University of Leeds **School of Computer Science**

COMP3011, 2024-2025

Web Services and Web Data

A RESTful API for Rating Professors

By

Shasha Nabila Binti Al Malek Faisal

sc22snba@leeds.ac.uk

201588070

Date: 10/3/2025

1. The Database

The database for the **Professor Rating Service** is implemented using **Django's ORM**, ensuring efficient data management and enforcement of constraints. The schema consists of three core tables: **Professor, Module, and Rating**, along with the built-in **User** table from Django for authentication.

Tables & Fields

1. Professor Table

Stores details of professors, ensuring each has a unique identifier and name.

Field	Data Type	Constraint
identifier	Char (3)	Primary Key, Unique, Validates 3 characters
name	Char (255)	Unique, Stores professor name

2. Module Table

Represents university modules, including the year and semester they are taught.

Unique Constraint: (code, year, semester) ensures a module is unique for a given academic term.

Field	Data Type	Constraint
code	Char (3)	Validates 3 characters
name	Char (255)	Module name
year	Integer	Validates 4-digit format (e.g., 2023)
semester	Integer	Validates values 1 or 2
professors	ManyToMany	Links multiple professors to a module

3. Rating Table

Stores ratings given by students for professors in specific module instances.

Unique Constraint: (user, professor, module, year, semester) ensures a user rates a professor only once per module instance.

Field	Data Type	Constraint
user	Foreign Key	Links to Django's User model
professor	Foreign Key	Links to Professor (deletes ratings if professor is removed)
module	Foreign Key	Links to Module
year	Integer	Must match the module's year
semester	Integer	Must match the module's semester
score	Integer	Allowed values: 1-5

Relationships Between Tables

- **Professor - Module (Many-to-Many):** A professor can teach multiple modules, and a module can have multiple professors.
- **User - Rating (One-to-Many):** Each user can rate multiple professors but only once per module instance.
- **Professor - Rating (One-to-Many):** A professor can have multiple ratings.
- **Module - Rating (One-to-Many):** A module instance can have multiple ratings.

2. The API

The API is developed using **Django** and **Django REST Framework**, providing functionalities for user registration, authentication, retrieving module and professor details, rating professors, and viewing average ratings. The API follows **RESTful principles**, ensuring consistency and usability.

API Endpoints

The API exposes several endpoints to handle different functionalities. The table below summarizes each endpoint, including its purpose, URL, HTTP method, request/response structure, and possible status codes.

1. User Registration

Purpose	Register a new user in the system
URL	/api/register/
Method	POST
Request Data	{ "username": "user1", "email": "user1@example.com", "password": "pass123" }
Response Data	{ "id": 1, "username": "user1", "email": "user1@example.com" }
Status Codes	201 Created (Success), 400 Bad Request (Invalid data)

2. User Login

Purpose	Authenticate user and return JWT tokens
URL	/api/login/
Method	POST
Request Data	{ "username": "user1", "password": "pass123" }
Response Data	{ "access": "jwt-token", "refresh": "refresh-token" }
Status Codes	200 (Success), 401 Unauthorized (Invalid credentials)

3. User Logout

Purpose	Log out the authenticated user
URL	/api/logout/
Method	POST
Request Data	None
Response Data	{ "message": "Logout successful!" }
Status Codes	205 Reset Content (Success), 401 Unauthorized (User not logged in)

4. List Modules

Purpose	Retrieve a list of all module instances and their professors
URL	/api/modules/
Method	GET
Request Data	None
Response Data	[{ "code": "CD1", "name": "Computing", "year": 2018, "semester": 1, "professors": ["JE1"] }]
Status Codes	200 (Success)

5. View Professors and Ratings

Purpose	Retrieve a list of all professors and their ratings
URL	/api/professors/
Method	GET
Request Data	None
Response Data	[{ "identifier": "JE1", "name": "J. Excellent", "average_rating": 5 }]

Status Codes	200 (Success)

6. Get Average Rating for a Professor in a Module

Purpose	Retrieve the average rating of a professor in a module
URL	/api/average/<professor_id>/<module_code>/
Method	GET
Request Data	None
Response Data	{ "professor": "JE1", "module": "CD1", "average_rating": 4 }
Status Codes	200 (Success), 404 Not Found (Invalid professor or module)

7. Rate a Professor

Purpose	Allow users to rate a professor for a module instance
URL	/api/rate/
Method	POST
Request Data	{ "professor_id": "JE1", "module_code": "CD1", "year": 2018, "semester": 1, "rating": 5 }
Response Data	{ "message": "Rating submitted successfully!" }
Status Codes	201 Created (Success), 400 Bad Request (Already rated), 401 Unauthorized (User not logged in)

Status Codes and Error Handling

The API ensures proper error handling by returning meaningful status codes:

- **200 OK:** Successful retrieval of data.
- **201 Created:** Resource successfully created (e.g., user registration, rating submission).
- **204 No Content:** Successful operation with no response body (e.g., logout).
- **400 Bad Request:** Invalid input data or request format.
- **401 Unauthorized:** User not authenticated.
- **403 Forbidden:** User lacks permission for the requested action.
- **404 Not Found:** Resource (e.g., professor, module) not found.

Implementation Highlights

- **Django REST Framework (DRF)** is used to handle API views, authentication, and response formatting.
- **JWT Authentication** secures sensitive actions (e.g., rating professors) while allowing public access to general data.
- **Efficient Querying** using Django ORM and annotate() ensures performance optimization when retrieving aggregated data like average ratings.
- **Validation and Error Handling** ensure data integrity, such as checking whether a professor teaches a given module before allowing a rating submission.

This API provides a robust and scalable system for students to rate professors, ensuring data integrity and secure access to protected operations.

3. The Client

The client application is a **command-line interface (CLI)** implemented in Python using the **requests** library. It interacts with the RESTful web service by sending HTTP requests and processing JSON responses. A session-based authentication system is used, where JWT tokens are stored for authenticated actions.

Key Features & Implementation

Feature	Implementation Details
Session Management	Uses requests.Session() for persistent connections.
Authentication	Retrieves and stores JWT tokens for authenticated actions.
Error Handling	Catches connection failures, timeouts, and HTTP errors.
User Input Validation	Ensures valid ratings (1-5) and prevents duplicate logins (Single Active Session).
Security Measures	Sends tokens in headers for authentication.

Handling User Input Errors & Exceptions

The client handles various errors, such as invalid credentials, network issues, and server-side validation failures. The safe_request() function wraps API calls, catching common exceptions like:

- **ConnectionError:** Displays a network failure message.
- **Timeout:** Notifies the user if the request takes too long.
- **HTTPError:** Parses and displays server error messages.
- **ValueError**: Prevents invalid rating inputs.

4. Using the client

Ensure you have Python 3.x installed and a stable internet connection for API requests.

Before running the client, install dependencies using: `pip install requests`

Then, start the client by executing: `python client.py`

Command	Description	Syntax
register	Registers a new user.	register
login <URL>	Logs in to the service.	login https://sc22snba.pythonanywhere.com
logout	Logs out the current user.	logout
list	Lists all module instances and professors.	list
view	Displays all professors and their ratings.	view
average	Gets the average rating for a professor in a module.	average
rate	Submits a professor rating.	rate
exit	Exits the client.	exit

For rating a professor, follow the prompts to enter professor ID, module code, year, semester, and rating (1-5). Authentication is required for rating.