Assignment 1, backend for high load

Put all deliverables into github repository in your profile. Share link to google form 24 hours before defense. Defend by explaining deliverables and answering questions.
Deliverables: code
Google form:
https://docs.google.com/forms/d/e/1FAIpQLSe0GyNdOYIvM1tX_I_CtIPod5jBf-ACLGdHYZq1gVZbUeBzIg/viewform?usp=sf_link

Exercise 1: Setting Up Your Django Project

1. **Objective**: Create a new Django project and set up your virtual environment.
   - Create a new directory for your Django project.
   - Set up a virtual environment.
   - Install Django in your virtual environment.
   - Create a new Django project named `my_blog`.
   - Run the server to ensure everything is set up correctly.
2. **Expected Outcome**: A Django project named `my_blog` should be running locally.

Exercise 2: Creating a Blog App

1. **Objective**: Create a new Django app named `blog` within your project.
   - Create the `blog` app using Django's `startapp` command.
   - Add the `blog` app to the `INSTALLED_APPS` list in `settings.py`.
   - Create a basic view in the `blog` app that returns a simple "Hello, Blog!" message.
   - Map this view to a URL in `urls.py`.
2. **Expected Outcome**: When you navigate to the blog's URL, you should see "Hello, Blog!" displayed in your browser.

Exercise 3: Creating Blog Models

1. **Objective**: Define the models for your blog posts.
   - Create a `Post` model in `models.py` with the following fields: `title`, `content`, `author`, `created_at`, and `updated_at`.
   - Add a `__str__` method to the model to return the title of the post.
   - Create and run migrations to apply the model to the database.
2. **Expected Outcome**: The `Post` model should be successfully created and applied to the database.

Exercise 4: Admin Interface for Blog Posts

1. **Objective**: Register your `Post` model with the Django admin interface.
   - In `admin.py`, register the `Post` model.
   - Customize the admin interface to display the title, author, and created date in the list view.
   - Add search functionality in the admin to search posts by title.
2. **Expected Outcome**: You should be able to manage blog posts through the Django admin interface.

Exercise 5: Creating Blog Views

1. **Objective**: Create views to list and display blog posts.
   - Create a view that lists all blog posts.
   - Create a detail view that displays the content of a single blog post based on its ID or slug.
   - Create corresponding templates for the list and detail views.
2. **Expected Outcome**: You should be able to view a list of all blog posts and click on individual posts to view their details.

Exercise 6: Adding URLs for Blog Views

1. **Objective**: Map the views you created to URLs.
   - Create URL patterns for the blog list and blog detail views.
   - Use Django's `path()` function to create dynamic URLs for individual blog posts.
2. **Expected Outcome**: The blog should have functioning URLs that display the list of posts and the details of each post.

Exercise 7: Creating a Blog Post Form

1. **Objective**: Allow users to create new blog posts through a form.
   - Create a `PostForm` in `forms.py`.
   - Create a view that displays the form and handles form submissions.
   - Update your templates to include the form for creating a new blog post.
2. **Expected Outcome**: Users should be able to create new blog posts through the web interface.

Exercise 8: Editing and Deleting Blog Posts

1. **Objective**: Add functionality to edit and delete blog posts.
   - Create views for editing and deleting blog posts.
   - Create corresponding templates for editing and deleting posts.
   - Ensure that only the author of the post can edit or delete it.
2. **Expected Outcome**: Users should be able to edit and delete their blog posts.

Exercise 9: User Authentication and Authorization

1. **Objective**: Add user authentication to your blog.
    - Implement user registration, login, and logout functionality.
    - Restrict access to creating, editing, and deleting posts to logged-in users.
    - Use Django's `User` model to associate posts with users.
2. **Expected Outcome**: Only logged-in users can create, edit, or delete blog posts.

Exercise 10: Adding Pagination

1. **Objective**: Implement pagination for the list of blog posts.
    - Modify the list view to include pagination.
    - Update the template to display page navigation controls.
2. **Expected Outcome**: The blog post list should be paginated, displaying a set number of posts per page.

Exercise 11: Adding Comments to Blog Posts

1. **Objective**: Allow users to add comments to blog posts.
    - Create a `Comment` model with fields for the comment text, author, post, and created date.
    - Create a form and view to handle comment submission.
    - Display comments below each blog post in the detail view.
2. **Expected Outcome**: Users should be able to leave comments on blog posts.