course-management-backend

Course Management Platform - Backend Service

This repository contains the backend service for a multi-feature Course Management Platform. The system is designed to support academic institutions by managing course allocations, tracking facilitator activities, and providing a secure, role-based API for all operations. This project fulfills the requirements of the summative assessment, demonstrating proficiency in Node.js, Express, Sequelize, MySQL, Redis, and RESTful API design.

Live Demo & Walkthrough

- Video Walkthrough: [PASTE YOUR 5-MINUTE VIDEO LINK HERE]
- Hosted i18n Frontend: [PASTE YOUR GITHUB PAGES LINK HERE]

Features

- Role-Based Access Control (RBAC): Secure authentication using JSON Web Tokens (JWT) with distinct roles and permissions for Managers and Facilitators.
- Course Allocation System (Full CRUD):
    - Managers can Create, Read, Update, and Delete course allocations.
    - Facilitators have read-only access to their *own* assigned courses.
- Facilitator Activity Tracker (FAT) (Full CRUD):
    - Facilitators can Create, Update, and Delete their weekly activity logs.
    - Optional file uploads are supported for log submissions.
    - Managers have read-only access to all logs.
- Asynchronous Notification System:
    - Utilizes a Redis-backed message queue for robust, non-blocking notifications.
    - Immediate Alerts: Managers are alerted via the worker console when a facilitator submits or updates a log.
    - Scheduled Reminders: A `node-cron` background job runs weekly to check for missing logs and queues reminders for facilitators.

- Comprehensive API Documentation: Live, interactive API documentation is available via Swagger UI.

Technology Stack

Category	Technology
Backend Framework	Node.js, Express.js
Database	MySQL (Hosted on Railway)
ORM	Sequelize
Authentication	JSON Web Tokens (JWT), bcryptjs
Task Queuing	Redis (Hosted on Upstash)
File Uploads	Multer
Scheduled Jobs	Node-Cron
Testing	Jest
API Documentation	Swagger (swagger-jsdoc, swagger-ui-express)

Setup and Installation

Follow these steps to set up the project locally.

Prerequisites

- Node.js (v18 or higher)
- An active internet connection (for connecting to cloud databases)

1. Clone the Repository

```
git clone https://github.com/your-username/your-repo-name.git
cd course-management-backend
```

2. Install Dependencies

```
npm install
```

3. Configure Environment Variables

Create a `.env` file in the root directory. This project is configured to use free-tier cloud services for the database and message queue. You will need to create your own free accounts at Railway (for MySQL) and Upstash (for Redis) and paste your connection URLs into the `.env` file.

```
# Server Configuration
PORT=3000

# Online Database URLs (from Railway and Upstash)
DATABASE_URL="mysql://..."
UPSTASH_REDIS_URL="rediss://..."

# JWT Configuration
JWT_SECRET=your_super_secret_key_here
JWT_EXPIRES_IN=1d
```

4. Start the Application

This project requires two separate processes to be run in two terminals.

Terminal 1: Start the main API server:

```
npm run dev
```

You should see output confirming the server is running and the database is synced.

Terminal 2: Start the background notification worker:

```
npm run worker
```

You should see output confirming the Redis client is connected and the cron job is scheduled.

API Documentation

Once the server is running, a complete and interactive API documentation is available via Swagger UI at:

http://localhost:3000/api-docs

Authentication Flow

1. Use the `POST /api/auth/register` endpoint to create a Manager and a Facilitator user.
2. Use the `POST /api/auth/login` endpoint with a user's credentials to receive a JWT token.
3. Click the Authorize button in the Swagger UI, type `Bearer` (with a space), and paste the token.
4. You can now access the protected (padlocked) endpoints corresponding to that user's role.

Author:Lina Iratwe