Exercise: Building a Task Management API with User Authentication, JWT, Celery, and Redis

Objective:

Build a Task Management API using Django REST Framework that allows users to create, update, and delete tasks. Each task will have a title, description, status, and assigned user. Implement user authentication using Django's built-in User model and JWT authentication for secure API access. Use Celery for asynchronous email notifications when the task status changes and Redis for caching task data.

Instructions:

1. Set up a new Django project and create a new Django app called "tasks".

2. Use Django's built-in User model for authentication. Configure the User model to include additional fields like email if required.

3. Implement JWT authentication in the API. Use the `djangorestframework-simplejwt` package to handle JWT authentication.

4. Create a Task model with the following fields:

   - title (CharField)

   - description (TextField)

   - status (CharField with choices: "TODO", "IN_PROGRESS", "DONE")

   - assigned_user (ForeignKey to the User model)

5. Implement serializers for both the Task and User models.

6. Create a DRF ViewSet for the Task model and configure it to use the serializers.

7. Implement a custom permission class that allows only authenticated users to access the API.

8. Configure the DRF router to handle the TaskViewSet.

9. Implement Celery and Redis for asynchronous email notifications:

- Create a Celery task called `send_task_status_email` that takes a task ID as an argument. This task will handle sending email notifications when the task status changes to "DONE".

- Configure Celery to use Redis as the message broker.

- In your Task model, override the `save` method to check if the task status has changed to "DONE". If it has, enqueue the `send_task_status_email` Celery task with the task ID.

10. Implement Redis-based caching for the Task API endpoints to improve performance:

- Configure Redis as the cache backend in your Django project settings.

- Use the `cache_page` decorator from Django's cache framework to cache the Task API endpoints.

11. Test the API using tools like Postman to ensure all CRUD operations are working correctly. Verify that only authenticated users can access the API endpoints. Check if email notifications are sent when the task status changes to "DONE" asynchronously using Celery.

12. Measure the API response times before and after implementing caching to verify its effectiveness.

13. Document the API endpoints, including their parameters and responses. Explain the JWT authentication mechanism, Celery for asynchronous email notifications, and Redis-based caching and their benefits.

14. Prepare a presentation to explain the implementation details and demonstrate the functionality of the Task Management API.

Note: Make sure to install and configure Celery, Redis, Django's email settings, Django's authentication system, Django REST Framework, and the `djangorestframework-simplejwt` package in your Django project before starting the exercise.