1. **What is Django REST Framework?**

Django REST Framework (DRF) is a powerful toolkit for building Web APIs in Python. It's built on top of Django, which is a high-level Python web framework, and provides a set of tools and functionalities for quickly building RESTful APIs.

DRF includes features such as:

1. Serialization: DRF provides serializers to convert complex data types, such as querysets and model instances, into native Python data types that can be easily rendered into JSON, XML, or other content types.

2. Viewsets and routers: DRF offers viewsets and routers for defining the logic for handling different HTTP methods (GET, POST, PUT, DELETE) on resources. Viewsets combine the logic for multiple related views into a single class, making it easier to organize and manage code.

3. Authentication and authorization: DRF supports various authentication methods, such as session authentication, token authentication, OAuth, etc., allowing you to control access to your APIs. It also provides authorization classes for defining permissions at the object level.

4. Pagination: DRF includes built-in support for paginating large data sets, helping to improve API performance and reduce response times.

5. Versioning: DRF allows you to version your APIs to support backward compatibility and manage changes over time.

6. Testing tools: DRF provides utilities for testing APIs, including tools for making requests, authenticating users, and validating responses.

1. **What is restful api ?**

A RESTful API, or Representational State Transfer Application Programming Interface, is an architectural style for designing networked applications, particularly web services, that allows different systems to communicate with each other over the internet. It's based on a set of principles that enable the creation of scalable, flexible, and maintainable APIs.

Key principles of RESTful APIs include:

1. \*\*Client-Server Architecture\*\*: The client and server are separate entities that communicate over HTTP or HTTPS. They operate independently, allowing for increased scalability and flexibility.

2. \*\*Statelessness\*\*: Each request from a client to the server must contain all the information necessary to understand and fulfill the request. The server doesn't store any client state between requests, which simplifies server implementation and improves scalability.

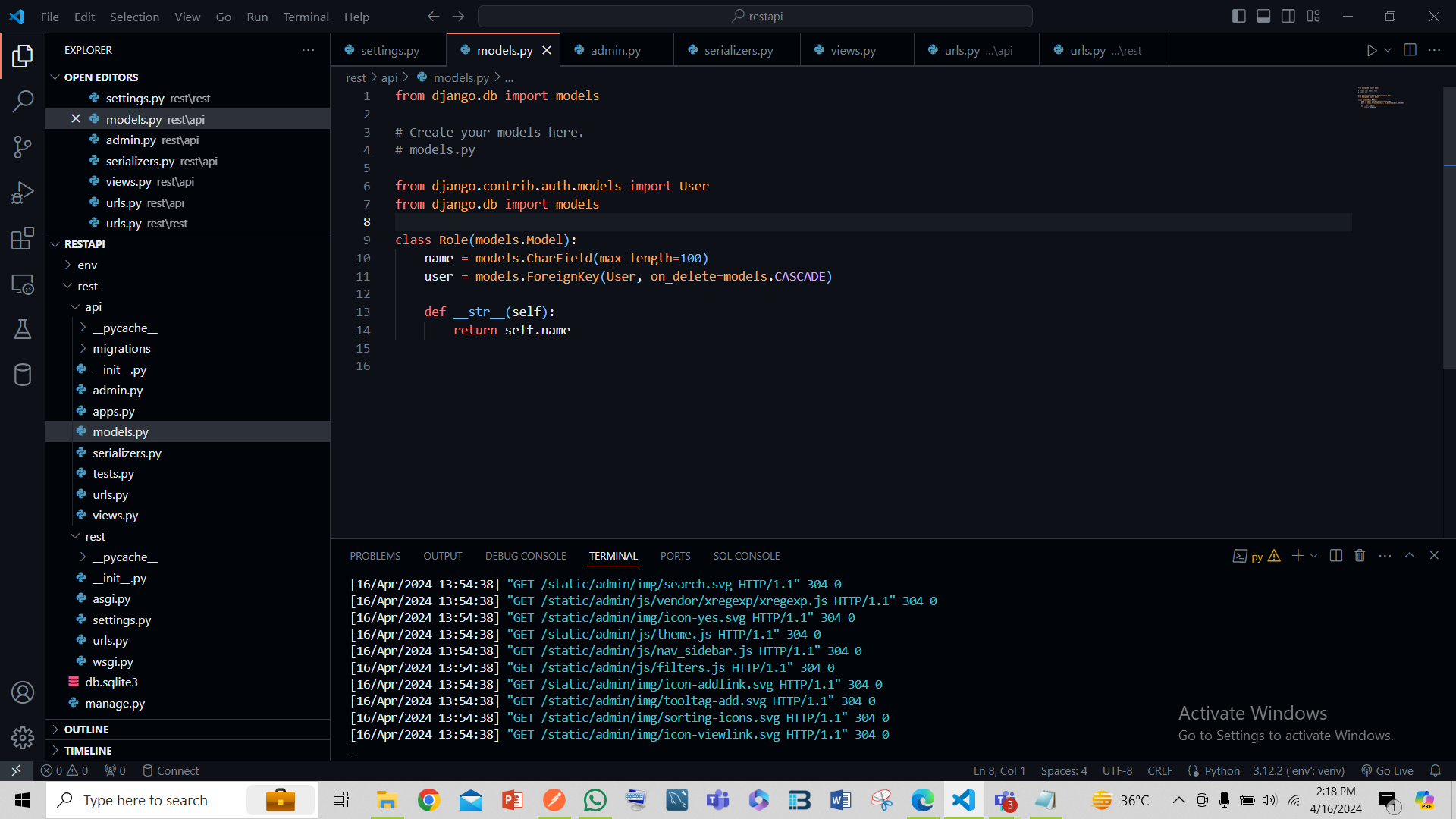
3. \*\*Uniform Interface\*\*: RESTful APIs have a uniform interface, which means that the same set of HTTP methods (GET, POST, PUT, DELETE, etc.) are used to manipulate resources. Additionally, resources are identified using URIs (Uniform Resource Identifiers).

4. \*\*Resource-Based\*\*: RESTful APIs treat data as resources that can be accessed and manipulated using standard HTTP methods. Each resource is represented by a unique URI, and clients interact with resources through these URIs.

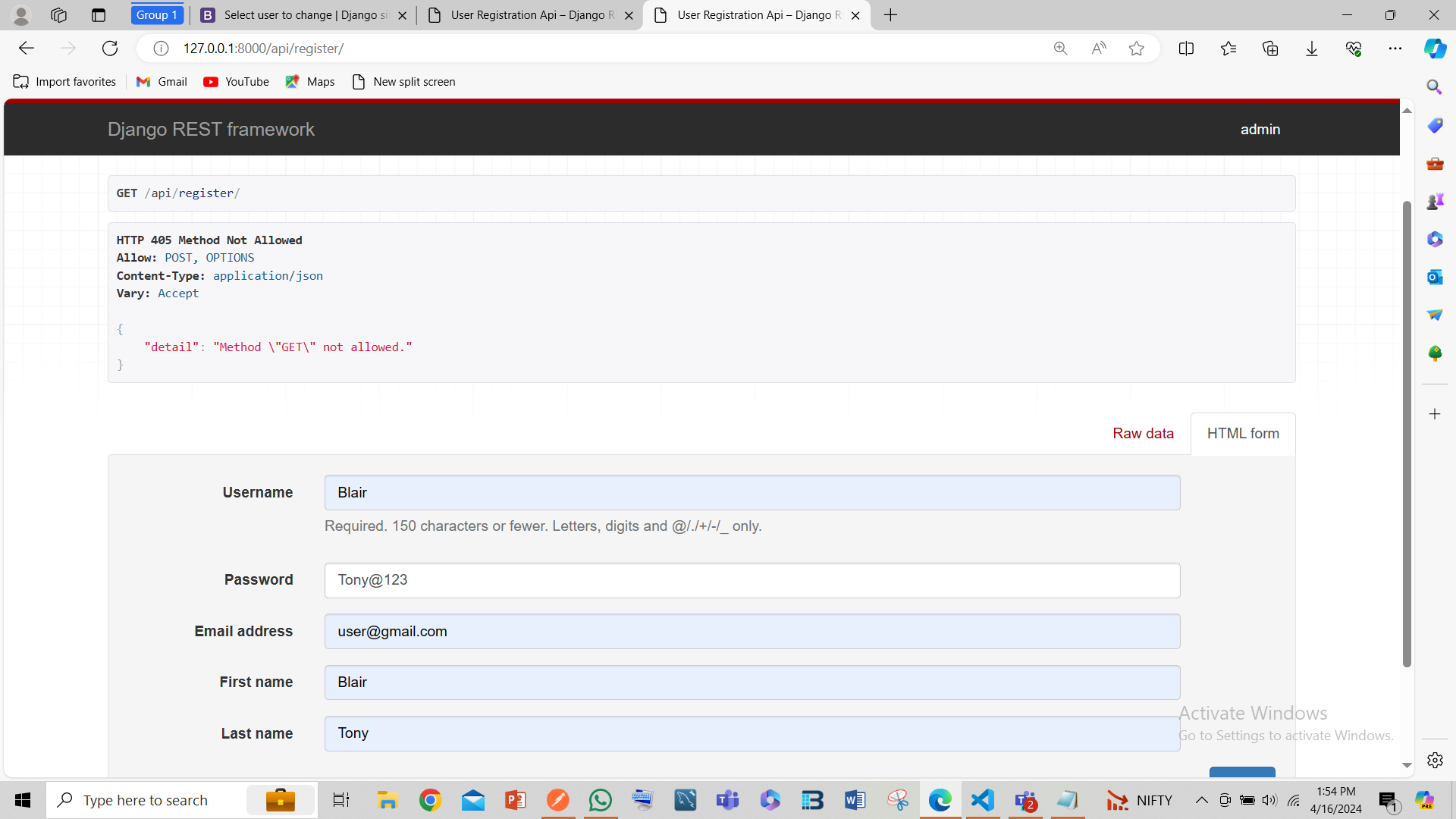
5. \*\*Representation\*\*: Resources can have multiple representations, such as JSON, XML, HTML, or others. Clients can specify the desired representation using HTTP headers, and servers respond with the appropriate representation.

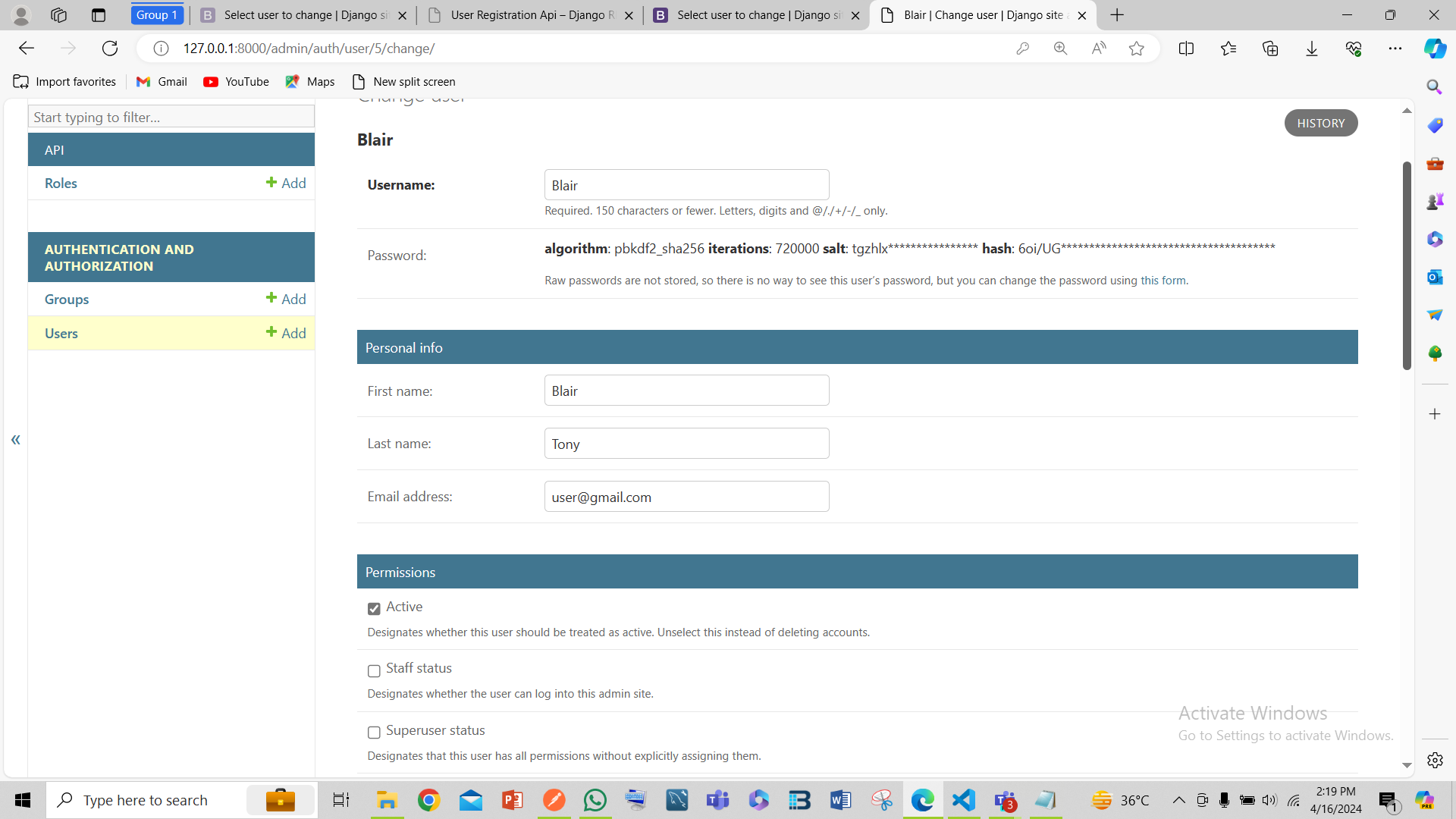
6. \*\*Stateless Communication\*\*: RESTful APIs communicate in a stateless manner, meaning that each request from a client to the server must contain all the information necessary to understand and fulfill the request. The server does not store any client state between requests, which simplifies server implementation and improves scalability.

**3) Create a model role and link with the user model**



4) **Create a api to register user**

****

****