












Backend CRUD operation

Student-Class Management CRUD API

A full-featured **RESTful API** built with **Node.js**, **Express**, and **MongoDB** (via Mongoose) to manage students and class records efficiently.

Core Features

-  Student & Class models with Mongoose
 -  Full **CRUD operations**
 -  Input validation with **Joi**
 -  Clean **RESTful API design**
 -  MongoDB relationships (Student ↔ Class)
 -  Pagination support
 -  Environment-based config
 -  Centralized error handling
 -  Modular folder structure
 -  Postman collection for API testing
 -  Comprehensive documentation
-

Getting Started

Prerequisites

- Node.js
- MongoDB (running locally)
- Postman

Installation

```
git clone https://github.com/yourusername/Student-Class-CRUD-API.git
cd Student-Class-CRUD-API
```

```
npm install
```

Setup Environment

Create a `.env` file in the root directory:

```
PORT=5000  
MONGO_URI=mongodb://localhost:27017/student_class_api
```

Start the Server

```
npm run dev
```

 **Make sure MongoDB is running** before starting:

```
mongod  
# or  
sudo service mongod start
```

Folder Structure

```
project-root/  
|  
├─ models/  
|   ├─ student.model.js  
|   └─ class.model.js  
|  
├─ controllers/  
|   ├─ student.controller.js  
|   └─ class.controller.js  
|  
├─ routes/  
|   ├─ student.routes.js  
|   └─ class.routes.js  
|  
├─ config/  
|   └─ db.js
```

```
|
|— middleware/
|   └─ errorHandler.js
|
|— app.js
|— server.js
|— .env
|— README.md
└─ Student-Class-CRUD-API.postman_collection.json
```

Environment Variables

Required in `.env` :

```
PORT=5000
MONGO_URI=mongodb://localhost:27017/student_class_api
```

Referenced in `config/db.js` .

Database Models

 **Student** (`student.model.js`)

Backend CRUD API Task | Assignment Data | Cloud: Mon... | Postman - Browser Based Auth | localhost:5000/classes

cloud.mongodb.com/v2/6849c244399dd422687e4c7a#/metrics/replicaSet/684a7e61fcb653535a75aac/explorer/test/students/find

Atlas MANIKAND... Access Manager Billing

Project 0 Data Services Charts

Overview DATABASE Clusters SERVICES Atlas Search Stream Processing Triggers Migration Data Federation SECURITY Quickstart Backup Database Access Network Access Advanced Goto

Search Namespaces

sample_mflix

test

classes

students

STORAGE SIZE: 36KB LOGICAL DATA SIZE: 306B TOTAL DOCUMENTS: 2 INDEXES TOTAL SIZE: 72KB

Find Indexes Schema Anti-Patterns Aggregation Search Indexes

Generate queries from natural language in Compass

Filter Type a query: { field: 'value' }

QUERY RESULTS: 1-2 OF 2

```
{
  "_id": ObjectId('684ad917da890567f6c1bf67'),
  "name": "Jane Doe",
  "rollNo": "2023002",
  "mobileNo": "9876543211",
  "classId": ObjectId('684ad7a2da890567f6c1bf63'),
  "createdAt": 2025-06-12T13:41:43.382+00:00,
  "updatedAt": 2025-06-12T13:41:43.382+00:00,
  "__v": 0
}
```

```
{
  "_id": ObjectId('684ae1e1393f836a52750e15'),
  "name": "Stitch",
  "rollNo": "2023091",
  "mobileNo": "9876543278A"
}
```

System Status: All Good

©2025 MongoDB, Inc. Status Terms Privacy Atlas Blog Contact Sales

Tasks

Go to my collection

✓ Mark as completed

3 Run a query

4 Explore search indexes

5 Query with an index

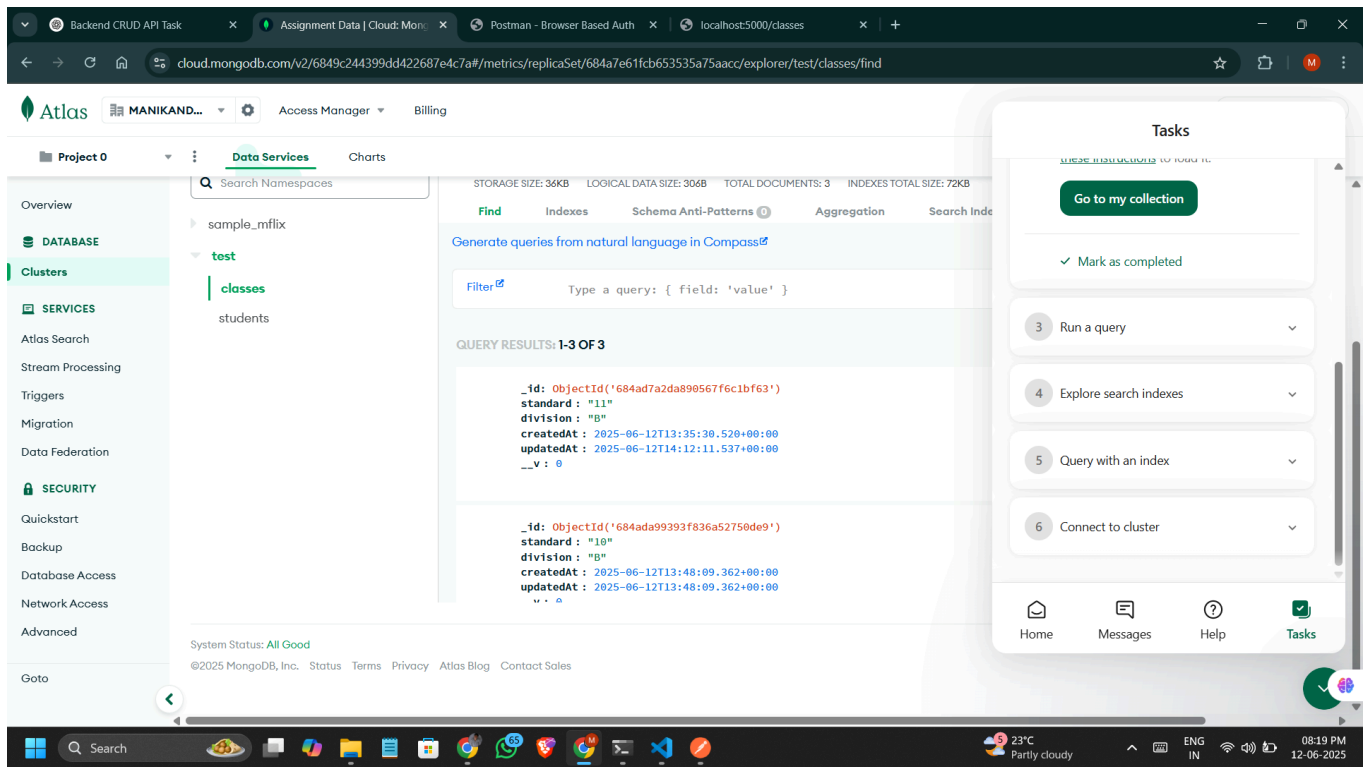
6 Connect to cluster

Home Messages Help Tasks

23°C Partly cloudy 08:19 PM 12-06-2025

Field	Type	Description
name	String	Required
email	String	Unique, required
age	Number	Optional
classId	ObjectId	Reference to Class

 **Class (class.model.js)**



Field	Type	Description
name	String	Required
standard	String/Int	Required
students	[ObjectID]	Array of Student references

API Endpoints

Class Routes

Method	Endpoint	Description
GET	/api/classes	Get all classes
GET	/api/classes/:id	Get class by ID
POST	/api/classes	Create new class
PUT	/api/classes/:id	Update class by ID
DELETE	/api/classes/:id	Delete class by ID

Student Routes

Method	Endpoint	Description
GET	/api/students	Get all students
GET	/api/students/:id	Get student by ID
POST	/api/students	Create new student
PUT	/api/students/:id	Update student by ID
DELETE	/api/students/:id	Delete student by ID

Error Handling

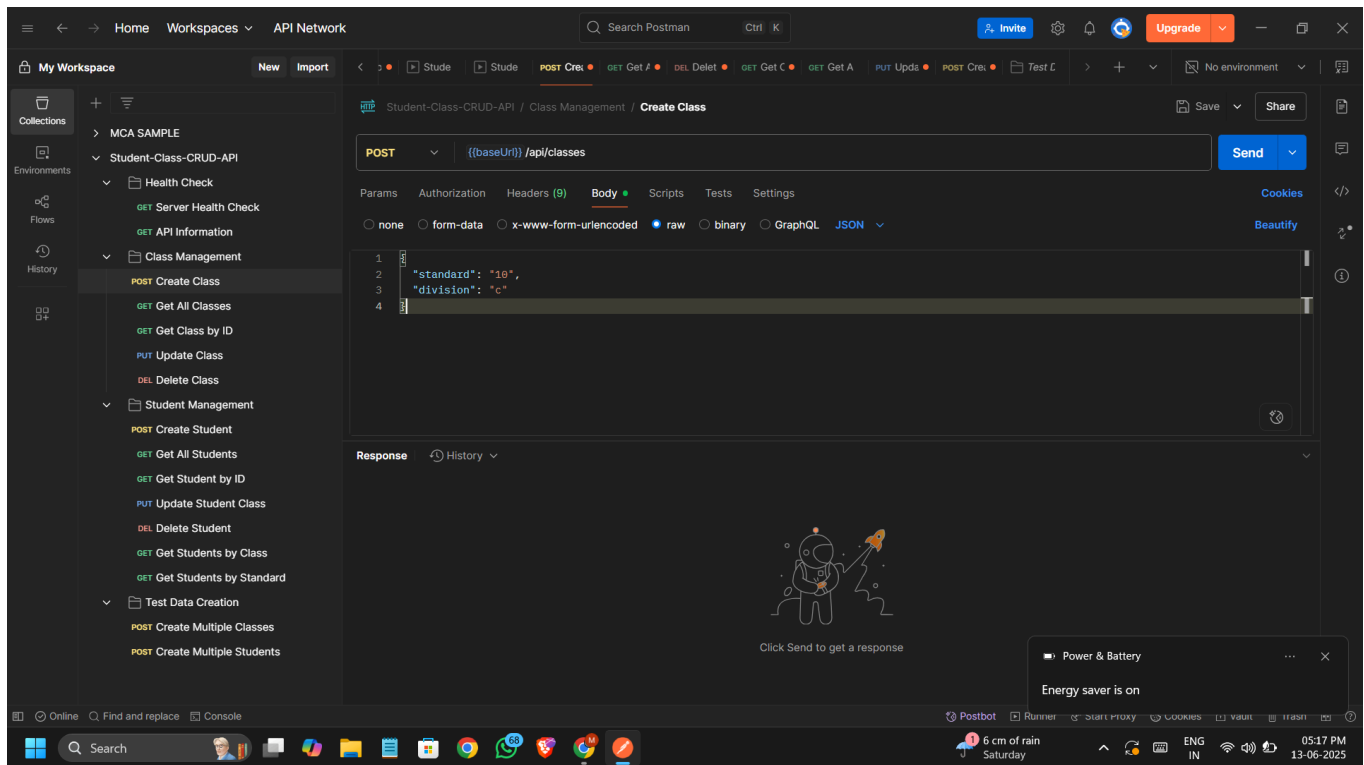
Centralized using custom middleware (`errorHandler.js`).

Example Error Response

```
{
  "success": false,
  "message": "Validation failed",
  "errors": [
    {
      "field": "email",
      "message": "Email must be valid"
    }
  ]
}
```

- All validations handled by **Joi**
- Errors are standardized as JSON

Postman Collection



✓ **File:** Student-Class-CRUD-API.postman_collection.json

To test the API:

1. Open **Postman**
2. Click **Import**
3. Upload the JSON file

💡 Pro Tip: First create a class → then add students using `classId`



Author

Manikandan M

Thank You