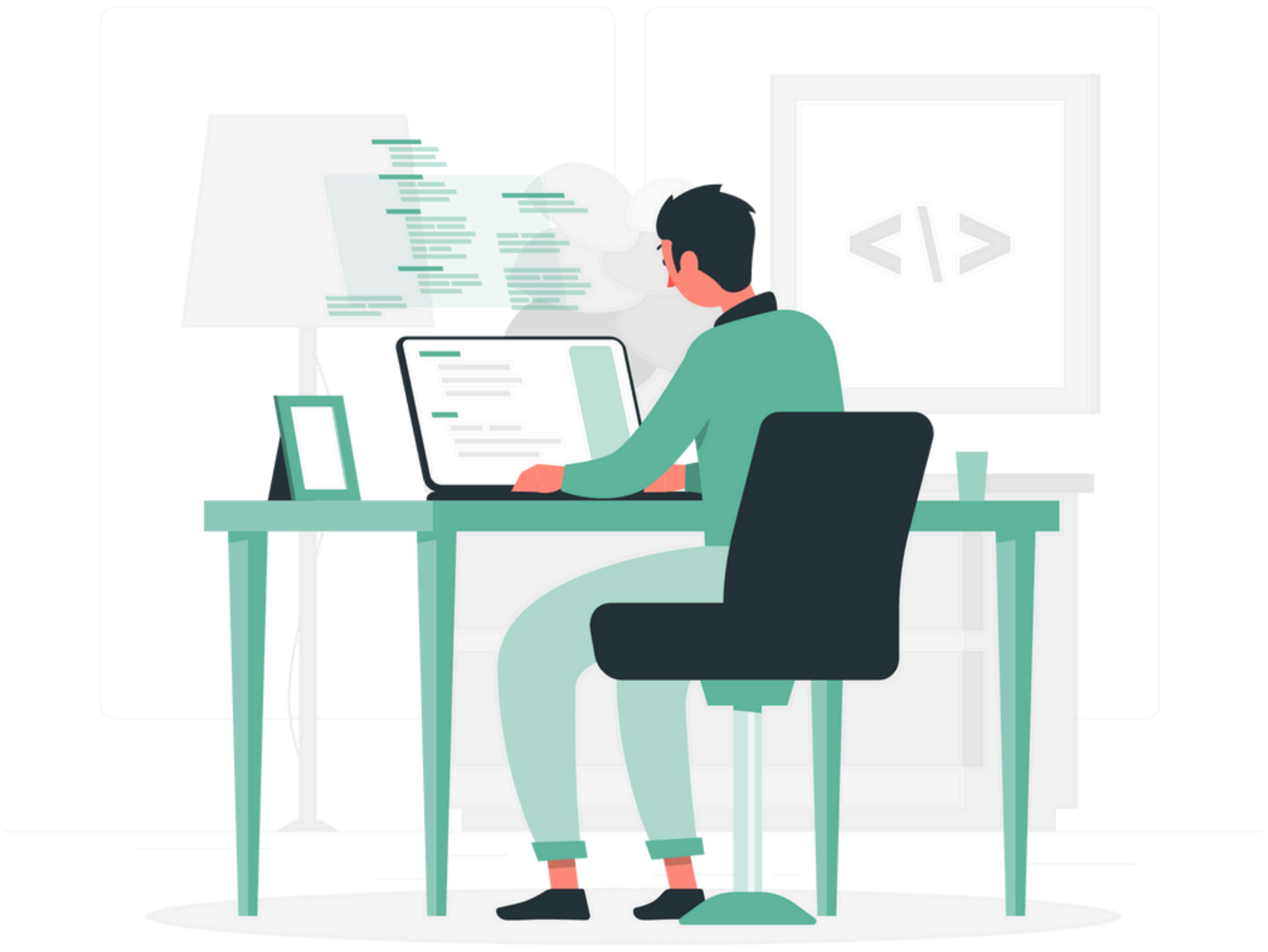
consultadd

django ASSIGNMENTS



Training Team
www.consultadd.com

consultadd

INSTRUCTION

Django Login System

Before starting the Assignments, Let’s set up our git repository.

1. Create Repository: DjangoProject:
   - Create a GitHub repository named “DjangoProject”.
2. Update the README file with project details and commit relevant messages.
3. Include screenshots of the tasks(templates and postman responses).

DJANGO LOGIN SYSTEM

Description:

- This Django project aims to create a robust system with features for user signup, login, and profile management.
- It includes functionalities such as user registration, user data retrieval, updating user details, and deleting user accounts.
- The project utilises Django's built-in features for model creation, views implementation, URL routing, and template rendering to achieve seamless user interaction and data management.
- Additionally, thorough testing with Postman ensures the reliability and functionality of the CRUD operations.

TASK- 1

SETTING UP PROJECT

Set up a Django project named "Login System" with a virtual environment and a Django application named "Loginify".

Microtasks:

1. Create a Virtual Environment: DjangoAssignment
- Create a virtual environment named "DjangoAssignment" to isolate project dependencies.

2. Activate the Virtual Environment
- Activate the virtual environment to use it for installing and running Django.

3. Install Django
- Install Django within the activated virtual environment to use it for the project.

4. Create a New Django Project: Login System
- Create a new Django project named "Login System" where all configurations and settings will reside.

5. Create a New Django Application: “Loginify”
- Create a new Django application within the project to handle the login functionality.

consultadd

TASK- 2

CREATE VIEWS
AND URLS FOR LOGIN SYSTEM

Create views and define URL patterns for the "Login System" Django application to handle login functionality.

Microtasks:

1. **Create Views**

- Create views within the "Loginify" Django application to handle login functionality.
- Create a view that returns an HTTP response with the text "Hello, world!" for testing purposes.

2. **Define URL Patterns**

- Define URL patterns in the "urls.py" file of the "Loginify" Django application to map views to specific URLs.
- Ensure that the URL patterns are properly configured to match the desired endpoints.

consultadd

TASK- 3

Define Models for Login System

Define models, implement views, and set up URLs and templates in Loginify.

Microtasks:

1. **Models :**

Create a “UserDetails” Model which has fields below

- `Username: Use models.CharField(max_length=50, primary_key=True)`
- `Email: Use models.EmailField(unique=True)`
- `Password: Use models.CharField(max_length=12, blank=True)`

Implement views in views.py for signup, login.

2. **Define URLs and Templates**

- Define URL patterns in urls.py for the implemented views.
- Create HTML templates for signup and login forms, confirmation page, and success message.
- Upon successful signup, redirect to the login page.
- Upon successful login, display a success message.

3.Signup view:

- Implement the Signup view in views.py, which handles user registration with inputs for name, email, and password.
- Ensure that the email field is unique.

4. Login view:

- Implement the Login view in views.py, which requires inputs for email and password.

TASK- 4

MODELS & ADMIN

Set up a superuser account using Django's manage.py command and verify the superuser endpoint by accessing the admin interface to ensure proper configuration and functionality.

1. Setup Superuser

- Create a superuser using Django's manage.py command.

```
python manage.py createsuperuser
```

- Verify the superuser endpoint by visiting the admin interface.

Lets Dive into the Django Shell to explore the Power of Command-Line Magic for Managing Your Django Project!

```
python manage.py shell
```

1.Create a new user instance :

```
new_user = User.objects.create(username="example_user",
email="user@example.com", password="example123")
```

2.Retrieve all Users :

```
all_users = User.objects.all()
```

3.Retrieve a single user by name:

For example :

```
username = “john”
user_by_name = User.objects.get(username=username)
```

4. Delete a user by username :

```
username_to_delete= “john”
user_to_delete =
User.objects.get(username=username_to_delete)
```

5.Create a new instance using object

```
obj = YourModel.objects.create(field1=value1,
field2=value2)
```

6.Query objects

```
queryset = YourModel.objects.filter(field1=value1)
```

7.Update an object

```
obj.field1 = new_value
obj.save()
```

8.Delete an object

```
obj.delete()
```

TASK- 5

CRUD OPERATIONS

Implement CRUD (Create, Read, Update, Delete) operations for managing user data within the Django login system.

Microtask:

1. Implement CRUD Operations -
Create four additional views functions for CRUD operations.

- Get all user details view: Retrieves and displays details of all users.
- Get a single user using by email view: Retrieves and displays details of a specific user based on their name .
- Update User details
- To delete a user using its email.

 - These views handle read, update, and delete operations for user data.
 - Use Postman to test and perform CRUD operations API's.

THANK YOU.

KEEP LEARNING!