Course Management System (Django-based)

📘 Project Overview

A Django-based Course Management System that allows users to:

- Browse and purchase online courses
- Access lecture content and course materials
- View and buy books
- Use an AI-powered chatbot
- Contact support
- Make secure payments

The system supports login, enrollment, and transaction management. It is collaboratively developed by four team members (referred to as A, B, C, D), using GitHub for version control and collaboration.

👥 Team Structure & Git Workflow

- **Team Members:** A (admin/owner), B, C, D
- **Branching Strategy:** Each member works in their own branch.
- **Rules Implemented:**
    - Only member A can merge into the main branch.
    - Members cannot push to each other's branches.
    - All pull requests are reviewed and merged only by A.
    - Default branch is main.

🔐 Security Measures (Issues Raised on GitHub)

1. **Enforce HTTPS in Production**
2. **Configure CSRF Protection.** (Future)
3. **Implement RBAC (Role-Based Access Control)**

4. **Enable Strong Passwords**
5. **JWT-based Authentication**
6. **Disable Debug Mode in Production.** (Future)
7. **Rate Limiting with django-axes.** (Future)
8. **Secure File Uploads.** (Future)
9. **Dependency Vulnerability Audit.** (Future)
10. **CSP & Security Headers.** (Future)

☁️ **Cloud Integration (GitHub Issues)**

1. Dockerize the Project
2. Deploy on AWS/GCP EC2
3. Setup CI/CD with GitHub Actions
4. Migrate DB to Cloud (RDS/Cloud SQL) (Future)
5. Use Object Storage (S3, GCS) for media (Future)
6. Add Load Balancer for traffic distribution. (Future)
7. Enable Auto-scaling (Future)
8. Enable Monitoring and Alerts (CloudWatch/Stackdriver). (Future)
9. Integrate CDN (CloudFront/Cloudflare). (Future)
10. Write Infrastructure as Code (Terraform). (Future)

🔑 **Authentication with JWT**

Implemented JWT-based authentication to secure API access. JWTs are generated after successful login and include expiration timestamps for secure session handling.

Example (Python, framework-agnostic):

import jwt, datetime

```
SECRET_KEY = "secret123"

def generate_jwt(user_id):
    payload = {
        "user_id": user_id,
        "exp": datetime.datetime.utcnow() + datetime.timedelta(minutes=5)
    }
    token = jwt.encode(payload, SECRET_KEY, algorithm="HS256")
    return token
```

🔄 Git Commands Used

```
# Create and switch to new branch
git checkout -b <branch-name>

# Push to remote
git push origin <branch-name>

# Set merge permissions and branch protections on GitHub repo
# via Repository Settings → Branches → Protection Rules
```

📦 Modules & Features

- **Courses:** CRUD operations, purchase & enroll
- **Books:** View & purchase educational books
- **Lecture:** Access video content and materials
- **AI Chatbot:** For course guidance and support
- **Payments:** Razorpay/Stripe integration (API-based)

- **User Roles:** Admin, Instructor, Student
- **Authentication:** JWT-based

📊 Deployment Considerations

- **Web Server:** Gunicorn
- **Reverse Proxy:** Nginx
- **SSL Certificate:** Let's Encrypt
- **Media Storage:** AWS S3 / GCP Storage
- **Database:** PostgreSQL/MySQL (Cloud Managed)
- **Logging:** Sentry / CloudWatch

🎯 Target Audience

- Students
- Instructors
- Institutions

📁 Tech Stack

- **Backend:** Django (Python)
- **Frontend:** HTML, CSS, JS (Vanilla or React)
- **Database:** MySQL/PostgreSQL
- **DevOps:** Docker, GitHub Actions
- **Cloud:** AWS / GCP / Azure

📬 Contact

For collaboration, feature requests or bug reports, open an issue or contact the owner (A) via GitHub.