...
sessionid	hy8m0ued5cf6rq446rft5ihrae82bfxc	127.0.0.1	/	2022-12-31	41	✓		Lax			Medium
csrftoken	cgAswlrDuH48nioRITSFLT10CDN8a9WMFC9oy...	127.0.0.1	/	2023-12-31	73			Lax			Medium

In regards to server side validation this is handled on the frontend and by the validators on the models mention earlier in section 3.

---

## Part D

# Meta-cognition

## 11. Comment on each of the 3rd party frameworks used, why was it chosen

The following third party frameworks were used to create this application:

Django

Django Rest Framework

Baton & Bootstrap

### 11.1 Django

Django was chosen as it supplied full stack functionality out of the box with a rich ecosystem of third party libraries with minium setups to achieve further functionality, because Django makes heavy use of Python's multiparadigm support by mainly using OOP design styles, abstracting a lot of bloat away from the user allowing for a much shallower learning curve then say react or express. Django's intent is that the mentality of the developers is to not reinvent the wheel and instead capitilise upon the mass of single purpose libraries available on <https://pypi.org>. with much of workload removed from the developer, it increase the speed in which applications can be developed.

I chose Django because i had used it before for personal projects that never came to fruition but allowed me to learn how to create applications in Django, I second biggest reason i chose Django's is because its developpt in Python which is by far one of my preferred programming languages. Django has superb documentation and a very friendly community allowing for easy access to help when needed. The main functionality of Django that i made use of is:

- Authentication System (HTTP\_ONLY\_COOKIE, users, permissions)
- CSRF mitigation
- Admin Panel
- HTTP statues codes, debugger, error pages, error messages pre configured
- testing framework

Another benefit of Django is that once all of my models, urls and admin configurations was done, there was only a small amount of tweaking and edge smoothing required to create a functional working application.

---

## 11..2 Django Rest Framework

Base Django can be used on its own as a REST API to some extent , However it lacks a lot of functionality in that area this is where Django Rest Framework comes in. With Django Rest Framework it takes only an hour or so to have a fully configured format of choice churning machine. Django Rest Framework comes with a token authentication system which only requires adding a line or two to its configuration to require users to provide a token with there requests in order to be able to use the various endpoints created in django. Django Rest Framework provides its own serializers to handle various file formats plus many third party packages to add support for different formats.

Django Rest Framework benefits from having an equally informative and robust documentation as Django does, making developing with it a dream.

## 11..3 Baton & Bootstrap

Baton offers pre setup admin interface which requires simply overriding the default admin url and adding two entries to installed apps and an some minor configuration and you have a great looking backend.

Baton under the hood uses Bootstrap and allows us to easily change out the Bootstrap stylesheet to one of our choosing, from for example bootswatch.com, which allows to change the color scheme and some design aspects of the admin panel.

## 12. What other technologies did you investigate in order to settle on a path

I had briefly considered Next.js, React, Vue, Express or php or my backend but from the beginning i had my sights set on Django, because from my past experiences with it in the past, i know how easy it is to develop an application in Django, and with such a short time to develop it in that was the main advantage.

## 13. Describe the rules by which your authentication restricts access. Comment in code.

There are several systems in play ensuring that only authenticated users can access and use the admin panel such as, Session ID / HTTP Only cookies, Token checks, Two-factor Authentication OTP check codes, IP whitelisting, and CSFR checks.

The session cookies are handled by Djangos in-built session middleware which is also extended by Django Rest Apis builtin session middleware which strengthens it even further.

Upon a user successfully logging in they are sent a session cookie which persists until either it expires or the user logs out Django flushes the session cookie completely from the browsers.

The token authentication works by assigning a unique token consiting of a randomized string of characters to an authenticated user, and upon request from the use their token is compared to the version saved in the database.



---

## 14. Describe why you chose this particular encryption technology

The password hashing is done using the PBKDF2 algorithm with a SHA256 hash iterated over 320000 times, which was chosen due to the fact that is the default in Django, The main benefit of using PBKDF2 is that it is considered slow which means that anyone who wants to break has to do so slowly aswell.

Given the current amount of time it takes to hash and save the password it was deemed acceptable for use in the application as it should have a very minor effect on account creation speed of accounts.