### ****Documentation for Rating Components****

| **HTTP Method** | **Endpoint** | **Description** |
| --- | --- | --- |
| POST | /api/ratings | Create a new movie rating |
| PUT | /api/ratings/{id} | Update an existing rating by ID |
| GET | /api/ratings/{id} | Get a rating by its UUID |
| GET | /api/ratings/ | Retrieve all movie ratings |
| DELETE | /api/ratings/{id} | Delete a rating by its UUID |

### RatingController.java

**Purpose:**  
RatingController is the main REST controller responsible for handling all HTTP requests related to movie ratings. It acts as a bridge between the client and the service layer (RatingService).

**Responsibilities:**

Accepts and validates incoming requests.

Maps requests to service layer calls.

Wraps responses in a consistent ResponseModel<T> format.

Handles UUID path variables and validation annotations.

**Endpoints Covered:**

POST /api/ratings – Create a new rating

PUT /api/ratings/{id} – Update a rating by ID

GET /api/ratings/{id} – Fetch a rating by ID

GET /api/ratings – List all ratings

DELETE /api/ratings/{id} – Delete a rating by ID

**Annotations Used:**

@RestController, @RequestMapping, @PostMapping, etc.

@Validated, @Valid for bean validation

@PathVariable, @RequestBody for request mapping

**Test Coverage Connection:**  
Each method in this controller is directly tested via RatingControllerTest, ensuring input/output correctness and integration with the mocked service.

### RatingService.java

**Purpose:**  
This is a **service interface** responsible for defining all business operations related to the **Rating** entity. It outlines the core functionalities such as creating, updating, retrieving, and deleting ratings. By abstracting the logic through an interface, it facilitates **decoupled architecture**, making the system easier to maintain, extend, and test.

### RatingServiceImpl.java

**Purpose:**  
This is the **concrete implementation** of the RatingService interface. It provides the **actual implementation logic** for managing rating data, such as validation, repository interaction, and enforcing business rules. It ensures a cohesive and maintainable business layer for all rating-related operations, adhering to clean architecture principles and supporting robust and scalable application development.

**RatingControllerTest.java**

**Purpose:**  
RatingControllerTest is a unit test class designed to validate the behavior of the RatingController using **Spring Boot's MockMvc framework**. It mocks the RatingService to isolate and test only the controller's logic.

**Tools Used:**

@WebMvcTest to test the controller layer only.

MockMvc for simulating HTTP requests and asserting responses.

Mockito for mocking the service layer.

ObjectMapper for JSON serialization/deserialization.

**Tested Endpoints:**

| **HTTP Method** | **Endpoint** | **Description** |
| --- | --- | --- |
| POST | /api/ratings | Create a new rating |
| PUT | /api/ratings/{id} | Update an existing rating |
| GET | /api/ratings/{id} | Retrieve a rating by ID |
| GET | /api/ratings | Retrieve all ratings |
| DELETE | /api/ratings/{id} | Delete a rating by ID |

**Test Coverage Provided:**

Happy-path test coverage for all CRUD endpoints.

JSON body assertions using jsonPath.

HTTP status validation (200, 201).

Service interaction verification through mocks.

### RatingDTO.java

**Purpose:**  
RatingDTO is a **Data Transfer Object** used to encapsulate and validate rating-related data exchanged between the client and server.

**Fields:**

UUID id – Unique ID of the rating (optional in creation)

Double rating – Must be between 0 and 10 (validated)

String review – Optional review message

UUID movieId – Required for linking rating to a movie

MovieDTO movieDTO – Optional embedded movie details

**Validations Applied:**

@NotNull for required fields

@Min and @Max for rating boundaries

**Conversion Methods:**

convertToRatingDTO(Rating rating) – Maps entity to DTO

convertToRating(RatingDTO dto, Movie movie) – Maps DTO to entity

**Test Coverage Connection:**

The RatingControllerTest uses this DTO as the request and response body.

Validations and field mappings are indirectly tested through JSON assertions in controller tests.

## Summary Table

| **Class** | **Type** | **Tested Components** | **Mocked Dependencies** | **Validations Tested** | **Responsibilities** |
| --- | --- | --- | --- | --- | --- |
| RatingController | Controller | All REST endpoints | Yes (RatingService) | Yes (via @Valid) | HTTP request handling and response mapping |
| RatingControllerTest | Test Class | Controller behavior and outputs | Yes | Partially | Unit test for all controller endpoints |
| RatingDTO | DTO | Used as request/response payloads | N/A | Yes (@Min, @NotNull) | Data structure and validation |