Django Training - 6. Django Extensions, Knox Token Authentication

Materials

1. django-extensions is a collection of custom extensions for the Django Framework. These include management commands, additional database fields, admin extensions and much more. For the purpose of the training we'll focus on management commands, namely, we'll focus on the commands that django-extensions will add to our `manage.py` program.
    - shell_plus usage: `python manage.py shell_plus`
        - `manage.py shell_plus` is like `manage.py shell`, but it saves you time by automatically importing all the defined models everytime you run the shell, in addition to other neat interactive features.
    - reset_db usage: `python manage.py reset_db`
        - `reset_db` resets your Django database, removing all data from all tables. This allows you to run all migrations again.
2. DRF TokenAuthentication
3. Knox TokenAuthentication
4. Extending the existing User model

Task

1. Remove any apps/views that we created from before that have to do with authenticating users
2. Create an app `users`
3. In the `users` app, extend Django's user model by inheriting from `AbstractUser` to include an optional `bio` `CharField` with a max length of 256 characters
    - On django admin, this field should be displayed as a `TextArea`
    - You might want to delete all the migration files you have and leverage `reset_db` shell command to extend the user model. Why?
    - *May I delete the migration files when working on a project?* **NEVER** delete the migrations files without consulting your team first, migrations files serve as version control for your models, sometimes you might want to set a model's state to a snapshot from that past, by deleting migration files you lose access to that. In addition to that, not all migration files can be automatically generated, there are manually created migration files like data migration files, if you delete data migration files you're deleting code from the codebase.
    - *Why are we allowed to delete migration files now then?* This purely for the purpose of this task.
4. Create an app `authentication`
5. In the `authentication` app, support a `POST authentication/register/` endpoint that creates users.
    - Think about the suitable permission class(es) for this endpoint.
    - This endpoint must accept the following fields formatted in JSON:
        - username
        - email
        - password1
        - password2 (confirmation of `password1`)
    - Perform proper validation on all fields **including** letting the user know if their password isn't strong enough or if password1 doesn't match password2.
    - Make sure passwords are being hashed and email domains are stored in lowercase. (hint: use `create_user`)
6. Create a `POST authentication/login/` that logs in users using their username and password and returns a `KnoxToken` and the user's data in a nested object.

```
{
    "token": <knox_token>
    "user": {
        "id": 1,
        "username": "my_user"
        "email": "email@email.com"
        "bio": "my sample bio"
}
```

6. Create a `POST authentication/logout/` endpoint that logs the user out from the app by invalidating the knox token
7. In the `users` app, create a user detail endpoint `/users/<pk>` that supports the following requests:
    - `GET` returns the user data matching the given `pk`, namely, it should return the user's `id`, `username`, `email`, and `bio`.
        - return 404 status code if the user with the given `pk` does not exist
    - Support updating the `bio`, `username`, and `email` fields via the following requests:
        - `PUT` This is exactly the same as when creating a user except that an ID of an existing user is provided in the URL, and that the request will overwrite the user's data with that given ID.
        - `PATCH` This is exactly the same as when updating a user except none of the fields are required, and that only fields given a value will be updated. (hint: see `partial_update` in serializers)
        - Allow update requests if the user making the request is the user in the `<pk>` of the url.
8. Add `TokenAuthentication` to the default authentication classes

Guidelines

1. Use this gitignore template
2. List your environment variables (if any) in a `.env.example` file
3. Don't forget to run `manage.py makemigrations` and `python manage.py migrate`
4. You are allowed to refer to any articles, documentation source, questions on StackOverFlow
5. You are not allowed to consult other people through any medium of communication
6. Your grade will be affected if your code isn't clean, you should maintain code cleanliness as much as you can