

LOGIN SYSTEM

A Project Report submitted in the partial fulfillment for the award of
Bachelor of Technology in computer Science & Engineering

Submitted By

Rima Das

Sahali Das

Sanjita Ray

Shramana Show

Soham Saha

In the department of

Computer Science & Engineering

at

Ardent Computech Pvt. Ltd.



Sur**Tech**



***College Name: Dr. Sudhir Chandra Sur Institute
of Technology and Sports Complex***



Ardent® Computech Pvt. Ltd.

Module – 132, Ground Floor, SDF Building,
Sector V, Salt Lake, Kolkata – 700091.

+91 33 40073507

✉ training@ardentcollaborations.com

CERTIFICATE FROM SUPERVISOR

This is to certify that, Student have successfully completed the project titled Login System. Under my supervision during the period from 11-03-24 to 16-03-24 which is in partial fulfilment of requirements for the award of the **B.Tech** degree and submitted to the Department of Computer Science and Engineering.

Signature of the Supervisor

Date:

Name of the Project Supervisor: **Saikat Dutta**



ACKNOWLEDGEMENT

The achievement that is associated with the successful completion of any task would be incomplete without mentioning the names of those people whose endless cooperation made it possible. Their constant guidance and encouragement made all our efforts successful.

We take this opportunity to express our deep gratitude towards our project mentor, SAIKAT DUTTA for giving such valuable suggestions, guidance and encouragement during the development of this project work.

Last but not the least we are grateful to all the faculty members of Ardent Computech Pvt. Ltd. for their support.

TABLE OF CONTENTS

- Abstract
- Introduction
- Objective
- Software requirements specification (SRS)
- Approach
- Snap shots of the project
- Snap shots of the code
- Conclusion
- Future Scope & Further enhancements
- Bibliography/References

ABSTRACT

At first the WELCOME page will be encountered by any user. Then they'll be getting three options such as SIGN IN ,SIGN UP & ARE YOU AN ADMINISTRATOR? there only. The SIGN IN option will redirect the user to sign in page, the SIGN UP option will redirect the user to sign up page and similarly the ARE YOU AN ADMINISTRATOR link will redirect the user to the admin page where the superuser can see the people who've signed up as well as their data.

The method which is used here in the case of SIGN UP is POST METHOD. The POST method is one of the HTTP request methods used to send data to a server to create or update a resource. By post method the credentials of the user is taken while they sign up in the page and the data of the users is stored in the database as well. In this way when the sign up of the user comes to an end it will show a message such as "You've created an account successfully". After this message is shown to the user the sign up page will redirect the user to the sign in page immediately. After completing this when the user signs in again with all the credentials it will redirect the user to a page where the user will be shown a message again such as "Hello! user's_name, you've logged in successfully". There a option will be shown which is SIGN OUT. When the user clicks on this option the user will be redirected to the WELCOME page only. Django's built-in authentication system is utilized for user management.

Authentication has been maintained throughout the entire project. Users provide their credentials (username, first name, last name, email and password) on the SIGN UP page. The credentials are then sent to the server for authentication. As for in the SIGN UP as well as SIGN IN page Username only has to be alphanumeric otherwise it'll be invalid. For email address the correct format of an email address must be given by the user otherwise it'll not be valid enough to create an account. For the SIGN IN page, On the server-side, Django's authentication logic verifies the received credentials against the stored user data in the database. This process involves comparing the provided password with the hashed password stored in the database to determine if the user is authenticated or not.

After successful authentication, authenticated users are redirected to their intended destination, such as home page. Unauthorized users are redirected back to the welcome page with an error message.

INTRODUCTION

In today's digital age, where online services are an integral part of our daily lives, having a secure login page is essential for protecting user accounts from unauthorized access. This project aims to address this need by implementing a login page using Django, a high-level Python web framework known for its security features and scalability. By incorporating HTML and CSS for front-end design, we aim to provide users with a visually appealing and intuitive interface while ensuring robust authentication mechanisms on the backend.

This report proposes the development of a modern login page system powered by Python and Django. This framework allows for a robust, scalable and user-friendly login system tailored to the specific needs of effective login page throughout the entire world.

By implementing this new login system, this project can contribute to a more efficient, user-centric, and data driven information management environment, ultimately supporting the academic success of its community.

OBJECTIVE

This project focuses on developing a secure and user-friendly login page using Django, Python, HTML, and CSS. With the increasing demand for web-based applications, providing a robust authentication system is crucial for safeguarding user data and ensuring a flawless user experience. By using Django's powerful features and combining them with HTML and CSS for front-end design, this project aims to create a reliable login page that meets modern security standards and offers an impressive interface for users.

The primary objective of this project is to develop a login page that encompasses the following key goals:

1. **Security:** The foremost objective is to ensure the security of user accounts and sensitive data. This involves implementing robust authentication mechanisms, such as password hashing and Cross-Site Request Forgery (CSRF) protection, to prevent unauthorized access and safeguard user information from potential security threats like brute-force attacks or data breaches.
2. **User Experience:** Another crucial objective is to provide users with an intuitive and seamless login experience. This includes designing a user-friendly interface using HTML and CSS that facilitates easy navigation and interaction with the login page. By prioritizing usability, we aim to enhance user

satisfaction and encourage continued engagement with the application.

3. Functionality: The login page should fulfill its core function effectively and efficiently. This involves enabling users to securely log in to their accounts by validating their credentials and handling error scenarios gracefully. Additionally, the login page should be compatible across different browsers and devices to ensure accessibility for all users.

4. Scalability: As a long-term objective, the login page should be designed with scalability in mind. This entails building a flexible and extensible system that can accommodate future enhancements and accommodate growing user demands. Whether it's integrating additional authentication methods, supporting multi-factor authentication, or scaling infrastructure to handle increased traffic, the login page should be adaptable to evolving requirements.

By achieving these objectives, we aim to deliver a login page that not only meets the immediate needs of users but also establishes a foundation for continued improvement and innovation. Through a combination of security, usability, functionality, and scalability considerations, we strive to create a login page that sets a high standard for authentication systems in web development.

SOFTWARE REQUIREMENTS SPECIFICATION (SRS)

The software requirements specification document enlists enough and necessary requirements that are required for the project development. To derive the requirements, we need to have clear and thorough understanding of the project to be developed.

1. **Django:** Django is a high-level Python web framework used for rapid development of web applications. It provides built-in features for user authentication, routing, database management, and more, making it an ideal choice for developing the login page.
2. **Python:** Python is a versatile programming language used for backend development in this project. It's used for writing the application logic, including authentication logic, data processing, and interacting with the database.
3. **HTML:** HTML (Hyper Text Markup Language) is used for structuring the content of the login page. It defines the layout, forms, and other elements that make up the user interface.
4. **CSS:** CSS (Cascading Style Sheets) is used for styling the login page to enhance its visual appeal and improve user experience. It defines the colors, fonts, layout, and other visual aspects of the page.

5. **Django Forms:** Django Forms are used to create HTML forms for capturing user input, such as login credentials. They provide built-in validation and error handling, simplifying the process of handling user input on the server-side.

6. **Django Authentication System:** Django's built-in authentication system provides features for user authentication, including user models, authentication backends, and login/logout views. It handles tasks such as user registration, password hashing, and session management, ensuring a secure authentication process.

7. **SQLite:** This is relational database management systems supported by Django for storing user data and application information. SQLite is often used for development and testing purposes.

8. **Git:** Git is a version control system used for managing the project's source code. It allows for collaboration among team members, version tracking, and rollback to previous versions if needed.

9. **Integrated Development Environment (IDE):** For IDE Visual Studio Code is used for writing, debugging, and managing the project code. They provide features like syntax highlighting, code completion, and debugging tools to streamline the development process.

APPROACH

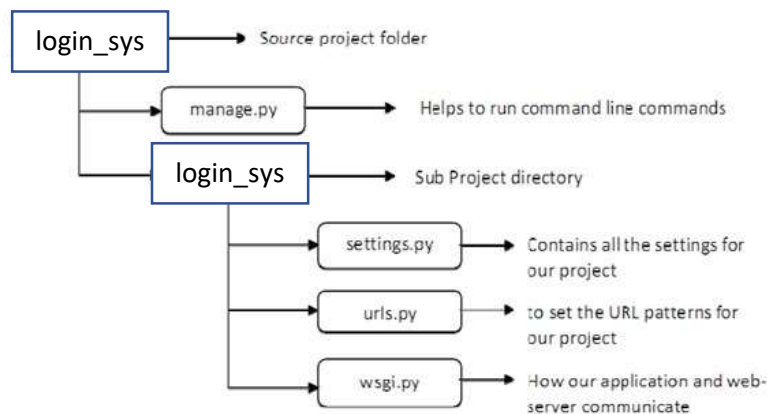
At first Django was installed and a new Django project was created.

Then, a virtual environment was set up to isolate dependencies.

Step 1:

Create a project `django-admin startproject login_sys`

This will create a `login_sys` folder in your current directory.



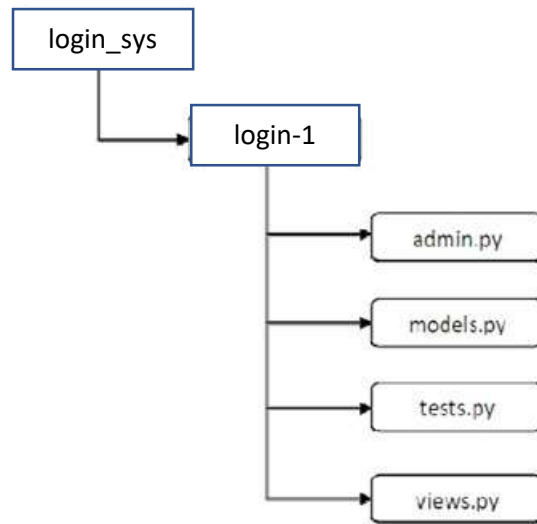
Step 2: Create a library app

Now change a directory to `login_sys` by following command:

```
cd login_sys
```

Now we will create an app by using following command:

```
python manage.py startapp login-1
```



Step 3: Now we will create a home page where we can display login, Sign Up and other button

Step 4: views.py

Step 5:urls.py

What is render():

Combines a given template with a given context dictionary and returns an HttpResponse object with that rendered text.

Django does not provide a shortcut function which returns a TemplateResponse because the constructor of TemplateResponse offers the safe level of convenience as render().

Required arguments:

request:

The request object used to generate this response.template_name:

The full name of a template to use or sequence of template names. If a sequence is given, the first template that exists will be used. See the template loading documentation for more information on how templates are found.

What is redirect():

Returns an HttpResponseRedirect arguments passed.

The arguments could be:

to the appropriate URL for the

- A model: the model's `get_absolute_url()` function will be called.
- A view name, possibly with reverse-resolve the name.

arguments: `reverse()` will be used to

- An absolute or relative URL, redirect location.

which will be used as-is for the By default issues a temporary redirect; pass `permanent=True` to issue a permanent redirect.

What are messages in Django?

The Django Web frameworks comes with a messaging system that allows us to store messages that we can check for on each page load. If there are some messages, we can display them to the user. For show them however we see these messages, we could fit. With `materialize`.

User authentication in Django: Django comes with a user authentication system. It handles user accounts, groups,

permissions and cookie-based user sessions.

Authentication support is bundled as a Django contrib module in `django.contrib.auth`. By default, the required configuration is already included in the `settings.py` generated by `django-admin startproject`, these consist of two items listed in your `INSTALLED_APPS` setting:

1. `'django.contrib.auth'` contains the its default models. core of the authentication framework, and
2. `'django.contrib.contenttypes'` is the Django content type system, which allow permissions to be associated with models you create. and these items in your `MIDDLEWARE` setting.

1. `SessionMiddleware` manages sessions across requests.
2. `AuthenticationMiddleware` associates users with requests using sessions.

The Django admin site:

One of the most powerful parts of Django is the automatic admin

interface. It reads metadata from your models to provide a quick, model centric interface where trusted users can manage content on your site. The admin's recommended use is limited to an organization's internal management tool.

What is path: `path` is a new function defined in `django 2.0`

It returns an element for inclusion in url patterns in `urls.py`

Request and response objects:

Django uses request and response objects to pass state through the system.

When a page is requested, Django creates an `HttpRequest` object that contains metadata about the request. Then Django loads the appropriate view, passing the `HttpRequest` as the first argument to the view function. Each view is responsible for returning an `HttpResponse` object.

First a folder named `template` is created then `index.html`, `signin.html` and `signup.html` these three files are inserted.

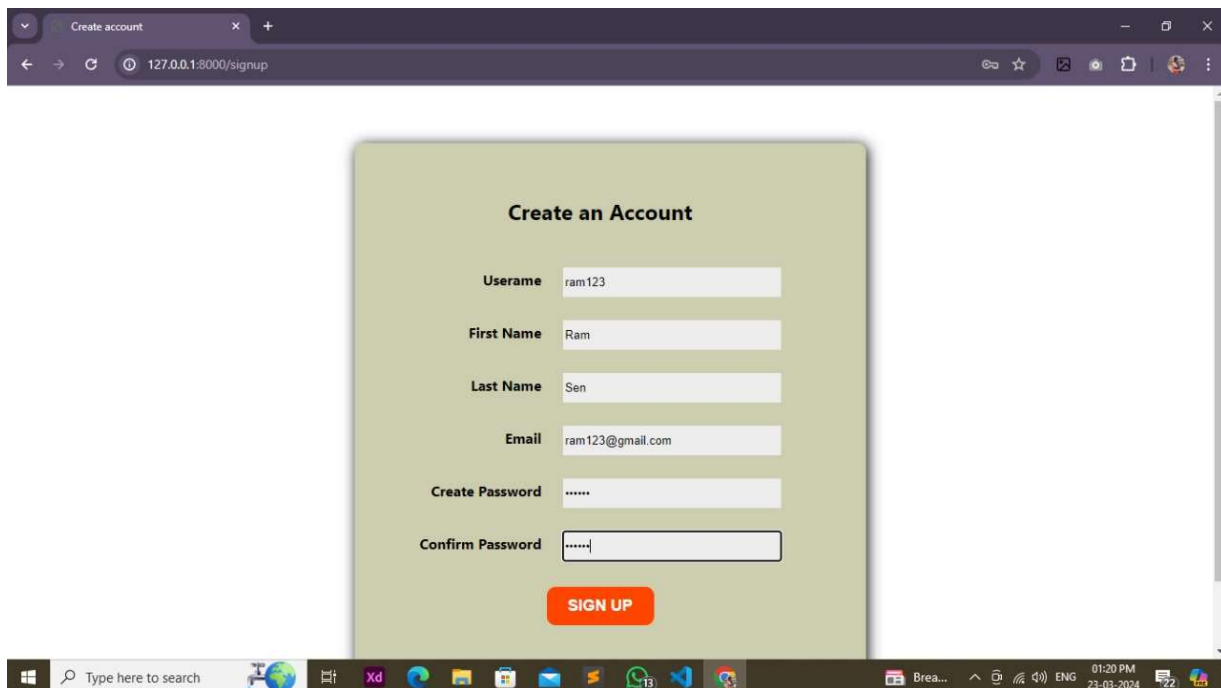
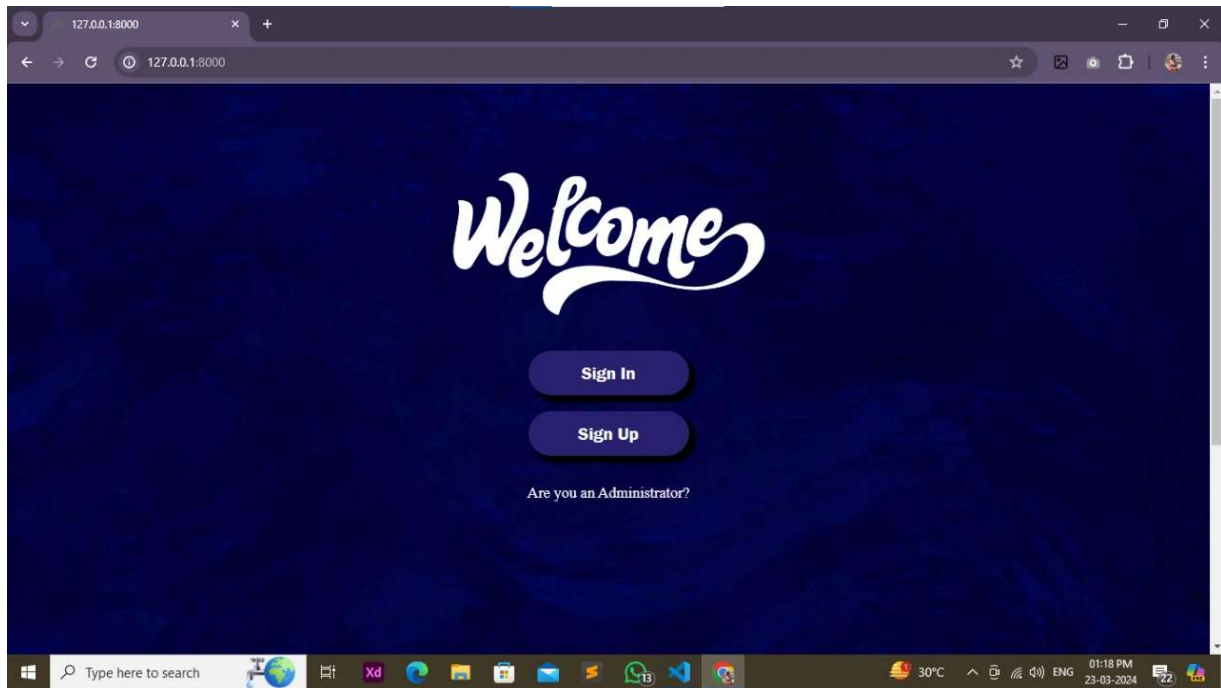
Then to add style in every page of HTML CSS code is written.

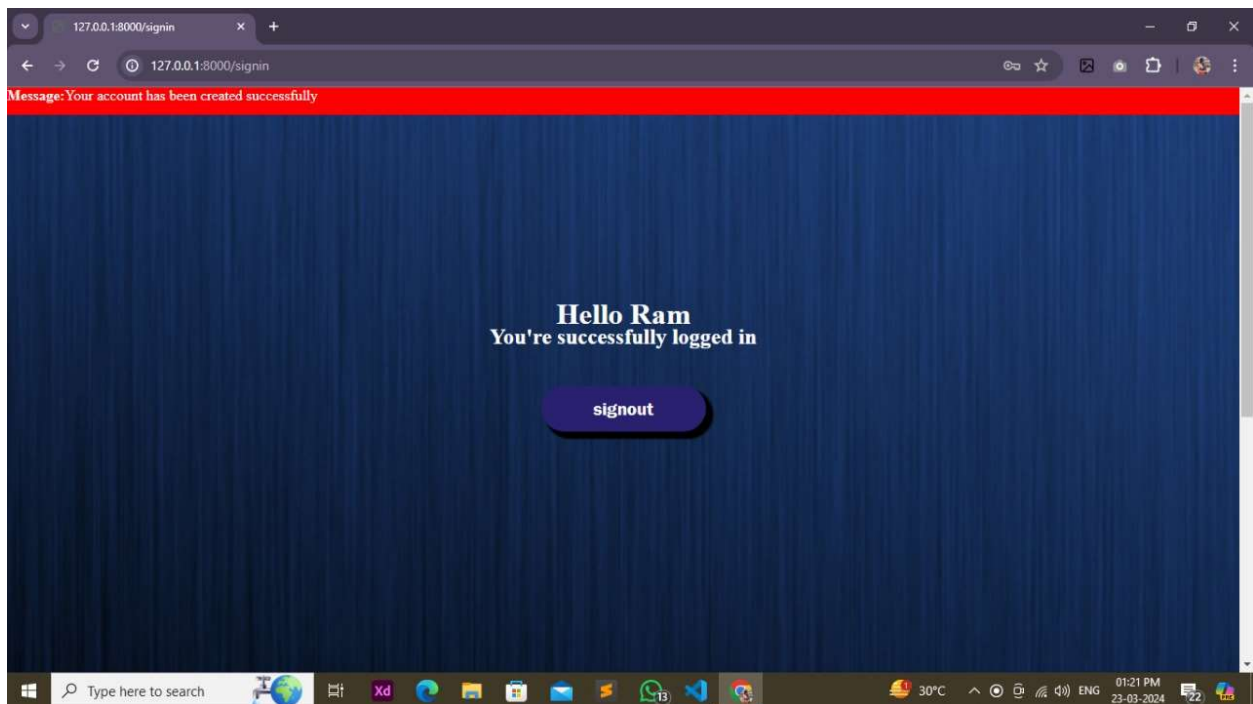
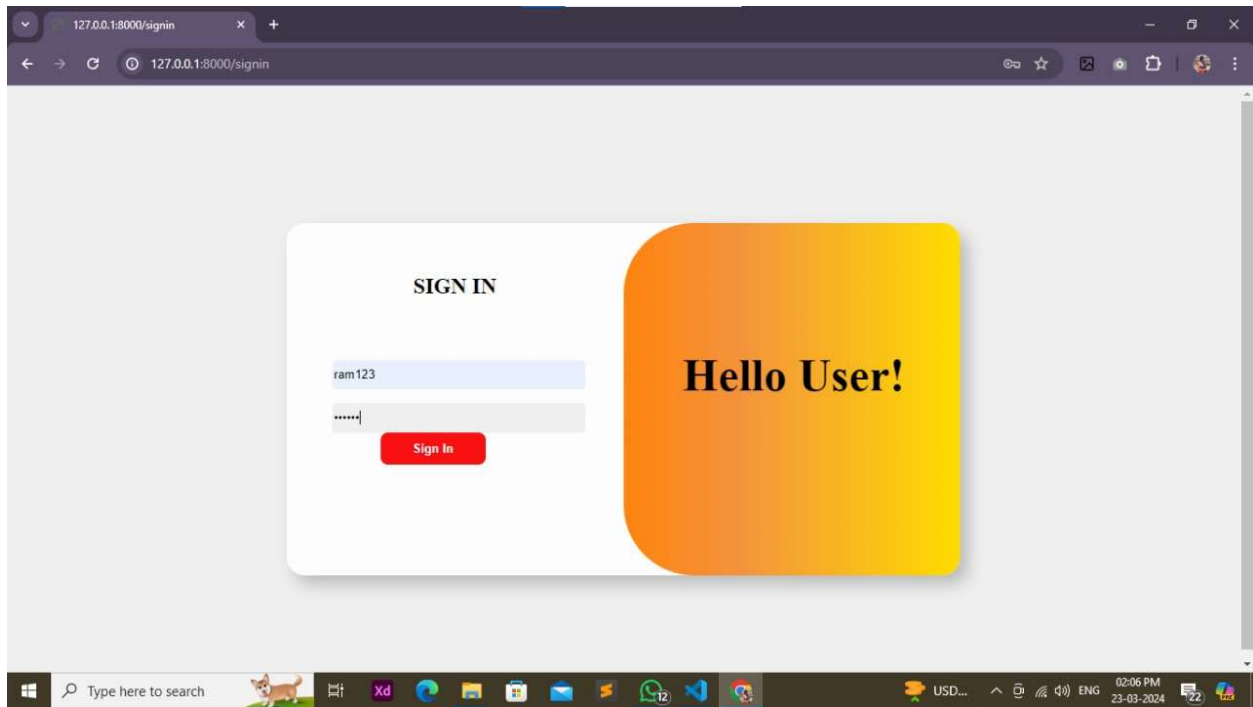
To make it responsive mediaquery is used (to make it perfectly visible in every kind screen).

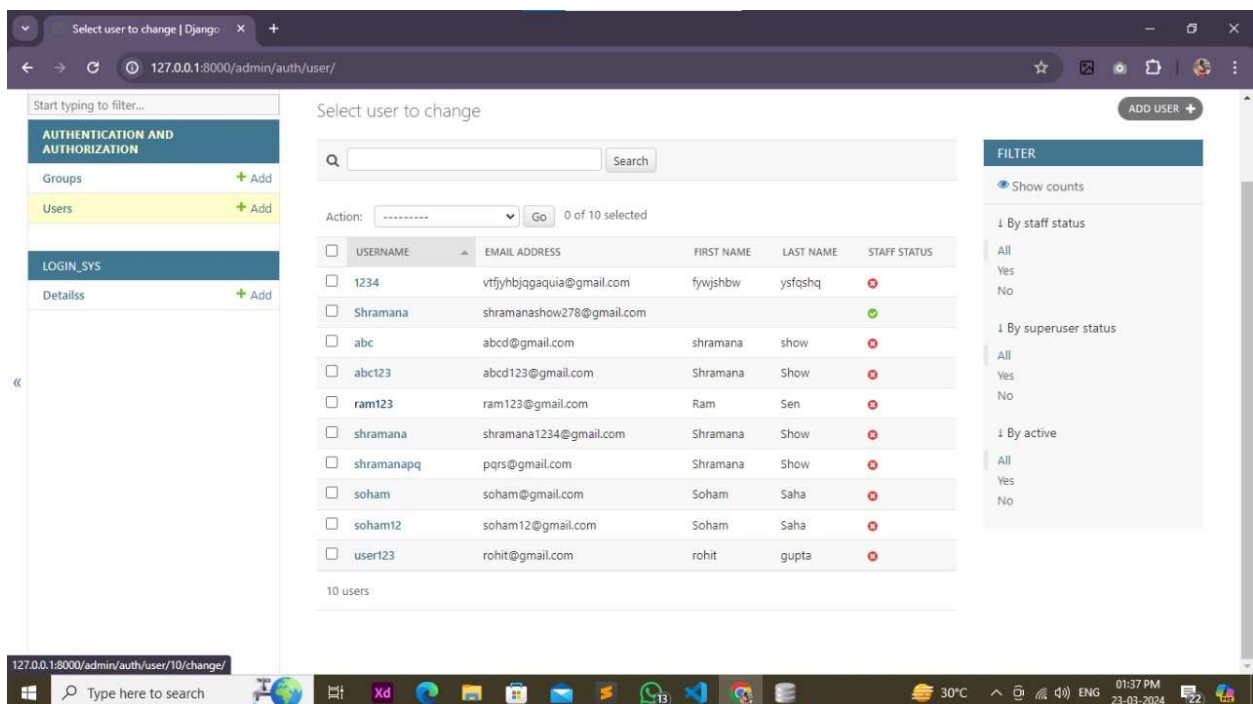
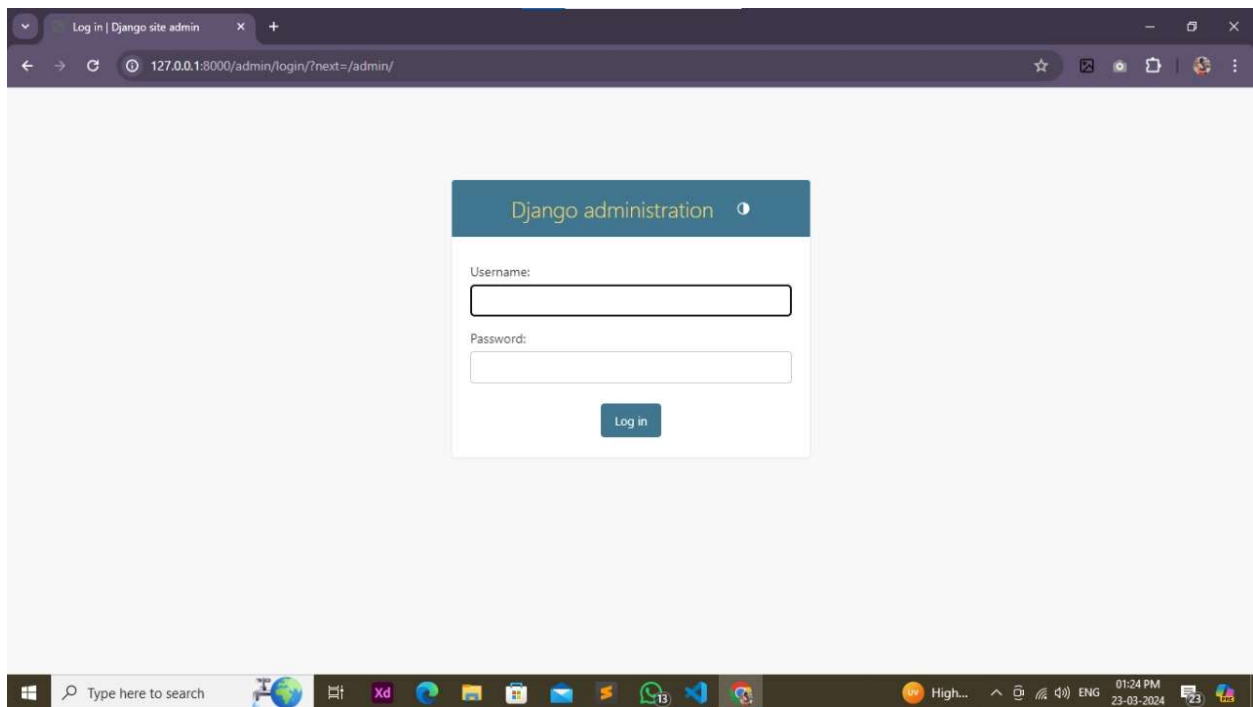
Our approach to developing the login page using Django with Python, HTML, and CSS involves a systematic and iterative process aimed at achieving security, usability, and functionality.

Requirement analysis, technology selection, design planning, implementation, testing and debugging, measuring security, and finally documentation and deployment, by following this approach, we aim to deliver a login page that meets the project objectives, adheres to best practices in web development, and provides a secure and user-friendly authentication experience for users.

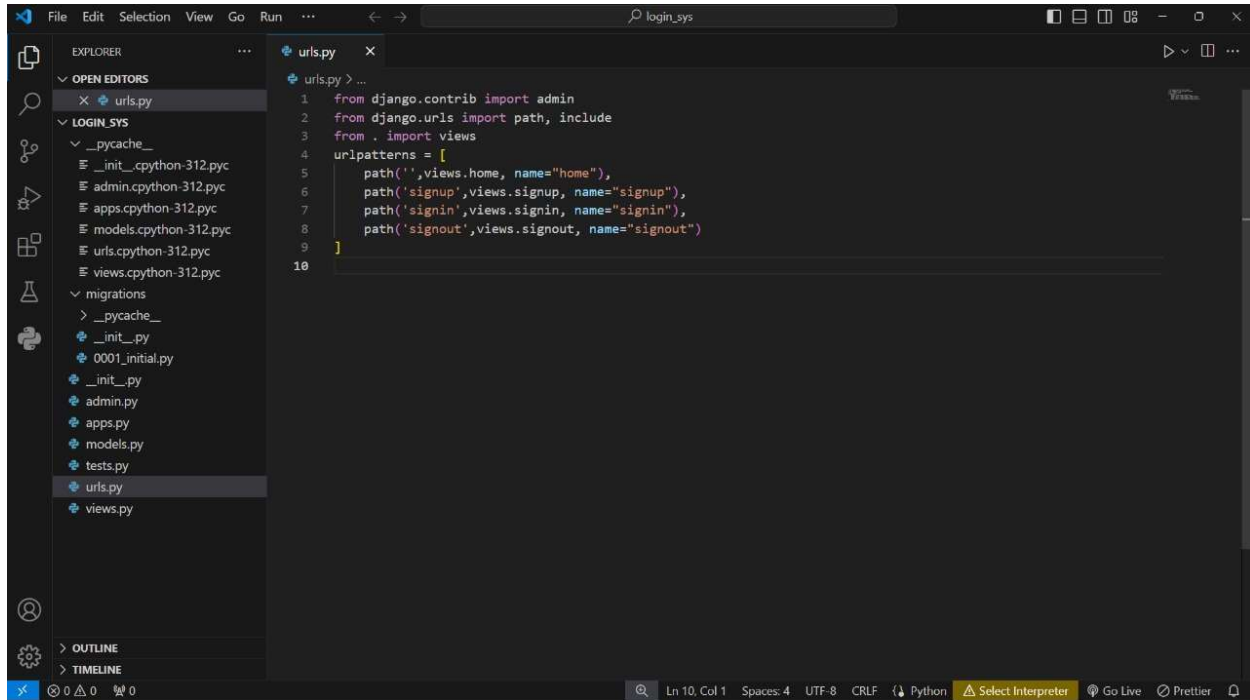
SNAPSHOTS OF PROJECT/OUTPUT SCREEN





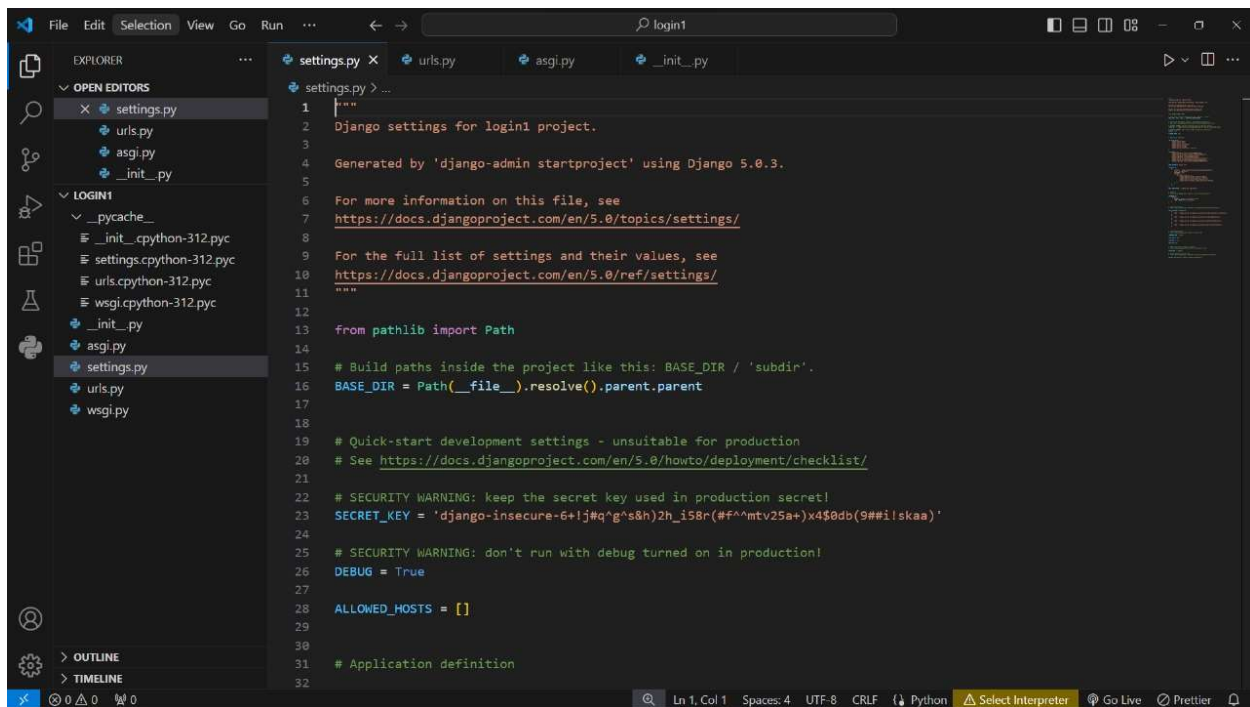


SNAPSHOTS OF CODE



This screenshot shows the Visual Studio Code editor with a Django project named 'login_sys'. The Explorer sidebar on the left shows the project structure, including files like urls.py and views.py. The main editor window displays the contents of urls.py, which defines the URL patterns for the project. The code includes imports for Django's admin, urls, and views modules, and a list of URL patterns for home, signup, signin, and signout views.

```
1 from django.contrib import admin
2 from django.urls import path, include
3 from . import views
4 urlpatterns = [
5     path('', views.home, name="home"),
6     path('signup', views.signup, name="signup"),
7     path('signin', views.signin, name="signin"),
8     path('signout', views.signout, name="signout")
9 ]
10
```

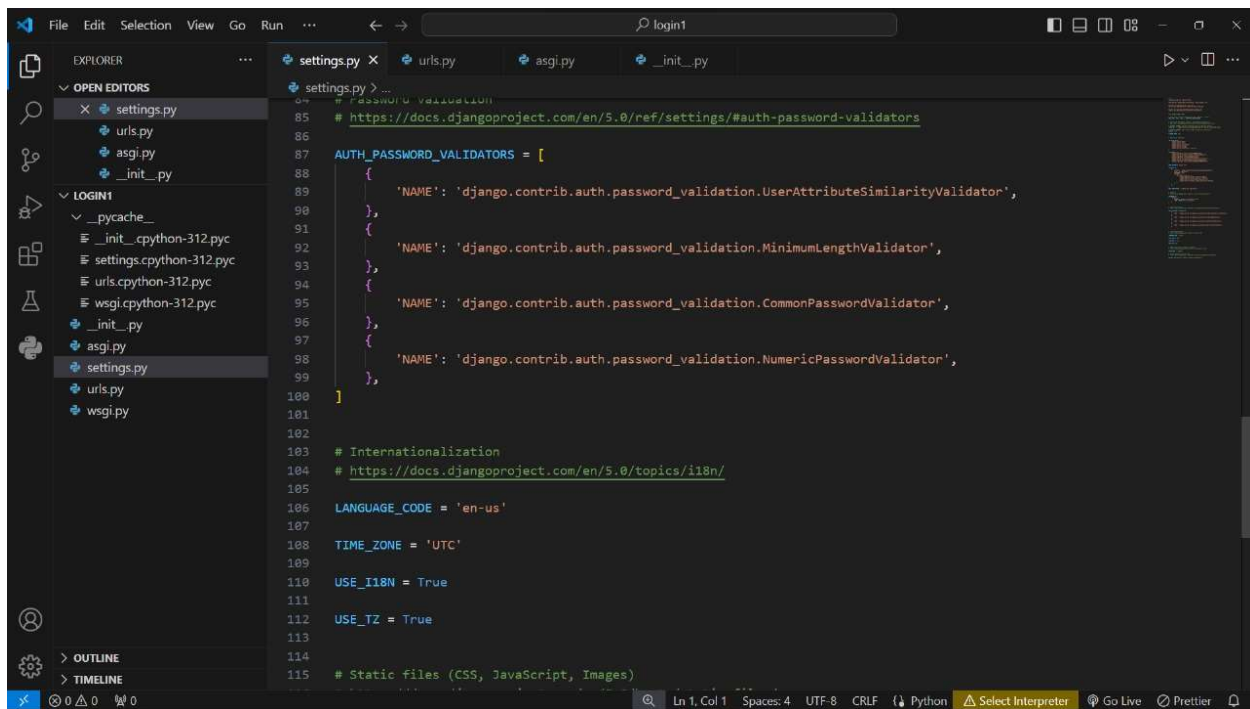


This screenshot shows the Visual Studio Code editor with a Django project named 'login1'. The Explorer sidebar on the left shows the project structure, including files like settings.py, urls.py, and wsgi.py. The main editor window displays the contents of settings.py, which defines the settings for the project. The code includes imports for Django's settings module and the Path class from pathlib. It also includes comments and code for setting the base directory, secret key, and other project-specific settings.

```
1 """
2 Django settings for login1 project.
3
4 Generated by 'django-admin startproject' using Django 5.0.3.
5
6 For more information on this file, see
7 https://docs.djangoproject.com/en/5.0/topics/settings/
8
9 For the full list of settings and their values, see
10 https://docs.djangoproject.com/en/5.0/ref/settings/
11 """
12
13 from pathlib import Path
14
15 # Build paths inside the project like this: BASE_DIR / 'subdir'.
16 BASE_DIR = Path(__file__).resolve().parent.parent
17
18 # Quick-start development settings - unsuitable for production
19 # See https://docs.djangoproject.com/en/5.0/howto/deployment/checklist/
20
21 # SECURITY WARNING: keep the secret key used in production secret!
22 SECRET_KEY = 'django-insecure-6+ljq^q^s&h)2h_i58r(#f^mtv25a+x4$0db9##!lskaa)'
23
24 # SECURITY WARNING: don't run with debug turned on in production!
25 DEBUG = True
26
27 ALLOWED_HOSTS = []
28
29
30
31 # Application definition
32
```

```
32
33 INSTALLED_APPS = [
34     'django.contrib.admin',
35     'django.contrib.auth',
36     'django.contrib.contenttypes',
37     'django.contrib.sessions',
38     'django.contrib.messages',
39     'django.contrib.staticfiles', 'login_sys'
40 ]
41
42 MIDDLEWARE = [
43     'django.middleware.security.SecurityMiddleware',
44     'django.contrib.sessions.middleware.SessionMiddleware',
45     'django.middleware.common.CommonMiddleware',
46     'django.middleware.csrf.CsrfViewMiddleware',
47     'django.contrib.auth.middleware.AuthenticationMiddleware',
48     'django.contrib.messages.middleware.MessageMiddleware',
49     'django.middleware.clickjacking.XFrameOptionsMiddleware',
50 ]
51
52 ROOT_URLCONF = 'login1.urls'
53
54 TEMPLATES = [
55     {
56         'BACKEND': 'django.template.backends.django.DjangoTemplates',
57         'DIRS': ["templates"],
58         'APP_DIRS': True,
59         'OPTIONS': {
60             'context_processors': [
61                 'django.template.context_processors.debug',
62                 'django.template.context_processors.request',
63                 'django.contrib.auth.context_processors.auth',
```

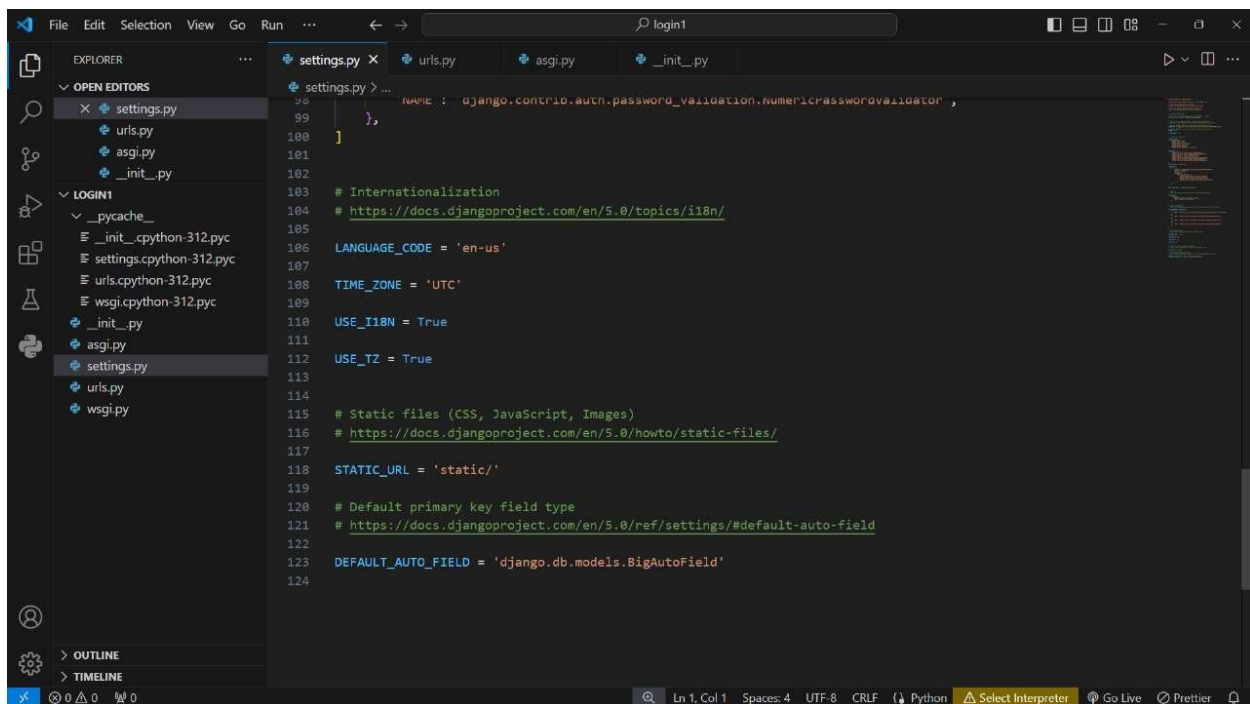
```
54 TEMPLATES = [
55     {
56         'BACKEND': 'django.template.backends.django.DjangoTemplates',
57         'DIRS': ["templates"],
58         'APP_DIRS': True,
59         'OPTIONS': {
60             'context_processors': [
61                 'django.template.context_processors.debug',
62                 'django.template.context_processors.request',
63                 'django.contrib.auth.context_processors.auth',
64                 'django.contrib.messages.context_processors.messages',
65             ],
66         },
67     ],
68 ]
69
70 WSGI_APPLICATION = 'login1.wsgi.application'
71
72
73 # Database
74 # https://docs.djangoproject.com/en/5.0/ref/settings/#databases
75
76 DATABASES = {
77     'default': {
78         'ENGINE': 'django.db.backends.sqlite3',
79         'NAME': BASE_DIR / 'db.sqlite3',
80     }
81 }
82
83
84 # Password validation
85 # https://docs.djangoproject.com/en/5.0/ref/settings/#auth-password-validators
```



This screenshot shows the Visual Studio Code editor with the Django settings.py file open. The Explorer sidebar on the left shows the project structure with files like settings.py, urls.py, asgi.py, and _init_.py. The main editor area displays the following code:

```
85 # Password validation
86 # https://docs.djangoproject.com/en/5.0/ref/settings/#auth-password-validators
87
88 AUTH_PASSWORD_VALIDATORS = [
89     {
90         'NAME': 'django.contrib.auth.password_validation.UserAttributeSimilarityValidator',
91     },
92     {
93         'NAME': 'django.contrib.auth.password_validation.MinimumLengthValidator',
94     },
95     {
96         'NAME': 'django.contrib.auth.password_validation.CommonPasswordValidator',
97     },
98     {
99         'NAME': 'django.contrib.auth.password_validation.NumericPasswordValidator',
100     },
101 ]
102
103 # Internationalization
104 # https://docs.djangoproject.com/en/5.0/topics/i18n/
105
106 LANGUAGE_CODE = 'en-us'
107
108 TIME_ZONE = 'UTC'
109
110 USE_I18N = True
111
112 USE_TZ = True
113
114
115 # Static files (CSS, JavaScript, Images)
```

The status bar at the bottom indicates the current position is Line 1, Column 1, with 4 spaces, UTF-8 encoding, CRLF line endings, and the Python interpreter selected.



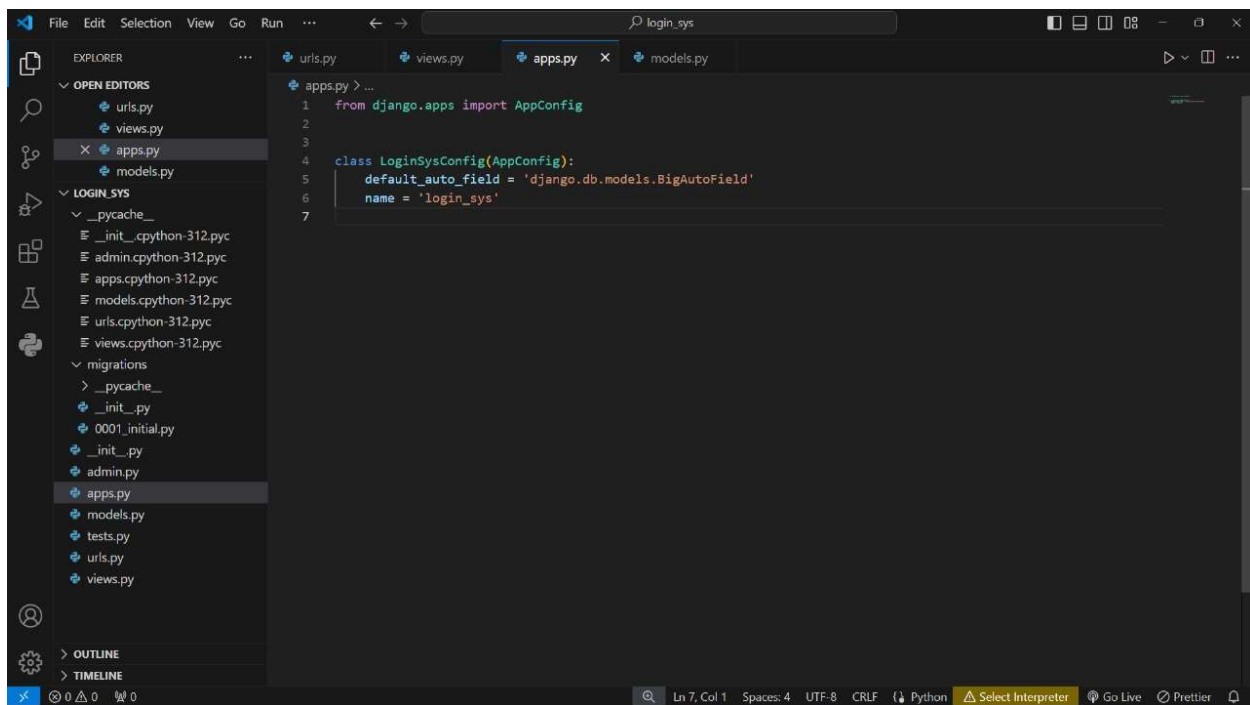
This screenshot shows the Visual Studio Code editor with the Django settings.py file open, displaying settings from line 99 to 124. The Explorer sidebar on the left shows the project structure. The main editor area displays the following code:

```
99     },
100 ]
101
102
103 # Internationalization
104 # https://docs.djangoproject.com/en/5.0/topics/i18n/
105
106 LANGUAGE_CODE = 'en-us'
107
108 TIME_ZONE = 'UTC'
109
110 USE_I18N = True
111
112 USE_TZ = True
113
114
115 # Static files (CSS, JavaScript, Images)
116 # https://docs.djangoproject.com/en/5.0/howto/static-files/
117
118 STATIC_URL = 'static/'
119
120 # Default primary key field type
121 # https://docs.djangoproject.com/en/5.0/ref/settings/#default-auto-field
122
123 DEFAULT_AUTO_FIELD = 'django.db.models.BigAutoField'
124
```

The status bar at the bottom indicates the current position is Line 1, Column 1, with 4 spaces, UTF-8 encoding, CRLF line endings, and the Python interpreter selected.

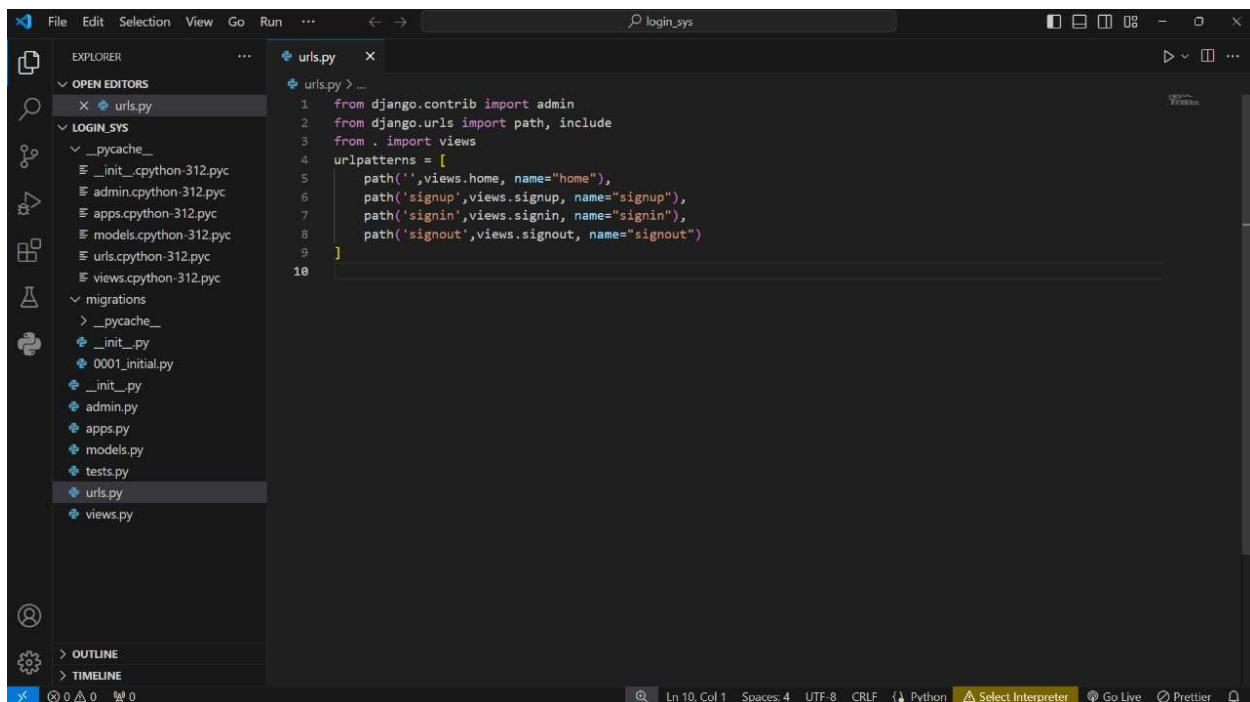

```
1 | Generated by Django 5.0.3 on 2024-03-15 14:15
2
3 import django.core.validators
4 from django.db import migrations, models
5
6
7 class Migration(migrations.Migration):
8
9     initial = True
10
11     dependencies = [
12     ]
13
14     operations = [
15         migrations.CreateModel(
16             name='details',
17             fields=[
18                 ('id', models.BigAutoField(auto_created=True, primary_key=True, serialize=False, verbose_name='id')),
19                 ('username', models.CharField(max_length=20, validators=[django.core.validators.RegexValidator('^[a-zA-Z0-9_]+$')])),
20                 ('first_name', models.CharField(max_length=20)),
21                 ('last_name', models.CharField(max_length=20)),
22                 ('email', models.EmailField(max_length=100, validators=[django.core.validators.RegexValidator('^[a-zA-Z0-9_+@-]+@[a-zA-Z0-9-]+.?[a-zA-Z0-9-.]+$')])),
23                 ('password', models.CharField(max_length=50, validators=[django.core.validators.RegexValidator('^[a-zA-Z0-9_]+$')])),
24             ],
25         ),
26     ],
27
```

```
1 from django.contrib import admin
2
3 # Register your models here.
4 from login_sys.models import details
5 admin.site.register(details)
```



```
File Edit Selection View Go Run ... login_sys
EXPLORER
  OPEN EDITORS
    urls.py
    views.py
    apps.py
    models.py
  LOGIN_SYS
    __pycache__
      __init__.cpython-312.pyc
      admin.cpython-312.pyc
      apps.cpython-312.pyc
      models.cpython-312.pyc
      urls.cpython-312.pyc
      views.cpython-312.pyc
    migrations
      __pycache__
        __init__.py
        0001_initial.py
        __init__.py
        admin.py
        apps.py
        models.py
        tests.py
        urls.py
        views.py
  OUTLINE
  TIMELINE

apps.py
1 from django.apps import AppConfig
2
3
4 class LoginSysConfig(AppConfig):
5     default_auto_field = 'django.db.models.BigAutoField'
6     name = 'login_sys'
7
```



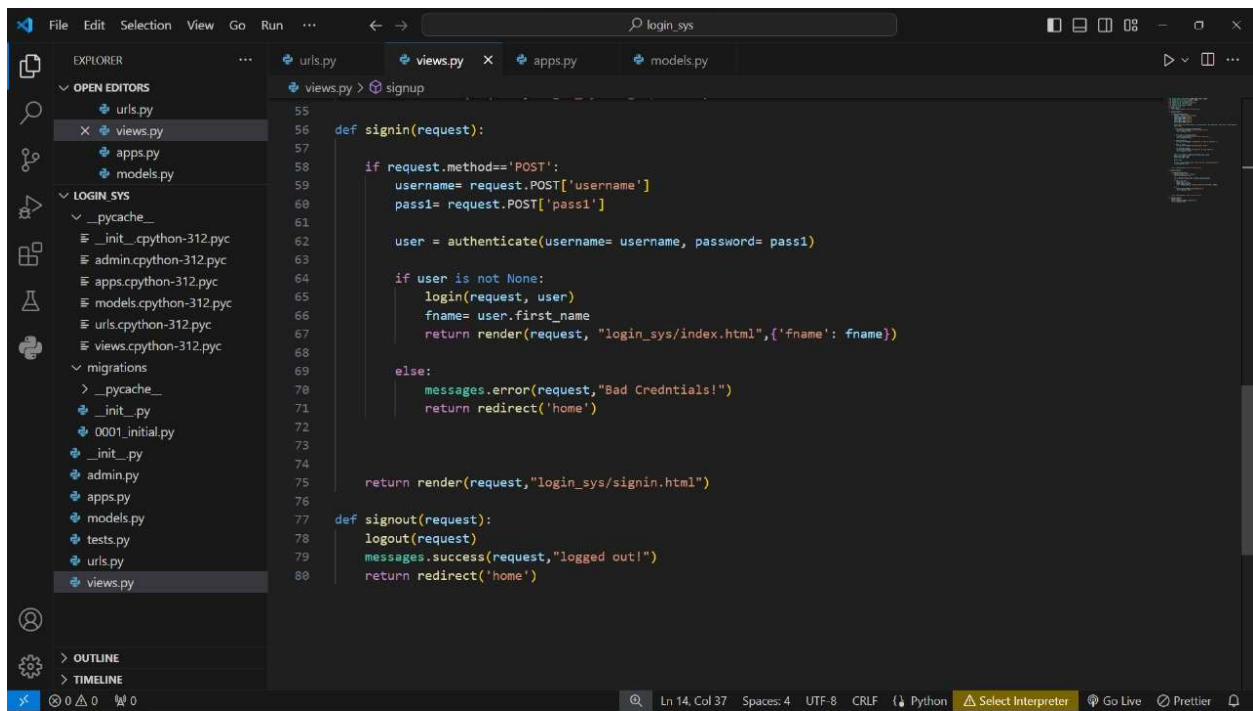
```
File Edit Selection View Go Run ... login_sys
EXPLORER
  OPEN EDITORS
    urls.py
    views.py
  LOGIN_SYS
    __pycache__
      __init__.cpython-312.pyc
      admin.cpython-312.pyc
      apps.cpython-312.pyc
      models.cpython-312.pyc
      urls.cpython-312.pyc
      views.cpython-312.pyc
    migrations
      __pycache__
        __init__.py
        0001_initial.py
        __init__.py
        admin.py
        apps.py
        models.py
        tests.py
        urls.py
        views.py
  OUTLINE
  TIMELINE

urls.py
1 from django.contrib import admin
2 from django.urls import path, include
3 from . import views
4 urlpatterns = [
5     path('', views.home, name="home"),
6     path('signup', views.signup, name="signup"),
7     path('signin', views.signin, name="signin"),
8     path('signout', views.signout, name="signout")
9 ]
10
```



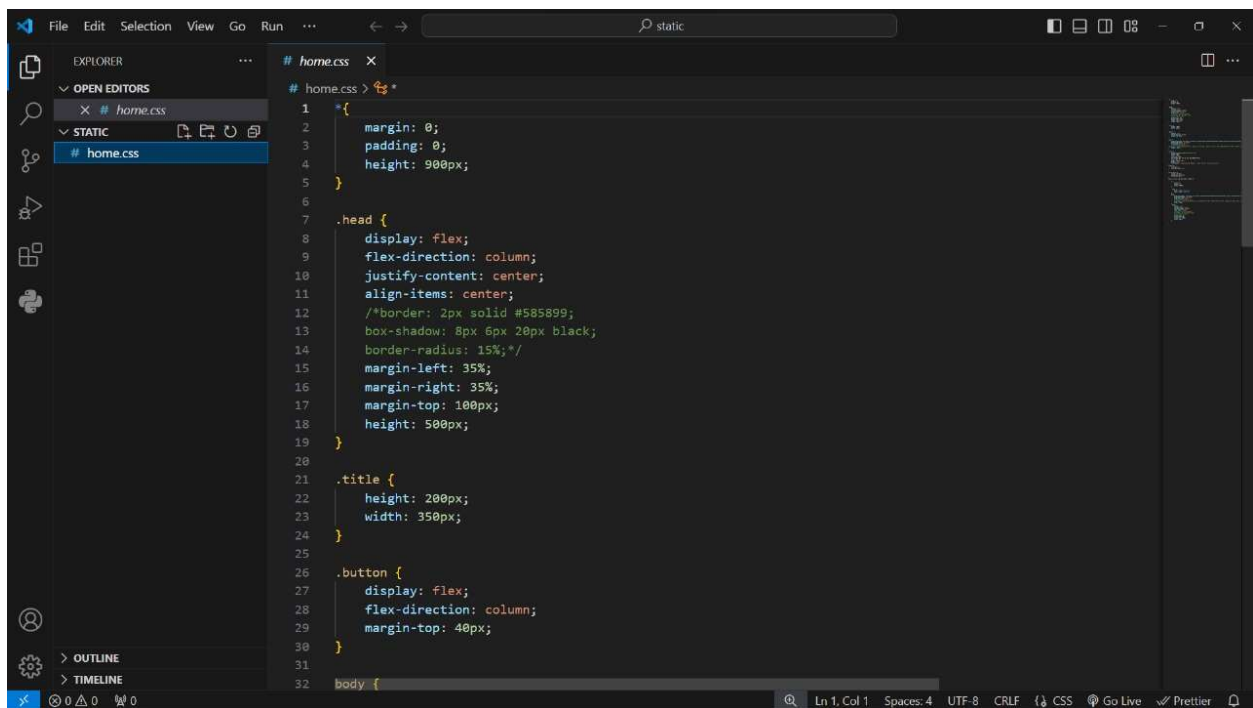
```
1 from django.contrib.auth import authenticate, login, logout
2 from django.shortcuts import redirect, render
3 from django.http import HttpResponseRedirect
4 from django.contrib.auth.models import User
5 from django.contrib import messages
6 from login_sys.models import details
7 # Create your views here.
8 def home(request):
9     return render(request, "login_sys/index.html")
10
11 def signup(request):
12     if request.method=="POST":
13         username = request.POST['username']
14         fname=request.POST['fname']
15         lname=request.POST['lname']
16         email=request.POST['email']
17         pass1=request.POST['pass1']
18         pass2=request.POST['pass2']
19
20         store= details(username=username, first_name=fname, last_name=lname, email=email, password=pass1)
21         store.save()
22
23         if User.objects.filter(username=username):
24             messages.error(request,"username already exist!")
25             return redirect('home')
26
27         if User.objects.filter(email=email):
28             messages.error(request,"Email already registered!")
29             return redirect('home')
```

```
11 def signup(request):
12     if len(username)>20:
13         messages.error(request,"Username must be under 20 characters.")
14
15     if pass1 != pass2:
16         messages.error(request,"Passwords didn't match")
17
18     if not username.isalnum():
19         messages.error(request,"Username must be alpha numeric")
20         return redirect('home')
21
22     myuser= User.objects.create_user(username,email, pass1)
23     myuser.first_name= fname
24     myuser.last_name= lname
25     myuser.save()
26
27     messages.success(request,"Your account has been created successfully")
28     return redirect('signin')
29
30 return render(request, "login_sys/signup.html")
31
32 def signin(request):
33     if request.method=='POST':
34         username= request.POST['username']
35         pass1= request.POST['pass1']
36
37         user = authenticate(username= username, password= pass1)
```



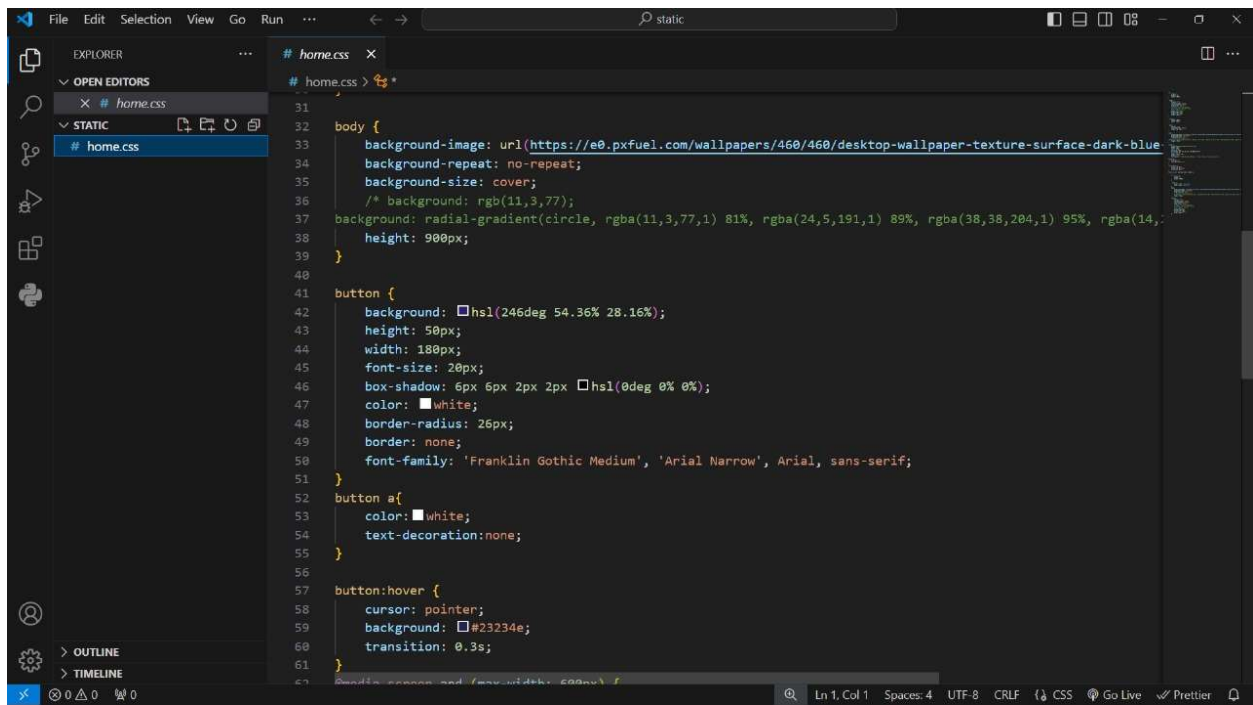
The screenshot shows the Visual Studio Code editor with a project named 'login_sys'. The Explorer sidebar on the left shows the file structure, including 'OPEN EDITORS' (urls.py, views.py, apps.py, models.py) and 'LOGIN_SYS' (various Python files). The main editor window displays the 'views.py' file, specifically the 'signin' and 'signout' functions. The 'signin' function handles a POST request, authenticates the user, and renders the 'login_sys/index.html' template if successful, or redirects to 'home' if credentials are bad. The 'signout' function simply logs out the user and redirects to 'home'.

```
55
56 def signin(request):
57     if request.method=='POST':
58         username= request.POST['username']
59         pass1= request.POST['pass1']
60
61         user = authenticate(username= username, password= pass1)
62
63         if user is not None:
64             login(request, user)
65             fname= user.first_name
66             return render(request, "login_sys/index.html",{'fname': fname})
67
68         else:
69             messages.error(request,"Bad Credentials!")
70             return redirect('home')
71
72
73
74
75 return render(request,"login_sys/signin.html")
76
77 def signout(request):
78     logout(request)
79     messages.success(request,"logged out!")
80     return redirect('home')
```

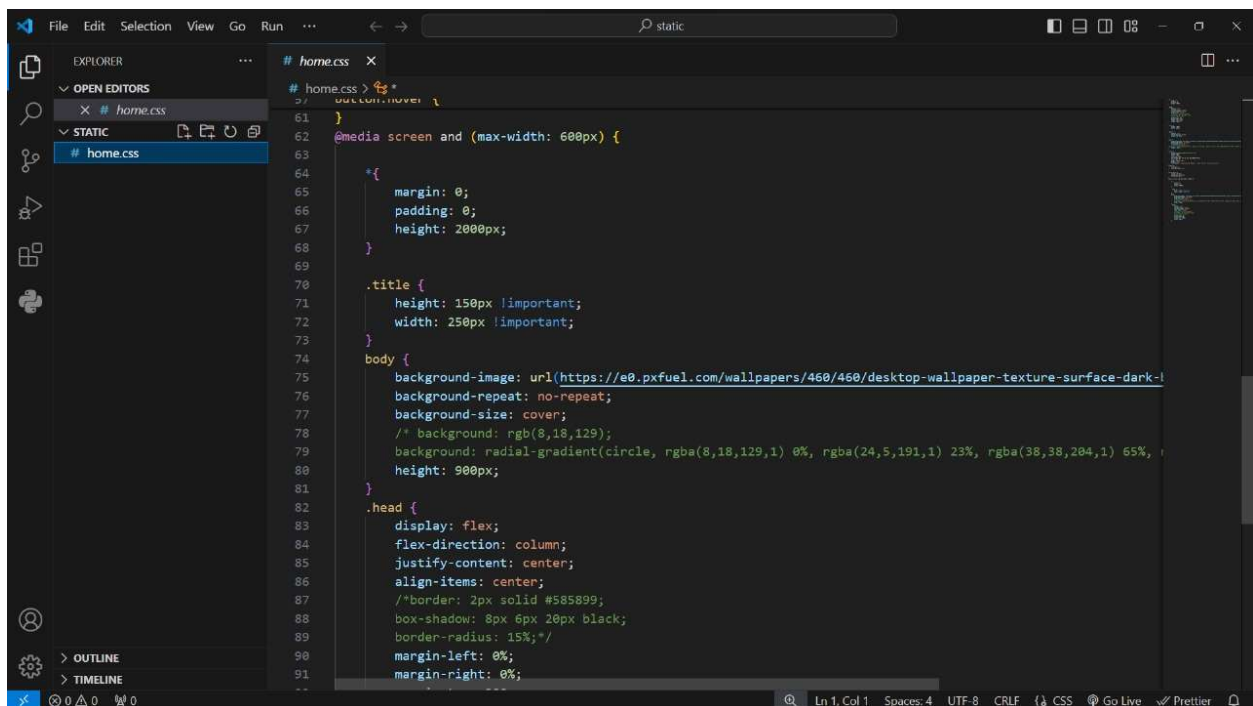


The screenshot shows the Visual Studio Code editor with a project named 'static'. The Explorer sidebar on the left shows the file structure, including 'OPEN EDITORS' (home.css) and 'STATIC' (home.css). The main editor window displays the 'home.css' file, which contains styles for a login page. The styles include a base style for the body, a head style for the header, a title style for the login form, and a button style for the login button.

```
1 *{
2     margin: 0;
3     padding: 0;
4     height: 900px;
5 }
6
7 .head {
8     display: flex;
9     flex-direction: column;
10    justify-content: center;
11    align-items: center;
12    /*border: 2px solid #585899;
13    box-shadow: 8px 6px 20px black;
14    border-radius: 15%;*/
15    margin-left: 35%;
16    margin-right: 35%;
17    margin-top: 100px;
18    height: 500px;
19 }
20
21 .title {
22     height: 200px;
23     width: 350px;
24 }
25
26 .button {
27     display: flex;
28     flex-direction: column;
29     margin-top: 40px;
30 }
31
32 body {
```



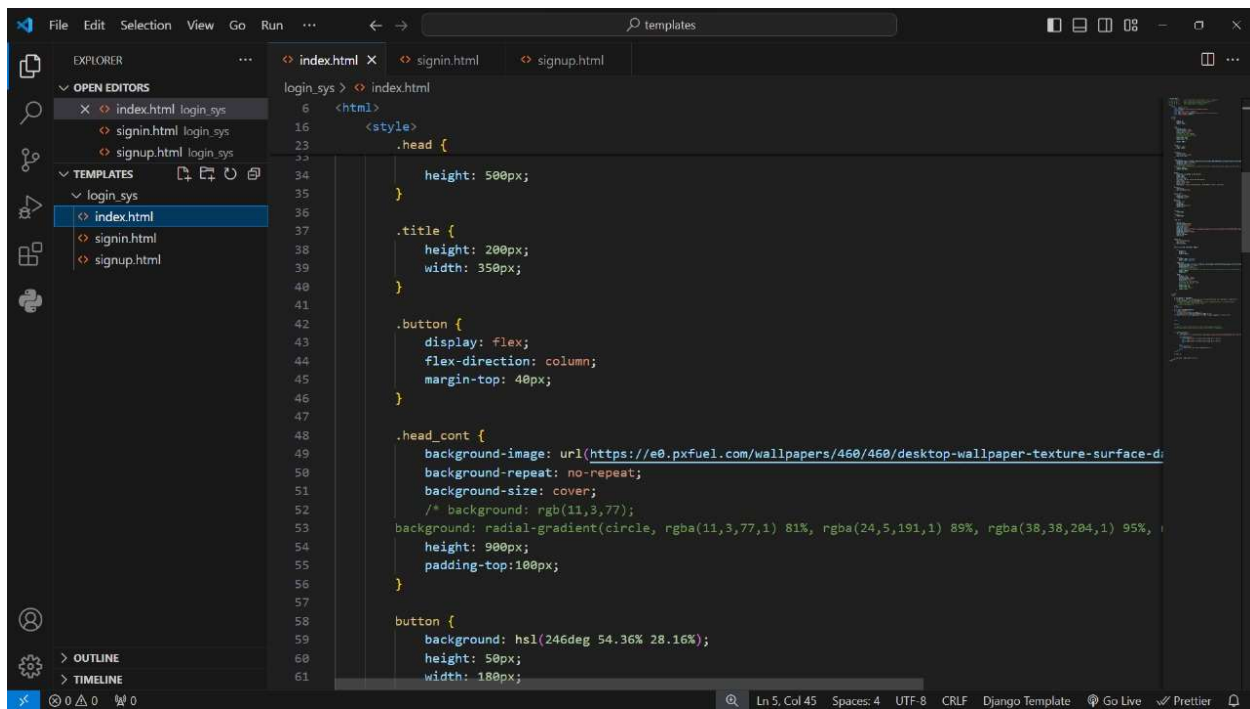
```
# home.css
31
32
33 body {
34   background-image: url(https://e0.pxfuel.com/wallpapers/460/460/desktop-wallpaper-texture-surface-dark-blue);
35   background-repeat: no-repeat;
36   background-size: cover;
37   /* background: rgb(11,3,77);
38   background: radial-gradient(circle, rgba(11,3,77,1) 81%, rgba(24,5,191,1) 89%, rgba(38,38,204,1) 95%, rgba(14,14,14,1) 100%);
39   height: 900px;
40 }
41
42 button {
43   background: hsl(246deg 54.36% 28.16%);
44   height: 50px;
45   width: 180px;
46   font-size: 20px;
47   box-shadow: 6px 6px 2px 2px hsl(0deg 0% 0%);
48   color: white;
49   border-radius: 26px;
50   border: none;
51   font-family: 'Franklin Gothic Medium', 'Arial Narrow', Arial, sans-serif;
52 }
53
54 button a {
55   color: white;
56   text-decoration: none;
57 }
58
59 button: hover {
60   cursor: pointer;
61   background: #23234e;
62   transition: 0.3s;
63 }
```



```
# home.css
61
62 @media screen and (max-width: 600px) {
63
64   * {
65     margin: 0;
66     padding: 0;
67     height: 2000px;
68   }
69
70   .title {
71     height: 150px !important;
72     width: 250px !important;
73   }
74
75   body {
76     background-image: url(https://e0.pxfuel.com/wallpapers/460/460/desktop-wallpaper-texture-surface-dark-l);
77     background-repeat: no-repeat;
78     background-size: cover;
79     /* background: rgb(8,18,129);
80     background: radial-gradient(circle, rgba(8,18,129,1) 0%, rgba(24,5,191,1) 23%, rgba(38,38,204,1) 65%,
81     height: 900px;
82   }
83
84   .head {
85     display: flex;
86     flex-direction: column;
87     justify-content: center;
88     align-items: center;
89     /*border: 2px solid #585899;
90     box-shadow: 8px 6px 20px black;
91     border-radius: 15%;*/
92     margin-left: 0%;
93     margin-right: 0%;
94   }
95 }
```

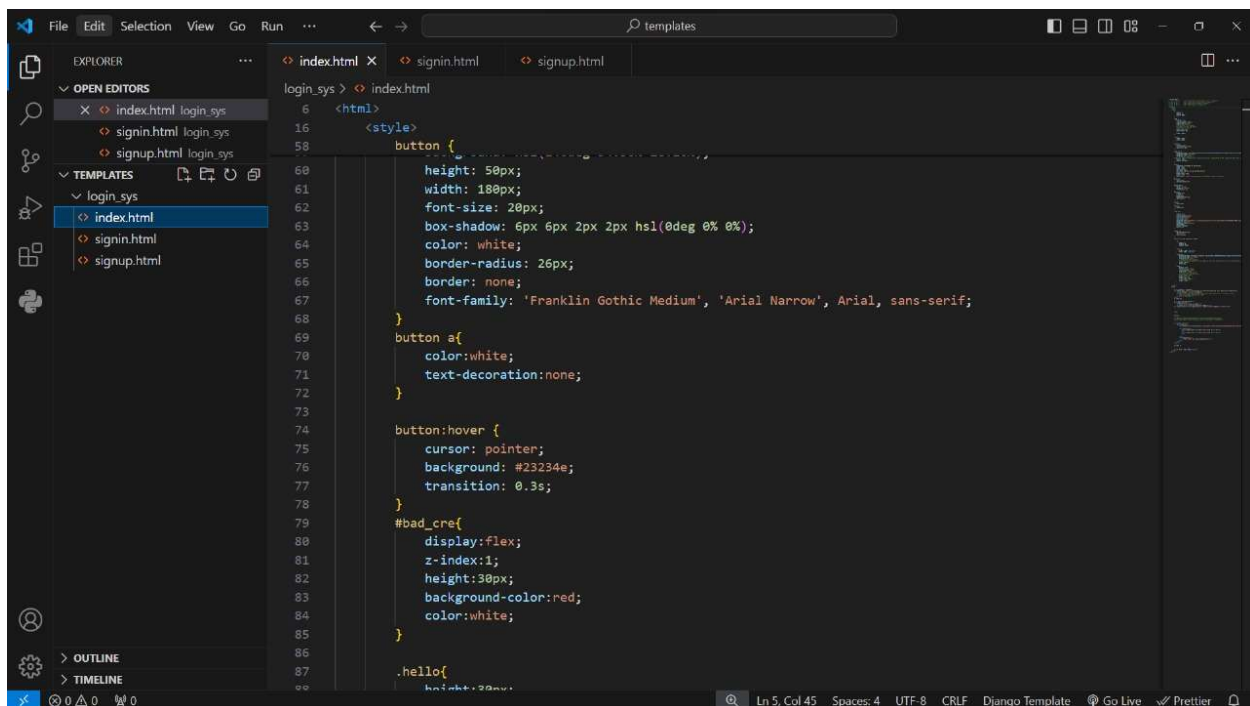
```
# home.css
# home.css
62 @media screen and (max-width: 600px) {
70   .title {
71   }
72 }
73 .body {
74   background-image: url(https://e0.pxfuel.com/wallpapers/460/460/desktop-wallpaper-texture-surface-dark-l
75   background-repeat: no-repeat;
76   background-size: cover;
77   /* background: rgb(8,18,129);
78   background: radial-gradient(circle, rgba(8,18,129,1) 0%, rgba(24,5,191,1) 23%, rgba(38,38,204,1) 65%,
79   height: 900px;
80 }
81 }
82 .head {
83   display: flex;
84   flex-direction: column;
85   justify-content: center;
86   align-items: center;
87   /*border: 2px solid #585899;
88   box-shadow: 8px 6px 20px black;
89   border-radius: 15%;*/
90   margin-left: 0%;
91   margin-right: 0%;
92   margin-top: 300px;
93   height: 600px;
94 }
95 }
96 }
```

```
login_sys > index.html
1 <!DOCTYPE html>
2 <!--[if lt IE 7]> <html class="no-js lt-ie7 lt-ie8 lt-ie9"> <![endif]-->
3 <!--[if IE 7]> <html class="no-js lt-ie9 lt-ie8"> <![endif]-->
4 <!--[if IE 8]> <html class="no-js lt-ie9"> <![endif]-->
5 <!--[if gt IE 8]> <html class="no-js"> <![endif]-->
6 <html>
7   <head>
8     <meta charset="utf-8">
9     <meta http-equiv="X-UA-Compatible" content="IE=edge">
10    <title></title>
11    <meta name="description" content="">
12    <meta name="viewport" content="width=device-width, initial-scale=1">
13    <link rel="stylesheet" href="">
14  </head>
15  <style>
16    *{
17      margin: 0;
18      padding: 0;
19      height: 900px;
20    }
21  }
22  .head {
23    display: flex;
24    flex-direction: column;
25    justify-content: center;
26    align-items: center;
27    /*border: 2px solid #585899;
28    box-shadow: 8px 6px 20px black;
29    border-radius: 15%;*/
30    margin-left: 35%;
31    margin-right: 35%;
32  }
```



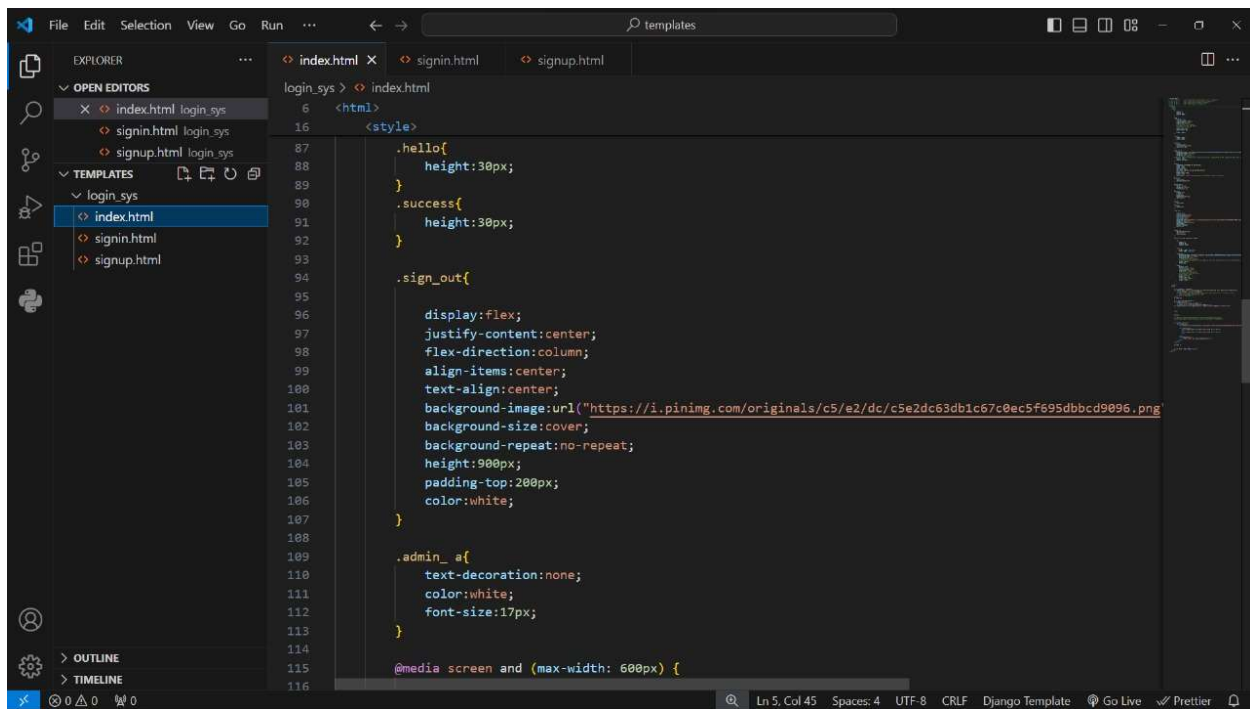
The image shows a VS Code editor window with the Explorer sidebar on the left. The Explorer sidebar has two sections: 'OPEN EDITORS' and 'TEMPLATES'. Under 'OPEN EDITORS', there are three files: 'index.html login_sys', 'signin.html login_sys', and 'signup.html login_sys'. Under 'TEMPLATES', there are three files: 'index.html', 'signin.html', and 'signup.html'. The 'index.html' file is selected. The main editor area shows the CSS code for 'index.html'. The code is as follows:

```
login_sys > index.html
6 <html>
16 <style>
23 .head {
33
34     height: 500px;
35
36
37
38     .title {
39         height: 200px;
40         width: 350px;
41     }
42
43     .button {
44         display: flex;
45         flex-direction: column;
46         margin-top: 40px;
47     }
48
49     .head_cont {
50         background-image: url(https://e0.pxfuel.com/wallpapers/460/460/desktop-wallpaper-texture-surface-d
51         background-repeat: no-repeat;
52         background-size: cover;
53         /* background: rgb(11,3,77);
54         background: radial-gradient(circle, rgba(11,3,77,1) 81%, rgba(24,5,191,1) 89%, rgba(38,38,204,1) 95%,
55         height: 900px;
56         padding-top: 100px;
57     }
58
59     button {
60         background: hsl(246deg 54.36% 28.16%);
61         height: 50px;
62         width: 180px;
```

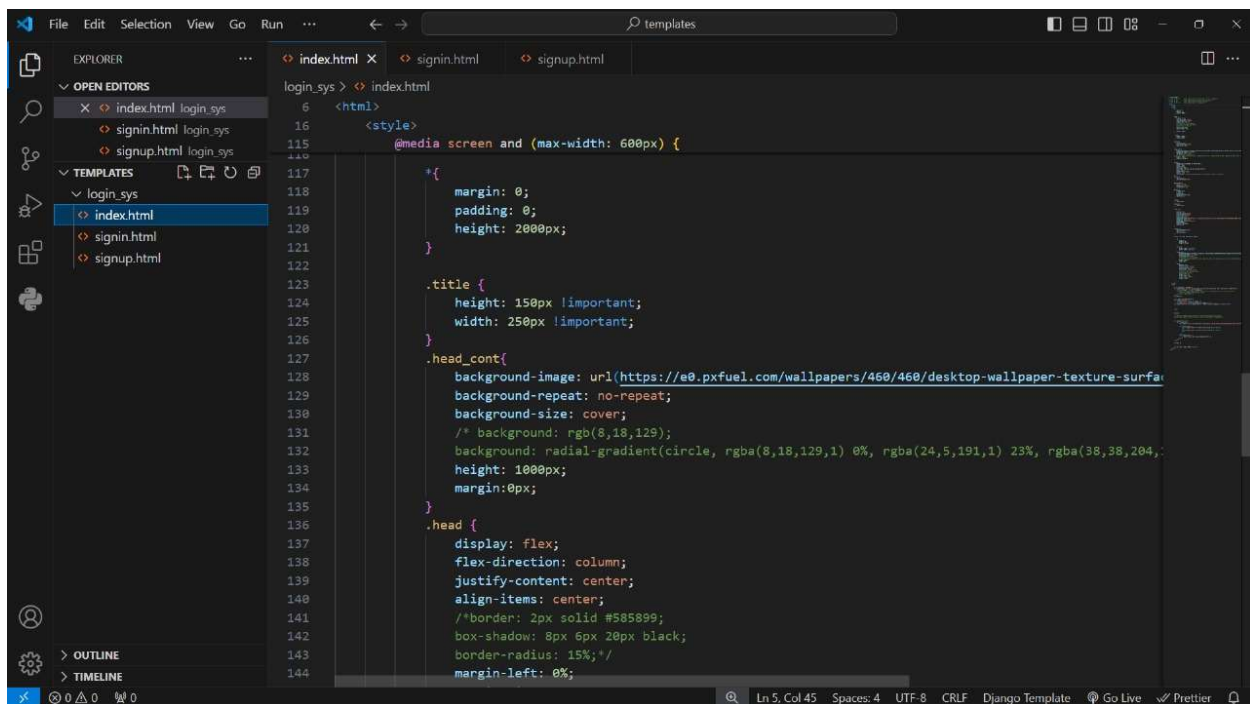


The image shows a VS Code editor window with the Explorer sidebar on the left. The Explorer sidebar has two sections: 'OPEN EDITORS' and 'TEMPLATES'. Under 'OPEN EDITORS', there are three files: 'index.html login_sys', 'signin.html login_sys', and 'signup.html login_sys'. Under 'TEMPLATES', there are three files: 'index.html', 'signin.html', and 'signup.html'. The 'index.html' file is selected. The main editor area shows the CSS code for 'index.html'. The code is as follows:

```
login_sys > index.html
6 <html>
16 <style>
58 button {
60     height: 50px;
61     width: 180px;
62     font-size: 20px;
63     box-shadow: 6px 6px 2px 2px hsl(0deg 0% 0%);
64     color: white;
65     border-radius: 26px;
66     border: none;
67     font-family: 'Franklin Gothic Medium', 'Arial Narrow', Arial, sans-serif;
68 }
69 button a{
70     color:white;
71     text-decoration:none;
72 }
73
74 button:hover {
75     cursor: pointer;
76     background: #23234e;
77     transition: 0.3s;
78 }
79 #bad_cre{
80     display: flex;
81     z-index: 1;
82     height: 30px;
83     background-color: red;
84     color: white;
85 }
86
87 .hello{
88     height: 20px;
```

```
login_sys > index.html
6 <html>
16 <style>
87 .hello{
88     height:30px;
89 }
90 .success{
91     height:30px;
92 }
93
94 .sign_out{
95
96     display:flex;
97     justify-content:center;
98     flex-direction:column;
99     align-items:center;
100     text-align:center;
101     background-image:url("https://i.pinimg.com/originals/c5/e2/dc/c5e2dc63db1c67c0ec5f695dbbcd9096.png");
102     background-size:cover;
103     background-repeat:no-repeat;
104     height:900px;
105     padding-top:200px;
106     color:white;
107 }
108
109 .admin_a{
110     text-decoration:none;
111     color:white;
112     font-size:17px;
113 }
114
115 @media screen and (max-width: 600px) {
116
```



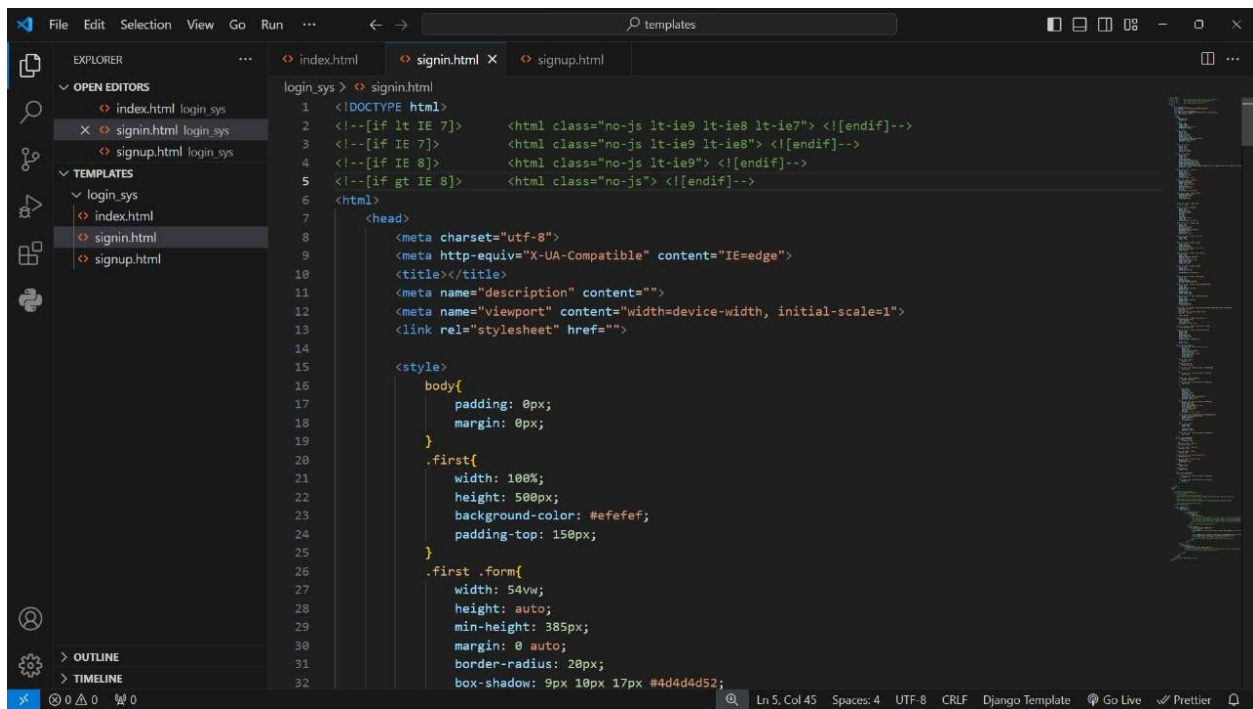
```
login_sys > index.html
115 @media screen and (max-width: 600px) {
116
117     *{
118         margin: 0;
119         padding: 0;
120         height: 2000px;
121     }
122
123     .title {
124         height: 150px !important;
125         width: 250px !important;
126     }
127
128     .head_cont{
129         background-image: url(https://e0.pxfuel.com/wallpapers/460/460/desktop-wallpaper-texture-surf);
130         background-repeat: no-repeat;
131         background-size: cover;
132         /* background: rgb(8,18,129); */
133         background: radial-gradient(circle, rgba(8,18,129,1) 0%, rgba(24,5,191,1) 23%, rgba(38,38,204,1) 100%);
134         height: 1000px;
135         margin:0px;
136     }
137
138     .head {
139         display: flex;
140         flex-direction: column;
141         justify-content: center;
142         align-items: center;
143         /*border: 2px solid #585899;
144         box-shadow: 8px 6px 20px black;
145         border-radius: 15%;*/
146         margin-left: 0%;
147     }
148
```

```
File Edit Selection View Go Run ... templates
EXPLORER
  OPEN EDITORS
    X index.html login_sys
    signin.html login_sys
    signup.html login_sys
  TEMPLATES
    login_sys
      index.html
      signin.html
      signup.html
  OUTLINE
  TIMELINE

login_sys > index.html
6 <html>
16 <style>
115 @media screen and (max-width: 600px) {
136 .head {
145     margin-right: 0%;
146     /*margin-top: 300px;*/
147     height: 600px;
148 }
150 }
151 </style>
152 <body>
153
154 {% for message in messages%}
155 <div class="alert alert-{{ message.tags }}" alert-dismissible fade show id="bad_cre" role="alert">
156 <strong>Message: </strong>{{ message }}
157 {% comment %} <button type="button" class="close" data-dismiss="alert" aria-label="Close">
158 <span aria-hidden="true">$times</span>
159 </button> {% endcomment %}
160 </div>
161 {% endfor %}
162
163 {% if user.is_authenticated %}
164 <div class="sign_out">
165 <h1 class="hello">Hello {{ fname}}</h1>
166 <h2 class="success">You're successfully logged in</h2>
167 <div class="button"><button type="submit"><a href="/signout">signout</a></button></div>
168
169
170
171
172 </div>
```

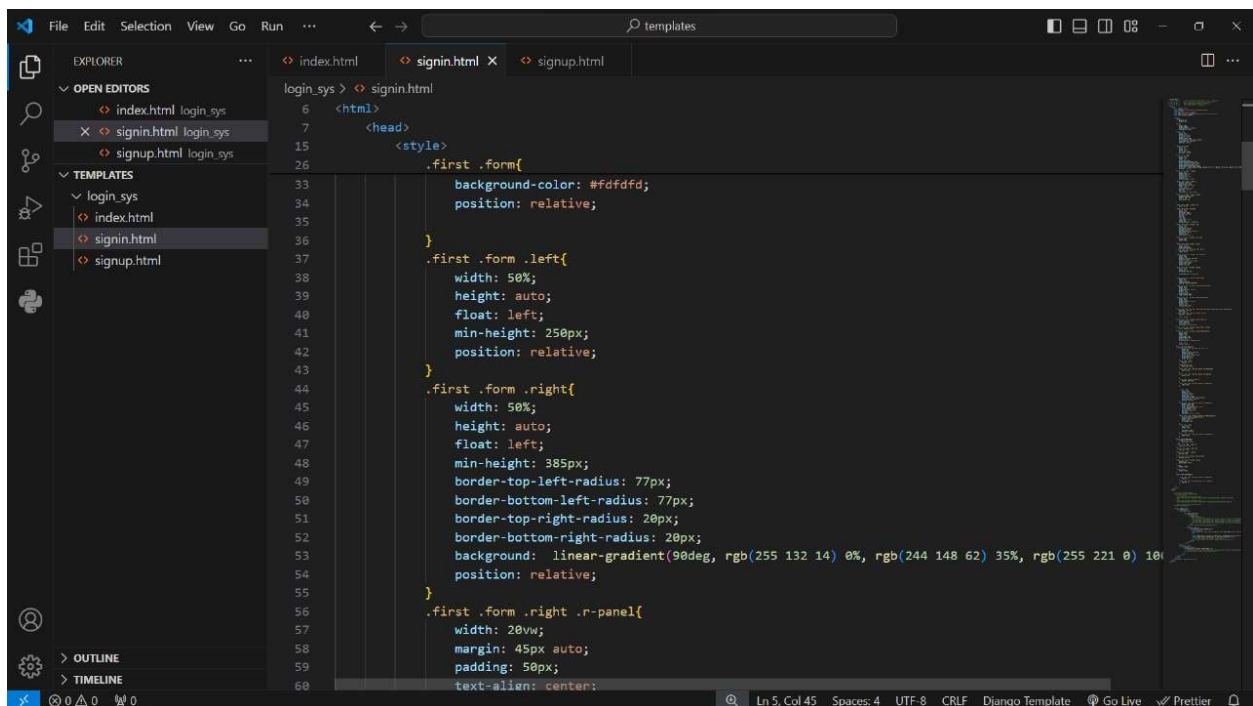
```
File Edit Selection View Go Run ... templates
EXPLORER
  OPEN EDITORS
    X index.html login_sys
    signin.html login_sys
    signup.html login_sys
  TEMPLATES
    login_sys
      index.html
      signin.html
      signup.html
  OUTLINE
  TIMELINE

login_sys > index.html
6 <html>
152 <body>
174
175 {%else%}
176
177 {% comment %} <button type="submit"><a href="/signup">signup</a></button>
178 <button type="submit"><a href="/signin">signin</a></button> {% endcomment %}
179
180
181 <div class="head_cont">
182 <div class="head">
183 <div class="title">
186 <button type="submit"><a href="/signin">Sign In</a></button>
187 <br>
188 <button type="submit"><a href="/signup">Sign Up</a></button>
189 <br>
190 </div>
191 <div class="admin_">
192 <a href="/admin">Are you an Administrator?</a>
193 </div>
194 </div>
195 </div>
196
197 {% endif %}
198
199
200 <script src="" async defer></script>
201 </body>
202 </html>
```



VS Code editor showing the `sign.html` file in the `templates` directory. The file contains HTML boilerplate with a head section and a body section with a form container.

```
1 <!DOCTYPE html>
2 <!--[if lt IE 7]> <html class="no-js lt-ie9 lt-ie8 lt-ie7"> <![endif-->
3 <!--[if IE 7]> <html class="no-js lt-ie9 lt-ie8"> <![endif-->
4 <!--[if IE 8]> <html class="no-js lt-ie9"> <![endif-->
5 <!--[if gt IE 8]> <html class="no-js"> <![endif-->
6 <html>
7 <head>
8 <meta charset="utf-8">
9 <meta http-equiv="X-UA-Compatible" content="IE=edge">
10 <title></title>
11 <meta name="description" content="">
12 <meta name="viewport" content="width=device-width, initial-scale=1">
13 <link rel="stylesheet" href="">
14
15 <body>
16 <div>
17 <div>
18 <div>
19 </div>
20 </div>
21 <div>
22 <div>
23 <div>
24 <div>
25 </div>
26 </div>
27 <div>
28 <div>
29 <div>
30 <div>
31 <div>
32 <div>
```



VS Code editor showing the `sign.html` file in the `templates` directory. The file contains CSS styles for the form container.

```
6 <html>
7 <head>
15 <style>
26
33 <div>
34 <div>
35 <div>
36 <div>
37 <div>
38 <div>
39 <div>
40 <div>
41 <div>
42 <div>
43 <div>
44 <div>
45 <div>
46 <div>
47 <div>
48 <div>
49 <div>
50 <div>
51 <div>
52 <div>
53 <div>
54 <div>
55 <div>
56 <div>
57 <div>
58 <div>
59 <div>
60 <div>
```


This screenshot shows the VS Code editor with the file explorer on the left. The 'login_sys' folder is expanded, showing 'index.html', 'signin.html', and 'signup.html'. The 'signin.html' file is open in the editor, displaying CSS code for a form. The code includes a `<html>` tag, a `<head>` tag with a `<style>` block, and several CSS rules. The `.first .form .right .r-panel` rule sets text alignment to center, display to flex, and justify-content to center. The `.first .form .right .r-panel a` rule sets padding, border-radius, border, color, text-decoration, position, top, and transition. The `.first .form .right .r-panel a:hover` rule sets background-color, padding, color, and font-weight. The `.first .form .right .r-panel h2,p` rule sets color to white. The `.first .form .left .form-box` rule sets width and height.

```
login_sys > > signin.html
6 <html>
7 <head>
15 <style>
56 .first .form .right .r-panel{
57   text-align: center;
61   display: flex;
62   justify-content: center;
63 }
64 .first .form .right .r-panel a{
65   padding: 10px 40px;
66   border-radius: 10px;
67   border: 2px white solid;
68   color: white;
69   text-decoration: none;
70   position: relative;
71   top: 25px;
72   transition: all 0.5s linear;
73 }
74 .first .form .right .r-panel a:hover{
75   background-color: white;
76   padding: 10px 40px;
77   color: black;
78   font-weight: 500;
79 }
80 }
81 .first .form .right .r-panel h2,p{
82   color: white;
83 }
84 }
85 .first .form .left .form-box{
86   width: 100%;
87   height: auto;
```

This screenshot shows the VS Code editor with the file explorer on the left. The 'login_sys' folder is expanded, showing 'index.html', 'signin.html', and 'signup.html'. The 'signin.html' file is open in the editor, displaying CSS code for a form. The code includes a `<html>` tag, a `<head>` tag with a `<style>` block, and several CSS rules. The `.first .form .left .form-box` rule sets height, min-height, position, z-index, top, left, text-align, right, and background-color. The `.first .form .left .form-box .top` rule sets width, height, min-height, padding, background-color, text-align, display, and place-items. The `.first .form .left .form-box .top .icon` rule sets width and height. The `.first .form .left .form-box .top p` rule sets color and margin-top.

```
login_sys > > signin.html
6 <html>
7 <head>
15 <style>
85 .first .form .left .form-box{
86   height: auto;
88   min-height: 307px;
89   position: absolute;
90   z-index: 1;
91   top: 0px;
92   left: 0px;
93   text-align: center;
94   right: 0px;
95   background-color: transparent;
96 }
97 .first .form .left .form-box .top{
98   width: 100%;
99   height: auto;
100   min-height: 115.5px;
101   padding: 12px 0px;
102   background-color: transparent;
103   text-align: center;
104   display: grid;
105   place-items: center;
106 }
107 .first .form .left .form-box .top .icon{
108   width: 100%;
109   height: 50px;
110 }
111 }
112 .first .form .left .form-box .top p{
113   color: #555555;
114   margin-top: 5px;
```

This screenshot shows the VS Code editor with the file explorer on the left. The 'login_sys' folder is expanded, showing 'index.html', 'signin.html', and 'signup.html'. The 'signin.html' file is open in the editor, displaying CSS code for a login form. The code includes a `<html>` tag, a `<head>` tag with a `<style>` block, and several CSS selectors for form elements. The status bar at the bottom indicates the cursor is at line 5, column 45, with 4 spaces, UTF-8 encoding, and CRLF line endings. The editor is using the Django Template and Prettier extensions.

```
login_sys > signin.html
6 <html>
7 <head>
15 <style>
112 .first .form .left .form-box .top p{
115     margin-bottom: 0px;
116 } .first .form .left .form-box .top .icon a{
117     text-decoration: none;
118     color: black;
119 }
120 .first .form .left .form-box .top .icon a i{
121     width: 20px;
122     height: 18px;
123     padding: 12px 10px 10px 10px;
124     font-size: 1rem;
125     border: 1.5px #afafaf solid;
126     border-radius: 12px;
127     margin-left: 10px;
128     cursor: pointer;
129 }
130 .first .form .left .form-box .bottom{
131     width: 100%;
132     height: auto;
133     min-height: 165.5px;
134 }
135     background-color: transparent;
136 }
137 }
138 .first .form .left .form-box .bottom form{
139     width: 60%;
140     height: auto;
141     min-height: 80px;
142     padding: 10px 0px 10px 50px;
```

This screenshot shows the VS Code editor with the file explorer on the left. The 'login_sys' folder is expanded, showing 'index.html', 'signin.html', and 'signup.html'. The 'signin.html' file is open in the editor, displaying CSS code for a login form. The code includes a `<html>` tag, a `<head>` tag with a `<style>` block, and several CSS selectors for form elements. The status bar at the bottom indicates the cursor is at line 5, column 45, with 4 spaces, UTF-8 encoding, and CRLF line endings. The editor is using the Django Template and Prettier extensions.

```
login_sys > signin.html
6 <html>
7 <head>
15 <style>
138 .first .form .left .form-box .bottom form{
143 }
144 .first .form .left .form-box .bottom form #username{
145     width: 20vw;
146     height: 30px;
147     background-color: #efefef;
148     border: none;
149     outline: none;
150     border-radius: 5px;
151     margin-bottom: 15px;
152 }
153 .first .form .left .form-box .bottom form #pass1{
154     width: 20vw;
155     height: 30px;
156     background-color: #efefef;
157     border: none;
158     outline: none;
159     border-radius: 5px;
160     /* margin-bottom: */
161 }
162 .first .form .left .form-box .bottom form #username::placeholder, #pass1::placeholder{
163     position: relative;
164     left: 10px;
165 } .first .form .left .form-box .bottom form p{
166     position: relative;
167 }
168     color: #555555;
169 }
170 .first .form .left .form-box .bottom form p a{
```

This screenshot shows the VS Code editor with the file explorer on the left. The 'login_sys' folder is expanded, showing 'index.html', 'signin.html', and 'signup.html'. The 'signin.html' file is open in the editor, displaying CSS code for a form. The code includes a style tag with rules for the form's appearance, including text decoration, color, font weight, and transition. It also includes a media query for a maximum width of 850px, which adjusts the width and padding of the form elements. The status bar at the bottom indicates the current position is line 5, column 45, with 4 spaces, UTF-8 encoding, and CRLF line endings. The Django Template engine is selected, and the Go Live and Prettier extensions are active.

```
login_sys > > signin.html
6 <html>
7 <head>
15 <style>
170 .first .form .left .form-box .button form p a{
171   text-decoration: none;
172   color: #555555;
173   font-weight: 800;
174   transition: all 0.5s linear;
175 }
176 .first .form .left .form-box .button form p a:hover{
177   color: rgb(248 17 17);
178 }
179 .first .form .left .form-box .button form #button{
180   width: 115px;
181   height: 35px;
182   border: none;
183   font-weight: 700;
184   border-radius: 8px;
185   outline: none;
186   background-color: rgb(248 17 17);
187 }
188   color: white;
189   cursor: pointer;
190 }
191 @media (max-width:850px) {
192   .first .form .left .form-box .top .icon a i {
193     width: 5px;
194     height: 5px;
195     padding: 6px 8px 10px 5px;
196     font-size: 0.7rem;
197     border: 1.5px #afafaf solid;
198     border-radius: 4px;
```

This screenshot shows the VS Code editor with the file explorer on the left. The 'login_sys' folder is expanded, showing 'index.html', 'signin.html', and 'signup.html'. The 'signin.html' file is open in the editor, displaying CSS code for a form. The code includes a media query for a maximum width of 850px, which adjusts the margin-left and cursor of the form elements. It also includes rules for the form's appearance, including font size, width, and padding. The status bar at the bottom indicates the current position is line 5, column 45, with 4 spaces, UTF-8 encoding, and CRLF line endings. The Django Template engine is selected, and the Go Live and Prettier extensions are active.

```
login_sys > > signin.html
6 <html>
7 <head>
15 <style>
191 @media (max-width:850px) {
192   .first .form .left .form-box .top .icon a i {
193     margin-left: 7px;
194     cursor: pointer;
195   }
196   .first .form .right{
197     opacity: 0;
198   }
199   .container form h1{
200     font-size: 1.3rem;
201   }
202   .first .form .left .form-box .button form #username{
203     width: 15vw;
204   }
205   .first .form .left .form-box .button form #pass1{
206     width: 32vw;
207   }
208   .first .form .right .p-panel a{
209     padding: 10px 14px;
210   }
211   .first .form .left .form-box .button form #button {
212     width: 80px;
213   }
214 }
215 }
216 }
217 }
218 }
219 }
220 }
221 }
222 }
223 }
224 }
225 }
```

```
login_sys > > signin.html
6  <html>
7  <head>
15  <style>
191  @media (max-width:850px) {
221  }
224  .first .form {
225  width: 54vw;
226  height: auto;
227  min-height: 500px;
228  margin: 0 auto;
229  border-radius: 20px;
230  box-shadow: 9px 10px 17px #4d4d4d52;
231  background-color: #fdfdfd;
232  position: relative;
233  }
234  .first .form .left .form-box .button form #design{
235  padding: 10px 14px;
236  border-radius: 10px;
237  border: 2px rgb(248 17 17) solid;
238  color: rgb(248 17 17);;
239  text-decoration: none;
240  position: relative;
241  top: 25px;
242  transition: all 0.5s linear;
243  }
244  .first .form .left .form-box .button form #design:hover{
245  background-color: rgb(248 17 17);;
246  padding: 10px 14px;
247  color: black;
248  font-weight: 500;
249  }
```

```
login_sys > > signin.html
6  <html>
7  <head>
15  <style>
191  @media (max-width:850px) {
244  .first .form .left .form-box .button form #design:hover{
250  }
251  .first .form .left{
252  width: 100%;
253  height: auto;
254  }
255  min-height: 250px;
256  position: relative;
257  }
258  .first .form .left .form-box .button form #username {
259  width: 32vw;
260  }
261  }
262  @media (max-width:850px){
263  .container form h1{
264  font-size: 1.1rem;
265  }
266  .first .form .right .r-panel p{
267  font-size: 0.8rem;
268  }
269  .first .form .right .r-panel h2{
270  font-size: 1rem;
271  }
272  .first .form .right .r-panel{
273  padding: 30px;
274  }
275  .first .form .left .form-box .button form {
276  padding-left:0px ;
}
```

The screenshot shows the VS Code editor with the file explorer on the left. The 'login_sys' folder is expanded, showing 'index.html', 'signin.html', and 'signup.html'. The 'signin.html' file is open in the editor. The code is as follows:

```
6 <html>
7 <head>
15 <style>
275 .first .form .left .form-box .bottom form {
277 }
278 .first .form .left .form-box .bottom {
279     display: grid;
280     place-items: center;
281 }
282 .first {
283     height: 550px;
284 }
285 .form {
286     height: 525px;
287 }
288 }
289 @media (min-width:850px){
291     .first .form .left .form-box .bottom form #design{
292         opacity: 0;
293     }
294     .first .form .left .form-box .bottom form .design{
295         opacity: 0;
296     }
297 }
298 </style>
299 </head>
301 <body>
302
303 {% comment %} <h1>Sign In</h1>
304 <form action="/signin" method="post">
```

The screenshot shows the VS Code editor with the file explorer on the left. The 'login_sys' folder is expanded, showing 'index.html', 'signin.html', and 'signup.html'. The 'signin.html' file is open in the editor. The code is as follows:

```
6 <html>
301 <body>
303 {% comment %} <h1>Sign In</h1>
304 <form action="/signin" method="post">
305     {% csrf_token %}
306     <label for="Username">Username</label>
307     <input type="text" id="username" name="username" placeholder="Enter username" required>
308     <br>
309     <label for="Firstname">Password</label>
310     <input type="password" id="pass1" name="pass1" placeholder="Enter Password" required>
311
312     <button type="submit">Sign In</button>
313 </form> {% endcomment %}
314
315 <section class="first">
316     <div class="form">
317         <div class="left">
318
319             <div class="form-box">
320                 <div class="top">
321                     <h2>SIGN IN</h2>
322                     <!--<div class="icon">
323                         <a href="https://www.facebook.com" target="_blank"><i class="fa fa-facebook">
324                         <a href="https://www.google.co.in" target="_blank"><i class="fa fa-google">
325                         <a href="https://www.twitter.com" target="_blank"><i class="fa fa-twitter">
326                         <a href="https://www.linkedin.com" target="_blank"><i class="fa fa-linkedin">
327                     </div>
328                     <p>or, use your email and password</p>!-->
329                 </div>
330                 <div class="bottom">
331                     <form action="/signin" method="post">
332                         {% csrf_token %}
```



```
login_sys > signin.html
6 <html>
301 <body>
315 <section class="first">
316 <div class="form">
317 <div class="left">
333 <input type="text" id="username" name="username" placeholder="Enter Username" />
334 <br>
335 <input type="password" id="pass1" name="pass1" placeholder="Enter password" />
336 <button type="submit" value="Sign in" id="button">Sign In</button>
337 <!--<p class="design">Or, create a new account</p>
338 <a href="index2.html" id="design" target="_blank">Sign Up</a>!-->
339 </div>
340 </div>
341 </div>
342 </div>
343 <div class="right">
344 <div class="r-panel">
345 <h1 style="font-size:50px;">Hello User!</h1>
346 <!--<p>Lorem ipsum dolor sit amet consectetur adipisicing elit. Officiis tempore ipsum,
347 <a href="index2.html" target="_blank">Sign Up</a>!-->
348 </div>
349 </div>
350 </div>
351 </div>
352 </div>
353 </section>
354 <script src="" async defer></script>
355 </body>
356 </html>
```

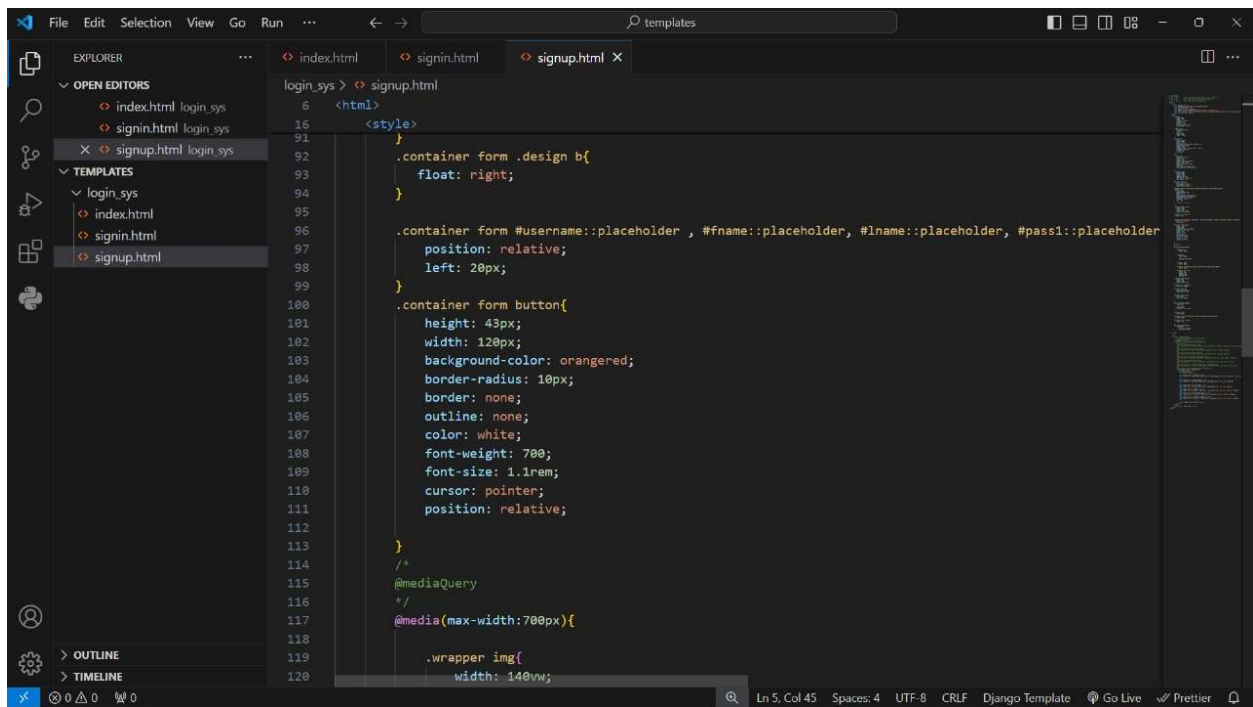
```
login_sys > signup.html
1 <!DOCTYPE html>
2 <!--[if lt IE 7]> <html class="no-js lt-ie9 lt-ie8 lt-ie7"> <![endif]-->
3 <!--[if IE 7]> <html class="no-js lt-ie9 lt-ie8"> <![endif]-->
4 <!--[if IE 8]> <html class="no-js lt-ie9"> <![endif]-->
5 <!--[if gt IE 8]> <html class="no-js"> <![endif]-->
6 <html>
7 <head>
8 <meta charset="utf-8">
9 <meta http-equiv="X-UA-Compatible" content="IE=edge">
10 <title>Create account</title>
11 <meta name="description" content="">
12 <meta name="viewport" content="width=device-width, initial-scale=1">
13 <link rel="stylesheet" href="https://cdnjs.cloudflare.com/ajax/libs/font-awesome/4.7.0/css/font-awesome.min.css">
14 <link rel="stylesheet" href="">
15 </head>
16 <style>
17 .wrapper{
18 width: 100%;
19 height: auto;
20 position: relative;
21 display: grid;
22 display: flex;
23 justify-content: center;
24 place-items: center;
25 }
26 .wrapper img{
27 position: absolute;
28 z-index: 1;
29 top: 0;
30 width: 100%;
31 height: 670px;
32 }
```

This screenshot shows the VS Code editor with the file explorer on the left. The 'login_sys' folder is expanded, showing 'index.html', 'signin.html', and 'signup.html'. The 'signup.html' file is open in the editor. The code is as follows:

```
login_sys > > signup.html
6 <html>
16 <style>
34 .container {
35     height: 560px;
36     width: 532px;
37     margin: 0 auto;
38     box-shadow: 2px 2px 15px rgb(0,0,0.2);
39     border-radius: 10px;
40     padding: 20px;
41     background-color: rgb(101 100 7 / 32%);
42     position: absolute;
43     z-index: 10;
44     top: 56px;
45 }
46 .fa {
47     top: -4px;
48     position: relative;
49     width: 14px;
50     height: 15px;
51     border: 2px solid #efefef;
52     border-radius: 5px;
53     padding: 7px 9px 7px 9px;
54     left: 7px;
55     border-top-left-radius: 4px;
56     border-bottom-left-radius: 4px;
57 }
58 }
59 .container form{
60     height: 600px;
61     width: 500px;
62     padding: 20px;
```

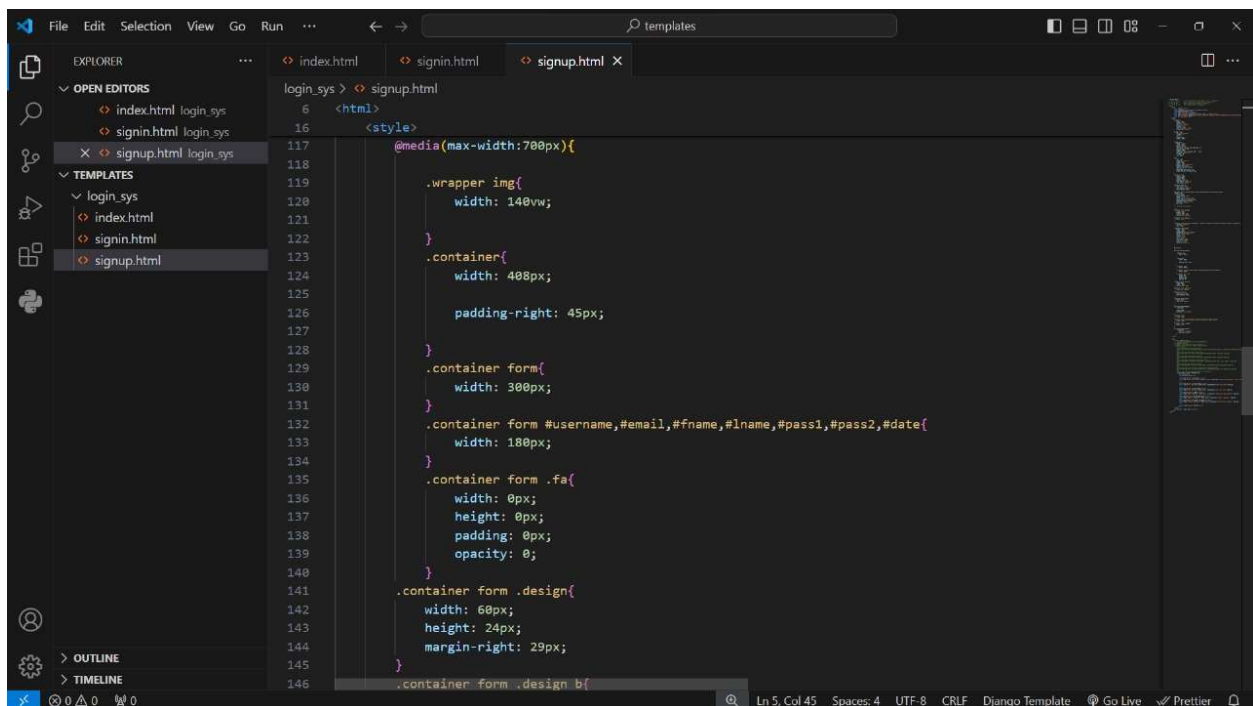
This screenshot shows the VS Code editor with the file explorer on the left. The 'login_sys' folder is expanded, showing 'index.html', 'signin.html', and 'signup.html'. The 'signup.html' file is open in the editor. The code is as follows:

```
login_sys > > signup.html
6 <html>
16 <style>
59 .container form{
64     font-family: system-ui;
65     text-align: center;
66 }
67 .container form h1{
68     text-align: center;
69     font-family: system-ui;
70     margin-bottom: 50px;
71 }
72 .container form #Username,#email,#fname,#lname,#pass1,#pass2,#date,#number{
73     height: 31px;
74     width: 240px;
75     margin-bottom: 23px;
76     border: none;
77     border-top-left-radius: 0px;
78     border-bottom-left-radius: 0px;
79     background-color: #ededed;
80     position: relative;
81     top: -6px;
82 }
83 /* .container form #radio{
84 } */
85 .container form .design{
86     width: 137px;
87     height: 24px;
88     margin-right: 20px;
89     display: inline-block;
90 }
91 }
92 .container form .design bf
```



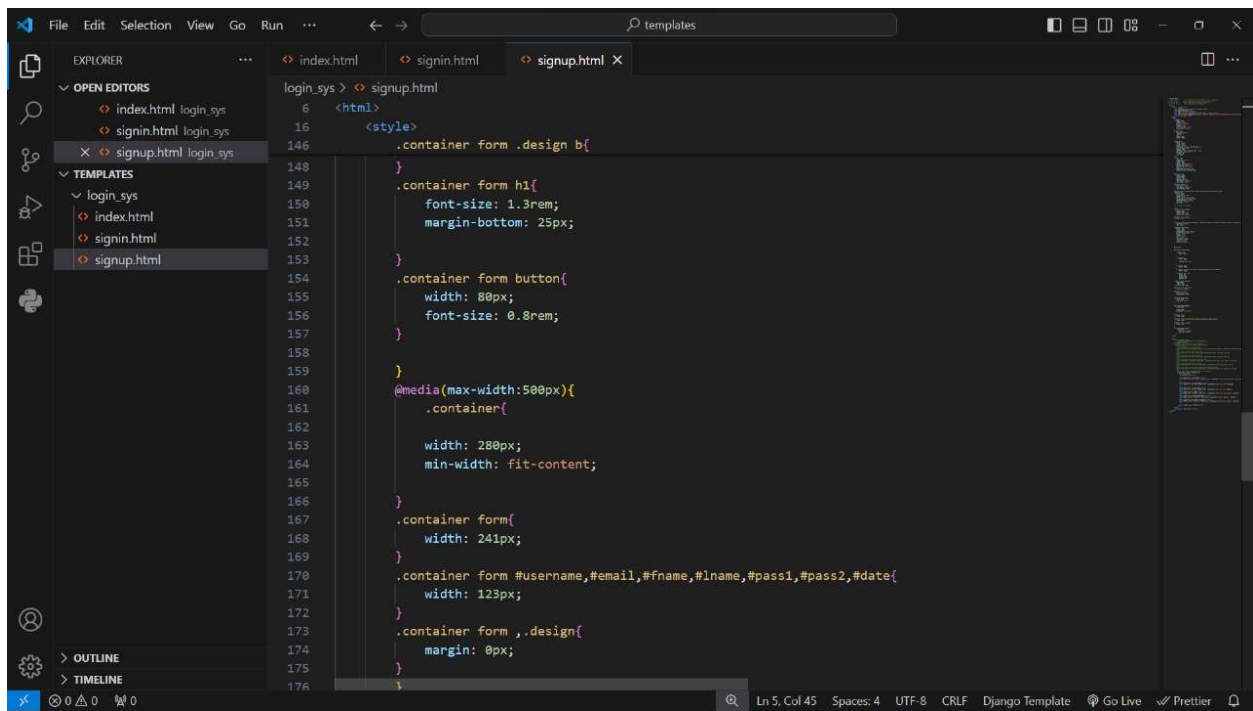
The screenshot shows the VS Code editor with the 'signup.html' file open. The Explorer sidebar on the left shows the project structure with 'login_sys' as a folder containing 'index.html', 'signin.html', and 'signup.html'. The 'TEMPLATES' sidebar shows the 'login_sys' template with the same files. The main editor area displays the following CSS code:

```
login_sys > signup.html
6 <html>
16 <style>
17
18 .container form .design b{
19     float: right;
20 }
21
22 .container form #username::placeholder, #fname::placeholder, #lname::placeholder, #pass1::placeholder
23     position: relative;
24     left: 20px;
25 }
26
27 .container form button{
28     height: 43px;
29     width: 120px;
30     background-color: orangered;
31     border-radius: 10px;
32     border: none;
33     outline: none;
34     color: white;
35     font-weight: 700;
36     font-size: 1.1rem;
37     cursor: pointer;
38     position: relative;
39 }
40
41 /*
42 @mediaQuery
43 */
44 @media(max-width:700px){
45
46     .wrapper img{
47         width: 140vw;
48     }
```



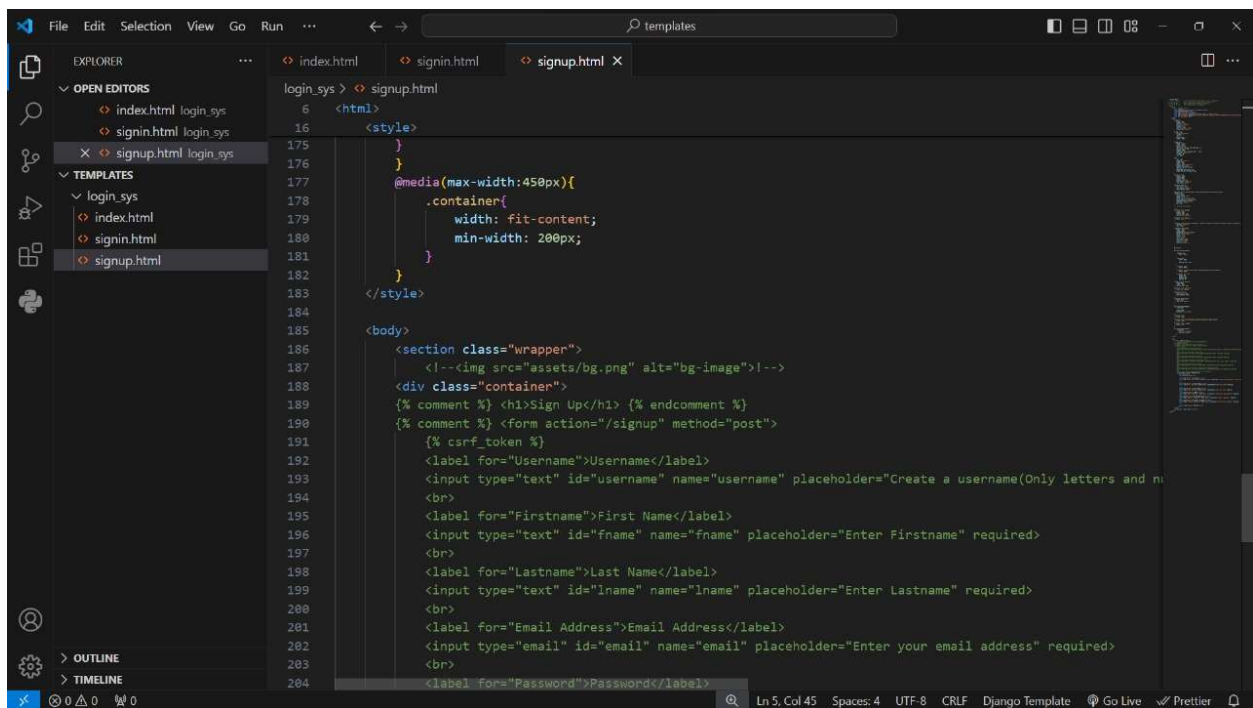
The screenshot shows the VS Code editor with the 'signup.html' file open. The Explorer sidebar on the left shows the project structure with 'login_sys' as a folder containing 'index.html', 'signin.html', and 'signup.html'. The 'TEMPLATES' sidebar shows the 'login_sys' template with the same files. The main editor area displays the following CSS code:

```
login_sys > signup.html
6 <html>
16 <style>
17
18 @media(max-width:700px){
19
20     .wrapper img{
21         width: 140vw;
22     }
23
24     .container{
25         width: 408px;
26         padding-right: 45px;
27     }
28
29     .container form{
30         width: 300px;
31     }
32
33     .container form #username,#email,#fname,#lname,#pass1,#pass2,#date{
34         width: 180px;
35     }
36
37     .container form .fa{
38         width: 0px;
39         height: 0px;
40         padding: 0px;
41         opacity: 0;
42     }
43
44     .container form .design{
45         width: 60px;
46         height: 24px;
47         margin-right: 29px;
48     }
49
50     .container form .design b{
```

This screenshot shows the VS Code editor with the file explorer on the left. The 'login_sys' folder is expanded, showing 'index.html', 'signin.html', and 'signup.html'. The 'signup.html' file is open in the editor, displaying CSS code for a sign-up form. The code includes a container with a design background, a heading, a button, and a form with various input fields. The status bar at the bottom indicates the current line and column.

```
login_sys > signup.html
6 <html>
16 <style>
146 .container form .design b{
148 }
149 .container form h1{
150 font-size: 1.3rem;
151 margin-bottom: 25px;
152 }
153
154 .container form button{
155 width: 80px;
156 font-size: 0.8rem;
157 }
158
159 }
160 @media(max-width:500px){
161 .container{
162 width: 280px;
163 min-width: fit-content;
164 }
165
166 .container form{
167 width: 241px;
168 }
169
170 .container form #username,#email,#fname,#lname,#pass1,#pass2,#date{
171 width: 123px;
172 }
173
174 .container form .design{
175 margin: 0px;
176 }
```



This screenshot shows the VS Code editor with the file explorer on the left. The 'login_sys' folder is expanded, showing 'index.html', 'signin.html', and 'signup.html'. The 'signup.html' file is open in the editor, displaying HTML code for a sign-up form. The code includes a wrapper section with a background image, a heading, and a form with various input fields. The status bar at the bottom indicates the current line and column.

```
login_sys > signup.html
6 <html>
16 <style>
175 }
176 }
177 @media(max-width:450px){
178 .container{
179 width: fit-content;
180 min-width: 200px;
181 }
182 }
183 </style>
184
185 <body>
186 <section class="wrapper">
187 <!--!-->
188 <div class="container">
189 {% comment %} <h1>Sign Up</h1> {% endcomment %}
190 {% comment %} <form action="/signup" method="post">
191 {% csrf_token %}
192 <label for="Username">Username</label>
193 <input type="text" id="username" name="username" placeholder="Create a username(Only letters and n
194 <br>
195 <label for="Firstname">First Name</label>
196 <input type="text" id="fname" name="fname" placeholder="Enter Firstname" required>
197 <br>
198 <label for="Lastname">Last Name</label>
199 <input type="text" id="lname" name="lname" placeholder="Enter Lastname" required>
200 <br>
201 <label for="Email Address">Email Address</label>
202 <input type="email" id="email" name="email" placeholder="Enter your email address" required>
203 <br>
204 <label for="Password">Password</label>
```

```
login_sys > signup.html
6 <html>
185 <body>
190 {% comment %} <form action="/signup" method="post">
211 <form action="/signup" method="post">
212 <div class="design"><b>Username</b></div>
213 <input type="text" id="username" name="username" placeholder="Create a username(Only letters a
214
215 <br>
216 <div class="design"><b>First Name</b></div>
217 <input type="text" id="fname" name="fname" placeholder="Enter first name" required>
218
219 <br>
220 <div class="design"><b>Last Name</b></div>
221 <input type="text" id="lname" name="lname" placeholder="Enter last name" required>
222
223 <br>
224 <div class="design"><b>Email</b></div>
225 <input type="email" id="email" name="email" placeholder="Enter your email address" required>
226
227 <br>
228 <div class="design"><b>Create Password</b></div>
229 <input type="Password" id="pass1" name="pass1" placeholder="Create a password" required>
230
231 <br>
232 <div class="design"><b>Confirm Password</b></div>
233 <input type="Password" id="pass2" name="pass2" placeholder="Confirm your password" required>
234
235 <br>
236 <button type="submit">SIGN UP</button>
237 </form>
238 </form>
239 <script src="" async defer></script>
```

```
login_sys > signup.html
6 <html>
185 <body>
190 {% comment %} <form action="/signup" method="post">
211 <form action="/signup" method="post">
221 <br>
222 <div class="design"><b>Last Name</b></div>
223 <input type="text" id="lname" name="lname" placeholder="Enter last name" required>
224
225 <br>
226 <div class="design"><b>Email</b></div>
227 <input type="email" id="email" name="email" placeholder="Enter your email address" required>
228
229 <br>
230 <div class="design"><b>Create Password</b></div>
231 <input type="Password" id="pass1" name="pass1" placeholder="Create a password" required>
232
233 <br>
234 <div class="design"><b>Confirm Password</b></div>
235 <input type="Password" id="pass2" name="pass2" placeholder="Confirm your password" required>
236
237 <br>
238 <button type="submit">SIGN UP</button>
239 </form>
240 </form>
241 <script src="" async defer></script>
242 </body>
243 </html>
```

CONCLUSION

systems that not only protect user data but also provide a seamless and enjoyable user experience. In conclusion, the login page project using Django with Python, HTML, and CSS has been a significant step towards creating a secure and user-friendly authentication system.

Through the implementation of secure authentication mechanisms, such as password hashing, we have safeguarded user accounts against unauthorized access and potential security threats. Additionally, the intuitive interface design ensures a seamless user experience, allowing users to easily navigate the login page and securely access their accounts.

Furthermore, our login page project lays a solid foundation for future enhancements and integration with additional features. Whether it's implementing multi-factor authentication or integrating social login options.

In essence, the completion of this project signifies our commitment to delivering secure and user-centric solutions. By prioritizing security, usability, and scalability, we have demonstrated our ability to create authentication systems that not only protect user data but also provide a seamless and enjoyable user experience.

FUTURE SCOPE & FURTHER ENHANCEMENTS

1. Multi-Factor Authentication (MFA): Implementing MFA adds an extra layer of security by requiring users to provide additional verification methods, such as SMS codes, email verification, or biometric authentication. Integrating MFA into the login page enhances security and protects user accounts from unauthorized access.

2. Social Login Integration: Integrating social login options, such as Google, Facebook, or Twitter, allows users to log in using their existing social media accounts. This not only simplifies the login process for users but also enhances user engagement by providing seamless access to the application.

3. Password less Authentication: Explore options for password less authentication methods, such as magic links or one-time passwords (OTPs), to offer users an alternative to traditional password-based authentication. This enhances user convenience and eliminates the need for users to remember and manage passwords.

4. Account Recovery Options: Implement account recovery mechanisms, such as email verification or security questions,

to allow users to regain access to their accounts in case they forget their passwords or encounter other login-related issues. Providing robust account recovery options enhances user trust and satisfaction.

5. User Profile Management: Extend the functionality of the login page to include user profile management features, such as profile editing, password reset, and account settings. This empowers users to update their information and manage their accounts conveniently within the application.

6. Audit Trails and Logging: Implement logging and audit trails to track user login activities, including successful logins, failed login attempts, and account lockouts. This helps administrators monitor user access patterns, identify suspicious activities, and enhance security measures proactively.

7. Localization and Internationalization: Add support for multiple languages and locales to make the login page accessible to users worldwide. Implementing localization and internationalization features allows users to customize the language and format of the login page based on their preferences or geographic location.

8. Performance Optimization: Optimize the performance of the login page by implementing caching mechanisms, optimizing database queries, and minimizing page load times. Improving performance ensures a smooth and responsive user experience, especially during periods of high traffic or resource usage.

9. Accessibility Improvements: Enhance the accessibility of the login page by adhering to web accessibility standards (e.g., WCAG) and implementing features such as keyboard navigation, screen reader compatibility, and text alternatives for non-text content. Improving accessibility ensures that the login page is usable by all users, including those with disabilities.

10. Continuous Security Updates: Stay updated with the latest security best practices and frameworks, and regularly review and update security measures to mitigate emerging threats and vulnerabilities. Conduct periodic security audits and penetration testing to identify and address potential security risks proactively.

BIBLIOGRAPHY/REFERENCES

- <https://www.youtube.com/>
- <https://realpython.com/>
- <https://github.com/>
- <https://docs.python.org/3/>
- <https://docs.djangoproject.com/en/stable/>