**Comparison between SpringBoot and Django**

High level component explanation:

1. Models.py – defines the databases tables using Python classes. It is similar to POJO class with @Entity and @Table annotations to define direct relation between DB and class.
2. Serializers.py – converts Django models ⬄ JSON for APIs. This is similar to DTO classes in Java which are used to get/send JSON api request/response. DataTransferObjects.
3. Views.py – This contains all the method that handles the hits of the URL. In Java, these are known as controllers.
4. Urls.py – maps each url to the correct view like GET, POST, PUT. Unlike python, Java directly maps Views(Controllers) and Urls(GetMapping, PostMapping).
5. Settings.py – Configuration for Databases, apps, authentication, etc. In java this file is know as application.properties where all the configurations are done.

Models.py

1. User

Explanation:

* AbstractUser: we are customizing Django’s built-in User model
* email: we make email required + unique (no two users with same email)
* profile\_pic: allows image upload; stored in /media/profile\_pics/
* phrase: a secret phrase that user must enter during login (extra security)

1. Message

Explanation:

* sender / receiver: links to the User table using a foreign key
* content: the actual message text
* attachment: if the message has a file (image, PDF, etc.)
* timestamp: date/time of message creation (set automatically)
* session\_id: unique ID for each login session, so we can delete messages for that session during logout

Serializers.py

Serializers convert Django models to JSON for APIs and vice versa.

1. Register Serializer: Handles user registration.
2. LoginWithPhraseSerializer: Validates username, password, and phrase during login.
3. Message Serializer: Used to send/receive chat message data through the API.

Views.py

This is where we define logic for each API (like what to do when a user registers, logs in, etc.).

* + Register View
  + post(): handles POST requests like user signup
  + serializer.save() => creates user if valid
  + LoginWithPhraseView
  + Validates login using username + password + phrase
  + Returns JWT tokens (access, refresh) for secure session handling
  + MessageView
  + GET: fetches all messages received by the current user
  + POST: sends a new message (sender is auto-filled from logged-in user)
  + LogoutView
* On logout, we delete all messages from this login session (identified by session\_id)

Urls.py

Just maps URLs like /api/accounts/register/ to the correct view function.

urlpatterns = [  
 path('register/', RegisterView.as\_view(), name='register'),  
 path('login/', LoginWithPhraseView.as\_view(), name='login'),  
 path('messages/', MessageView.as\_view(), name='messages'),  
 path('logout/', LogoutView.as\_view(), name='logout'),  
]