Vul AI
Vulnerability Detection Report
File Name: views (1).py
Date: 2024-09-11 03:29:33.990533

Our System has analyzed the file views (1).py and found total of 8 functions in the file
Each function is analyzed by our fined tuned model to detect for possible vulnerabilities
A Final report has been generated covering all functions

GENERATED REPORT

TOTAL FUNCTIONS IN FILE 8

No.	Function CODE	Secure Code	Vulnerability Status	CWE IDS
1	def task_list(request): tasks = Task.objects.all() print(tasks) return render(request, 'task_list.html', {'tasks': tasks})	{"tasks": Task.objects.all().values_list()}	The code is vulnerable to a Cross-Site Scripting (XSS) attack because it directly prints the output of the database query without any sanitization or validation. An attacker could inject malicious HTML or JavaScript code into the database, which would be executed by the browser when the page is rendered.	CWE-434: Unrestricted Include in Library File
2	def task_details(request,t_id): task = Task.objects.get(t_id=t_id) return render(request, 'task_details.html', {'task': task})	{"task": Task.objects.get(t_id=t_id.id)}; # Use t_id.id to prevent SQL injection	The code is vulnerable to SQL injection because it directly uses user input (t_id) to query the database without proper sanitization or parameterization.	CWE-89: Improper Neutralization of Input During Web Page Generation ('Cross-site Scripting')
3	def task_create(request): if request.method == 'POST': try: title = request.POST.get('title') description = request.POST.get('description') status = request.POST.get('status') priority = request.POST.get('priority') due_date = request.POST.get('due_date') # category = request.POST.get('category') category = TaskCategoy.objects.get(name=request.POST.get('category')) # assigned_to = request.POST.get('assigned_to') assigned_to = Member.objects.get(name=request.POST.get('assigned_to')) print("TASK CATEGORY,---",category) task = Task.objects.create(title=title,description=description,status=status,priority=priority,due_date=due_date,category=category,assigned_to=assigned_to) task.save() return redirect('task_list') except Exception as e: print(e) member = Member.objects.all() task_category = TaskCategoy.objects.all() print("MEMBER---",member) return render(request, 'task_create.html',{'member':member,'task_category':task_category})	def task_create(request): if request.method == 'POST': try: title = request.POST.get('title', '') description = request.POST.get('description', '') status = request.POST.get('status', '') priority = request.POST.get('priority', '') due_date = request.POST.get('due_date', '') category = request.POST.get('category', '') assigned_to = request.POST.get('assigned_to', '') # Validate user input if not all([title, description, status, priority, due_date, category, assigned_to]): return render(request, 'task_create.html', {'error': 'Please fill in all fields.'}) category = TaskCategoy.objects.filter(name=category).first() if not category: return render(request, 'task_create.html', {'error': 'Invalid category.'}) assigned_to = Member.objects.filter(name=assigned_to).first() if not assigned_to: return render(request, 'task_create.html', {'error': 'Invalid assigned to.'}) task = Task.objects.create(title=title, description=description, status=status, priority=priority, due_date=due_date, category=category, assigned_to=assigned_to) task.save() return redirect('task_list') except Exception as e: print(e) member = Member.objects.all() task_category = TaskCategoy.objects.all() return render(request, 'task_create.html', {'member': member, 'task_category': task_category})	The code does not validate user input properly, which can lead to potential security vulnerabilities such as SQL injection and cross-site scripting (XSS).	CWE-20: Improper Input Validation

4	def task_update(request,t_id): try: task = Task.objects.get(t_id=t_id) members = Member.objects.all() category = TaskCategoy.objects.all() print("TASK",task.t_id) if request.method == 'POST': task.title = request.POST.get('title') task.description = request.POST.get('description') task.status = request.POST.get('status') task.priority = request.POST.get('priority') task.due_date = request.POST.get('due_date') task.category = TaskCategoy.objects.get(name=request.POST.get('category')) task.assigned_to = Member.objects.get(name=request.POST.get('assigned_to')) task.save() print("SUCESSS_____") return redirect('task_list') return render(request, 'task_update.html',{'task':task,'member':members,'category':category}) except Exception as e: print('EXCEPTTION+==========',e)	def task_update(request, t_id): try: task = Task.objects.get(t_id=t_id) members = Member.objects.all() category = TaskCategoy.objects.all() print("TASK", task.t_id) if request.method == 'POST': # Validate input here task.title = request.POST.get('title', '') task.description = request.POST.get('description', '') task.status = request.POST.get('status', '') task.priority = request.POST.get('priority', '') task.due_date = request.POST.get('due_date', '') task.category = TaskCategoy.objects.get(name=request.POST.get('category', '')) task.assigned_to = Member.objects.get(name=request.POST.get('assigned_to', '')) task.save() print("SUCESSS_____") return redirect('task_list') return render(request, 'task_update.html', {'task': task,'member': members, 'category': category}) except Exception as e: print('EXCEPTTION+==========', e)	The code does not validate the input from the request, which can lead to potential security vulnerabilities such as SQL injection or cross-site scripting (XSS).	CWE-20: Improper Input Validation
5	def task_delete(request,t_id): task = Task.objects.get(t_id=t_id) task.delete() return redirect('task_list')	def task_delete(request, t_id): task = Task.objects.get(t_id=t_id) task.delete() return redirect('task_list') # Secure version from django.core.exceptions import ObjectDoesNotExist def task_delete(request, t_id): try: task = Task.objects.get(t_id=t_id) task.delete() except ObjectDoesNotExist: # Handle the case when the task does not exist pass return redirect('task_list')	The task_delete function does not validate the input t_id, which can lead to a potential SQL injection attack. An attacker can manipulate the t_id parameter to delete arbitrary tasks.	CWE-89: Improper Neutralization of Input During Web Page Generation ('Cross-site Scripting')
6	def users_create(request): try: if request.method == 'POST': print("USER") name = request.POST.get('name') email = request.POST.get('email') password = request.POST.get('password') user = User.objects.create_user(first_name=name,username=email,password=password) Member.objects.create(name=name, email=email,password=password,) print("USER CREATED") return redirect('user_list') return render(request, 'user_create.html') except Exception as e: print('EXCEPTTION+==========',e)	{"try": "if request.method == 'POST':", "name": request.POST.get('name'), "email": request.POST.get('email'), "password": request.POST.get('password'), "user": User.objects.create_user(first_name=name,username=email,password=password), "Member": Member.objects.create(name=name, email=email,password=password,)}; return redirect('user_list')	The code does not validate or sanitize user input, making it vulnerable to XSS attacks. Additionally, storing passwords in plain text is insecure.	CWE-89: Improper Neutralization of Input During Web Page Generation ('Cross-site Scripting')
7	def user_list(request): members = Member.objects.all() return render(request, 'member_list.html',{'members':members})	def user_list(request): members = Member.objects.filter(name=request.user.name) return render(request,'member_list.html',{'members':members})	The code does not validate or sanitize user input, making it vulnerable to cross-site scripting (XSS) attacks.	CWE-89: Improper Neutralization of Input During Web Page Generation ('Cross-site Scripting')

8	def category_create(request): try: if request.method == 'POST': name = request.POST.get('name') description = request.POST.get('description') category = TaskCategoy.objects.create(name=name,description=description) print("CATEGORY CREATED") return redirect('categories') return render(request, 'category_create.html') except Exception as e: print('EXCEPTTION+=============',e)	def category_create(request): try: if request.method == 'POST': name = request.POST.get('name', '') description = request.POST.get('description', '') # Validate the input if not name or not description: return render(request, 'category_create.html', {'error': 'Please fill in all fields'}) category = TaskCategoy.objects.create(name=name, description=description) print("CATEGORY CREATED") return redirect('categories') return render(request, 'category_create.html') except Exception as e: print('EXCEPTION=============', e)	The code does not validate the input from the request, which can lead to potential security vulnerabilities such as SQL injection or cross-site scripting (XSS).	CWE-20: Improper Input Validation
Total Vulneratble Functions				2

Vulnerability Description And Effects on Robotic Systems