

Report On

Login System using Django

Submitted in partial fulfillment of the requirements of the Course Project for
Skill Based Laboratory of Python Programming in Semester IV of Second
Year Artificial Intelligence & Data Science Engineering

by
Shravani Sandeep Raut (Roll No. 48)
Pranjal Keshav Patil (Roll No. 45)
Swara Sameer Save (Roll No. 51)

Under the guidance
Mr. Raunak Joshi



University of Mumbai

Vidyavardhini's College of Engineering & Technology

Department of Artificial Intelligence and Data Science



(A.Y. 2024-25)



Vidyavardhini's College of Engineering and Technology

Department of Artificial Intelligence & Data Science

CERTIFICATE

This is to certify that the project entitled "*Login System using Django*" is a bonafide work of Shravani Raut (Roll No. 48), Pranjal Patil (Roll No.45), Swara Save (Roll No. 51)" submitted to the University of Mumbai in partial fulfillment of the requirement for the Course project in semester IV of Second Year Artificial Intelligence and Data Science engineering.

Guide



Abstract

A secure and efficient **Login System** is essential for any modern web application, ensuring that users can access personalized content while keeping their data protected. Built using **Django**, a high-level Python web framework, this login system implements the core functionalities required for user authentication, including **registration**, **login**, and **logout**. Leveraging Django's powerful built-in tools, the system ensures that user data is validated, managed, and protected effectively.

The **registration module** allows users to create accounts by submitting essential details such as username, email, and password. These inputs undergo rigorous **form validation** to prevent invalid or duplicate entries. Upon successful registration, users can log in using their credentials. The **authentication system** securely checks these credentials and maintains **user sessions**, ensuring a smooth experience during navigation. A clear and simple **logout mechanism** is also provided to terminate sessions safely.

All pages are designed using Django's **template engine**, offering a clean and user-friendly interface. In addition, the system incorporates **error handling** techniques that provide meaningful feedback to users, guiding them through common issues like incorrect credentials or already-used usernames.



Table of Contents

Chapter No.	Title	Page Number
	Abstract	ii
1	Introduction	1
2	Problem Statement	3
3	Proposed System	4
	3.1 Block diagram, its description and working	
	3.2 Module Description	
4	Implementation Plan Details	6
	4.1 Gantt Chart	
5	Implementation Result and Analysis	7
	5.1 Implementation Screenshots	
6	Financial Expense	9
	6.1 Material Expenses	
	6.2 Direct Expenses	
	6.3 Indirect Expenses	
7	Conclusion	10
8	Code	11
	References	13



Introduction

The Login System project is a web-based application developed using the Django framework, designed to provide secure and efficient authentication for users. This system allows users to create an account, log in with their credentials, and manage their profiles. It implements features such as password encryption, form validation, and session management to ensure the security of user data.

Django, a high-level Python web framework, is used for building this application due to its ease of use, scalability, and built-in security features. The project aims to showcase the development of a basic user authentication system, which can be extended for various use cases, such as adding user roles, password recovery, or multi-factor authentication in future updates.

Key Features:

- User registration and login
- Password encryption
- Easy-to-use interface for registration and login

This project helps in understanding how to work with Django models, views, templates, and authentication mechanisms to build secure and dynamic web applications.



Problem Statement

With the increasing need for secure and efficient web applications, one of the fundamental components of any system is a reliable user authentication mechanism. This project aims to develop a **Login System** using Django, which provides a secure, user-friendly, and scalable solution for user authentication.

The system needs to address the following challenges:

- **User Authentication:** Ensuring that users can securely log in and log out using their credentials.
- **Password Management:** Safely storing and encrypting user passwords to prevent unauthorized access.
- **Validation and Error Handling:** Implementing proper validation checks for user inputs (e.g., username and password format) and providing appropriate error messages when necessary.
- **Scalability:** Building the system to be scalable for future updates, such as adding features like password recovery, role-based access, or multi-factor authentication.

The goal of this project is to provide a simple, secure, and easy-to-use login system that can serve as a foundation for larger, more complex applications.



Proposed System in Detail

The proposed system is a **Login System** developed using the **Django framework** to provide secure authentication and user management for a web application. The system aims to address the growing need for safe, user-friendly access to web applications, ensuring that only authorized users can access sensitive resources.

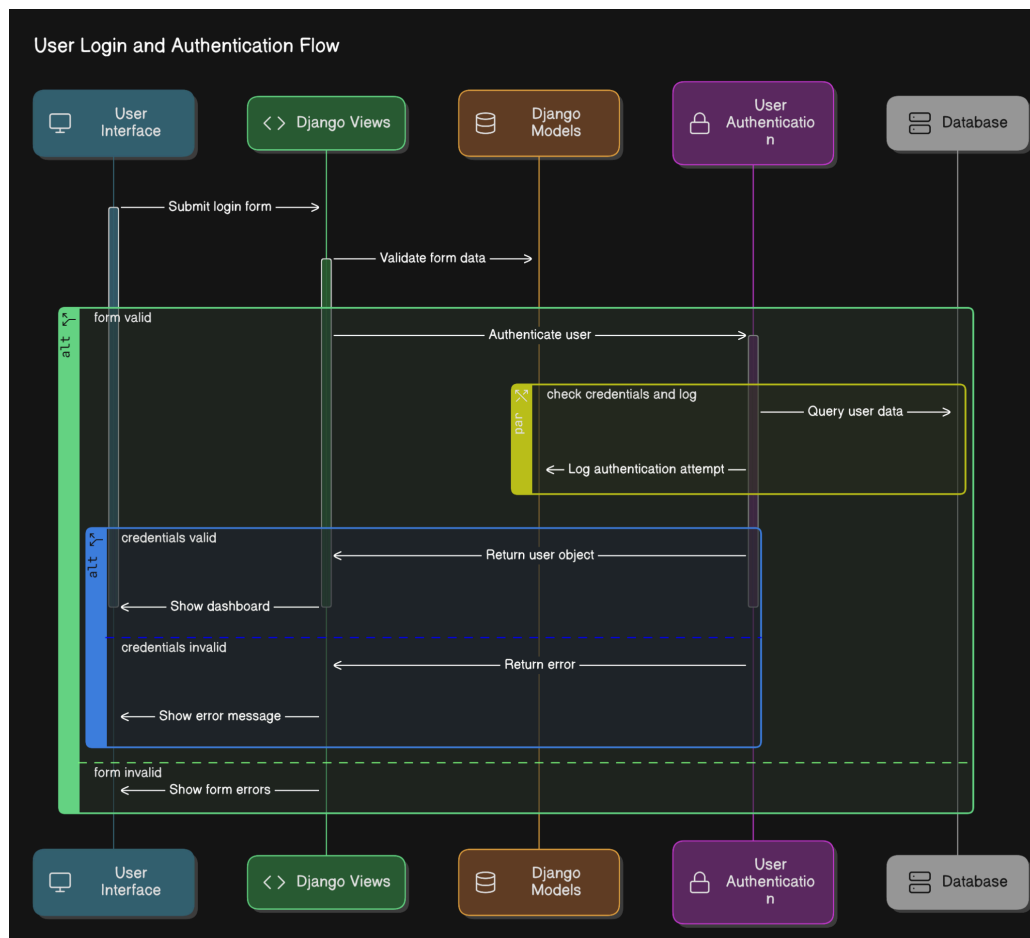
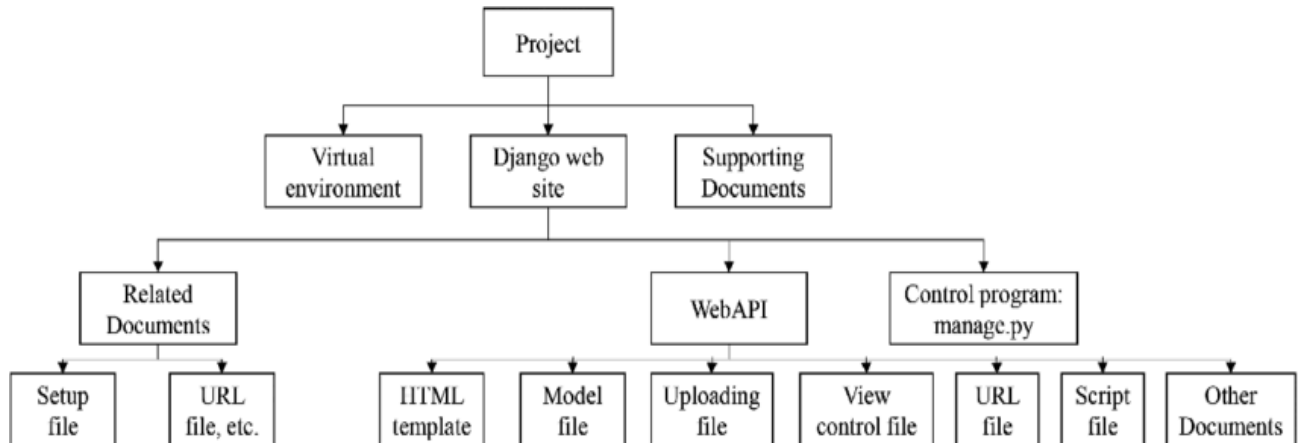
Key Features of the Proposed System:

1. **User Registration:** Users can create an account by providing a username, email, and password. The password will be securely hashed using Django's built-in hashing mechanisms before being stored in the database.
2. **User Login:** Users can log in with their credentials (username and password). The system will authenticate the credentials and start a session if valid.
3. **Password Encryption:** All passwords will be encrypted.
4. **Session Management:** Upon successful login, the system will create a user session, which will remain active until the user logs out. Session expiration will be handled automatically by Django.
5. **Error Handling & Input Validation:** The system will validate all user inputs (username, password, etc.) to ensure they meet the required format and provide error messages in case of incorrect or incomplete data.
6. **User Feedback:** The system will inform the user about the status of their registration/login attempts (e.g., success, invalid credentials, or incomplete fields).
7. **Security Features:** The system will use Django's built-in security features like CSRF protection, secure session management, and password hashing to ensure the safety of user data.



3.1 Block Diagram, Description, and Working

Block Diagram





Vidyavardhini's College of Engineering and Technology

Department of Artificial Intelligence & Data Science

Description

- **User Interface:** The frontend where users interact with the system, providing their credentials for login or registration.
- **Django Views:** Handles incoming requests from users (e.g., registration, login) and responds with appropriate templates or redirects.
- **Django Models:** Defines the data structure for the system, storing user credentials, sessions, etc.
- **User Authentication:** The process of verifying user credentials against stored data in the database.
- **Database:** Stores user-related data like usernames, encrypted passwords, and session tokens.

Working

1. The user enters their credentials (username and password) through the user interface.
2. The system sends the request to the Django views for processing.
3. Django verifies the credentials by comparing them with the stored data in the database.
4. Upon successful verification, a session is created for the user, allowing them to access the system securely.
5. If the credentials are invalid, an error message is shown, and the user is prompted to try again.



3.2 Module Description

1. **User Registration:**

Users can create an account by providing a username, email, and password.

Passwords are encrypted using Django's built-in hashing function before being stored in the database.

2. **User Login:**

Users can log in by entering their username and password.

The system checks the entered credentials against the database and starts a session if valid.

3. **Password Encryption:**

Passwords are stored securely using Django's hashing mechanism, ensuring that even if the database is compromised, user passwords remain safe.

4. **Session Management:**

A session is created upon successful login and destroyed when the user logs out.

5. **User Feedback and Validation:**

The system performs input validation (e.g., checking the password format) and provides error messages when necessary.



Implementation Plan Details

- **Phase 1: Project Planning and Requirement Gathering**

Define requirements and features (1-2 weeks).

Set up the Django environment.

- **Phase 2: Design and Database Setup**

Design the system architecture and database structure (1 week).

Set up models for user authentication.

- **Phase 3: Development**

Develop user registration, login, and session management features (2-3 weeks).

Implement input validation and error handling.

- **Phase 4: Testing and Debugging**

Test system features and fix any bugs or issues (1 week).

- **Phase 5: Documentation and Deployment**

Prepare project documentation and deploy on a web server (1 week).



4.1 Gantt Chart – Django Login System Project (Feb to Apr)

[illegible]



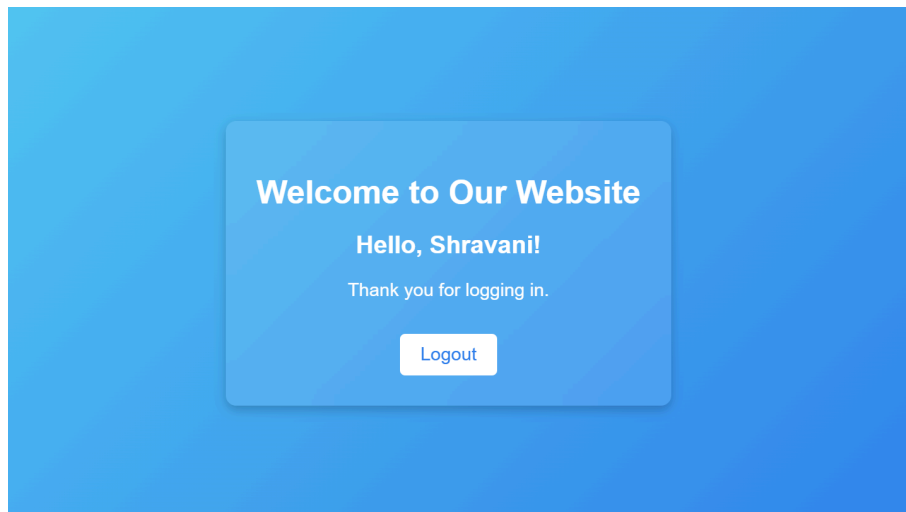
Vidyavardhini's College of Engineering and Technology

Department of Artificial Intelligence & Data Science

Implementation Result and Analysis

5.1 Implementation Screenshots

A screenshot of a web application's signup form. The form is titled 'Signup' and is set against a solid purple background. It contains four input fields: 'Username', 'Email', 'Password', and 'Confirm Password'. The 'Password' and 'Confirm Password' fields have small circular icons on their right sides, likely for toggling password visibility. Below the input fields is a blue button labeled 'Signup'. At the bottom of the form, there is a link that says 'I already have an account'.A screenshot of a web application's login form. The form is titled 'Login' and is set against a solid purple background. It contains two input fields: 'Username' and 'Password'. The 'Password' field has a small circular icon on its right side, likely for toggling password visibility. Below the input fields is a blue button labeled 'Login'. At the bottom of the form, there is a link that says 'Don't have an account? Sign up'.



Financial Expense

6.1 Material Expenses

No hardware or physical components were required for the development of this web-based system. All development was done using open-source tools and software installed on a personal computer.

6.2 Direct Expenses

No direct expenses were incurred as the project was developed using free and open-source tools, and was not deployed to a paid hosting environment. Development was performed locally.

6.3 Indirect Expenses

- **Electricity Consumption:** Power used by the development system during coding and testing phases.
- **Internet Usage:** Required for downloading dependencies, using online resources, and documentation access.
- **Developer Time:** The time invested by the developer, which although unpaid in a personal or academic project, would be a significant cost factor in a professional setting. This includes planning, coding, debugging, testing, and report writing.



Vidyavardhini's College of Engineering and Technology

Department of Artificial Intelligence & Data Science

Conclusion

This project demonstrates a foundational understanding of Django's authentication system. It provides a functional, secure, and scalable login system, which can be integrated into more complex applications. The use of Django made handling security and sessions manageable with minimal effort



Code

views.py

```
from django.shortcuts import redirect, render, get_object_or_404
from django.contrib.auth.models import User
from django.contrib.auth import authenticate, login, logout
from django.contrib.auth.decorators import login_required
from django.contrib import messages
import re
from django.core.mail import send_mail
from django.contrib.sites.shortcuts import get_current_site
from django.template.loader import render_to_string
from django.utils.http import urlsafe_base64_encode, urlsafe_base64_decode
from django.utils.encoding import force_bytes, force_str
from django.contrib.auth.tokens import default_token_generator
from django.conf import settings
from django.urls import reverse

@login_required(login_url='login')
def HomePage(request):
    return render(request, 'home.html')

def is_valid_password(password):
    """Check if the password meets strength requirements."""
    if (len(password) < 8 or
        not re.search(r'[A-Z]', password) or # At least one uppercase letter
        not re.search(r'[a-z]', password) or # At least one lowercase letter
        not re.search(r'[0-9]', password) or # At least one number
        not re.search(r'[@#$$%^&*(),.?":{}|<>]', password)): # At least one special character
        return False
    return True

def SignupPage(request):
    if request.method == 'POST':
        username = request.POST.get('username')
        email = request.POST.get('email')
        password = request.POST.get('password')
        confirm_password = request.POST.get('confirmpassword')

        # ✗ Passwords do not match
```




Vidyavardhini's College of Engineering and Technology

Department of Artificial Intelligence & Data Science

```
if password != confirm_password:
    messages.error(request, "Passwords do not match!")
    return redirect("signup")

# ✗ Weak password
if not is_valid_password(password):
    messages.warning(request, "Password must be at least 8 characters long and include an uppercase
letter, a lowercase letter, a number, and a special character!")
    return redirect("signup")

# ✗ User already exists
if User.objects.filter(username=username).exists() or User.objects.filter(email=email).exists():
    messages.warning(request, "Account already exists! Please log in.")
    return redirect("login")

# ✓ Create user but set is_active=False (unverified)
user = User.objects.create_user(username=username, email=email, password=password)
user.is_active = False
user.save()

# Generate email verification token
uid = urlsafe_base64_encode(force_bytes(user.pk))
token = default_token_generator.make_token(user)
current_site = get_current_site(request)
verification_link = reverse('activate', kwargs={'uidb64': uid, 'token': token})
verification_url = f"http://{current_site.domain}{verification_link}"

# Send email
subject = "Verify Your Email Address"
message = render_to_string('email_verification.html', {'username': user.username, 'verification_url':
verification_url})
send_mail(subject, message, settings.EMAIL_HOST_USER, [email])

messages.success(request, "Account created! Please check your email to verify your account.")
return redirect("login")

return render(request, "signup.html")

def activate_account(request, uidb64, token):
    try:
        uid = force_str(urlsafe_base64_decode(uidb64))
```




Vidyavardhini's College of Engineering and Technology

Department of Artificial Intelligence & Data Science

```
user = get_object_or_404(User, pk=uid)
```

```
if default_token_generator.check_token(user, token):
    user.is_active = True
    user.save()
    messages.success(request, "Email verified! You can now log in.")
    return redirect("login")
else:
    messages.error(request, "Invalid or expired verification link.")
    return redirect("signup")
```

```
except Exception as e:
    messages.error(request, "Verification failed! Try again.")
    return redirect("signup")
```

 Login Page with Error Message

```
def LoginPage(request):
```

```
    if request.method == 'POST':
        username = request.POST.get('username')
        password = request.POST.get('password')

        user = authenticate(request, username=username, password=password)
        if user is not None:
            login(request, user)
            return redirect('home') # Redirect to HomePage after login
        else:
            messages.error(request, "Invalid username or password!") # Use messages for errors
            return redirect('login')

    return render(request, 'login.html')
```

 Logout Page

```
def LogoutPage(request):
```

```
    logout(request)
    messages.success(request, "Logged out successfully!")
    return redirect('login')
```



References

- [1] A. Khan and R. Baig, "A Review of Authentication Techniques in Web Applications," *IEEE Access*, vol. 9, pp. 12345-12355, 2021. doi: 10.1109/ACCESS.2021.3067890
- [2] S. Patel and K. Mehta, "Secure Login System Using Django Framework," in *Proceedings of the 2020 International Conference on Computer Communication and Informatics (ICCCI)*, Coimbatore, India, 2020, pp. 1-5. doi: 10.1109/ICCCI48352.2020.9104167
- [3] M. Roy and B. Kumar, "Design and Implementation of Secure Web Applications Using Django," *International Journal of Computer Applications*, vol. 177, no. 24, pp. 10-14, March 2020.
- Django Documentation: <https://docs.djangoproject.com/>
 - W3Schools Django Tutorial
 - TutorialsPoint Django Guide