Reflective Piece

The Secure Software Development module covered a wide range of topics including: waterfall and agile approaches to software development, cryptography, UML modelling, standards that are beneficial to secure a software, testing for vulnerabilities, future trends, concepts of programming languages like python, etc.

One of the most rewarding aspects of the module was learning about secure coding practices, how to implement them in projects and testing. It was gratifying to see how following best practices can enforce security of a system. Evidence of the tasks performed are after References.

Other than covering the module, we were grouped into different teams so as to apply the skillsets acquired throughout the module. In Group2 we chose to do a project on Dutch Police Internet Forensics (Government of the Netherlands, N.D.), where we were to design a secure software proposal report and a coding output for them.

We started off by threat modeling using STRIDE AND OWASP (CWE Content Team, 2021), then used the findings to build a more secure system that has features and development strategies that promotes Confidentiality, Integrity, Availability (CIA), Non-Reputability and mitigates cyberattacks. Such features included:

- Web architecture and cloud hosting so as for easier accessibility on the internet
- GDPR implementation
- The Model-View-Controller (MVC) architectural pattern and UML designing
- User controls and Permissions
- Authentication and authorization

- Data encryption and use of SQLite database
- Firewalls, SSL certificates, session management, HTTPS, etc.

As my teammates performed other tasks, I took part in the designing of UML diagrams and also offered some cybersecurity solutions that could be applied in the proposal report (my submission is included after References).

Even though coding was the most challenging aspect of the project due to lack of experiences in Python web development, my groupmates and I still opted to using Django framework instead of Web2Py framework or Flask.

After researching on Django, Python, CSS, HTML and Bootstrap from sources such as: Codio exercises on Essex's student page, (Django, 2015), (T, 2013), (Django, N.D.), (W3Schools, N.D.), (Dauzon, et al., 2016), (Saabith, et al., 2019) YouTube videos, etc.- I came to realization on how Django framework:

- supports MVC pattern and has robust built-in security features against common web vulnerabilities.
- allows integration of other Django modules and all python libraries,
- fully customizable, more automated, efficient and scalable

thus, guaranteeing productivity, security and code quality: which is a plus in secure software development.

As a result, I became motivated and confident enough to develop a Django web application that focused on our 1st Sprint's scopes. In doing so, I got to have a first-hand experience and gained skills on:

- Django and Python installation and their different libraries that needs to be imported first for a function to work.
- How to connect and integrate the Model (SQLite database), View (HTML, user Interface) and Controller.
- How to extend a Django's default user model/database using one-to-one relationship.
- How to secure a web application by setting permissions (decorators), groups and user roles so as to control access, implementing custom password validators, authentication, adding csrf_token on login page to protect data, debugging, etc.
- HTML page inheritance (extend and include) which helps in minimizing code redundancy through out the View part of the system.
- How it is not a good idea, security-wise, to implement a user account deactivation after 3 or more failed login attempts. This is because an attacker can purposefully initiate a DDOS attack by blocking users. Therefore, a better solution is to block an IP address after 3 or more failed login attempts.

(Screenshot evidence of the above are after References)

Afterwards, my teammates performed tests and did the documentation since that is also essential in secure software development.

Being part of a team has been a rewarding and educational experience. Some of the factors that contributed to the success of this project and individual growth were:

- Daily reviewal discussions with my team throughout the development process so as to present and explain work progress and ensure we all on the same page.
- Team-playing, respect and self-discipline,
- A lot of patience and endurance during coding and debugging.
- Motivating, supporting, helping and learning from each other to have clear understanding on the project

I have come to understand the importance of secure software development in protecting sensitive information (GDPR) and preventing damage to individuals and organizations, and also got an opportunity to apply the skills on an actual project and see its fruition.

I plan to apply my learning in the future by being more mindful of and prioritizing security when building software, and also by staying up-to-date on the latest best practices, trends and technologies.

Additionally, it is crucial to prioritize securing a software from the earliest development stage all through to the end. That been the case, I will continue to practice and build upon this knowledge so as to ensure security is prioritised and fully-covered in future software development and projects.

References

CWE Content Team, 2021. *CWE VIEW: Weaknesses in OWASP Top Ten (2021).* [Online]
Available at: https://cwe.mitre.org/data/definitions/1344.html
[Accessed 15 December 2022].

Dauzon, S., Bendoraitis, A. & Ravindran, A., 2016. *Django: Web Develoment with Python.* Mumbai: Packt Publishing Ltd.

Django, 2015. *Documentation.* [Online]
Available at: https://docs.djangoproject.com/en/4.1/
[Accessed 15 December 2022].

Django, N.D.. *Django 1.7.11 documentation.* [Online]
Available at: https://django.readthedocs.io/en/1.7.x/index.html
[Accessed 12 December 2022].

Government of the Netherlands, N.D.. *Fighting cybercrime in the Netherlands.* [Online]
Available at: https://www.government.nl/topics/cybercrime/fighting-cybercrime-in-the-netherlands
[Accessed 15 December 2022].

Saabith, A. S., Fareez, M. & Vinothraj, T., 2019. PYTHON CURRENT TREND APPLICATIONS- AN OVERVIEW. *International Journal of Advance Engineering and Research Development,* 6(10), pp. 6-12.

T, N., 2013. *Lock out users after too many failed login attempts.* [Online]
Available at: https://stackoverflow.com/questions/9033287/lock-out-users-after-too-many-failed-login-attempts
[Accessed 12 December 2022].

W3Schools, N.D.. *CSS Tutorial.* [Online]
Available at: https://www.w3schools.com/css/default.asp
[Accessed 12 December 2022].

Below are screenshots of my contributions throughout this module as from my E-portfolio

https://mutegibeatrice.github.io/module4.html

Secure Software Development November 2022

« Collaborative Discussion 1: UML flowchart

Beatrice Mutegi

Initial Post

7 days ago

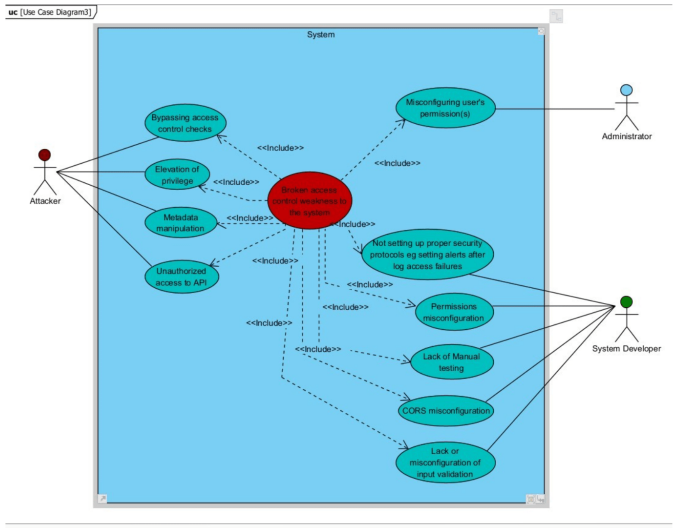
The Open Web Application Security Project (OWASP) is an online community that enables organizations to develop, purchase, and maintain secure applications and APIs by providing free and open methodologies, tools, documentation, cheat sheets, technologies, etc. (Anon, 2017).

As from (Anon, 2017), Broken Access Control is one of the OWASP top 10 most critical Web Application Security Risks (2017) (placed in category A5).

Before a user get access to a feature, some web applications check the user's access so as to control access, however, if requests are not checked, attackers will be able to gain access to features and even servers without the proper permission(s) (Fredj, et al., 2021)

(Anon, 2017) further describes that one of the common causes of this weakness been due to the lack of automated detection and effective functional testing by application developers. Therefore, manual testing is necessary and the most effective way to detect missing or ineffective access control.

Below is a use case diagram that shows some of the possible causes that may have led to occurrence of a broken access control.



Other than the use case diagram shown above, (which has been created with the use of one of the Open-source tools, Visual Paradigm), UML diagrams such as: sequence diagrams and activity diagrams, can give a more in-depth graphical description of this attack due to their ability to show interaction of operations or processes and show dynamic aspects of a system respectively.

References

Anon, 2017. *OWASP Top 10 - 2017*. [Online]
Available at: https://owasp.org/www-pdf-archive/OWASP_Top_10-2017_%28en%29.pdf.pdf
[Accessed 28 November 2022].

Fredj, O. B., Cheikhrouhou, O. & Krichen, M., 2021. *An OWASP Top Ten Driven Survey on Web Application Protection Methods*. s.l., Springer, Cham, pp. 235-252.

Reply

Maximum rating: -

0 replies

Add your reply

Your subject

Type your post

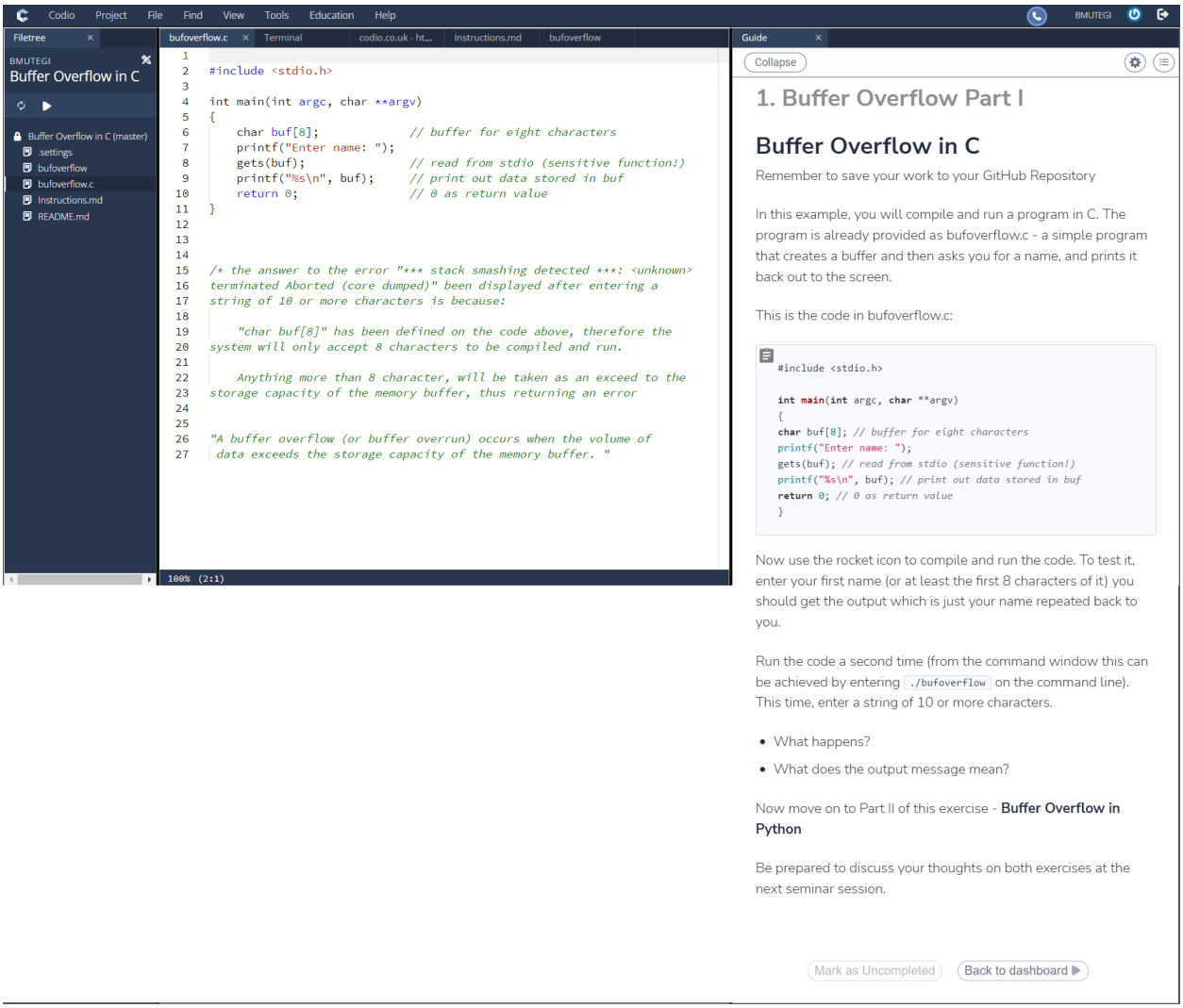
Choose files No file chosen

Submit

Use advanced editor and additional options

OLDER DISCUSSION
Initial Post

NEWER DISCUSSION
Initial Post


Codio
Project
File
Find
View
Tools
Education
Help
BMUTEGI
Filetree
BMUTEGI
Buffer Overflow in C
Buffer Overflow in C (master)
.settings
bufoverflow
bufoverflow.c
Instructions.md
README.md
bufoverflow.c
Terminal
codio.co.uk - ht...
Instructions.md
bufoverflow
#include <stdio.h>
int main(int argc, char **argv)
{
char buf[8]; // buffer for eight characters
printf("Enter name: ");
gets(buf); // read from stdio (sensitive function!)
printf("%s\n", buf); // print out data stored in buf
return 0; // 0 as return value
}
/* the answer to the error "*** stack smashing detected ***: <unknown>
terminated Aborted (core dumped)" been displayed after entering a
string of 10 or more characters is because:
"char buf[8]" has been defined on the code above, therefore the
system will only accept 8 characters to be compiled and run.
Anything more than 8 character, will be taken as an exceed to the
storage capacity of the memory buffer, thus returning an error
"A buffer overflow (or buffer overrun) occurs when the volume of
data exceeds the storage capacity of the memory buffer. "
100% (2:1)
Guide
Collapse
1. Buffer Overflow Part I
Buffer Overflow in C
Remember to save your work to your GitHub Repository
In this example, you will compile and run a program in C. The program is already provided as bufoverflow.c - a simple program that creates a buffer and then asks you for a name, and prints it back out to the screen.
This is the code in bufoverflow.c:
#include <stdio.h>
int main(int argc, char **argv)
{
char buf[8]; // buffer for eight characters
printf("Enter name: ");
gets(buf); // read from stdio (sensitive function!)
printf("%s\n", buf); // print out data stored in buf
return 0; // 0 as return value
}
Now use the rocket icon to compile and run the code. To test it, enter your first name (or at least the first 8 characters of it) you should get the output which is just your name repeated back to you.
Run the code a second time (from the command window this can be achieved by entering ./bufoverflow on the command line). This time, enter a string of 10 or more characters.
What happens?
What does the output message mean?
Now move on to Part II of this exercise - Buffer Overflow in Python
Be prepared to discuss your thoughts on both exercises at the next seminar session.
Mark as Uncompleted
Back to dashboard

submit4.pdf 3 / 12 66%

My Input:

Design Decisions and Approaches That You Have Adopted to Create a Secure Software Solution, and the Libraries Needed.

1. Custom-designed Input Validating Code

Regular Expressions (RegEx or re) library (to enable the use of 're.search()' method to search and detect a series of strings within a given criteria (Lenka, 2020).

2. One-Time Pin/Password (OTP) Generator and Password Generator for MFA

Importation of Random library (to allow generation of random numbers) and also Math library (to give access to common mathematical functions) (Oliphant, 2007).

3. Time based login of 3 attempts then blocks user after been unsuccessful

Mitigates cyber-attacks like brute-force, etc.

4. Google translate library

5. Use of html, css, javascript and python for developing the web app

6. A database application SQL

7. Python Imaging Library: commonly used to handle all types of image manipulation (Dauzon, et al., 2016)

3

8. Django forms library: it joins 3 main components of the framework: database fields, html form tags and ability of the user's input to be validated and display error messages (Dauzon, et al., 2016).

9. Python library: to interface with SQLite (Dauzon, et al., 2016).

submit4.pdf 4 / 12 66%

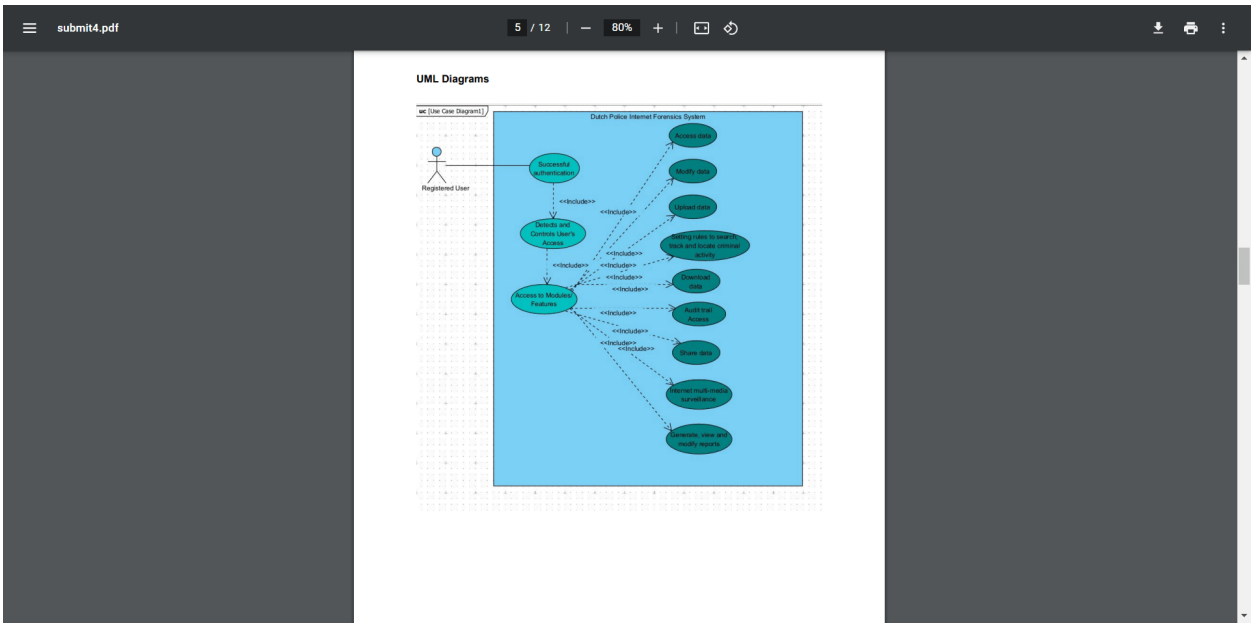
html form tags and ability of the user's input to be validated and display error messages (Dauzon, et al., 2016).

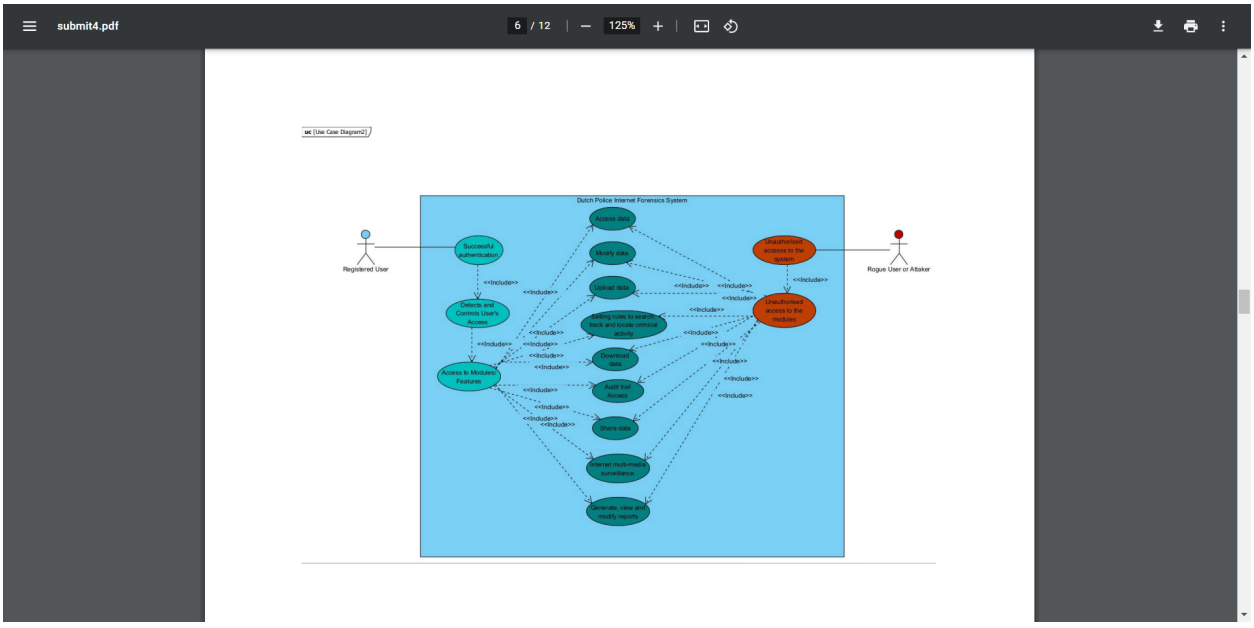
9. Python library: to interface with SQLite (Dauzon, et al., 2016).

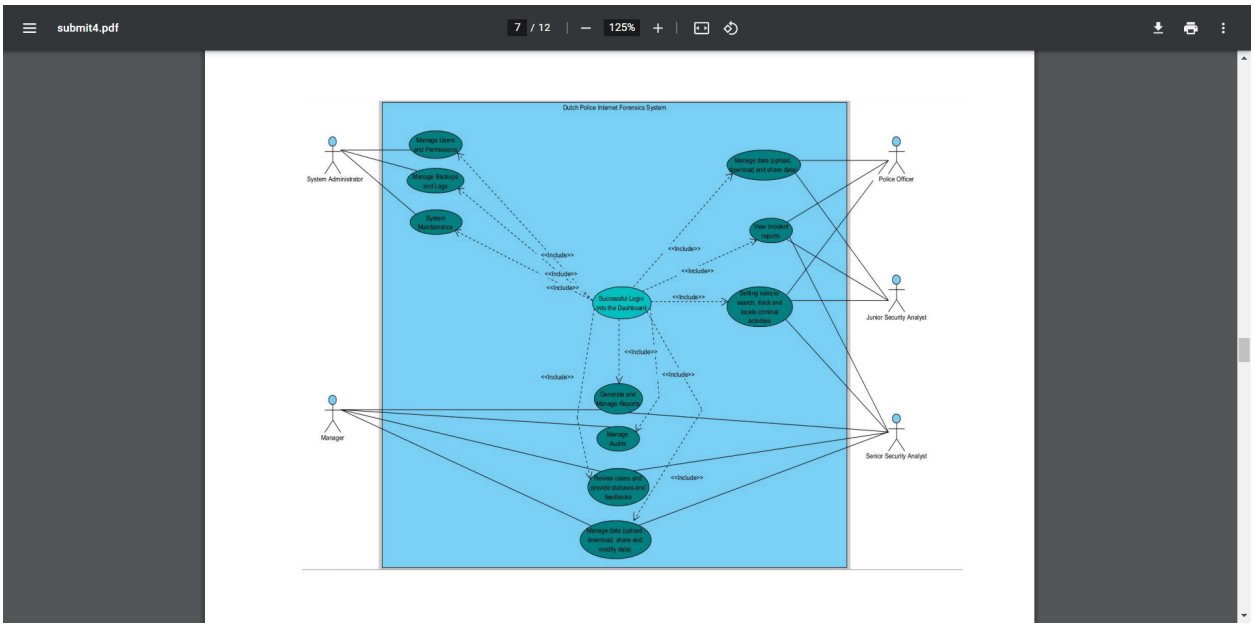
10. **Pycha library:** is a charting and graphic python library (Dauzon, et al., 2016).
11. **Javascript library:** allows a faster creation of client-side functionality (Dauzon, et al., 2016).
12. **ReportLab library:** allows generation of PDFs with python code (Melé, 2020)
13. **Venv library:** allows creation of lightweight virtual environments (Melé, 2020)
14. **Django-allauth:** used for authentication (local and social).
15. **Django import export:** allows backup of databases.
16. **Requests library:** allows HTTP basic authentication (Melé, 2020)
17. **Pillow library:** for handling images (Melé, 2020)
18.

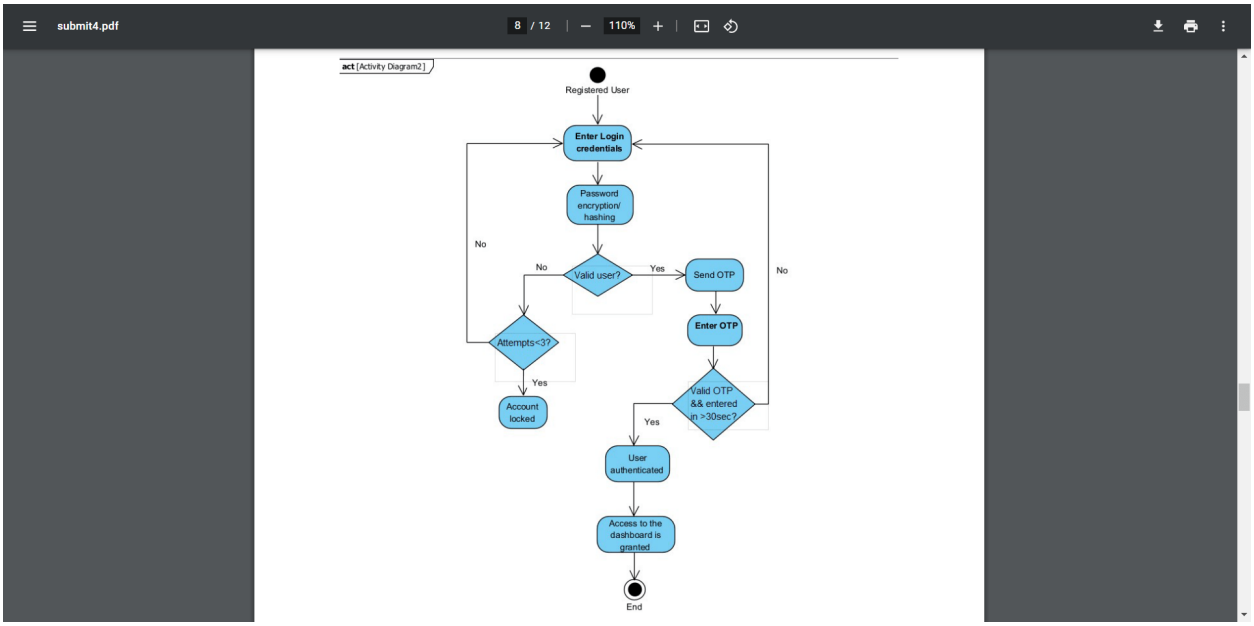
4

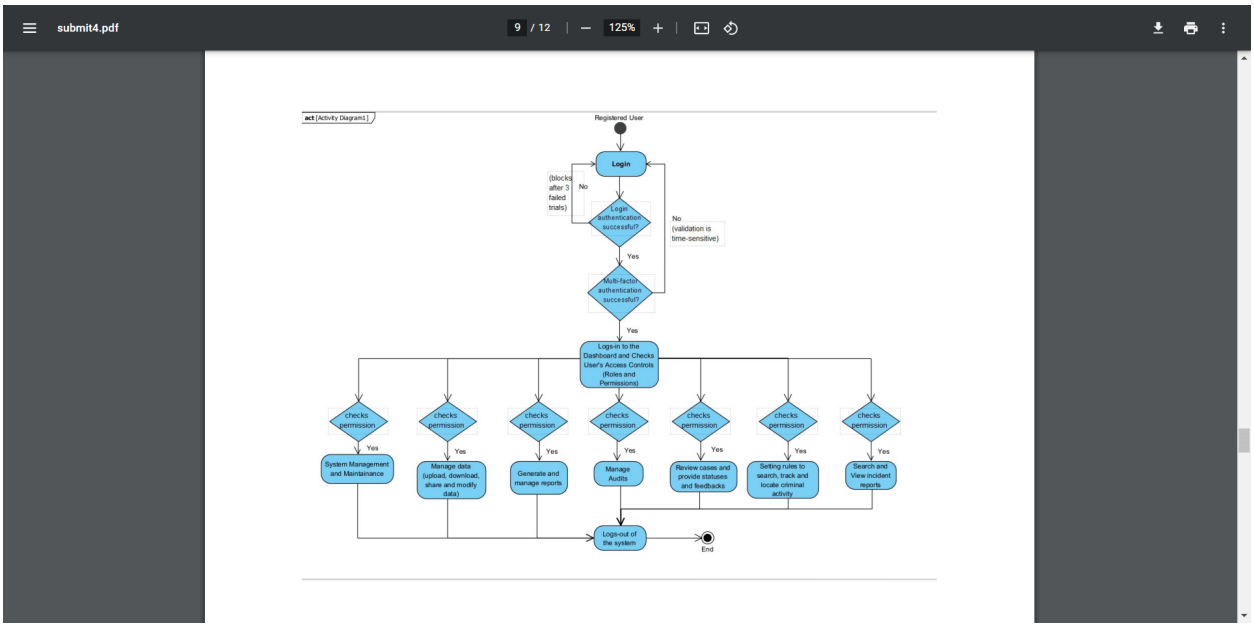
UML Diagrams

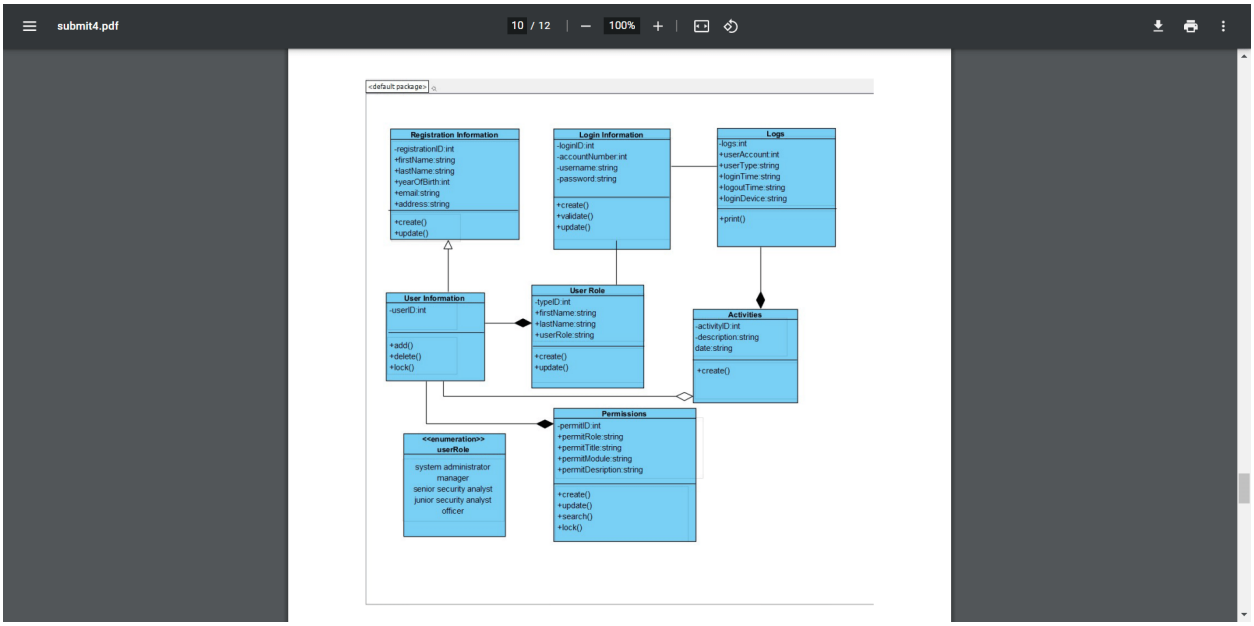

submit4.pdf
5 / 12
80%
UML Diagrams
uc [Use Case Diagram1]
Dutch Police Internet Forensics System
Registered User
Successful authentication
Detects and Controls User's Access
Access to Modules/ Features
Access data
Modify data
Upload data
Skilling rules to search, track and locate criminal activity
Download data
Audit trail Access
Share data
Internet multi-media surveillance
Generate, view and modify reports
<<Include>>

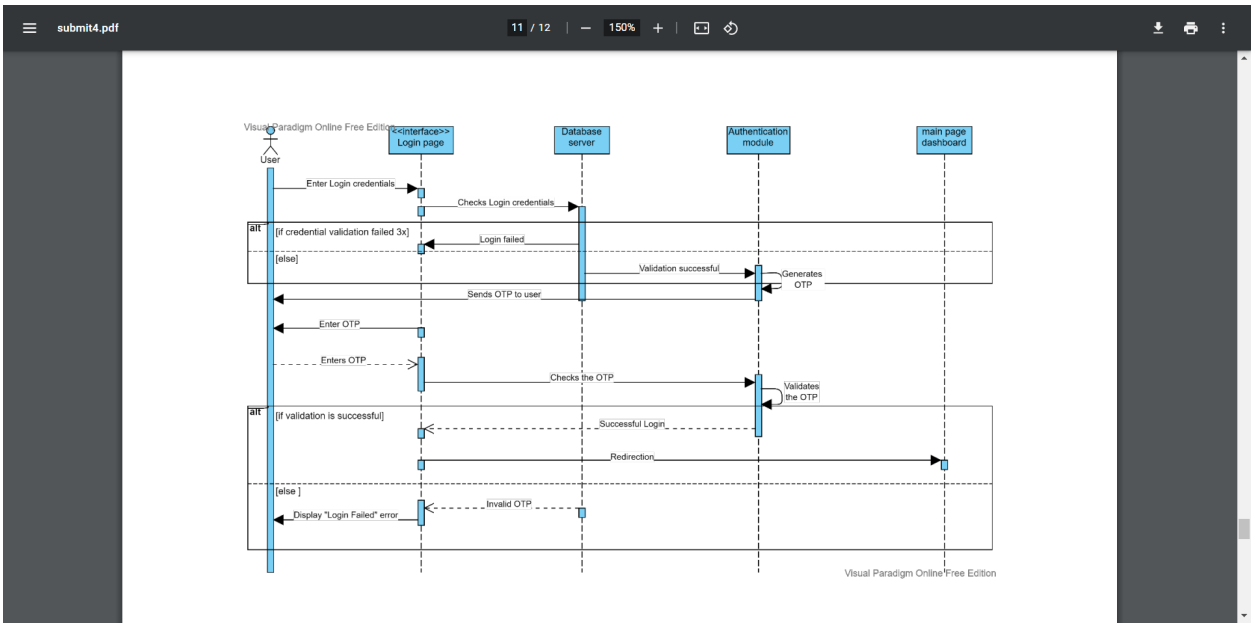

submit4.pdf
6 / 12
125%
uc [Use Case Diagram2]
Dutch Police Internet Forensics System
Registered User
Rogue User or Attaker
Successful authentication
Detects and Controls User's Access
Access to Modules/ Features
Access data
Modify data
Upload data
Skilling rules to search, track and locate criminal activity
Download data
Audit trail Access
Share data
Internet multi-media surveillance
Generate, view and modify reports
Unauthorised access to the system
Unauthorised access to the modules
<<Include>>

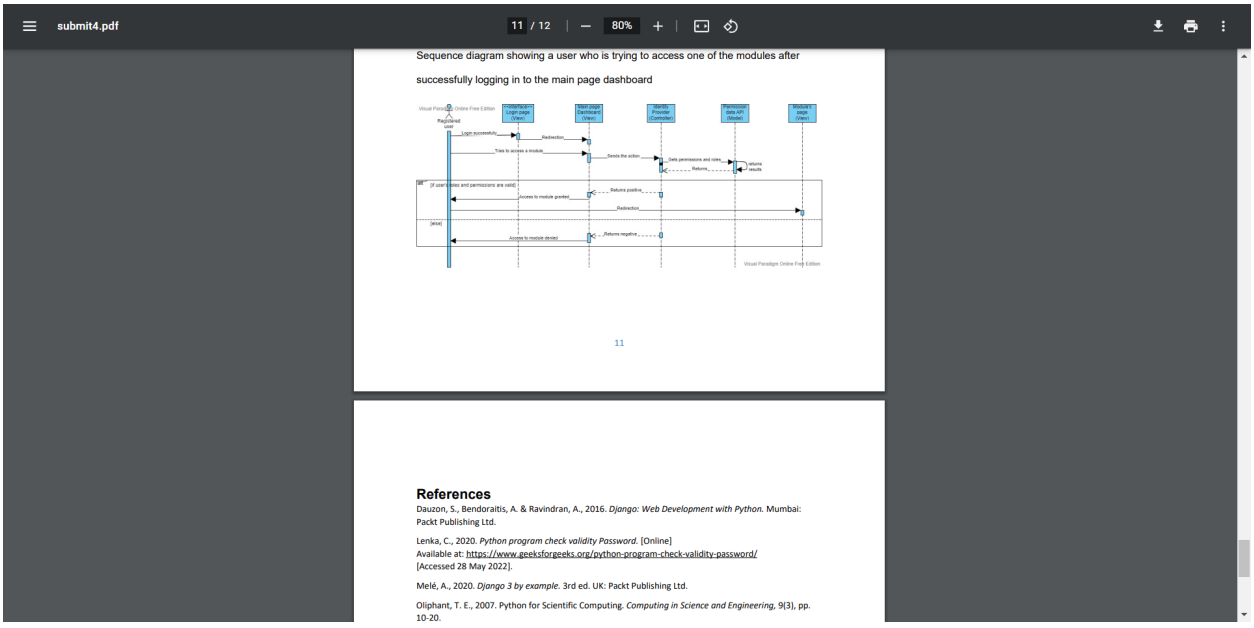

submit4.pdf
7 / 12
125%
Dutch Police Internet Forensics System
System Administrator
Manage Users and Permissions
Manage Backups and Logs
System Maintainance
Manage data (upload, download and share data)
Police Officer
View Incident reports
Junior Security Analyst
Successful Login into the Dashboard
Setting rules to search, track and locate criminal activities
<<Include>>
Manager
Generate and Manage Reports
Manage Audits
Review cases and provide statuses and feedbacks
Senior Security Analyst
Manage data (upload, download, share and modify data)


submit4.pdf
8 / 12
110%
act [Activity Diagram2]
Registered User
Enter Login credentials
Password encryption/ hashing
Valid user?
No
Yes
Send OTP
Enter OTP
Attempts<3?
Account locked
Valid OTP && entered in >30sec?
User authenticated
Access to the dashboard is granted
End


submit4.pdf
9 / 12
125%
act [Activity Diagram1]
Registered User
Login
(blocks after 3 failed trials)
No
Login authentication successful?
No (validation is time-sensitive)
Yes
Multi-factor authentication successful?
Yes
Logs-in to the Dashboard and Checks User's Access Controls (Roles and Permissions)
checks permission
Yes
System Management and Maintainance
Manage data (upload, download, share and modify data)
Generate and manage reports
Manage Audits
Review cases and provide statuses and feedbacks
Setting rules to search, track and locate criminal activity
Search and View incident reports
Logs-out of the system
End


submit4.pdf
10 / 12
100%
<default package>
Registration Information
-registrationID:int
+firstName:string
+lastName:string
+yearOfBirth:int
+email:string
+address:string
+create()
+update()
Login Information
-loginID:int
-accountNumber:int
-username:string
-password:string
+create()
+validate()
+update()
Logs
-logs:int
+userAccount:int
+userType:string
+loginTime:string
+logoutTime:string
+loginDevice:string
+print()
User Information
-userID:int
+add()
+delete()
+lock()
User Role
-typeID:int
+firstName:string
+lastName:string
+userRole:string
+create()
+update()
Activities
-activityID:int
-description:string
date:string
+create()
Permissions
-permitID:int
+permitRole:string
+permitTitle:string
+permitModule:string
+permitDesription:string
+create()
+update()
+search()
+lock()
<<enumeration>>
userRole
system administrator
manager
senior security analyst
junior security analyst
officer





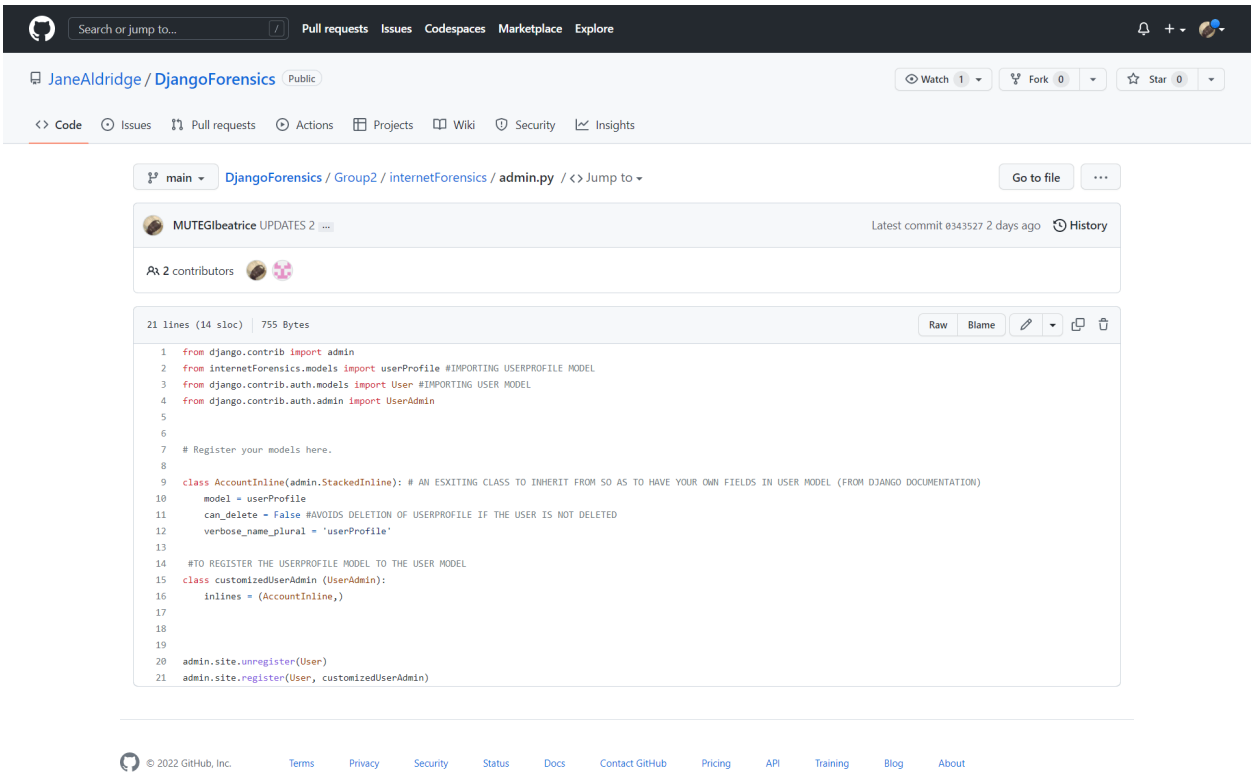
References

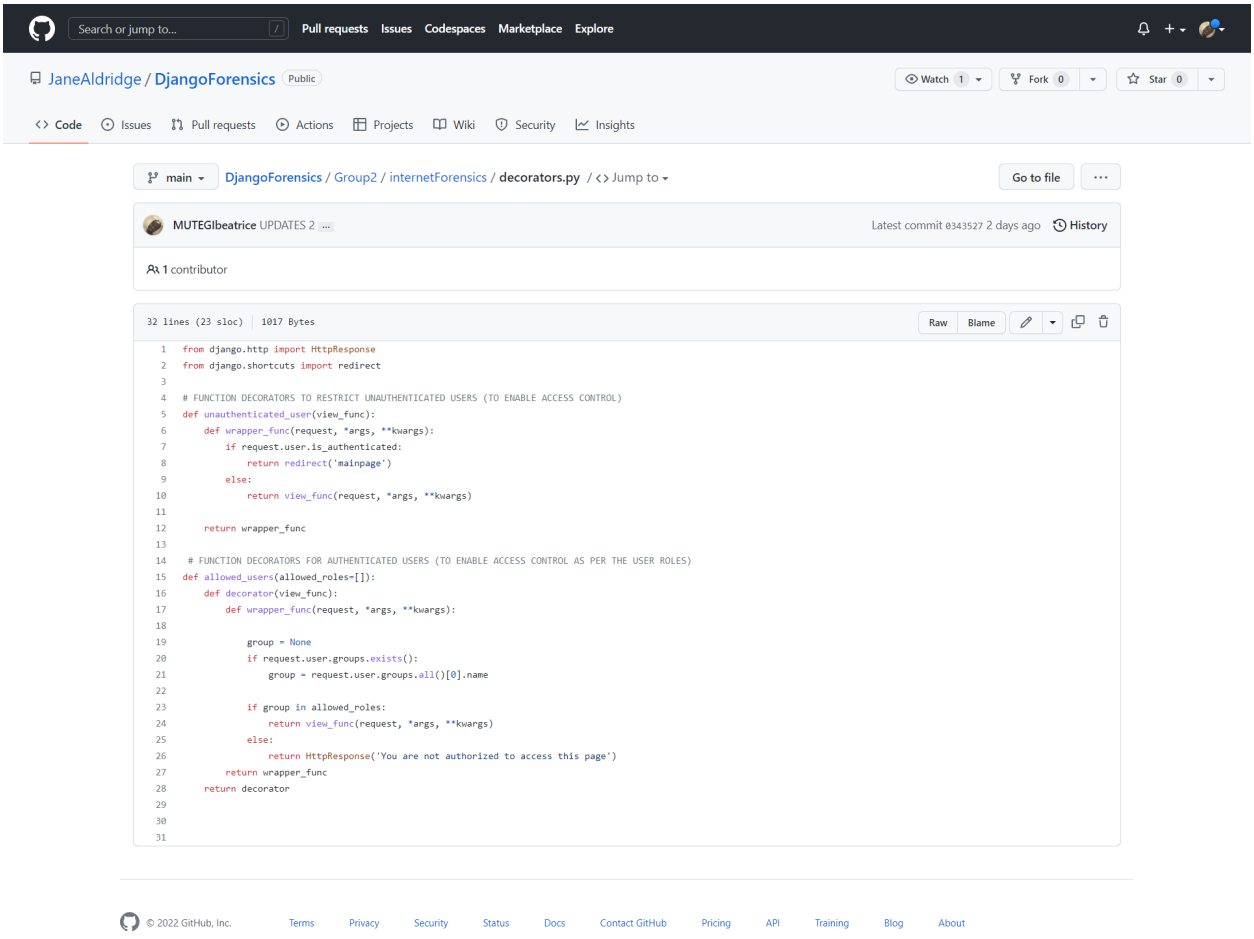
Dauzon, S., Bendoraitis, A. & Ravindran, A., 2016. *Django: Web Development with Python.* Mumbai: Packt Publishing Ltd.

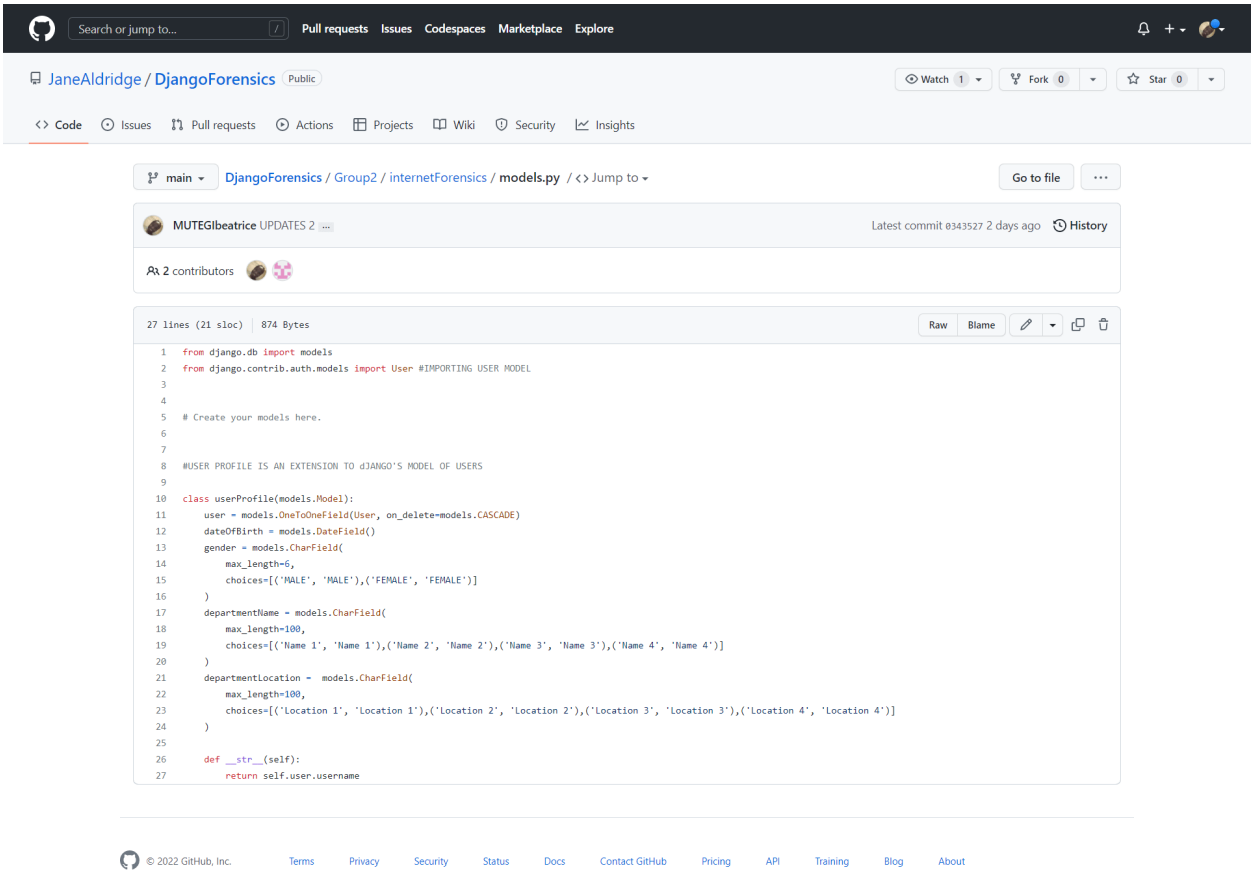
Lenka, C., 2020. *Python program check validity Password.* [Online]
Available at: https://www.geeksforgeeks.org/python-program-check-validity-password/
[Accessed 28 May 2022].

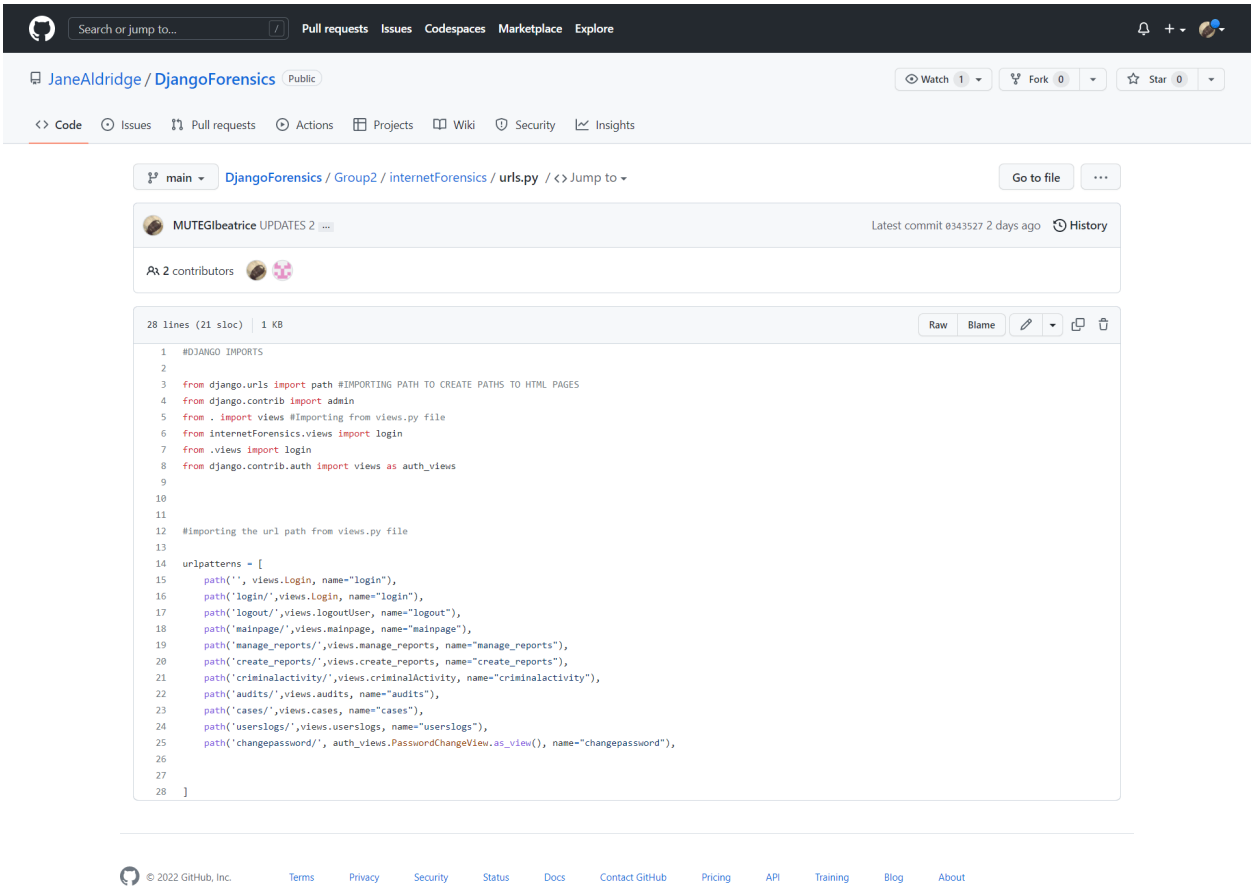
Melé, A., 2020. *Django 3 by example.* 3rd ed. UK: Packt Publishing Ltd.

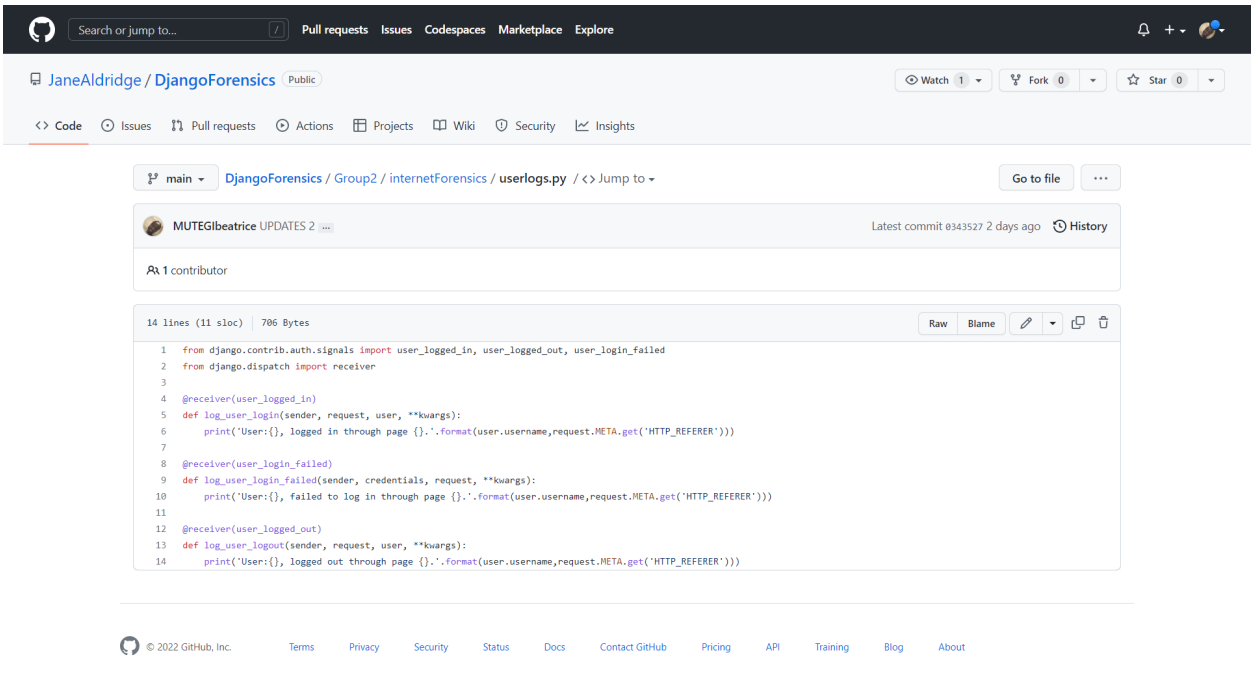
Oliphant, T. E., 2007. Python for Scientific Computing. *Computing in Science and Engineering,* 9(3), pp. 10-20.


Search or jump to...
Pull requests
Issues
Codespaces
Marketplace
Explore
JaneAldridge / DjangoForensics
Public
Watch 1
Fork 0
Star 0
Code
Issues
Pull requests
Actions
Projects
Wiki
Security
Insights
main
DjangoForensics / Group2 / internetForensics / admin.py
Jump to
Go to file
MUTEGIbeatrice UPDATES 2
Latest commit 0343527 2 days ago
History
2 contributors
21 lines (14 sloc)
755 Bytes
Raw
Blame
from django.contrib import admin
from internetForensics.models import userProfile #IMPORTING USERPROFILE MODEL
from django.contrib.auth.models import User #IMPORTING USER MODEL
from django.contrib.auth.admin import UserAdmin
# Register your models here.
class AccountInline(admin.StackedInline): # AN EXSITING CLASS TO INHERIT FROM SO AS TO HAVE YOUR OWN FIELDS IN USER MODEL (FROM DJANGO DOCUMENTATION)
model = userProfile
can_delete = False #AVOIDS DELETION OF USERPROFILE IF THE USER IS NOT DELETED
verbose_name_plural = 'userProfile'
#TO REGISTER THE USERPROFILE MODEL TO THE USER MODEL
class customizedUserAdmin (UserAdmin):
inlines = (AccountInline,)
admin.site.unregister(User)
admin.site.register(User, customizedUserAdmin)
© 2022 GitHub, Inc.
Terms
Privacy
Security
Status
Docs
Contact GitHub
Pricing
API
Training
Blog
About

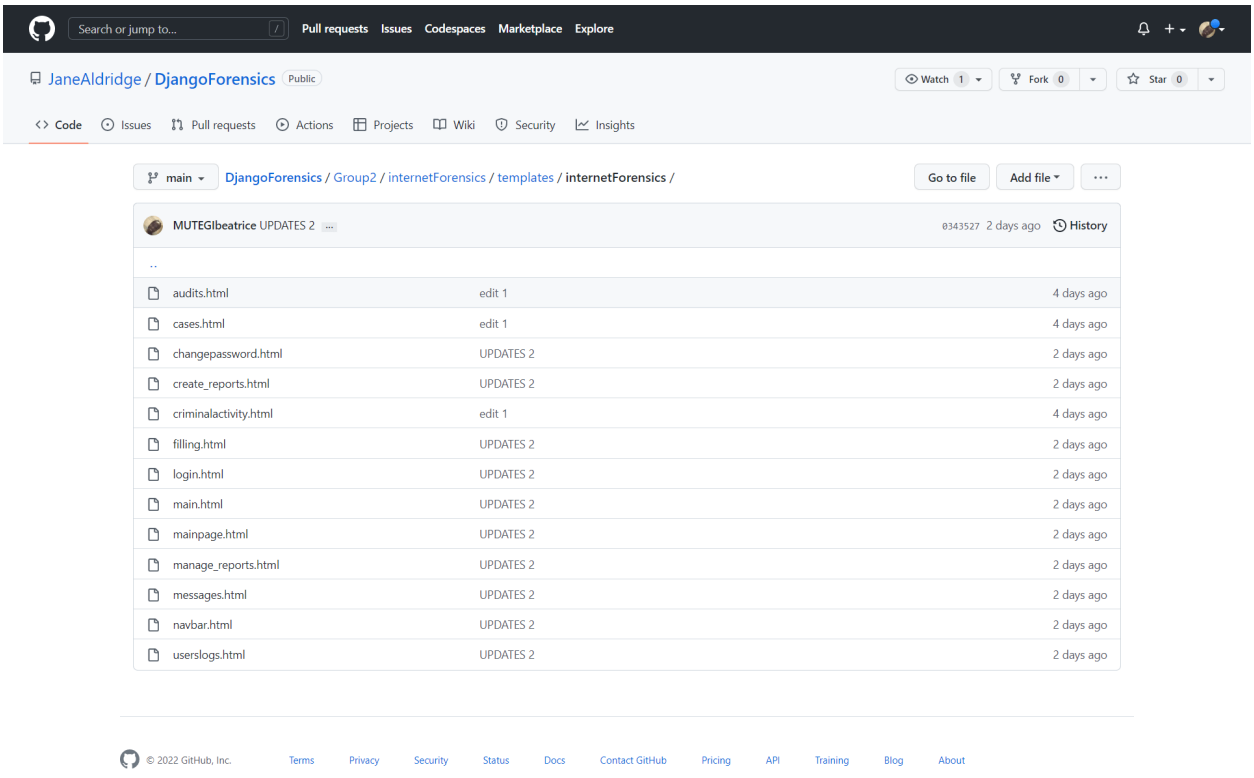

Search or jump to...
Pull requests
Issues
Codespaces
Marketplace
Explore
JaneAldridge / DjangoForensics
Public
Watch 1
Fork 0
Star 0
Code
Issues
Pull requests
Actions
Projects
Wiki
Security
Insights
main
DjangoForensics / Group2 / internetForensics / decorators.py
Jump to
Go to file
MUTEGIbeatrice UPDATES 2
Latest commit 0343527 2 days ago
History
1 contributor
32 lines (23 sloc)
1017 Bytes
Raw
Blame
from django.http import HttpResponse
from django.shortcuts import redirect
# FUNCTION DECORATORS TO RESTRICT UNAUTHENTICATED USERS (TO ENABLE ACCESS CONTROL)
def unauthenticated_user(view_func):
def wrapper_func(request, *args, **kwargs):
if request.user.is_authenticated:
return redirect('mainpage')
else:
return view_func(request, *args, **kwargs)
return wrapper_func
# FUNCTION DECORATORS FOR AUTHENTICATED USERS (TO ENABLE ACCESS CONTROL AS PER THE USER ROLES)
def allowed_users(allowed_roles=[]):
def decorator(view_func):
def wrapper_func(request, *args, **kwargs):
group = None
if request.user.groups.exists():
group = request.user.groups.all()[0].name
if group in allowed_roles:
return view_func(request, *args, **kwargs)
else:
return HttpResponse('You are not authorized to access this page')
return wrapper_func
return decorator
© 2022 GitHub, Inc.
Terms
Privacy
Security
Status
Docs
Contact GitHub
Pricing
API
Training
Blog
About


Search or jump to...
Pull requests
Issues
Codespaces
Marketplace
Explore
JaneAldridge / DjangoForensics
Public
Watch 1
Fork 0
Star 0
Code
Issues
Pull requests
Actions
Projects
Wiki
Security
Insights
main
DjangoForensics / Group2 / internetForensics / models.py
Jump to
Go to file
MUTEGIbeatrice UPDATES 2
Latest commit 0343527 2 days ago
History
2 contributors
27 lines (21 sloc)
874 Bytes
Raw
Blame
from django.db import models
from django.contrib.auth.models import User #IMPORTING USER MODEL
# Create your models here.
#USER PROFILE IS AN EXTENSION TO dJANGO'S MODEL OF USERS
class userProfile(models.Model):
user = models.OneToOneField(User, on_delete=models.CASCADE)
dateOfBirth = models.DateField()
gender = models.CharField(
max_length=6,
choices=[('MALE', 'MALE'),('FEMALE', 'FEMALE')]
)
departmentName = models.CharField(
max_length=100,
choices=[('Name 1', 'Name 1'),('Name 2', 'Name 2'),('Name 3', 'Name 3'),('Name 4', 'Name 4')]
)
departmentLocation = models.CharField(
max_length=100,
choices=[('Location 1', 'Location 1'),('Location 2', 'Location 2'),('Location 3', 'Location 3'),('Location 4', 'Location 4')]
)
def __str__(self):
return self.user.username
© 2022 GitHub, Inc.
Terms
Privacy
Security
Status
Docs
Contact GitHub
Pricing
API
Training
Blog
About


Search or jump to...
Pull requests
Issues
Codespaces
Marketplace
Explore
JaneAldridge / DjangoForensics
Public
Watch 1
Fork 0
Star 0
Code
Issues
Pull requests
Actions
Projects
Wiki
Security
Insights
main
DjangoForensics / Group2 / internetForensics / urls.py
Jump to
Go to file
MUTEGIbeatrice UPDATES 2
Latest commit 0343527 2 days ago
History
2 contributors
28 lines (21 sloc) 1 KB
Raw
Blame
#DJANGO IMPORTS
from django.urls import path #IMPORTING PATH TO CREATE PATHS TO HTML PAGES
from django.contrib import admin
from . import views #Importing from views.py file
from internetForensics.views import login
from .views import login
from django.contrib.auth import views as auth_views
#importing the url path from views.py file
urlpatterns = [
path('', views.Login, name="login"),
path('login/',views.Login, name="login"),
path('logout/',views.logoutUser, name="logout"),
path('mainpage/',views.mainpage, name="mainpage"),
path('manage_reports/',views.manage_reports, name="manage_reports"),
path('create_reports/',views.create_reports, name="create_reports"),
path('criminalactivity/',views.criminalActivity, name="criminalactivity"),
path('audits/',views.audits, name="audits"),
path('cases/',views.cases, name="cases"),
path('userslogs/',views.userslogs, name="userslogs"),
path('changepassword/', auth_views.PasswordChangeView.as_view(), name="changepassword"),
]
© 2022 GitHub, Inc.
Terms
Privacy
Security
Status
Docs
Contact GitHub
Pricing
API
Training
Blog
About


Search or jump to...
Pull requests
Issues
Codespaces
Marketplace
Explore
JaneAldridge / DjangoForensics
Public
Watch 1
Fork 0
Star 0
Code
Issues
Pull requests
Actions
Projects
Wiki
Security
Insights
main
DjangoForensics / Group2 / internetForensics / userlogs.py
Jump to
Go to file
MUTEGIbeatrice UPDATES 2
Latest commit 0343527 2 days ago
History
1 contributor
14 lines (11 sloc) 706 Bytes
Raw
Blame
from django.contrib.auth.signals import user_logged_in, user_logged_out, user_login_failed
from django.dispatch import receiver
@receiver(user_logged_in)
def log_user_login(sender, request, user, **kwargs):
print('User:{}, logged in through page {}.'.format(user.username,request.META.get('HTTP_REFERER')))
@receiver(user_login_failed)
def log_user_login_failed(sender, credentials, request, **kwargs):
print('User:{}, failed to log in through page {}.'.format(user.username,request.META.get('HTTP_REFERER')))
@receiver(user_logged_out)
def log_user_logout(sender, request, user, **kwargs):
print('User:{}, logged out through page {}.'.format(user.username,request.META.get('HTTP_REFERER')))
© 2022 GitHub, Inc.
Terms
Privacy
Security
Status
Docs
Contact GitHub
Pricing
API
Training
Blog
About


Search or jump to...
Pull requests
Issues
Codespaces
Marketplace
Explore
JaneAldridge / DjangoForensics
Public
Watch 1
Fork 0
Star 0
Code
Issues
Pull requests
Actions
Projects
Wiki
Security
Insights
main
DjangoForensics / Group2 / internetForensics / views.py
Jump to
Go to file
MUTEGIbeatrice UPDATES 2
Latest commit 0343527 2 days ago
History
2 contributors
105 lines (64 sloc) | 4.22 KB
Raw
Blame
#DJANGO IMPORTS
from django.contrib import admin
from django.conf import settings
from django.shortcuts import render, redirect #to allow redirection from one page to another pages
from django.http import HttpResponse #FOR HTTP RESPONSE
from django.contrib.auth.models import User #FOR IMPORTING USERS DATABASE
from django.contrib.auth import authenticate, login, logout #FOR AUTH AND LOGOUT
from django.contrib.auth.models import Group #FOR IMPORTING USERS GROUP MODELS
from django.contrib import messages #FOR ERROR AND INFORMATION/STATUS MESSAGES
from django.contrib.auth.decorators import login_required #FOR ENABLING VIEW RESTRICTIONS UNTIL LOGGED IN
from datetime import date, timedelta
# Create your views here.
from .models import * #FOR IMPORTING ALL MODELS
from .decorators import unauthenticated_user, allowed_users #FOR IMPORTING THE USER ACCESS CONTROL DECORATORS
from django.contrib.auth import views as auth_views
# LOGIN REQUEST, AUTHENTICATION AND RESTRICTION TO AN ALREADY AUTHENTICATED USER
@unauthenticated_user # only accessed to unauthenticated users and restriction put on validated users who try to go back to login page via the URL
def Login(request): #DEFINING THE LOGIN AUTHENTICATION
if request.method == 'POST': #authentication
username = request.POST.get("username")
password = request.POST.get("password")
user = authenticate(request, username=username, password=password)
if user is not None:
messages.success(request, 'Login Successful')
login(request,user)
return redirect('mainpage')
else:
messages.info(request, 'Invalid Username or Password')# Return an 'invalid login' error message.
return render(request, 'internetForensics/login.html')
# PASSWORD CHANGING
class PasswordChangeView(auth_views.PasswordChangeView):
template_name = 'internetForensics/changepassword.html'
# LOG OUT REQUEST
def logoutUser(request):
logout(request)
return redirect('login')
VIEW REQUESTS FOR THE REST OF THE PAGES AND ACCESS RESTRICTIONS
@login_required(login_url='login') # RESTRICTION ON UNAUTHENTICATED USERS AND DERIVED FROM 'IMPORT LOGIN_REQUIRED'
def mainpage(request):
return render(request, 'internetForensics/mainpage.html')
@login_required(login_url='login') # RESTRICTION ON UNAUTHENTICATED USERS AND DERIVED FROM 'IMPORT LOGIN_REQUIRED'
@allowed_users(allowed_roles=['admin','managers']) # ACCESS CONTROL AS PER THE USER ROLES
def manage_reports(request): #to view manage_report page
return render(request, 'internetForensics/manage_reports.html')
@login_required(login_url='login') # RESTRICTION ON UNAUTHENTICATED USERS AND DERIVED FROM 'IMPORT LOGIN_REQUIRED'
def create_reports(request): #to view create_report page
return render(request, 'internetForensics/create_reports.html')
@login_required(login_url='login') # RESTRICTION ON UNAUTHENTICATED USERS AND DERIVED FROM 'IMPORT LOGIN_REQUIRED'
def criminalActivity(request): #to view search and locate criminal activities page
return render(request, 'internetForensics/criminalactivity.html')
@login_required(login_url='login') # RESTRICTION ON UNAUTHENTICATED USERS AND DERIVED FROM 'IMPORT LOGIN_REQUIRED'
@allowed_users(allowed_roles=['admin','managers','senior_security_analysts']) # ACCESS CONTROL AS PER THE USER ROLES
def audits(request): #to view audits page
return render(request, 'internetForensics/audits.html')
@login_required(login_url='login') # RESTRICTION ON UNAUTHENTICATED USERS AND DERIVED FROM 'IMPORT LOGIN_REQUIRED'
def cases(request): #to view cases page
return render(request, 'internetForensics/cases.html')
@login_required(login_url='login') # RESTRICTION ON UNAUTHENTICATED USERS AND DERIVED FROM 'IMPORT LOGIN_REQUIRED'
@allowed_users(allowed_roles=['admin','managers']) # ACCESS CONTROL AS PER THE USER ROLES
def userslogs(request): #to view userslogs page
return render(request, 'internetForensics/userslogs.html')
@login_required(login_url='login') # RESTRICTION ON UNAUTHENTICATED USERS AND DERIVED FROM 'IMPORT LOGIN_REQUIRED'
def changepassword(request): #to view userslogs page
return render(request, 'internetForensics/changepassword.html')
© 2022 GitHub, Inc.
Terms
Privacy
Security
Status
Docs
Contact GitHub
Pricing
API
Training
Blog
About


Search or jump to...
Pull requests
Issues
Codespaces
Marketplace
Explore
JaneAldridge / DjangoForensics
Public
Watch 1
Fork 0
Star 0
Code
Issues
Pull requests
Actions
Projects
Wiki
Security
Insights
main
DjangoForensics / Group2 / internetForensics / templates / internetForensics /
Go to file
Add file
MUTEGIbeatrice UPDATES 2
0343527 2 days ago
History
..
audits.html edit 1 4 days ago
cases.html edit 1 4 days ago
changepassword.html UPDATES 2 2 days ago
create_reports.html UPDATES 2 2 days ago
criminalactivity.html edit 1 4 days ago
filling.html UPDATES 2 2 days ago
login.html UPDATES 2 2 days ago
main.html UPDATES 2 2 days ago
mainpage.html UPDATES 2 2 days ago
manage_reports.html UPDATES 2 2 days ago
messages.html UPDATES 2 2 days ago
navbar.html UPDATES 2 2 days ago
userslogs.html UPDATES 2 2 days ago
© 2022 GitHub, Inc.
Terms
Privacy
Security
Status
Docs
Contact GitHub
Pricing
API
Training
Blog
About

Search or jump to... Pull requests Issues Codespaces Marketplace Explore

JaneAldridge / DjangoForensics Public

Watch 1 Fork 0 Star 0

Code Issues Pull requests Actions Projects Wiki Security Insights

main DjangoForensics / Group2 / Group2 / settings.py / <> Jump to

Go to file ...

MUTEGIbeatrice UPDATES 2 ... Latest commit e343527 2 days ago History

2 contributors

167 lines (120 sloc) 4.32 KB

Raw Blame

```python
"""
Django settings for Group2 project.

Generated by 'django-admin startproject' using Django 4.1.3.

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

# 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/checklist/

# SECURITY WARNING: keep the secret key used in production secret!
SECRET_KEY = 'django-insecure-89v3do6quy)n(f!x51)=g!04shpghl-7vf^q@fuk-alxs4)!*$'

# SECURITY WARNING: don't run with debug turned on in production!
DEBUG = True

ALLOWED_HOSTS = []


# Application definition

INSTALLED_APPS = [
    'django.contrib.admin',
    'django.contrib.auth',
    'django.contrib.contenttypes',
    'django.contrib.sessions',
    'django.contrib.messages',
    'django.contrib.staticfiles',

    'django_password_validators',
     'django_password_validators.password_history',

    'internetForensics'
]

MIDDLEWARE = [
    '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 = 'Group2.urls'

TEMPLATES = [
    {
        'BACKEND': 'django.template.backends.django.DjangoTemplates',
        'DIRS': ['templates'],
        '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.messages',
            ],
        },
    },
]

WSGI_APPLICATION = 'Group2.wsgi.application'


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

DATABASES = {
    'default': {
        'ENGINE': 'django.db.backends.sqlite3',
        'NAME': BASE_DIR / 'db.sqlite3',
    }
}


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

AUTH_PASSWORD_VALIDATORS = [
    {
        'NAME': 'django.contrib.auth.password_validation.UserAttributeSimilarityValidator',
        'OPTIONS': {
         'max_similarity': 0.7,
         'user_attributes': ("username", "first_name", "last_name", "email")
      }
    },
    {
        'NAME': 'django.contrib.auth.password_validation.MinimumLengthValidator',
           'OPTIONS': {
            'min_length': 8,
            }
    },
    {
        'NAME': 'django.contrib.auth.password_validation.CommonPasswordValidator',
    },

    {
        'NAME': 'django.contrib.auth.password_validation.NumericPasswordValidator',
    },
    {
        'NAME': 'django_password_validators.password_history.password_validation.UniquePasswordsValidator',
        'OPTIONS': {
            'last_passwords': 5 # Only the last 5 passwords entered by the user
        }
    },
    {
        'NAME': 'django_password_validators.password_character_requirements.password_validation.PasswordCharacterValidator',
        'OPTIONS': {
             'min_length_digit': 1,
             'min_length_special': 1,
             'min_length_lower': 1,
             'min_length_upper': 1,
             'special_characters': "~!@#$%^&*()_+:;'[]"
         }
    },
]


# So as to keep updating the expiry time with every request

SESSION_SAVE_EVERY_REQUEST = True

PASSWORD_EXPIRE_FORCE = True #to force password change


PASSWORD_EXPIRE_EXCLUDE_SUPERUSERS = True #to exclude superusers from the password expiration


# 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 = 'static/'

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

DEFAULT_AUTO_FIELD = 'django.db.models.BigAutoField'

import os
```