Django Developer Assignment

Objective: Develop specific functionalities using Django as outlined in our assignment document.

This includes creating API endpoints and implementing backend logic

Core features

1. 1. **Understand the Requirements**: Review the assignment document carefully to understand the specific functionalities you need to implement with API endpoints.
2. **Set Up Your Django Project**: If you haven't already, create a Django project using `django-admin startproject project_name`.
3. **Create Django Apps**: Divide your project into apps based on the functionalities they serve. You can create a new app using `python manage.py startapp app_name`.
4. **Define Models**: Define Django models that represent your data. Ensure that your models capture all the necessary fields and relationships required for your functionalities.
5. **Write Migrations**: After defining your models, create database migrations using `python manage.py makemigrations` and apply them using `python manage.py migrate`.
6. **Implement Business Logic**: Write Python code to implement the backend logic of your application. This includes any calculations, data manipulation, or other operations required by your application.
7. **Install Django REST Framework (DRF)**: If you haven't already, install Django REST Framework using pip:
8. **Create Serializers**: Define serializers to convert Django model instances to JSON and vice versa. Serializers in DRF help you to control the representation of your data.
9. **Create Views for API Endpoints**: Define views for your API endpoints. You can use class-based views provided by DRF or write function-based views. Ensure that your views handle different HTTP methods (GET, POST, PUT, DELETE) as required.
10. **Set Up URL Patterns for API Endpoints**: Map URLs to your API views by configuring URL patterns in your Django app's `urls.py` file. You can use Django's `path` or `re_path` functions to define URL patterns.
11. **Implement Authentication and Authorization (if required)**: Decide on the authentication and authorization mechanisms for your API endpoints. DRF provides various authentication classes and permission classes that you can use to secure your endpoints.
12. **Test Your API Endpoints**: Use tools like Postman, cURL, or Django's built-in test client to test your API endpoints and ensure they behave as expected. Test different scenarios, including success cases and error cases.
13. **Handle Errors and Edge Cases**: Make sure your API endpoints handle errors gracefully and cover edge cases to provide a good user experience. Use appropriate HTTP status codes and error messages in your responses.
14. **Document Your API**: Document your API endpoints, including their URLs, input parameters, output formats, and any authentication requirements. You can use tools like Swagger or Django Rest Swagger for API documentation.
15. **Review and Refactor**: Review your code to ensure it follows best practices and refactor it if necessary to improve readability, performance, or maintainability.
16. **Deploy Your Application**: Once your application is ready, deploy it to a server using a platform like Heroku, AWS, or DigitalOcean, and configure it for production use.

By following these steps, you should be able to develop specific functionalities using Django with API endpoint.Django REST Framework documentation for more detailed information on building APIs with DRF.

Deliverables:

1. **Source Code**: Submit the source code of your Django project, including all necessary files such as models, views, serializers, URLs, and tests.
2. **Documentation**: Provide documentation for your API endpoints. This documentation should include details about each endpoint, such as its purpose, URL, request and response formats, and any authentication requirements.
3. **Test Cases**: Include test cases to demonstrate that your API endpoints work correctly. Ensure that your test cases cover different scenarios and edge cases.

Submission Guidelines:

1. **Deadline**: Ensure that you submit your assignment before the deadline specified in your assignment document.
2. **Format**: Submit your assignment in a format specified by your instructor or assignment guidelines. This may include submitting a zip file containing your project source code and documentation, or using a version control system like Git and providing a link to your repository.
3. **Instructions**: Follow any specific instructions provided in your assignment document regarding submission format, naming conventions, or any other requirements.
4. **Contact Information**: Provide your contact information in case your instructor needs to reach out to you regarding your submission.
5. **Plagiarism**: Make sure that your submission is your original work and does not contain any plagiarized code or content. Cite any external sources used in your documentation or code comments.