Documentation

The steps for setting up the project on a Linux machine can be found in the README.md file .
The details of the models can be found in models file in src folder or README.md
MongoDB connection details can be found in README.md

Deployed URL: localhost:3000

This API allows you to do the following tasks:

1. As a **Student**:
   1.1. Sign Up
   1.2. Sign In
   1.3. View available courses
   1.4. Register for a course
   1.5. View registered courses

2. As an **Admin**:
   2.1. Sign In
   2.2. Add a course
   2.3. Delete a course
   2.4. Enroll a student to a course by admin
   2.5. Disenroll a student from a course by admin

1.As a **Student**:

1.1 **Sign Up**:

First of all, the student should sign up to perform the above mentioned tasks.

Send the **POST** request to: "**localhost:3000/api/students/signup**" with the fields mentioned below as request body.

```
{
        "registerno":<number>
        "password":<string>
}
```

Example request body:

```
{
  "registerno": 12546011,
  "password":"samplepassword"
}
```

The response after creating student successfully will be:


```
{
  "data": {
    "regno": 12546011,
    "courses": [],
    "id": "6416e8ffec1fe0a0a20da0f6"
  },
  "success": true
}
```


If mandatory fields are missing:

Ex:


```
{
  "registerno":12515600111
}
```


Response:


```
{
  "message": "Missing mandatory field, check request body",
  "success": false
}
```


If the user is already registered, the response will be:


```
{
  "message": "User already exists",
  "success": false
}
```


If the registerno field is not number, then then response will be:

Ex:


```
{
  "registerno":"ASDSD",
  "password":"samplepassword"
}
```


Response:


```
{
  "message": "Student validation failed: regno: Cast to Number failed for value \"ASDSD\" (type string) at path \"regno\"",
  "success": false
}
```

1.2 **Sign In**:

The above mentioned tasks can be performed only when the student signs in using registerno and password.

Send the **POST** request to:"**localhost:3000/api/students/signup**" with below mentioned fields in request body:


```
{
  "registerno":12546011,
  "password":"samplepassword"
}
```


If the **sign in is successful**, a **unique token** will be returned which should be used for performing the tasks.

The token should be passed as **Bearer Token** . The token created will be valid for **1 hour**, after which the user should login again.

Successful response:


```
{
  "token":
"eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJyZWdubyI6MTI1NDYwMTEsImlkIjoiNjQxNmU4ZmZl
YzFmZTBhMGEyMGRhMGY2IiwiaWF0IjoxNjc5MjI0MTQxLCJleHAiOjE2NzkyMjc3NDF9.-_tSJJ94
5QEhP09RoPYZSbBkyw2FCpZ8ae0uXKOrEVE",
  "success": true
}
```


Response for Invalid Password:


```
{
  "message": "Invalid password",
  "success": false
}
```


If the user is not registered, the response will be:


```
{
  "message": "Unregistered user",
  "success": false
}
```

If the input field data type doesn't match:


```
{
  "message": "Cast to Number failed for value \"rams88\" (type string) at path \"regno\" for model \"Student\"",
  "success": false
}
```


1.3 **View Available courses:**

To perform this task, the user should send a **GET** request to: "localhost:3000/api/students/courses" along with the token as Bearer token.

The successful response, when the token is valid looks like this:


```
{
  "courses": [
    {
      "courseName": "DAA",
      "courseCode": "DA108",
      "capacity": 50
    },
    {
      "courseName": "DSA",
      "courseCode": "DS109",
      "capacity": 48
    },
    {
      "courseName": "Flutter",
      "courseCode": "APK101",
      "capacity": 195
    },
    {
      "courseName": "Not learning",
      "courseCode": "NL101",
      "capacity": 100
    }
  ],
  "success": true
}
```

Token Errors:

Response when token is not passed:


```
{
  "message": "No token received",
  "success": false
}
```


Response when token is tampered:


```
{
  "message": "Invalid/Malformed/Missing signature of the token",
  "success": false
}
```


Response when token has expired:


```
{
  "message": "Token expired, login again",
  "success": false
}
```


1.4 **Register for a course**:

To register for a course, send **POST** request to: "localhost:3000/api/students/courses" along with the token as bearer token and request body as follows:


```
{
  "courseName":"Flutter",
  "courseCode":"APK101"
}
```


Successful response will look like this:


```
{
  "data": {
    "courseName": "Flutter",
    "courseCode": "APK101"
  },
  "success": true
}
```

Response if user already registered for the course:


```
{
  "error": "Already registered",
  "success": false
}
```


Response if courseCode or courseName is not correct:


```
{
  "error": "Course Not Found",
  "success": false
}
```


Response if mandatory fields in body are missing:


```
{
  "message": "Missing mandatory fields, check request body",
  "success": false
}
```


1.5 **View registered courses**:

To view the registered courses by a student, **GET** request should be sent to: "localhost:3000/api/students/registeredcourses" along with the token as bearer token.

Successful response will be:


```
{
  "data": [
    {
      "courseName": "Flutter",
      "courseCode": "APK101"
    },
    {
      "courseName": "DSA",
      "courseCode": "DS109"
    },
    {
      "courseName": "DAA",
      "courseCode": "DA108"
    }
  ],
  "success": true
}
```

If the token is invalid/expired, the error responses mentioned Token Errors in 1.3 will be sent

2. As an **Admin**:

2.1 **Sign In**:

For this task,the admin credentials are hard coded in the .env file, which is used for verifying the credentials. The admin credentials can be made dynamic in the backend.

For Signing In, the user should send **POST** request to: "localhost:3000/api/admin/signin" with request body as follows:

```
{
  "userid":"admin@google.com",
  "password":"admin"
}
```

If signing in is successful, the response will contain the token:

```
{
  "token":
"eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJhZG1pbnVzZXJpZCI6ImFkbWluQGdvb2dsZS5jb20iLC
JhZG1pbnBhc3N3b3JkIjoiYWRtaW4iLCJpYXQiOjE2NzkyMjgyNzgsImV4cCI6MTY3OTIzMTg3OH0.S
927RLmoT8vi-gG8xs6Fdt_OeEnBbYMnjK3e7qk1p0A",
  "success": true
}
```

This token can be used for further admin tasks by passing the token as Bearer token.

If mandatory fields are missing:

```
{
  "message": "Missing mandatory fields, check request body",
  "success": false
}
```

If invalid username or password is invalid:

```
{
  "message": "Unauthorized user",
  "success": false
}
```

2.2 **Adding a course**:

To add a course, the admin should pass a valid token and the request body as **POST** request to:"localhost:3000/api/admin/courses"

```
{
  "courseName":"Machine Learning",
  "courseCode":"ML101",
  "capacity":100
}
```

Successful response will look like this:


```
{
  "data": {
    "courseName": "Machine Learning",
    "courseCode": "ML101",
    "capacity": 100,
    "students": [],
    "_id": "64170396a74cb3fdad3e77cf",
    "__v": 0
  },
  "success": true
}
```


If course is already present, response will be:


```
{
  "message": "Duplicate course entry",
  "success": false
}
```


If mandatory fields are missing, then response will be:


```
{
  "message": "Missing mandatory fields",
  "success": false
}
```


If the token is invalid/expired, the error responses mentioned Token Errors in 1.3 will be sent

2.3 **Delete a course**:

To delete a course, the admin should pass a valid token and the request body as **DELETE** request to:"localhost:3000/api/admin/courses"

```
{
```

```
  "courseName":"Machine Learning",
  "courseCode":"ML101"
}
```


If deleted successfully, response will be like this:


```
{
  "message": "Deleted course",
  "success": true
}
```


If course does not exist, response will be:


```
{
  "message": "Course Not Found",
  "success": false
}
```


If the token is invalid/expired, the error responses mentioned Token Errors in 1.3 will be sent

2.4 **Add student to a course:**

Admin can change the registration made by a student by changing the course.

For this the user should send PATCH request to: "localhost:3000/api/admin/courses" along with a valid token and request body as mentioned below:


```
{
  "registerno":125156001,
  "courseName":"Not learning",
  "courseCode":"NL101"
}
```


If added successfully, response will be:


```
{
  "data": {
    "courseName": "Not learning",
    "courseCode": "NL101"
  },
```

```
  "success": true
}
```


If mandatory fields are missing, response will be:


```
{
  "message": "Missing mandatory fields, check request body",
  "success": false
}
```


If the entered registerno (Student) does not exist then response will be:


```
{
  "message": "Student Not Found",
  "success": false
}
```


If the token is invalid/expired, the error responses mentioned Token Errors in 1.3 will be sent

2.5 Removing a student from a course:

Admin can remove a student from a course by sending a **PATCH** request to :"localhost:3000/api/admin/removecourses" along with the token and request body as follows:


```
{
  "registerno":125156001,
  "courseName":"Not learning",
  "courseCode":"NL101"
}
```


Successful Response will be:


```
{
  "message": "Successfully removed student from course",
  "success": true
}
```


If student is not registered for the course you are trying to remove, the response will be:


```
{
  "message": "Student not registered for the course",
```

```
  "success": false
}
```


If mandatory field missing:


```
{
  "message": "Missing mandatory fields, check request body",
  "success": false
}
```


If course not found:


```
{
  "message": "Course Not Found",
  "success": false
}
```


If the token is invalid/expired, the error responses mentioned Token Errors in 1.3 will be sent