Course Management API - Documentation Report

Project Overview

A RESTful API for course management with validation, rate limiting, and file-based persistence. This documentation covers the implementation approach for each user story with code snippets and outputs.

User Story 1: CRUD API Setup

Implementation Approach

Built RESTful endpoints for course management using Express.js with in-memory storage initially, then migrated to file-based JSON storage for data persistence. Implemented proper HTTP status codes and REST conventions.

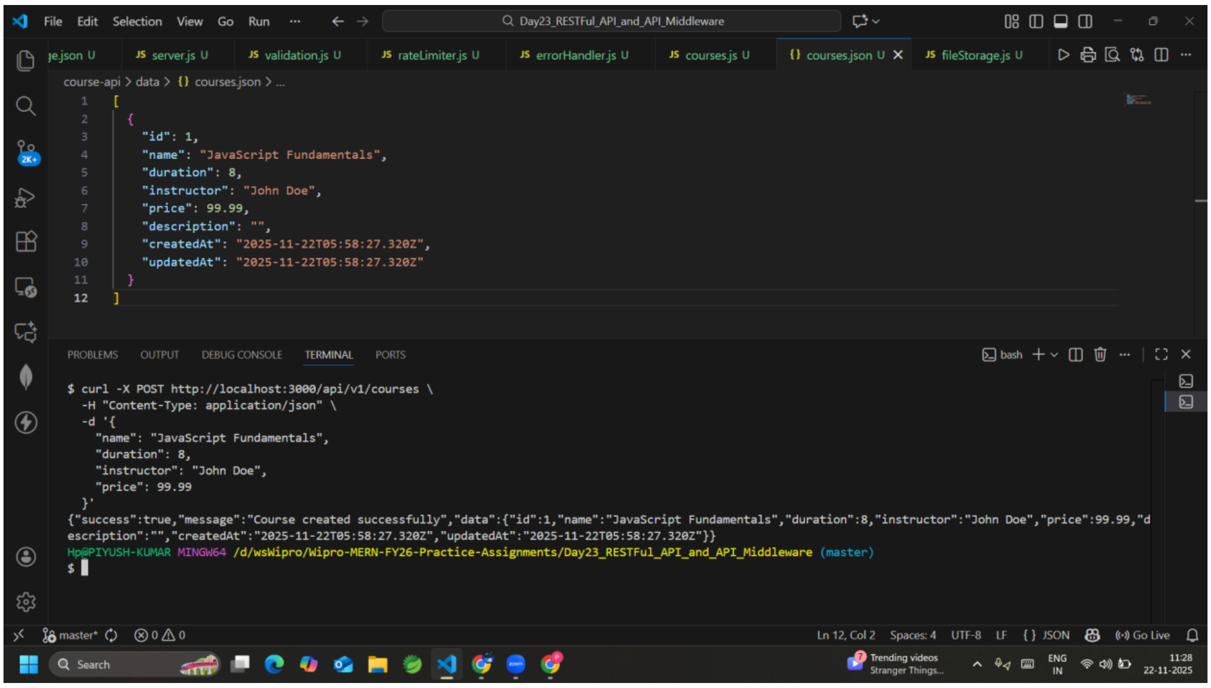
Key Code Snippet

```
// routes/courses.js - CRUD operations
router.get('/', async (req, res) => {
    const courses = await readCourses();
    res.status(200).json({ success: true, data: courses });
});

router.post('/', createCourseLimiter, validateCourseCreate,
handleValidationErrors,
    async (req, res) => {
        const newCourse = { id: generateNewId(courses), ...req.body };
        courses.push(newCourse);
        await writeCourses(courses);
        res.status(201).json({ success: true, data: newCourse });
    });
```

Output

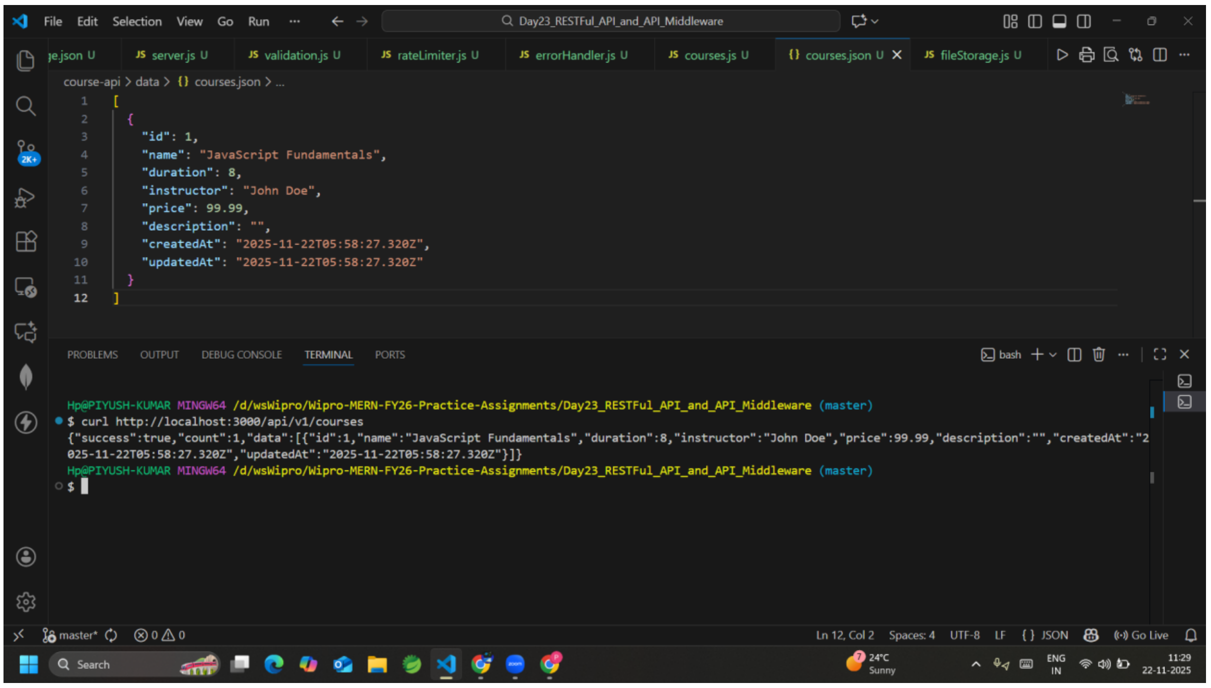
[POST]



```
[
  {
    "id": 1,
    "name": "JavaScript Fundamentals",
    "duration": 8,
    "instructor": "John Doe",
    "price": 99.99,
    "description": "",
    "createdAt": "2025-11-22T05:58:27.320Z",
    "updatedAt": "2025-11-22T05:58:27.320Z"
  }
]
```

```
$ curl -X POST http://localhost:3000/api/v1/courses \
  -H "Content-Type: application/json" \
  -d '{
    "name": "JavaScript Fundamentals",
    "duration": 8,
    "instructor": "John Doe",
    "price": 99.99
  }'
{"success":true,"message":"Course created successfully","data":{"id":1,"name":"JavaScript Fundamentals","duration":8,"instructor":"John Doe","price":99.99,"d
escription":"","createdAt":"2025-11-22T05:58:27.320Z","updatedAt":"2025-11-22T05:58:27.320Z"}}
Hp@PIYUSH-KUMAR MINGW64 /d/wsWipro/Wipro-MERN-FY26-Practice-Assignments/Day23_RESTFul_API_and_API_Middleware (master)
$ 
```

[GET]



```
[
  {
    "id": 1,
    "name": "JavaScript Fundamentals",
    "duration": 8,
    "instructor": "John Doe",
    "price": 99.99,
    "description": "",
    "createdAt": "2025-11-22T05:58:27.320Z",
    "updatedAt": "2025-11-22T05:58:27.320Z"
  }
]
```

```
Hp@PIYUSH-KUMAR MINGW64 /d/wsWipro/Wipro-MERN-FY26-Practice-Assignments/Day23_RESTFul_API_and_API_Middleware (master)
$ curl http://localhost:3000/api/v1/courses
{"success":true,"count":1,"data":[{"id":1,"name":"JavaScript Fundamentals","duration":8,"instructor":"John Doe","price":99.99,"description":"","createdAt":"2
025-11-22T05:58:27.320Z","updatedAt":"2025-11-22T05:58:27.320Z"}]}
Hp@PIYUSH-KUMAR MINGW64 /d/wsWipro/Wipro-MERN-FY26-Practice-Assignments/Day23_RESTFul_API_and_API_Middleware (master)
$ 
```

User Story 2: Input Validation

Implementation Approach

Implemented server-side validation using express-validator middleware to ensure data integrity. Added validation rules for required fields, data types, and constraints with meaningful error messages.

Key Code Snippet

```
// middleware/validation.js
```

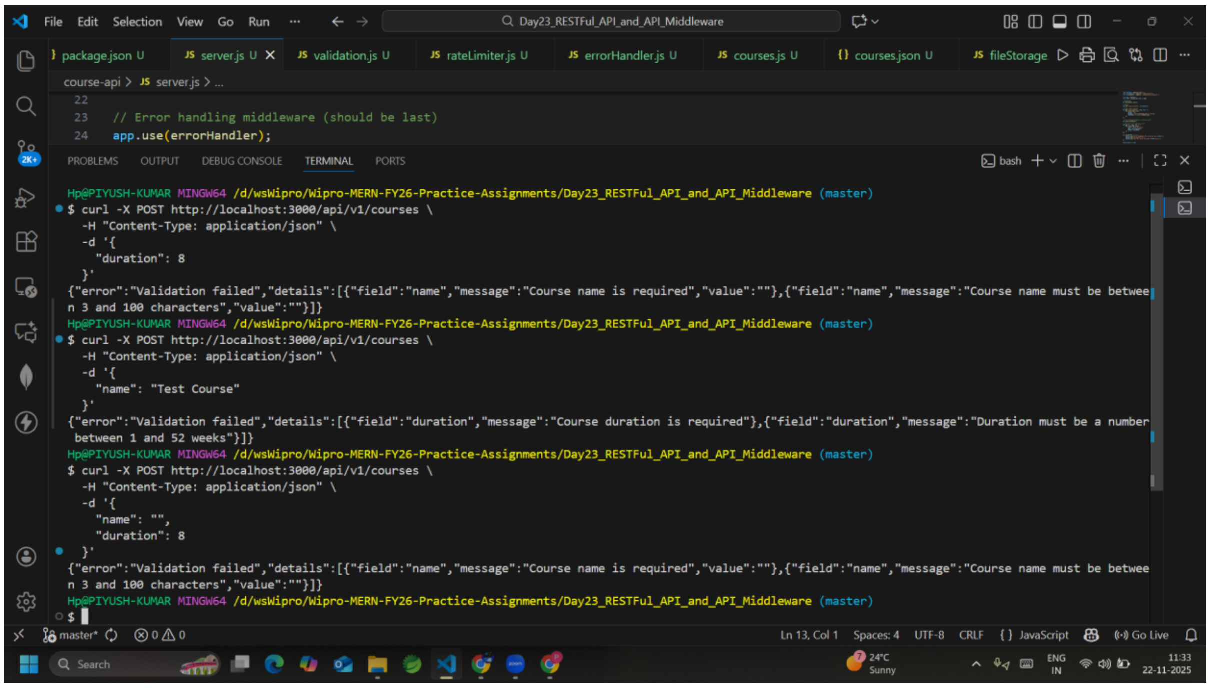
```
const validateCourseCreate = [
    body('name')
        .trim()
        .notEmpty().withMessage('Course name is required')
        .isLength({ min: 3, max: 100 }),

    body('duration')
        .notEmpty().withMessage('Course duration is required')
        .isInt({ min: 1, max: 52 }),

    body('instructor')
        .optional()
        .isLength({ min: 2, max: 50 })
];

const handleValidationErrors = (req, res, next) => {
    const errors = validationResult(req);
    if (!errors.isEmpty()) {
        return res.status(400).json({
            error: 'Validation failed',
            details: errors.array()
        });
    }
    next();
};
```

Output


File Edit Selection View Go Run ···
Day23_RESTFul_API_and_API_Middleware
package.json U
server.js U
validation.js U
rateLimiter.js U
errorHandler.js U
courses.js U
courses.json U
fileStorage
course-api > server.js > ...
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS
bash
master* ⊗ 0 ⚠ 0
Ln 13, Col 1 Spaces: 4 UTF-8 CRLF { } JavaScript Go Live
Search
24°C Sunny
ENG IN
11:33 22-11-2025

```
    // Error handling middleware (should be last)
    app.use(errorHandler);
```

```
Hp@PIYUSH-KUMAR MINGW64 /d/wsWipro/Wipro-MERN-FY26-Practice-Assignments/Day23_RESTFul_API_and_API_Middleware (master)
$ curl -X POST http://localhost:3000/api/v1/courses \
  -H "Content-Type: application/json" \
  -d '{
    "duration": 8
  }'
{"error":"Validation failed","details":[{"field":"name","message":"Course name is required","value":""},{"field":"name","message":"Course name must be betwee
n 3 and 100 characters","value":""}]}
Hp@PIYUSH-KUMAR MINGW64 /d/wsWipro/Wipro-MERN-FY26-Practice-Assignments/Day23_RESTFul_API_and_API_Middleware (master)
$ curl -X POST http://localhost:3000/api/v1/courses \
  -H "Content-Type: application/json" \
  -d '{
    "name": "Test Course"
  }'
{"error":"Validation failed","details":[{"field":"duration","message":"Course duration is required"},{"field":"duration","message":"Duration must be a number
 between 1 and 52 weeks"}]}
Hp@PIYUSH-KUMAR MINGW64 /d/wsWipro/Wipro-MERN-FY26-Practice-Assignments/Day23_RESTFul_API_and_API_Middleware (master)
$ curl -X POST http://localhost:3000/api/v1/courses \
  -H "Content-Type: application/json" \
  -d '{
    "name": "",
    "duration": 8
  }'
{"error":"Validation failed","details":[{"field":"name","message":"Course name is required","value":""},{"field":"name","message":"Course name must be betwee
n 3 and 100 characters","value":""}]}
Hp@PIYUSH-KUMAR MINGW64 /d/wsWipro/Wipro-MERN-FY26-Practice-Assignments/Day23_RESTFul_API_and_API_Middleware (master)
$
```

User Story 3: API Rate Limiting

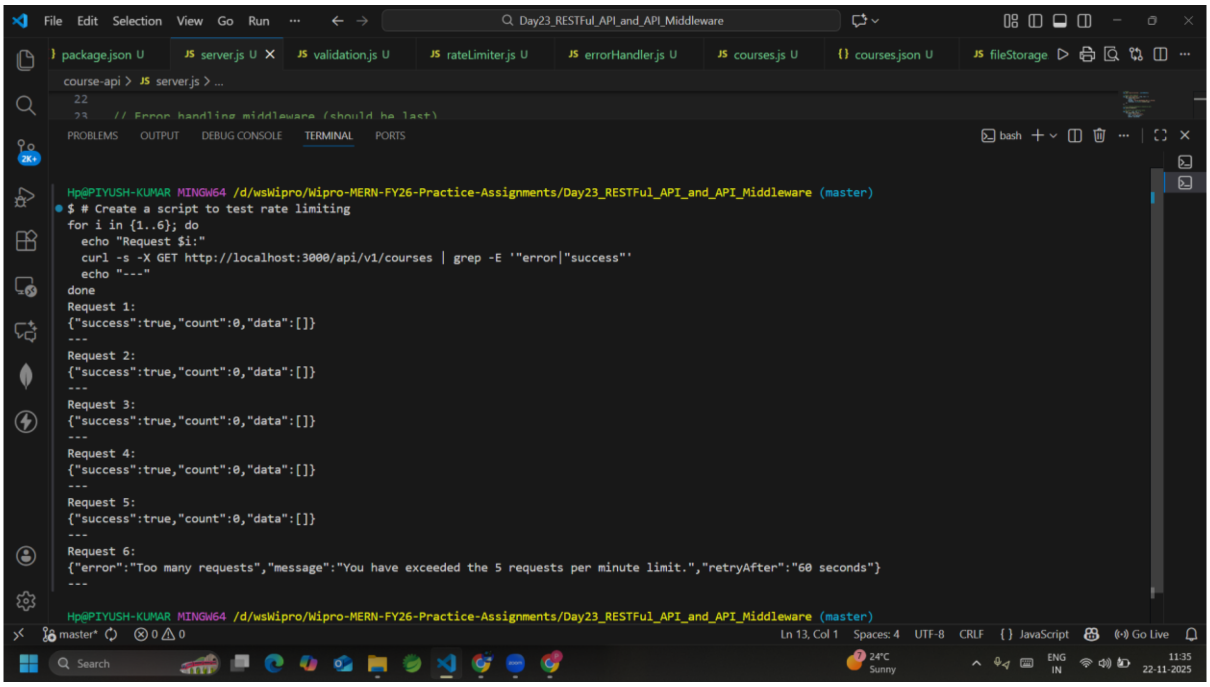
Implemented rate limiting using express-rate-limit middleware to prevent API abuse. Configured different limits for general API usage (5 req/min) and course creation (3 req/min) with appropriate error responses.

Key Code Snippet

```javascript
// middleware/rateLimiter.js
const apiLimiter = rateLimit({
    windowMs: 1 * 60 * 1000, // 1 minute
    max: 5, // 5 requests per minute
    message: {
        error: 'Too many requests',
        message: 'You have exceeded the 5 requests per minute limit.',
        retryAfter: '60 seconds'
    }
});

const createCourseLimiter = rateLimit({
    windowMs: 1 * 60 * 1000,
    max: 3, // 3 creation requests per minute
    message: {
        error: 'Too many course creation requests',
        message: 'Please slow down when creating new courses.'
    }
});
```

Output


File Edit Selection View Go Run
Day23_RESTFul_API_and_API_Middleware
package.json U
server.js U
validation.js U
rateLimiter.js U
errorHandler.js U
courses.js U
courses.json U
fileStorage
course-api > server.js > ...
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS
bash
master* 0 0
Ln 13, Col 1 Spaces: 4 UTF-8 CRLF { } JavaScript Go Live
Search
24°C Sunny
ENG IN
11:35 22-11-2025

```
// Error handling middleware (should be last)
```

```
Hp@PIYUSH-KUMAR MINGW64 /d/wsWipro/Wipro-MERN-FY26-Practice-Assignments/Day23_RESTFul_API_and_API_Middleware (master)
$ # Create a script to test rate limiting
for i in {1..6}; do
  echo "Request $i:"
  curl -s -X GET http://localhost:3000/api/v1/courses | grep -E '"error|"success"'
  echo "---"
done
Request 1:
{"success":true,"count":0,"data":[]}
---
Request 2:
{"success":true,"count":0,"data":[]}
---
Request 3:
{"success":true,"count":0,"data":[]}
---
Request 4:
{"success":true,"count":0,"data":[]}
---
Request 5:
{"success":true,"count":0,"data":[]}
---
Request 6:
{"error":"Too many requests","message":"You have exceeded the 5 requests per minute limit.","retryAfter":"60 seconds"}
---

Hp@PIYUSH-KUMAR MINGW64 /d/wsWipro/Wipro-MERN-FY26-Practice-Assignments/Day23_RESTFul_API_and_API_Middleware (master)
```