Admin App

Views.py:

from django.shortcuts import render, redirect

from adminsapp.models import Admins

from adminsapp.models import Employee

from adminsapp.forms import EmployeeForm

from adminsapp.models import Project

from adminsapp.forms import ProjectForm

from employeeapp.models import Bug, Testcase

from employeeapp.forms import BugForm, TestcaseForm

from adminsapp.models import News

from adminsapp.forms import NewsForm

# Create your views here.

def admins(request):

if request.method == "POST":

username = request.POST["username"]

password = request.POST["password"]

request.session["username"] = username

try:

admin = Admins.objects.get(username=username, password=password)

msg = "Data Found"

print("msg")

return render(request, "admins\_home.html", {"admin": admin, "msg": msg})

except Exception as e:

msg = "Invalid Credentials"

print(e)

return render(request, "admins.html", {"msg": msg})

return render(request, "admins.html", {})

def admins\_home(request):

return render(request, "admins\_home.html", {})

def admins\_change\_pwd(request):

username = request.session["username"]

if islogin(request):

if request.method == "POST":

username = request.session["username"]

password = request.POST["password"]

new\_password = request.POST["new\_password"]

try:

admins = Admins.objects.get(username=username, password=password)

admins.password = new\_password

admins.save()

msg = "Successfully Password Update"

return redirect("/admins\_logout",{"msg":msg})

except:

return render(request, "admins\_chpwd.html", {"msg": "Invalid Data", "username":username})

return render(request, "admins\_chpwd.html", {"username":username})

else:

return render(request, "admins.html", {"username":username})

def admins\_logout(request):

request.session["username"] = ""

del request.session["username"]

return render(request, "admins.html", {})

def islogin(request):

if request.session.\_\_contains\_\_("username"):

return True

else:

return False

def add\_employee(request):

if request.method == "POST":

print("hi")

employee = EmployeeForm(request.POST)

print("hi1")

try:

if employee.is\_valid():

print("hi2")

employee.save()

return render(request, "add\_employee.html", {"msg": "Success"})

print("h3")

except Exception as e:

print(e)

return render(request, "add\_employee.html", {})

return render(request, "add\_employee.html", {})

def view\_employee(request):

if request.method == 'GET':

employee = Employee.objects.all()

return render(request, "view\_employee.html", {"employee": employee})

def delete\_employee(request, id):

employee = Employee.objects.get(id=id)

employee.delete()

return redirect("/view\_employee")

def edit\_employee(request, id):

employee = Employee.objects.get(id=id)

return render(request, "edit\_employee.html", {"employee": employee})

def update\_employee(request):

if request.method == "POST":

print("h")

userid = request.POST["id"]

print("hi")

employee = Employee.objects.get(id=userid)

print("HI")

employee = EmployeeForm(request.POST, instance=employee)

print("hii")

if employee.is\_valid():

print("hiii")

employee.save()

print("hiiiii")

return redirect("/view\_employee")

return redirect("/view\_employee")

def add\_project(request):

managers = Employee.objects.filter(role="manager")

return render(request, "add\_project.html", {"managers": managers})

def add\_new\_project(request):

if request.method == "POST":

project = ProjectForm(request.POST)

try:

if project.is\_valid():

print("hi2")

project.save()

print("h3")

return render(request, "admins\_home.html", {})

except Exception as e:

print(e)

return render(request, "add\_project.html", {})

return render(request, "add\_project.html", {})

def view\_project(request):

if request.method == 'GET':

projects = Project.objects.all()

return render(request, "view\_project.html", {"projects": projects})

def delete\_project(request, id):

project = Project.objects.get(id=id)

project.delete()

return redirect("/view\_project")

def edit\_project(request, id):

project = Project.objects.get(id=id)

managers = Employee.objects.filter(role="manager")

return render(request, "edit\_project.html", {"project": project, "managers": managers})

def update\_project(request):

if request.method == "POST":

print("h")

userid = request.POST["id"]

print("hi")

project = Project.objects.get(id=userid)

print("HI")

project = EmployeeForm(request.POST, instance=project)

print("hii")

if project.is\_valid():

print("hiii")

project.save()

print("hiiiii")

return redirect("/view\_project")

return redirect("/view\_project")

def view\_testcases(request):

if request.method == "GET":

testcases = Testcase.objects.all()

return render(request, "view\_testcases.html", {"testcases": testcases})

def view\_bugs(request):

if request.method == "GET":

bugs = Bug.objects.all()

return render(request, "view\_bugs.html", {"bugs": bugs})

def add\_news(request):

if request.method == "POST":

news = NewsForm(request.POST)

if news.is\_valid():

news.save()

return render(request, "add\_news.html", {"msg": "Success"})

return render(request, "add\_news.html", {})

def view\_news(request):

if request.method == "GET":

news = News.objects.all()

return render(request, "view\_news.html", {"news": news})

def delete\_news(request,id):

news=News.objects.get(id=id)

news.delete()

return redirect("/view\_news")

def edit\_news(request, id):

news = News.objects.get(id=id)

return render(request, "edit\_news.html", {"news": news})

def update\_news(request):

if request.method == "POST":

userid = request.POST["id"]

news = News.objects.get(id=userid)

news = NewsForm(request.POST, instance=news)

if news.is\_valid():

news.save()

return redirect("/view\_news")

return redirect("/view\_news")

**Employee App:**

**Views.py:**

from django.shortcuts import render, redirect

from adminsapp.models import Employee, Project, News

from adminsapp.forms import EmployeeForm, ProjectForm

from employeeapp.models import Testcase, Bug, Requirements

from employeeapp.forms import TestcaseForm, BugForm, RequirementsForm

# Create your views here.

def employee(request):

if request.method == "POST":

email = request.POST["email"]

password = request.POST["password"]

role = request.POST["role"]

try:

employee = Employee.objects.get(email=email, password=password, role=role)

print(role)

request.session["email"] = email

request.session["role"] = role

return render(request, "employee\_home.html", {"employee":employee,"role":role})

except Exception as e:

msg = "Invalid Credentials"

print(e)

return render(request, "employee\_login.html", {"msg": msg})

return render(request, "employee\_login.html", {})

def employee\_home(request):

return render(request, "employee\_home.html", {})

def employee\_change\_pwd(request):

email = request.session["email"]

if islogin(request):

# role = request.session["role"]

if request.method == "POST":

email = request.session["email"]

password = request.POST["password"]

new\_password = request.POST["new\_password"]

try:

employee = Employee.objects.get(email=email, password=password)

employee.password = new\_password

employee.save()

msg = "Successfully Password Update"

return render(request, "index.html", {"msg": msg})

except:

msg = "Invalid Data"

return render(request, "employee\_chpwd.html", {"msg": msg,"email":email})

return render(request, "employee\_chpwd.html", {"email":email})

else:

return render(request, "index.html", {"email":email})

def employee\_logout(request):

request.session["email"] = ""

request.session["role"] = ""

del request.session["email"]

del request.session["role"]

return render(request, "employee\_login.html", {})

def islogin(request):

if request.session.\_\_contains\_\_("email"):

return True

else:

return False

def employee\_edit\_profile(request):

email = request.session["email"]

role = request.session["role"]

employ = Employee.objects.get(email=email, role=role)

print(employ)

return render(request, "employee\_edit\_profile.html", {"employ": employ, "role":role})

def employee\_update\_profile(request):

email = request.session['email']

role = request.session["role"]

data = Employee.objects.get(email=email, role=role)

if request.method == "POST":

print("hi1")

email = request.session["email"]

print(email)

emp = Employee.objects.get(email=email, role=role)

print("hi3")

employee = EmployeeForm(request.POST, instance=emp)

print("hi4")

try:

if employee.is\_valid():

print("hi6")

employee.save()

print("hi")

return render(request, "employee\_home.html", {"msg": "Successfully Update", "role": role})

except Exception as e:

print(e)

return render(request, "employee\_home.html", {"msg": "your details not updated", "role":role})

return render(request, "employee\_edit\_profile.html", {"employee":data, "role":role})

def add\_testcase(request):

if request.method == "POST":

testcase = TestcaseForm(request.POST)

print("hi1")

try:

if testcase.is\_valid():

print("hi2")

testcase.save()

print("h3")

except Exception as e:

print(e)

return render(request, "add\_testcase.html", {"msg": "success"})

tester\_emails = request.session["email"]

details\_project = Project.objects.all()

requirements = Requirements.objects.all()

return render(request, "add\_testcase.html", {"tester\_emails": tester\_emails, "details\_project": details\_project,"requirements":requirements})

def view\_testcase(request):

if request.method == "GET":

testcases = Testcase.objects.all()

role=request.session["role"]

return render(request, "view\_testcase.html", {"testcases": testcases,"role": role})

def delete\_testcase(request, id):

testcase = Testcase.objects.get(id=id)

testcase.delete()

return redirect("/view\_testcase")

def edit\_testcase(request, id):

testcases = Testcase.objects.get(id=id)

tester\_emails = request.session["email"]

details\_project = Project.objects.all()

requirements = Requirements.objects.all()

return render(request, "edit\_testcase.html",

{"testcases": testcases, "tester\_emails": tester\_emails, "details\_project": details\_project,"requirements":requirements})

def update\_testcase(request):

if request.method == "POST":

userid = request.POST["id"]

testcase = Testcase.objects.get(id=userid)

testcase = TestcaseForm(request.POST, instance=testcase)

print("hi1")

try:

if testcase.is\_valid():

print("hi2")

testcase.save()

print("h3")

except Exception as e:

print(e)

return redirect("/view\_testcase", {"msg": "success"})

return render(request, "edit\_testcase.html", {})

def add\_bug(request):

if request.method == "POST":

print("hi")

bugs = BugForm(request.POST,request.FILES)

print(bugs.errors)

print("hi1")

try:

if bugs.is\_valid():

print("hi2")

bugs.save()

print("h3")

except Exception as e:

print(e)

return render(request, "add\_bug.html", {"msg": "success"})

tested\_by = request.session["email"]

testcaseids = Testcase.objects.all()

return render(request, "add\_bug.html",

{"tested\_by": tested\_by, "testcaseids": testcaseids})

def view\_bug(request):

if request.method=="GET":

bugs=Bug.objects.all()

role=request.session["role"]

return render(request,"view\_bug.html",{"bugs":bugs,"role":role})

def delete\_bug(request, id):

bugs = Bug.objects.get(id=id)

bugs.delete()

return redirect("/view\_bug")

def edit\_bug(request, id):

bugs = Bug.objects.get(id=id)

tested\_by = request.session["email"]

testcaseids = Testcase.objects.all()

return render(request, "edit\_bug.html",

{"tested\_by": tested\_by, "testcaseids": testcaseids,"bugs":bugs})

def update\_bug(request):

if request.method == "POST":

userid = request.POST["id"]

bugs = Bug.objects.get(id=userid)

print("hi")

bugs = BugForm(request.POST,request.FILES,instance=bugs)

print(bugs.errors)

print("hi1")

try:

if bugs.is\_valid():

print("hi2")

bugs.save()

print("h3")

except Exception as e:

print(e)

return redirect("/view\_bug", {"msg": "success"})

return render(request, "edit\_bug.html", {})

def developer\_view\_bugs(request):

if request.method=="GET":

role=request.session["role"]

print(role)

bugs=Bug.objects.all()

return render(request,"developer\_view\_bugs.html",{"bugs":bugs,"role":role})

def developer\_view\_testcases(request):

if request.method == "GET":

role=request.session["role"]

testcases = Testcase.objects.all()

return render(request, "developer\_view\_testcases.html", {"testcases": testcases,"role":role})

def developer\_update\_status(request,id):

bugs = Bug.objects.get(id=id)

email= request.session["email"]

return render(request,"developer\_status\_update.html",{"bugs":bugs,"email":email,"id":id})

def developer\_status\_insert(request):

if request.method == "POST":

try:

tested\_by\_email = request.session["email"]

status=request.POST['status']

id=request.POST['id']

print("hi1")

print("hi2")

bug=Bug.objects.get(id=id)

print("hi3")

bug.status=status

bug.tested\_by\_email=tested\_by\_email

print("hi4")

bug.save()

print("hi")

except Exception as e:

print(e)

return render(request, "developer\_view\_bugs.html", {})

return render(request, "developer\_status\_update.html", {})

def tester\_update\_status(request,id):

bugs = Bug.objects.get(id=id)

email= request.session["email"]

return render(request,"tester\_update\_status.html",{"bugs":bugs,"email":email,"id":id})

def tester\_status\_insert(request):

if request.method == "POST":

try:

tested\_by\_email = request.session["email"]

status=request.POST['status']

id=request.POST['id']

print("hi1")

print("hi2")

bug=Bug.objects.get(id=id)

print("hi3")

bug.status=status

bug.tested\_by\_email=tested\_by\_email

print("hi4")

bug.save()

print("hi")

except Exception as e:

print(e)

return render(request, "view\_bug.html", {})

return render(request, "tester\_update\_status.html", {})

def analyst\_view\_bugs(request):

if request.method=="GET":

bugs=Bug.objects.all()

return render(request,"analyst\_view\_bugs.html",{"bugs":bugs})

def analyst\_view\_testcases(request):

if request.method == "GET":

testcases = Testcase.objects.all()

return render(request, "analyst\_view\_testcases.html", {"testcases": testcases})

def developer(request):

if request.method == "POST":

email = request.POST["email"]

password = request.POST["password"]

role = request.POST["role"]

try:

employee = Employee.objects.get(email=email, password=password, role=role)

print(role)

request.session["email"] = email

request.session["role"] = role

return render(request, "developer\_home.html", {"employee":employee,"role":role})

except Exception as e:

msg = "Invalid Credentials"

print(e)

return render(request, "developer.html", {"msg": msg})

return render(request, "developer.html", {})

def developer\_home(request):

return render(request, "developer\_home.html", {})

def developer\_change\_pwd(request):

email = request.session["email"]

if islogin(request):

# role = request.session["role"]

if request.method == "POST":

email = request.session["email"]

password = request.POST["password"]

new\_password = request.POST["new\_password"]

try:

employee = Employee.objects.get(email=email, password=password)

employee.password = new\_password

employee.save()

msg = "Successfully Password Update"

return render(request, "index.html", {"msg": msg})

except:

msg = "Invalid Data"

return render(request, "developer\_change\_pwd.html", {"msg": msg,"email":email})

return render(request, "developer\_change\_pwd.html", {"email":email})

else:

return render(request, "index.html", {"email":email})

def developer\_logout(request):

request.session["email"] = ""

request.session["role"] = ""

del request.session["email"]

del request.session["role"]

return render(request, "developer.html", {})

def developer\_edit\_profile(request):

email = request.session["email"]

role = request.session["role"]

employ = Employee.objects.get(email=email, role=role)

print(employ)

return render(request, "developer\_edit\_profile.html", {"employ": employ, "role":role})

def developer\_update\_profile(request):

email = request.session['email']

role = request.session["role"]

data = Employee.objects.get(email=email, role=role)

if request.method == "POST":

print("hi1")

email = request.session["email"]

print(email)

emp = Employee.objects.get(email=email, role=role)

print("hi3")

employee = EmployeeForm(request.POST, instance=emp)

print("hi4")

try:

if employee.is\_valid():

print("hi6")

employee.save()

print("hi")

return render(request, "developer.html", {"msg": "Successfully Update ", "role": role})

except Exception as e:

print(e)

return render(request, "developer\_edit\_profile.html", {"msg": "Your Details Not Updated", "role":role})

return render(request, "developer\_edit\_profile.html", {"employee":data, "role":role})

def analyst\_login(request):

if request.method == "POST":

email = request.POST["email"]

password = request.POST["password"]

role = request.POST["role"]

try:

employee = Employee.objects.get(email=email, password=password, role=role)

print(role)

request.session["email"] = email

request.session["role"] = role

return render(request, "analyst\_home.html", {"employee":employee,"role":role})

except Exception as e:

msg = "Invalid Credentials"

print(e)

return render(request, "analyst\_login.html", {"msg": msg})

return render(request, "analyst\_login.html", {})

def analyst\_home(request):

return render(request, "analyst\_home.html", {})

def analyst\_change\_pwd(request):

email = request.session["email"]

if islogin(request):

# role = request.session["role"]

if request.method == "POST":

email = request.session["email"]

password = request.POST["password"]

new\_password = request.POST["new\_password"]

try:

employee = Employee.objects.get(email=email, password=password)

employee.password = new\_password

employee.save()

msg = "Successfully Password Update"

return render(request, "index.html", {"msg": msg})

except:

msg = "Invalid Data"

return render(request, "analyst\_change\_pwd.html", {"msg": msg,"email":email})

return render(request, "analyst\_change\_pwd.html", {"email":email})

else:

return render(request, "index.html", {"email":email})

def analyst\_logout(request):

request.session["email"] = ""

request.session["role"] = ""

del request.session["email"]

del request.session["role"]

return render(request, "analyst\_login.html", {})

def analyst\_edit\_profile(request):

email = request.session["email"]

role = request.session["role"]

employ = Employee.objects.get(email=email, role=role)

print(employ)

return render(request, "analyst\_edit\_profile.html", {"employ": employ, "role":role})

def analyst\_update\_profile(request):

email = request.session['email']

role = request.session["role"]

data = Employee.objects.get(email=email, role=role)

if request.method == "POST":

print("hi1")

email = request.session["email"]

print(email)

emp = Employee.objects.get(email=email, role=role)

print("hi3")

employee = EmployeeForm(request.POST, instance=emp)

print("hi4")

try:

if employee.is\_valid():

print("hi6")

employee.save()

print("hi")

return render(request, "analyst\_login.html", {"msg": "Successfully Update ", "role": role})

except Exception as e:

print(e)

return render(request, "analyst\_edit\_profile.html", {"msg": "Your Details Not Updated", "role":role})

return render(request, "analyst\_edit\_profile.html", {"employee":data, "role":role})

def developer\_view\_projects(request):

projects = Project.objects.all()

return render(request, "developer\_view\_projects.html", {"projects": projects})

def employee\_view\_projects(request):

projects = Project.objects.all()

return render(request, "employee\_view\_projects.html", {"projects": projects})

def analyst\_view\_projects(request):

projects = Project.objects.all()

return render(request, "analyst\_view\_projects.html", {"projects": projects})

def add\_requirements(request):

email = request.session['email']

projects = Project.objects.all()

if request.method == "POST":

requirements = RequirementsForm(request.POST)

if requirements.is\_valid():

requirements.save()

return render(request, "add\_requirements.html", {"msg": "Success","projects":projects,"email":email})

return render(request, "add\_requirements.html", {"projects":projects,"email":email})

def analyst\_view\_requirements(request,id):

projects = Project.objects.get(id=id)

requirements = Requirements.objects.filter(projects\_id=projects)

return render(request, "analyst\_view\_requirements.html", {"requirements": requirements})

def developer\_view\_requirements(request,id):

projects = Project.objects.get(id=id)

requirements = Requirements.objects.filter(projects\_id=projects)

return render(request, "developer\_view\_requirements.html", {"requirements": requirements})

def employee\_view\_requirements(request,id):

projects = Project.objects.get(id=id)

requirements = Requirements.objects.filter(projects\_id=projects)

return render(request, "employee\_view\_requirements.html", {"requirements": requirements})

def testcase\_bug\_piechart(request):

"""

Generates data and renders the template with testcase and bug counts.

"""

try:

requirements\_count = Requirements.objects.count()

testcase\_count = Testcase.objects.count()

bug\_count = Bug.objects.count()

return render(request, "testcase\_bug\_piechart.html", {"testcase\_count": testcase\_count, "bug\_count": bug\_count,"requirements\_count":requirements\_count})

except Exception as e:

print(e)

return render(request, "testcase\_bug\_piechart.html", {"msg": "Error retrieving data."})

def developer\_view\_notifications(request):

notifications = News.objects.all()

return render(request, "developer\_view\_notifications.html", {"notifications": notifications})

def analyst\_view\_notifications(request):

notifications = News.objects.all()

return render(request, "analyst\_view\_notifications.html", {"notifications": notifications})

def employee\_view\_notifications(request):

notifications = News.objects.all()

return render(request, "employee\_view\_notifications.html", {"notifications": notifications})

**Management App:**

**Views.py:**

from django.shortcuts import render, redirect

from managementapp.models import Contact

from managementapp.forms import ContactForm

from adminsapp.models import Employee, Project, News

from adminsapp.forms import EmployeeForm, ProjectForm

from managementapp.models import Task

from managementapp.forms import TaskForm

from employeeapp.models import Bug, Testcase, Requirements

from employeeapp.forms import BugForm, TestcaseForm

# Create your views here.

def index(request):

return render(request, "index.html", {})

def about(request):

return render(request, "about.html", {})

def blog(request):

return render(request, "blog.html", {})

def contact(request):

if request.method == "POST":

contact = ContactForm(request.POST)

if contact.is\_valid():

contact.save()

return render(request, "contact.html", {"msg": "Success"})

return render(request, "contact.html", {})

# def management(request):

# if request.method == "POST":

# email = request.POST['email']

# p\_word = request.POST['password']

# request.session['email'] = email

# role = request.session["role"]

# if Employee.objects.filter(email=email, password=p\_word, role="manager"):

# return render(request, "management\_home.html", {"msg": "Great! login Successfully"})

# else:

# return render(request, "management\_login.html", {"msg": "Sorry! you are not a manager"})

# return render(request, "management\_login.html", {})

def management(request):

if request.method == "POST":

email = request.POST["email"]

password = request.POST["password"]

role = request.POST["role"]

try:

employee = Employee.objects.get(email=email, password=password, role=role)

print(role)

request.session["email"] = email

request.session["role"] = role

return render(request, "management\_home.html", {"employee":employee,"role":role})

except Exception as e:

msg = "Invalid Credentials"

print(e)

return render(request, "management\_login.html", {"msg": msg})

return render(request, "management\_login.html", {})

def islogin(request):

if request.session.\_\_contains\_\_("email"):

return True

else:

return False

def manager\_change\_pwd(request):

email = request.session["email"]

if islogin(request):

# role = request.session["role"]

if request.method == "POST":

email = request.session["email"]

password = request.POST["password"]

new\_password = request.POST["new\_password"]

try:

employee = Employee.objects.get(email=email, password=password)

employee.password = new\_password

employee.save()

msg = "Successfully Password Update"

return render(request, "index.html", {"msg": msg})

except:

msg = "Invalid Data"

return render(request, "manager\_change\_pwd.html", {"msg": msg,"email":email})

return render(request, "manager\_change\_pwd.html", {"email":email})

else:

return render(request, "index.html", {"email":email})

def manager\_edit\_profile(request):

email = request.session["email"]

role = request.session["role"]

employ = Employee.objects.get(email=email, role=role)

print(employ)

return render(request, "manager\_edit\_profile.html", {"employ": employ, "role":role})

def manager\_update\_profile(request):

email = request.session['email']

role = request.session["role"]

data = Employee.objects.get(email=email, role=role)

if request.method == "POST":

print("hi1")

email = request.session["email"]

print(email)

emp = Employee.objects.get(email=email, role=role)

print("hi3")

employee = EmployeeForm(request.POST, instance=emp)

print("hi4")

try:

if employee.is\_valid():

print("hi6")

employee.save()

print("hi")

return render(request, "manager\_home.html", {"msg": "Successfully Update ", "role": role})

except Exception as e:

print(e)

return render(request, "manager\_edit\_profile.html", {"msg": "Your Details Not Updated", "role":role})

return render(request, "manager\_edit\_profile.html", {"employee":data, "role":role})

def management\_home(request):

return render(request, "management\_home.html", {})

def management\_logout(request):

request.session["email"] = ""

del request.session["email"]

return render(request, "management\_login.html", {})

def manager\_logout(request):

request.session["email"] = ""

request.session["role"] = ""

del request.session["email"]

del request.session["role"]

return render(request, "management\_login.html", {})

def islogin(request):

if request.session.\_\_contains\_\_("email"):

return True

else:

return False

def add\_task(request):

if request.method == "POST":

task = TaskForm(request.POST)

print("hi1")

try:

if task.is\_valid():

print("hi2")

task.save()

return render(request, "add\_task.html", {"msg": "Success"})

print("h3")

except Exception as e:

print(e)

return render(request, "add\_task.html", {})

mngr\_email = request.session["email"]

details\_emply = Employee.objects.all().exclude(role="manager")

details\_project = Project.objects.all()

return render(request, "add\_task.html",

{"mngr\_email": mngr\_email, "details\_emply": details\_emply, "details\_project": details\_project})

def view\_task(request):

if request.method == "GET":

tasks = Task.objects.all()

return render(request, "view\_task.html", {"tasks": tasks})

def delete\_task(request, id):

task = Task.objects.get(id=id)

task.delete()

return redirect("/view\_task")

def edit\_task(request, id):

tasks = Task.objects.get(id=id)

mngr\_email = request.session["email"]

details\_emply = Employee.objects.all().exclude(role="manager")

details\_project = Project.objects.all()

return render(request, "edit\_task.html", {"mngr\_email": mngr\_email, "details\_emply": details\_emply, "tasks": tasks,

"details\_project": details\_project})

def update\_task(request):

if request.method == "POST":

userid = request.POST["id"]

task = Task.objects.get(id=userid)

task = TaskForm(request.POST, instance=task)

print("hi1")

try:

if task.is\_valid():

print("hi2")

task.save()

print("h3")

except Exception as e:

print(e)

return redirect("/view\_task", {"msg": "success"})

return render(request, "edit\_task.html", {})

def manager\_view\_testcases(request):

if request.method == "GET":

testcases = Testcase.objects.all()

return render(request, "manager\_view\_testcases.html", {"testcases": testcases})

def manager\_view\_bugs(request):

if request.method == "GET":

bugs = Bug.objects.all()

return render(request, "manager\_view\_bugs.html", {"bugs": bugs})

def task\_update\_status(request, id):

tasks = Task.objects.get(id=id)

# email= request.session["email"]

return render(request, "task\_update\_status.html", {"tasks": tasks, "id": id})

def task\_status\_insert(request):

if request.method == "POST":

try:

# tested\_by\_email = request.session["email"]

status = request.POST['status']

id = request.POST['id']

print("hi1")

print("hi2")

task = Task.objects.get(id=id)

print("hi3")

task.status = status

# task.tested\_by\_email=tested\_by\_email

print("hi4")

task.save()

print("hi")

except Exception as e:

print(e)

return render(request, "view\_task.html", {})

return render(request, "task\_update\_status.html", {})

def manager\_view\_projects(request):

projects = Project.objects.all()

return render(request, "manager\_view\_projects.html", {"projects": projects})

def manager\_view\_requirements(request,id):

projects = Project.objects.get(id=id)

requirements = Requirements.objects.filter(projects\_id=projects)

return render(request, "manager\_view\_requirements.html", {"requirements": requirements})

def manager\_view\_notifications(request):

notifications = News.objects.all()

return render(request, "manager\_view\_notifications.html", {"notifications": notifications})

models.py:

from django.db import models

from employeeapp.models import Bug,Testcase

# Create your models here.

class Contact(models.Model):

message=models.CharField(max\_length=100)

name=models.CharField(max\_length=100)

email=models.EmailField(max\_length=100)

subject=models.CharField(max\_length=100)

class Task(models.Model):

manager\_email=models.CharField(max\_length=100)

status=models.CharField(max\_length=100)

employee\_email=models.EmailField()

project=models.CharField(max\_length=100)

description=models.CharField(max\_length=100)

enddate=models.DateField()

from django.db import models

from adminsapp.models import \*

# Create your models here.

class Requirements(models.Model):

email=models.EmailField()

projects=models.ForeignKey(Project,on\_delete=models.CASCADE)

title=models.CharField(max\_length=100)

description=models.CharField(max\_length=350)

priority=models.CharField(max\_length=300)

ndatetime = models.DateTimeField(auto\_now=True)

class Testcase(models.Model):

requirements=models.CharField(max\_length=350)

project\_name=models.CharField(max\_length=350)

module\_name=models.CharField(max\_length=350)

senario=models.CharField(max\_length=350)

description=models.CharField(max\_length=350)

input\_data=models.CharField(max\_length=350)

type\_of\_exception=models.CharField(max\_length=350)

pre\_condition=models.CharField(max\_length=350)

expected\_actual\_result=models.CharField(max\_length=350)

status=models.CharField(max\_length=300)

date\_of\_creation=models.DateField()

tester\_email=models.EmailField()

class Bug(models.Model):

testcase\_id=models.IntegerField()

description=models.CharField(max\_length=350)

test\_path=models.CharField(max\_length=300)

screenshot=models.ImageField(upload\_to="images/")

sevearity=models.CharField(max\_length=300)

priority=models.CharField(max\_length=300)

status=models.CharField(max\_length=300)

date=models.DateField()

tested\_by\_email=models.EmailField()

from django.db import models

# Create your models here.

class Admins(models.Model):

username = models.CharField(max\_length=50)

password = models.IntegerField()

class Employee(models.Model):

empid=models.IntegerField()

empname=models.CharField(max\_length=100)

job=models.CharField(max\_length=100)

email=models.EmailField()

password=models.CharField(max\_length=100)

dateofjoin=models.DateField()

salary=models.IntegerField()

gender=models.CharField(max\_length=100)

role=models.CharField(max\_length=100)

class Project(models.Model):

title=models.CharField(max\_length=100)

code=models.CharField(max\_length=100)

technology=models.CharField(max\_length=100)

domain=models.CharField(max\_length=100)

description=models.CharField(max\_length=100)

fromdate=models.DateField()

todate=models.DateField()

manager=models.CharField(max\_length=100)

class News(models.Model):

title=models.CharField(max\_length=200)

description=models.CharField(max\_length=200)

date=models.DateTimeField(auto\_now=True)

**Urls.py:**

"""BugAndTestcase URL Configuration

The `urlpatterns` list routes URLs to views. For more information please see:

https://docs.djangoproject.com/en/3.2/topics/http/urls/

Examples:

Function views

1. Add an import: from my\_app import views

2. Add a URL to urlpatterns: path('', views.home, name='home')

Class-based views

1. Add an import: from other\_app.views import Home

2. Add a URL to urlpatterns: path('', Home.as\_view(), name='home')

Including another URLconf

1. Import the include() function: from django.urls import include, path

2. Add a URL to urlpatterns: path('blog/', include('blog.urls'))

"""

from django.contrib import admin

from django.urls import path

from django.conf import settings

from django.conf.urls.static import static

from managementapp import views as management\_view

from adminsapp import views as admins\_view

from employeeapp import views as employee\_view

urlpatterns = [

#----------------management---------------------

path("",management\_view.index,name="index"),

path("index/", management\_view.index, name="index"),

path("about/",management\_view.about,name="about"),

path("blog/",management\_view.blog,name="blog"),

path("contact/",management\_view.contact,name="contact"),

path("management/",management\_view.management,name="management"),

path("management\_home/",management\_view.management\_home,name="management\_home"),

path("manager\_logout/",management\_view.manager\_logout,name="manager\_logout"),

path("management\_logout/", management\_view.management\_logout, name="management\_logout"),

path("add\_task/",management\_view.add\_task,name="add\_task"),

path("view\_task/",management\_view.view\_task,name="view\_task"),

path("delete\_task/<id>",management\_view.delete\_task,name="delete\_task"),

path("edit\_task/<int:id>",management\_view.edit\_task,name="edit\_task"),

path("update\_task/",management\_view.update\_task,name="update\_task"),

#path("view\_testcases/",management\_view.view\_testcases,name="view\_testcases"),

#path("view\_bugs/",management\_view.view\_bugs,name="view\_bugs"),

path("task\_update\_status/<id>",management\_view.task\_update\_status,name="task\_update\_status"),

path("task\_status\_insert/",management\_view.task\_status\_insert,name="task\_status\_insert"),

path("manager\_view\_testcases/",management\_view.manager\_view\_testcases,name="manager\_view\_testcases"),

path("manager\_view\_bugs/",management\_view.manager\_view\_bugs,name="manager\_view\_bugs"),

path("manager\_change\_pwd/", management\_view.manager\_change\_pwd, name="manager\_change\_pwd"),

path("manager\_edit\_profile/", management\_view.manager\_edit\_profile, name="manager\_edit\_profile"),

path("manager\_update\_profile/", management\_view.manager\_update\_profile, name="manager\_update\_profile"),

path("manager\_view\_projects/", management\_view.manager\_view\_projects, name="manager\_view\_projects"),

path("manager\_view\_requirements/<int:id>", management\_view.manager\_view\_requirements, name="manager\_view\_requirements"),

path("manager\_view\_notifications/", management\_view.manager\_view\_notifications, name="manager\_view\_notifications"),

#----------------------admin-----------------

path('admin/', admin.site.urls),

path("admins/",admins\_view.admins,name="admins"),

path("admins\_home/",admins\_view.admins\_home,name="admins\_home"),

path("admins\_change\_pwd/",admins\_view.admins\_change\_pwd,name="admins\_change\_pwd"),

path("admins\_logout/",admins\_view.admins\_logout,name="admins\_logout"),

path("add\_employee/",admins\_view.add\_employee,name="add\_employee"),

path("view\_employee/",admins\_view.view\_employee,name="view\_employee"),

path("delete\_employee/<id>",admins\_view.delete\_employee,name="delete\_employee"),

path("edit\_employee/<int:id>",admins\_view.edit\_employee,name="edit\_employee"),

path("update\_employee/",admins\_view.update\_employee,name="update\_employee"),

path("add\_project/",admins\_view.add\_project,name="add\_project"),

path("add\_new\_project/",admins\_view.add\_new\_project,name="add\_new\_project"),

path("view\_project/",admins\_view.view\_project,name="view\_project"),

path("delete\_project/<id>",admins\_view.delete\_project,name="delete\_project"),

path("edit\_project/<id>",admins\_view.edit\_project,name="edit\_project"),

path("update\_project/",admins\_view.update\_project,name="update\_project"),

path("view\_testcases/",admins\_view.view\_testcases,name="view\_testcases"),

path("view\_bugs/",admins\_view.view\_bugs,name="view\_bugs"),

path("add\_news/",admins\_view.add\_news,name="add\_news"),

path("view\_news/",admins\_view.view\_news,name="view\_news"),

path("delete\_news/<id>",admins\_view.delete\_news,name="delete\_news"),

path("edit\_news/<id>",admins\_view.edit\_news,name="edit\_news"),

path("update\_news/",admins\_view.update\_news,name="update\_news"),

#----------------------employee------------------------

path("employee/",employee\_view.employee,name="employee"),

path("employee\_home/",employee\_view.employee\_home,name="employee\_home"),

path("employee\_change\_pwd/",employee\_view.employee\_change\_pwd,name="employee\_change\_pwd"),

path("employee\_logout/",employee\_view.employee\_logout,name="employee\_logout"),

path("employee\_edit\_profile/",employee\_view.employee\_edit\_profile,name="employee\_edit\_profile"),

path("employee\_update\_profile/",employee\_view.employee\_update\_profile,name="employee\_update\_profile"),

path("add\_testcase/",employee\_view.add\_testcase,name="add\_testcase"),

path("view\_testcase/",employee\_view.view\_testcase,name="view\_testcase"),

path("delete\_testcase/<id>",employee\_view.delete\_testcase,name="delete\_testcase"),

path("edit\_testcase/<id>",employee\_view.edit\_testcase,name="edit\_testcase"),

path("update\_testcase/",employee\_view.update\_testcase,name="update\_testcase"),

path("add\_bug/",employee\_view.add\_bug,name="add\_bug"),

path("view\_bug/",employee\_view.view\_bug,name="view\_bug"),

path("delete\_bug/<id>",employee\_view.delete\_bug,name="delete\_bug"),

path("edit\_bug/<id>",employee\_view.edit\_bug,name="edit\_bug"),

path("update\_bug/",employee\_view.update\_bug,name="update\_bug"),

path("developer\_view\_bugs/",employee\_view.developer\_view\_bugs,name="developer\_view\_bugs"),

path("developer\_view\_testcases/",employee\_view.developer\_view\_testcases,name="developer\_view\_testcases"),

path("developer\_update\_status/<id>",employee\_view.developer\_update\_status,name="developer\_update\_status"),

path("developer\_status\_insert/",employee\_view.developer\_status\_insert,name="developer\_status\_insert"),

path("tester\_update\_status/<id>",employee\_view.tester\_update\_status,name="tester\_update\_status"),

path("tester\_status\_insert/",employee\_view.tester\_status\_insert,name="tester\_status\_insert"),

path("analyst\_view\_bugs/",employee\_view.analyst\_view\_bugs,name="analyst\_view\_bugs"),

path("analyst\_view\_testcases/",employee\_view.analyst\_view\_testcases,name="analyst\_view\_testcases"),

path("developer/", employee\_view.developer, name="developer"),

path("developer\_home/", employee\_view.developer\_home, name="developer\_home"),

path("developer\_change\_pwd/", employee\_view.developer\_change\_pwd, name="developer\_change\_pwd"),

path("developer\_logout/", employee\_view.developer\_logout, name="developer\_logout"),

path("developer\_edit\_profile/", employee\_view.developer\_edit\_profile, name="developer\_edit\_profile"),

path("developer\_update\_profile/", employee\_view.developer\_update\_profile, name="developer\_update\_profile"),

path("analyst\_login/", employee\_view.analyst\_login, name="analyst\_login"),

path("analyst\_home/", employee\_view.analyst\_home, name="analyst\_home"),

path("analyst\_change\_pwd/", employee\_view.analyst\_change\_pwd, name="analyst\_change\_pwd"),

path("analyst\_logout/", employee\_view.analyst\_logout, name="analyst\_logout"),

path("analyst\_edit\_profile/", employee\_view.analyst\_edit\_profile, name="analyst\_edit\_profile"),

path("analyst\_update\_profile/", employee\_view.analyst\_update\_profile, name="analyst\_update\_profile"),

path("developer\_view\_projects/", employee\_view.developer\_view\_projects, name="developer\_view\_projects"),

path("employee\_view\_projects/", employee\_view.employee\_view\_projects, name="employee\_view\_projects"),

path("analyst\_view\_projects/", employee\_view.analyst\_view\_projects, name="analyst\_view\_projects"),

path("add\_requirements/", employee\_view.add\_requirements, name="add\_requirements"),

path("analyst\_view\_requirements/<int:id>", employee\_view.analyst\_view\_requirements, name="analyst\_view\_requirements"),

path("developer\_view\_requirements/<int:id>", employee\_view.developer\_view\_requirements,

name="developer\_view\_requirements"),

path("employee\_view\_requirements/<int:id>", employee\_view.employee\_view\_requirements,

name="employee\_view\_requirements"),

path("testcase\_bug\_piechart/", employee\_view.testcase\_bug\_piechart,

name="testcase\_bug\_piechart"),

path("developer\_view\_notifications/", employee\_view.developer\_view\_notifications,

name="developer\_view\_notifications"),

path("analyst\_view\_notifications/", employee\_view.analyst\_view\_notifications,

name="analyst\_view\_notifications"),

path("employee\_view\_notifications/", employee\_view.employee\_view\_notifications,

name="employee\_view\_notifications"),

]

if settings.DEBUG:

urlpatterns += static(settings.MEDIA\_URL,

document\_root=settings.MEDIA\_ROOT)

# developer=Home,profile,developer\_view\_bugs,developer\_view\_testcases

# test=Home,profile,testcase-add\_testcase,view\_testcase,bug-add\_bug,view\_bug

# manager=Home,profile,task-add\_task,view\_task,view\_bugs,view\_testcase

# requiremen=projectid-all,title,de,priori-high,medium,low,