Task Description

You are required to build a basic **Library Management System** API using Django, Django REST Framework (DRF), and Celery. The system should allow users to manage books and authors, handle book borrowing records, and include a background task for generating periodic reports on library activity.

Requirements

1. Models:

- **Author:**
  - `id`: Auto-generated (Primary Key)
  - `name`: CharField (max_length=255)
  - `bio`: TextField (optional)

- **Book:**
  - `id`: Auto-generated (Primary Key)
  - `title`: CharField (max_length=255)
  - `author`: ForeignKey to `Author`
  - `isbn`: CharField (unique, max_length=13)
  - `available_copies`: IntegerField (default=0)

- **BorrowRecord:**
  - `id`: Auto-generated (Primary Key)
  - `book`: ForeignKey to `Book`
  - `borrowed_by`: CharField (max_length=255)
  - `borrow_date`: DateField (auto_now_add=True)
  - `return_date`: DateField (nullable)

---

2. API Endpoints:

1. **Authors**
   - `GET /authors/` - List all authors
   - `POST /authors/` - Create a new author
   - `GET /authors/<id>/` - Retrieve a specific author
   - `PUT /authors/<id>/` - Update a specific author
   - `DELETE /authors/<id>/` - Delete a specific author

2. **Books**
   - `GET /books/` - List all books
   - `POST /books/` - Add a new book
   - `GET /books/<id>/` - Retrieve a specific book

- `PUT /books/<id>/` - Update a specific book
- `DELETE /books/<id>/` - Delete a specific book

3. **Borrow Records**
   - `POST /borrow/` - Create a new borrow record (reduce the `available_copies` of the book by 1 if copies are available)
   - `PUT /borrow/<id>/return/` - Mark a book as returned (set `return_date` and increase `available_copies` by 1)

4. **Reports**

   - `GET /reports/` - Retrieve the latest report generated by the background task.
   - `POST /reports/` - Generate new report using celery delay function in background.
   -

---

3. Background Task with Celery:

- Implement a Celery task that runs on api call POST /reports/
- The report should include:
  - Total number of authors.
  - Total number of books.
  - Total books currently borrowed.
- Save the report in a JSON file and store it in a directory named `reports/`. The file should have a timestamp in its name (e.g., `report_YYYYMMDD.json`).
- The latest report should be accessible via the GET `/reports/` endpoint.

Optional Bonus Features

- Include proper error handling and meaningful response messages for all operations.
- Write **unit tests** for at least 2 endpoints of your choice.
- Use **Git** for version control and share the repository link. Ensure clear commit messages.
- API Documentation using **Swagger** or **DRF-YASG**.
- Dockerize the application for easy deployment.
- Add user authentication (JWT or session-based) for secured endpoints.

Submission Guidelines

1. Submit your GitHub repository link with
    - a README file with:
    - Steps to run the project (including Celery setup).
    - Brief explanation of your approach.
    - API documentation (if implemented).
2. Deployed link of application on heroku or similar paltform with
    - Admin username and password

Evaluation Criteria

- Code Quality: Readability, structure, and adherence to Django best practices.
- Functionality: The assignment must meet the requirements and handle edge cases effectively.
- Background Tasks: Proper integration of Celery and report generation.
- Test Coverage: Presence and quality of unit tests.
- Bonus Points: Implementation of optional features or additional improvements.