Courses API Design Documentation

STEP 1: Create a New API

Title: OpenAPI Specification for CMS – Courses API
Description: API Specification document for managing courses in the CMS system
Contact: Praveenkumar Bouna (http://myorganization.com/staff/praveenkumar-bouna)
Version: 1.0

STEP 2: Identify the Type of API

Type: Public
Style: RESTful API

STEP 3: Identify the Server Base URL

http://{hostname}:{portnumber}/{directory}
http://localhost:44333

STEP 4: Identify the Resources

courses, course-subjects, colleges

STEP 5: Use Plural Resources

/api/courses
/api/courses/{courseId}
/api/courses/{courseId}/students
/api/course-subjects
/api/colleges

STEP 6: Define the Resource Model

Course Model:
- courseId: int
- courseName: string
- courseDuration: int
- courseType: string

STEP 7: Select the Identifier

courseId (unique identifier)

STEP 8: Associations

/api/courses
/api/courses/{courseId}

/api/courses/{courseId}/students
/api/courses/{courseId}/course-subjects

STEP 9: URL Complexity

Collection/Item pattern kept simple

STEP 10: Identify Operations

/api/courses → GET (all), POST (new)
/api/courses/{courseId} → GET (by ID), PUT (update), DELETE (remove)
/api/courses/{courseId}/students → GET (list students), POST (add student)

STEP 11: Parameters

Path: courseId
Query: page, size, sortBy, courseType
Headers: None

STEP 12: Content-Type of Request

application/json

STEP 13: Request Body

POST /api/courses:
```
{
"courseName": "B.Tech",
"courseDuration": 4,
"courseType": "Full-time"
}
```

PUT /api/courses/{courseId}:
```
{
"courseName": "B.Sc",
"courseDuration": 3,
"courseType": "Part-time"
}
```

STEP 14: Status Codes

GET → 200 OK
POST → 201 CREATED, 400 BAD REQUEST
GET by ID → 200 OK, 404 NOT FOUND
PUT → 200 OK, 404 NOT FOUND
DELETE → 204 NO CONTENT, 404 NOT FOUND
GET students → 200 OK
POST student → 201 CREATED, 400 INVALID INPUT

STEP 15: Response Content-Type

application/json

STEP 16: Response Body

GET /api/courses:

```
[
{
"courseId": 1,
"courseName": "B.Tech",
"courseDuration": 4,
"courseType": "Full-time"
}
]
```

GET /api/courses/{courseId}:

```
{
"courseId": 1,
"courseName": "B.Tech",
"courseDuration": 4,
"courseType": "Full-time"
}
```

STEP 17: Error Handling

400 BAD REQUEST:

```
{
"error": {
"code": "INVALID_INPUT",
"message": "One or more input arguments are invalid",
"target": "Course",
"details": [
{ "code": "MISSING_FIELD", "target": "courseName", "message": "Course name is required" }
]
}
}
```

STEP 18–20: Filtering, Pagination, Sorting

Filtering: GET /api/courses?courseType=Full-time
Pagination: GET /api/courses?page=1&size;=20
Sorting: GET /api/courses?sortBy=courseName

STEP 21: API Versioning

Scheme: URL Versioning
Version: 1.0
/api/v1/courses
/api/v1/courses/{courseId}
/api/v1/courses/{courseId}/students