Secure code Review for App\_Backend:

<https://github.com/alexbaar/app_Backend/tree/f4f77552d8269407d5f6c5f100068765dc6567dc>

# Introduction

This document includes report and finding through secure code review for the backend components of App Dev Team of Redback Operations. This backend serves as the core for various operations, including user authentication, file handling, and data management. This review identifies potential security vulnerabilities and provides best practices to mitigate risks. This review primarily focuses on ensuring secure password handling, preventing Cross-Site Request Forgery (CSRF), and addressing insecure file upload handling.

# Code Overview

The codebase is structured to support a Django application. It encompasses various modules that interact to provide backend functionalities. Here's an overview of the key components and their roles in the application:

Models.py: It defines the data structure in the application. The models.py file contains definitions for user accounts, file uploads, and other database entities. Django's ORM (Object-Relational Mapping) facilitates these relationships and ensures proper data management.

Views.py: This handles the HTTP requests and responses. The views.py file includes functions that manage user authentication, data retrieval, and other business logic. These functions interact with the models to retrieve and manipulate data, returning appropriate responses to clients.

Urls.py: The urls.py file defines the routing for HTTP requests. It maps specific URLs to corresponding view functions, ensuring proper request handling based on the URL structure.

ASGI/WSGI Configuration: These files configure Django's interaction with the web server, which enables the asynchronous and synchronous request handling.

# Vulnerability Findings:

Some finding that I found within the code has been added to the secure code review template, in which I have explained the impact, location and advice for each vulnerability. I have combined different findings to one documentation below:

A logo with a person running

Description automatically generated**REDBACK OPERTAIONS**

**VULNERABILITY FOUNDED**

**Insecure Password Handling (CWE-319)**

|  |  |  |  |
| --- | --- | --- | --- |
| **Name** | **Team** | **Role** | **Is this a re-tested Finding?** |
| Tushar Sharma | Cybersecurity | Secure Code Review | No |

|  |
| --- |
| **Was this Finding Successful?** |
| Yes |

**Risk Rating**  
Impact: Major  
Likelihood: Moderate

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Impact values** | | | | |
| **Very Minor** | **Minor** | **Significant** | **Major** | **Severe** |
| Risk that holds little to no impact. Will not cause damage and regular activity can continue. | Risk that holds minor form of impact, but not significant enough to be of threat. Can cause some damage but not enough to impede regular activity. | Risk that holds enough impact to be somewhat of a threat. Will cause damage that can impede regular activity but will be able to run normally. | Risk that holds major impact to be of threat. Will cause damage that will impede regular activity and will not be able to run normally. | Risk that holds severe impact and is a threat. Will cause critical damage that can cease activity to be run. |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Likelihood** | | | | |
| **Rare** | **Unlikely** | **Moderate** | **High** | **Certain** |
| Event may occur and/or if it did, it happens in specific circumstances. | Event could occur occasionally and/or could happen (at some point) | Event may occur and/or happens. | Event occurs at times and/or probably happens a lot. | Event is occurring now and/or happens frequently. |

**Impact**

This vulnerability can lead to major security breaches because passwords are being compared in plain text. An attacker with access to the code or logs could potentially retrieve sensitive information, compromising user accounts.

**Location of vulnerability**

Path:

app\_Backend/backend\_server/views.py:178

app\_Backend/backend\_server/views.py:210

**Evidence**

**A screen shot of a computer program

Description automatically generated**

**A screen shot of a computer program

Description automatically generated**

**Remediation Advice**

To rectify this, instead of comparing plain-text passwords, implement Django's `authenticate()` and `check\_password()` to verify passwords securely.

These function securely checks for user credentials, as it hashes the password, and then compares with the stored hash in the database, this eliminates the risk and this approach ensures that even if someone get access to source code or logs, they cannot retrieve the actual passwords.

from django.contrib.auth import authenticate, login

@api\_view(['POST'])

def login\_view(request):

    # Extract email and password from the POST request

    email = request.data.get('email')

    password = request.data.get('password')

    # Use Django's built-in authentication method to verify user credentials

    user = authenticate(username=email, password=password)

    if user is not None:

        # If the authentication is successful, log the user in

        # This creates a session for the user and sets the necessary session cookies

        login(request, user)

        # Return a successful response with additional information if needed

        return Response({

            'message': 'Login successful',

            'user\_id': user.id,  # Return the user's ID for reference

        }, status=status.HTTP\_200\_OK)

    else:

        # If authentication fails, return a 401 Unauthorized status

        # This indicates that the credentials were incorrect

        return Response({'message': 'Incorrect email or password'}, status=status.HTTP\_401\_UNAUTHORIZED)

**References**

<https://docs.djangoproject.com/en/5.0/topics/auth/>

<https://cwe.mitre.org/data/definitions/319.html>

**Contact Details**

Student Name: Tushar Sharma

Student Id: 222197136

A logo with a person running

Description automatically generated**REDBACK OPERTAIONS**

**VULNERABILITY FOUNDED**

**Cross-Site Request Forgery (CWE-352)**

|  |  |  |  |
| --- | --- | --- | --- |
| **Name** | **Team** | **Role** | **Is this a re-tested Finding?** |
| Tushar Sharma | Cybersecuirty | Secure Code Review | No |

|  |
| --- |
| **Was this Finding Successful?** |
| Yes |

**Risk Rating**  
Impact: Significant  
Likelihood: High

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Impact values** | | | | |
| **Very Minor** | **Minor** | **Significant** | **Major** | **Severe** |
| Risk that holds little to no impact. Will not cause damage and regular activity can continue. | Risk that holds minor form of impact, but not significant enough to be of threat. Can cause some damage but not enough to impede regular activity. | Risk that holds enough impact to be somewhat of a threat. Will cause damage that can impede regular activity but will be able to run normally. | Risk that holds major impact to be of threat. Will cause damage that will impede regular activity and will not be able to run normally. | Risk that holds severe impact and is a threat. Will cause critical damage that can cease activity to be run. |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Likelihood** | | | | |
| **Rare** | **Unlikely** | **Moderate** | **High** | **Certain** |
| Event may occur and/or if it did, it happens in specific circumstances. | Event could occur occasionally and/or could happen (at some point) | Event may occur and/or happens. | Event occurs at times and/or probably happens a lot. | Event is occurring now and/or happens frequently. |

**Business Impact**

This vulnerability can lead to unauthorized actions on behalf of authenticated users. By disabling CSRF protection with the @csrf\_exempt decorator, the application becomes susceptible to attacks where malicious actors could submit requests on behalf of users without their consent, leading to unauthorized data modification or other harmful effects.

**Location of vulnerability**

Path:

app\_Backend/backend\_server/views.py:170

app\_Backend/backend\_server/views.py:201

app\_Backend/backend\_server/views.py:228

**Evidence**

A screen shot of a computer program

Description automatically generated

A screenshot of a computer program

Description automatically generated

A screen shot of a computer program

Description automatically generated

**Remediation Advice**

To mitigate this vulnerability, remove the @csrf\_exempt decorator to ensure CSRF protection is active. Django's CSRF protection uses a token-based mechanism to prevent unauthorized requests. When CSRF protection is enabled, requests that don't include a valid token are rejected, significantly reducing the risk of CSRF attacks.

**References**

<https://docs.djangoproject.com/en/5.0/ref/csrf/>

<https://cwe.mitre.org/data/definitions/352.html>

**Contact Details**

Student Name: Tushar Sharma

Student Id: 222197136

A logo with a person running

Description automatically generated**REDBACK OPERTAIONS**

**VULNERABILITY FOUNDED**

**Insecure File Upload Handling (CWE-434)**

|  |  |  |  |
| --- | --- | --- | --- |
| **Name** | **Team** | **Role** | **Is this a re-tested Finding?** |
| Tushar Sharma | Cybersecurity | Secure Code Review | No |

|  |
| --- |
| **Was this Finding Successful?** |
| Yes |

**Risk Rating**  
Impact: Major  
Likelihood: Moderate

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Impact values** | | | | |
| **Very Minor** | **Minor** | **Significant** | **Major** | **Severe** |
| Risk that holds little to no impact. Will not cause damage and regular activity can continue. | Risk that holds minor form of impact, but not significant enough to be of threat. Can cause some damage but not enough to impede regular activity. | Risk that holds enough impact to be somewhat of a threat. Will cause damage that can impede regular activity but will be able to run normally. | Risk that holds major impact to be of threat. Will cause damage that will impede regular activity and will not be able to run normally. | Risk that holds severe impact and is a threat. Will cause critical damage that can cease activity to be run. |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Likelihood** | | | | |
| **Rare** | **Unlikely** | **Moderate** | **High** | **Certain** |
| Event may occur and/or if it did, it happens in specific circumstances. | Event could occur occasionally and/or could happen (at some point) | Event may occur and/or happens. | Event occurs at times and/or probably happens a lot. | Event is occurring now and/or happens frequently. |

**Business Impact**

Insecure file upload handling can lead to serious security risks. If user-uploaded files are not properly validated, malicious actors can upload files containing malware, scripts, or other harmful content. This can result in unauthorized code execution, data loss, or system compromise.

**Location of vulnerbility**

**Path:** app\_Backend/backend\_server/models.py:47

**Evidence**

A screen shot of a computer program

Description automatically generated

**Remediation Advice**

To mitigate this vulnerability, implementing strict validation and sanitization is necessary for all user-uploaded files. This involves checking the file type, ensuring it's a valid image or document, and restricting potentially harmful file types.

from django.core.exceptions import ValidationError

# Function to validate file types

def validate\_file\_extension(value):

    valid\_extensions = ['jpg', 'jpeg', 'png', 'gif']  # List of acceptable extensions

    # Extract file extension by splitting on the dot and taking the last element

    ext = value.name.split('.')[-1].lower()

    if ext not in valid\_extensions:

        raise ValidationError(f'Unsupported file extension: {ext}')

class AccountDetails(models.Model):

    ...

    # Apply the validator to ensure only specific file types are accepted

    image = models.ImageField(null=True, blank=True, upload\_to='images/', validators=[validate\_file\_extension])

**References**

<https://docs.djangoproject.com/en/5.0/topics/files/>

<https://cwe.mitre.org/data/definitions/434.html>

**Contact Details**

Student Name: Tushar Sharma

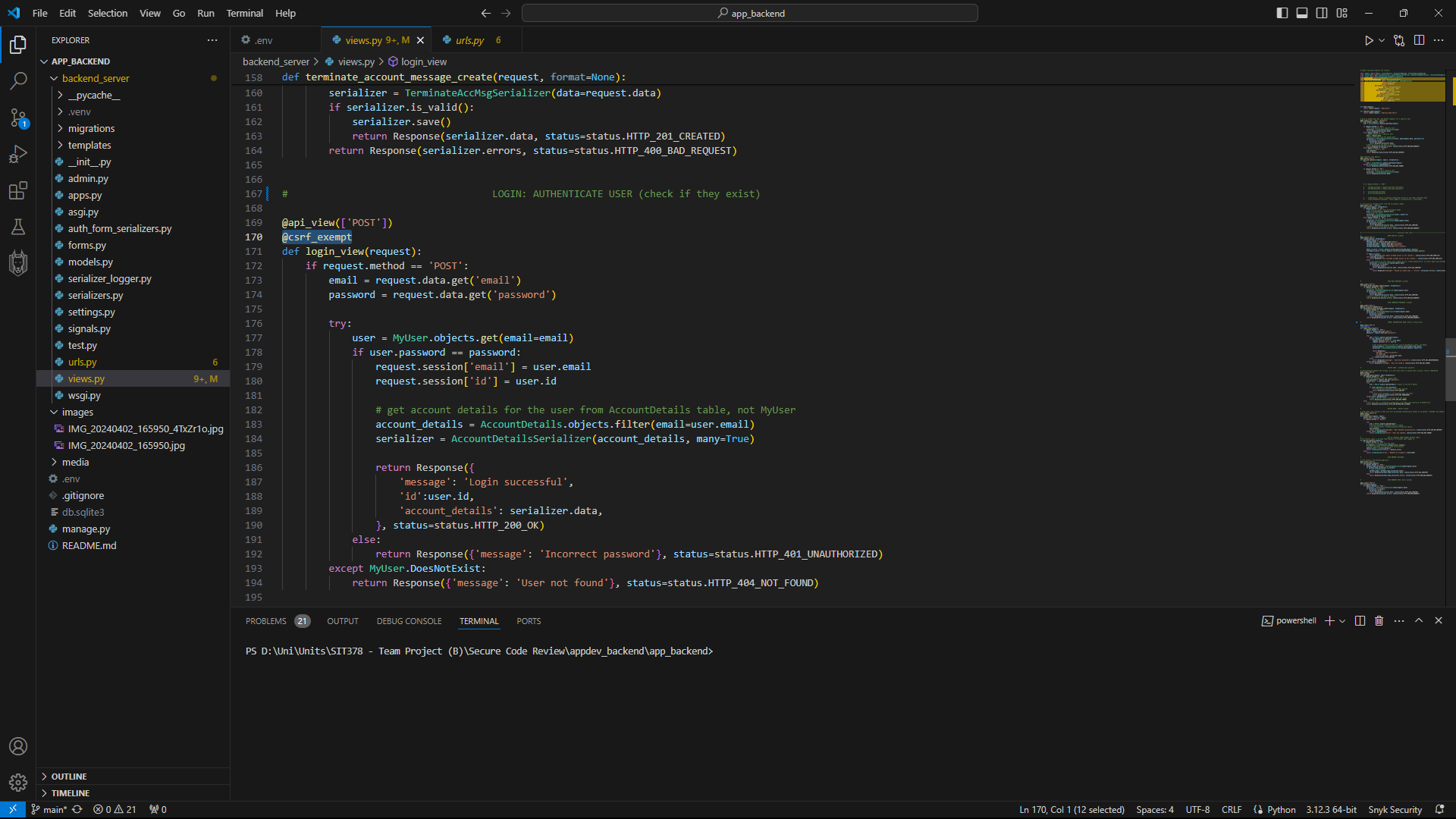
Student Id: 222197136

# Possible Areas for improvement:

## CSRF Attacks:

In the backend\_server/view.py file, @csrf\_exempt has been used three times, which could lead to significant risk as using @csrf\_exempt is used to bypass Cross-Site Request Forgery (CSRF) protection in Django. CSRF attacks occur when a malicious website tricks a user's browser into submitting unauthorized requests to another website where the user is authenticated. This can lead to unauthorized actions, data modification.

Django provides robust CSRF protection through a token-based mechanism. This ensures that only requests with a valid CSRF token are accepted, significantly reducing the risk of unauthorized actions.



To prevent CSRF attacks, remove `@csrf\_exempt` and ensure CSRF protection is active for all POST requests. If exceptions are necessary, it needs to implement with additional security measures like custom authentication or validation checks.

For reference: <https://docs.djangoproject.com/en/5.0/ref/csrf/>

## Session Handling

Session handling means to manage the user sessions to maintain user authentication, track user activity, and preserve state between requests. Proper session handling is crucial for ensuring user security and preventing unauthorized access to sensitive information. It secure the system from session hijacking, session fixation and improper session validation.

Session handling can be done in Backend\_servers/settings.py.

For Secure Session Handling,

* we can configure session cookies to ensure they are transmitted securely. Use SESSION\_COOKIE\_SECURE to enforce HTTPS-only transmission, preventing session hijacking through unsecured channels.
* Use of SESSION\_COOKIE\_AGE to set a reasonable session expiration time, minimizing the risk of long-lived sessions. Shorter session durations reduce the risk of unauthorized access due to session hijacking.
* Implementing the SESSION\_EXPIRE\_AT\_BROWSER\_CLOSE to ensure sessions are terminated when the user closes their browser.

Reference: <https://docs.djangoproject.com/en/5.0/ref/settings/#sessions>