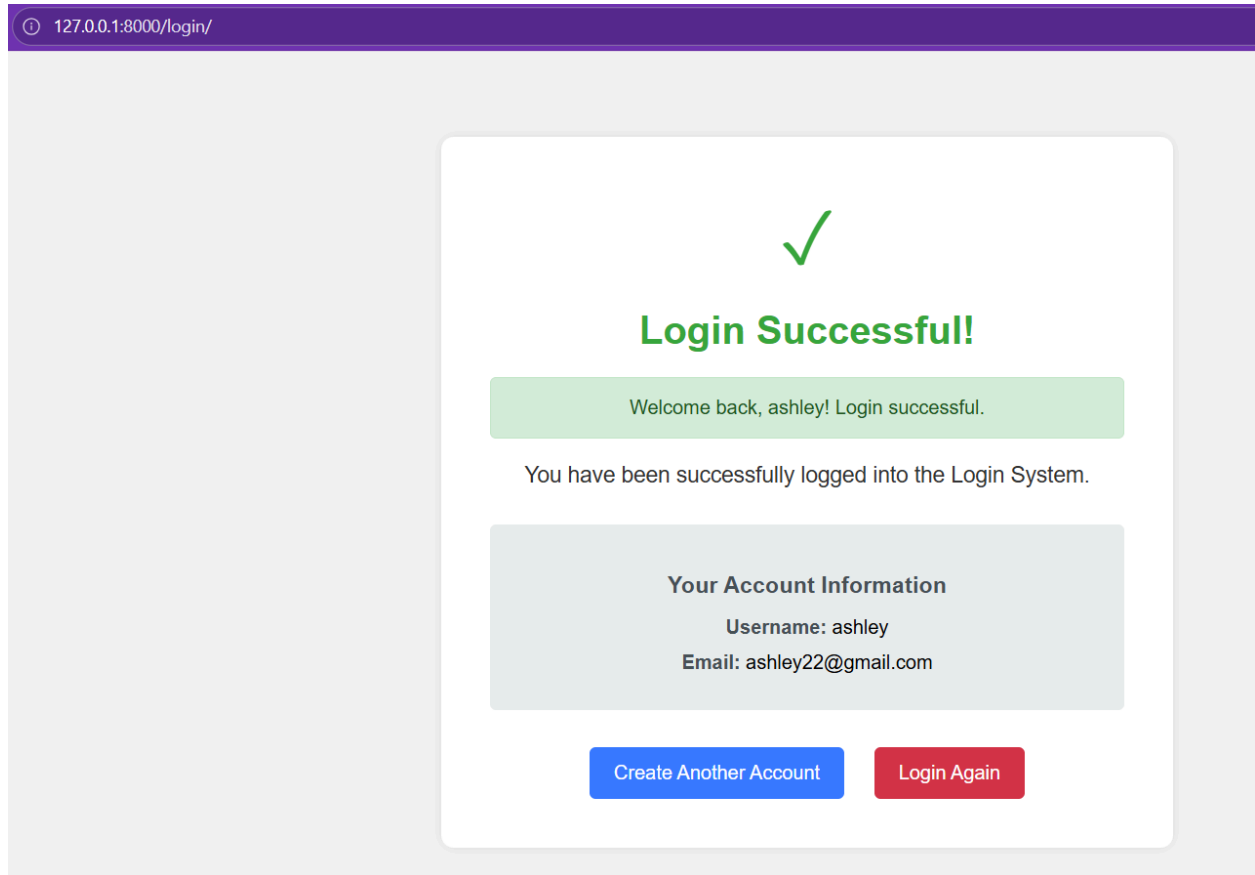


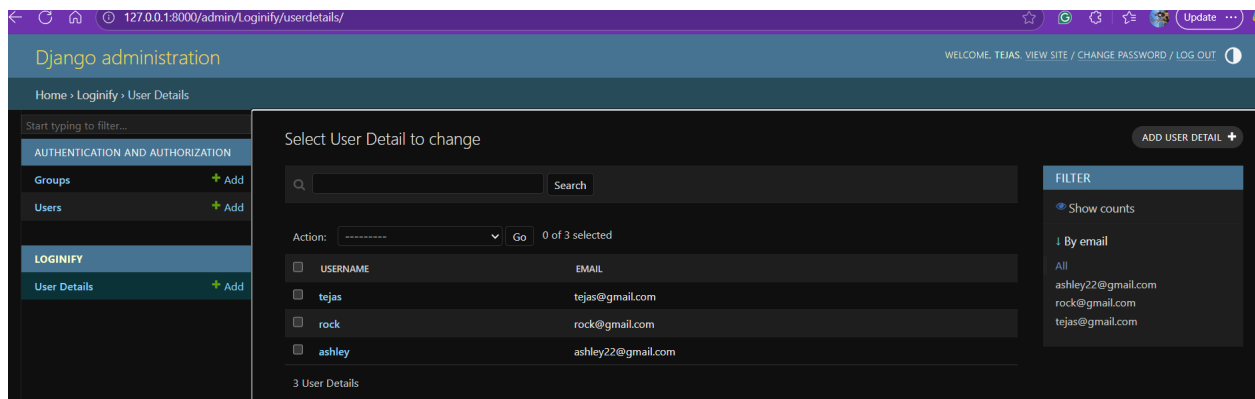
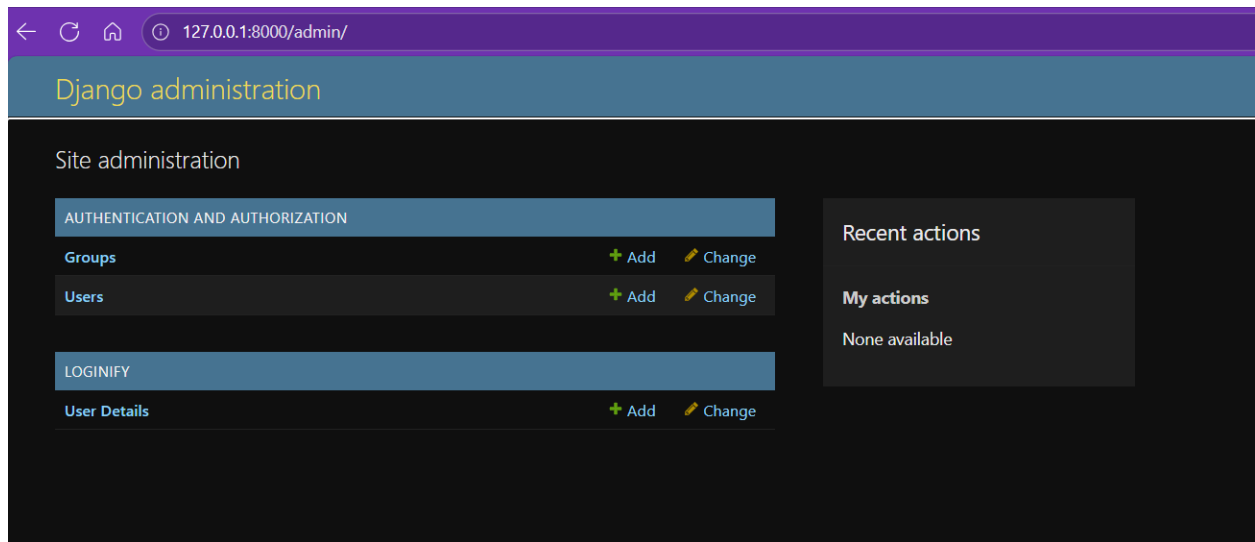
Django Assignment Screenshots

(tejas bachhav)

Login view success:



Django Admin panel:



Python Shell Django implementation:

1. Create new user instance
2. Retrieve all users

```
PS D:\projects\training\DjangoProject>LoginSystem> python manage.py shell
7 objects imported automatically (use -v 2 for details).

Python 3.12.0 (tags/v3.12.0:0fb18b0, Oct 2 2023, 13:03:39) [MSC v.1935 64 bit (AMD64)] on
win32
Type "help", "copyright", "credits" or "license" for more information.
(InteractiveConsole)
>>>
>>> new_user = User.objects.create(username="user1",email="user1@gmail.com", password="user
1@123")
>>> all_users = User.objects.all()
>>> print(all_users)
<QuerySet [<User: tejas>, <User: user1>]>
>>> print(all_users.count())
2
```

- 3 retrieve a single user by username

```
>>> username = "Mike"
>>> user_by_name = UserDetails.objects.get(username=username)
>>> print(user_by_name)
Mike
>>>
```

- 4 username to delete

```
>>> username_to_delete = "Mike"
>>> user_to_delete = ""
>>> user_to_delete = UserDetails.objects.get(username=username_to_delete)
>>> print("deleted user is",username_to_delete)
deleted user is Mike
>>>
```

- 5 Create a new instance using an object

```
>>> obj = UserDetails.objects.create(username="shell_user", email="shell@example.com", password="shell123")
>>> print(obj)
shell_user
>>>
```

- 6 query objects

```
>>> queryset = UserDetails.objects.filter(email__contains="gmail")
>>> print(queryset)
<QuerySet [<UserDetails: tejas>, <UserDetails: rock>, <UserDetails: ashley>, <UserDetails: Mike>, <UserDetails: kyle>]>
>>> queryset = UserDetails.objects.filter(email__contains="example")
>>> print(queryset)
<QuerySet [<UserDetails: shell_user>]>
>>>
```

7 update an object

```
>>> obj.email = "updated_shell@example.com"
>>> obj.save()
>>> print("updated email to:", obj.email)
updated email to: updated_shell@example.com
>>>
```

8 Delete object

```
>>> obj.delete()
(1, {'Loginify.UserDetails': 1})
>>>
>>>
```

Task 5 - CRUD operations - Screenshots

1. Create test user

The screenshot displays a REST client interface for a request to `http://localhost:8000/signup/`. The request is a `POST` with a JSON body containing user details. The response is a `201 Created` status with a success message and the created user's data.

Request:

```
POST http://localhost:8000/signup/

{
  "username": "testuser",
  "email": "test@example.com",
  "password": "test123"
}
```

Response:

```
201 Created • 27 ms • 427 B

{
  "status": "success",
  "message": "User created successfully",
  "user": {
    "username": "testuser",
    "email": "test@example.com",
    "password": "test123"
  }
}
```

1. Get all users views:

GET
http://localhost:8000/users

Params
Authorization
Headers (6)
Body
Scripts
Settings

Query Params

	Key	Value	Description
	Key	Value	Description

Body
Cookies
Headers (8)
Test Results

200 OK
9 ms
77

JSON
Preview
Visualize

```

1  {
2    "status": "success",
3    "count": 6,
4    "users": [
5      {
6        "username": "tejas",
7        "email": "tejas@gmail.com",
8        "password": "tejas@123"
9      },
10     {
11       "username": "rock",
12       "email": "rock@gmail.com",
13       "password": "rock@123"
14     },
15     {
16       "username": "ashley",
17       "email": "ashley22@gmail.com",
18       "password": "ashley@123"
19     },
20     {
21       "username": "Mike",
22       "email": "mike@gmail.com",
23       "password": "mike123"
24     },
25     {
26       "username": "kyle",

```

2.

Get a single user using by email view:

GET http://localhost:8000/user/rock@gmail.com/

Params Authorization Headers (6) Body Scripts Settings

Query Params

	Key	Value	Description
	Key	Value	Description

Body Cookies Headers (8) Test Results

200 OK • 10 ms • 377 B •

{ } JSON Preview Visualize

```

1  {
2    "status": "success",
3    "user": {
4      "username": "rock",
5      "email": "rock@gmail.com",
6      "password": "rock@123"
7    }
8  }
```

Update User details

PUT http://127.0.0.1:8000/user/test@example.com/update/ Send

Params Authorization Headers (9) Body Scripts Settings

none form-data x-www-form-urlencoded raw binary GraphQL JSON

```

1  {
2    "username": "updated_user",
3    "password": "newpass123"
4  }
```

Body Cookies Headers (8) Test Results

200 OK • 32 ms • 429 B • Save Response

{ } JSON Preview Visualize

```

1  {
2    "status": "success",
3    "message": "User updated successfully",
4    "user": {
5      "username": "updated_user",
6      "email": "test@example.com",
7      "password": "newpass123"
8    }
9  }
```

To delete a user using its email.

django-crud-ops / New Request

Save Share

DELETE http://localhost:8000/user/test@example.com/delete/ Send

Params Authorization Headers (7) Body Scripts Settings

☒ none ☐ form-data ☐ x-www-form-urlencoded ☐ raw ☐ binary ☐ GraphQL

This request does not have a body

Body Cookies Headers (8) Test Results 200 OK 16 ms 372 B Save Response

{ } JSON Preview Visualize

```
1 {
2   "status": "success",
3   "message": "User testuser with email test@example.com deleted successfully"
4 }
```

Thank you
