Loan Management System REST API - Django

Objective

The Loan Management System is a Django-based REST API designed to manage loans with **user-defined monthly compound interest**. The system provides role-based authentication, automatic interest calculations, and loan repayment schedules. It supports loan foreclosure, allowing users to settle loans before tenure completion with adjusted interest calculations. The system is built using Django and Django REST Framework (DRF), with authentication managed via JWT and OTP verification through Nodemailer.

Tech Stack

- **Backend:** Django, Django REST Framework
- **Authentication:** JWT (Simple JWT)
- **OTP Email Service:** Nodemailer (via SMTP)
- **Database:** PostgreSQL (Preferred) or MongoDB (Optional with Djongo)
- **Deployment:** Render (Free Tier)

Features & Requirements

1. Authentication & Role-Based Access Control

- Implement JWT authentication using Simple JWT.
- Support two user roles: **Admin & User**.
- Register users with OTP email verification.
- Each API call must include a valid JWT token in the request header.
- Determine role (**Admin/User**) from token and allow/restrict access accordingly.

2. Core Loan Features

Users can:

- Add a new loan by specifying the amount, tenure, and interest rate.
- View their active and past loans.
- View loan details with monthly installments and interest breakdown.
- Foreclose a loan before tenure completion (adjusted interest calculations apply).

Admins can:

- View all loans in the system.
- View all user loan details.

- Delete loan records.

3. Loan Calculation

- **User-defined yearly compound interest.**
- Monthly installments and total payable amount are calculated automatically.
- Foreclosure allows users to pay off the loan early with adjusted interest.
- The system stores total amount payable, interest amount, and payment schedules.

API Endpoints & Examples

1. Add Loan

POST /api/loans/

Request:

```
{
    "amount": 10000,
    "tenure": 12,
    "interest_rate": 10  // User-defined yearly interest percentage
}
```

Success Response:

```
{
    "status": "success",
    "data": {
        "loan_id": "LOAN001",
        "amount": 10000,
        "tenure": 12,
        "interest_rate": "10% yearly",
        "monthly_installment": 879.16,
        "total_interest": 1549.92,
        "total_amount": 11549.92,
        "payment_schedule": [
            {
                "installment_no": 1,
                "due_date": "2024-03-24",
                "amount": 879.16
            }
            // ... remaining installments
        ]
    }
}
```

2. List Loans

GET /api/loans/

User Response:

```
{
    "status": "success",
    "data": {
        "loans": [
            {
                "loan_id": "LOAN001",
                "amount": 10000,
                "tenure": 12,
                "monthly_installment": 879.16,
                "total_amount": 11549.92,
                "amount_paid": 1758.32,
                "amount_remaining": 9791.60,
                "next_due_date": "2024-04-24",
                "status": "ACTIVE",
                "created_at": "2024-02-24T10:30:00Z"
            }
        ]
    }
}
```

3. Loan Foreclosure

POST /api/loans/{loan_id}/foreclose/

Request:

```
{
    "loan_id": "LOAN001"
}
```

Success Response:

```
{
    "status": "success",
    "message": "Loan foreclosed successfully.",
    "data": {
        "loan_id": "LOAN001",
        "amount_paid": 11000.00,
        "foreclosure_discount": 500.00,
        "final_settlement_amount": 10500.00,
        "status": "CLOSED"
    }
}
```

Deployment Instructions (Render Free Tier)

1. **Set Up GitHub Repository**
2. **Create Render Web Service:**
   - Choose Python runtime
   - Set start command: `gunicorn loan_management.wsgi:application`

Add environment variables:
DATABASE_URL=<Render PostgreSQL URL>
SECRET_KEY=<your-secret-key>
DEBUG=False

○

3. **Set Up PostgreSQL Database** on Render
4. **Deploy the Web Service**

Evaluation Criteria

- Proper implementation of JWT authentication & role-based access
- Correct compound loan calculations with user-defined yearly interest
- Proper API response handling & validation
- Database design & query optimization
- Code cleanliness, structure, and documentation
- Successful deployment on Render

Validation Rules

Amount Validation

- Minimum: ₹1,000
- Maximum: ₹100,000
- Must be a number

Tenure Validation

- Minimum: 3 months
- Maximum: 24 months
- Must be a whole number

Interest Calculation

For a loan of ₹10,000 for 12 months at **10% yearly compound interest**:

1. **Yearly Interest Rate** = 10%
2. **Monthly Interest Rate** = 0.833% (10% ÷ 12)
3. **Total Interest** = ₹1,549.92
4. **Total Amount** = ₹11,549.92
5. **Monthly Installment** = ₹879.16

   EMI Calculator - Calculate EMI on Home, Car and Personal Loans

Submission Guidelines

- Share GitHub repository link
- Provide Postman/Thunder Client collection for testing APIs
- Deploy on Render (Free Tier) and share live API URL
- Include README.md with setup steps

Important Notes for Developers

1. **Authentication:**
   - Always validate JWT tokens
   - Implement proper role checks
   - Handle token expiry
2. **API Response Format:**
   - Always use a consistent response structure
   - Include proper status codes
   - Handle errors gracefully
3. **Code Organization:**
   - Use proper Django project structure
   - Follow PEP 8 guidelines
   - Add comments for complex logic
4. **Testing:**
   - Test all API endpoints

- Verify calculations
- Test error scenarios

5. **Documentation:**
    - Document API endpoints
    - Include setup instructions
    - Add code comments

Expected Deliverables

1. **GitHub Repository containing:**
    - Complete source code
    - README.md with setup instructions
    - requirements.txt
    - Postman/Thunder Client collection
2. **Live API URL** on Render
3. **Documentation:**
    - API endpoints
    - Setup guide
    - Testing instructions