**1. Login Api**

Curl :- curl --location 'http://localhost:8000/sms/auth/login' \

--header 'Content-Type: application/json' \

--data-raw '{

"email": "superuser@yopmail.com",

"password": "superuser@123"

}'

Request :- {

"email": "superuser@yopmail.com",

"password": "superuser@123"

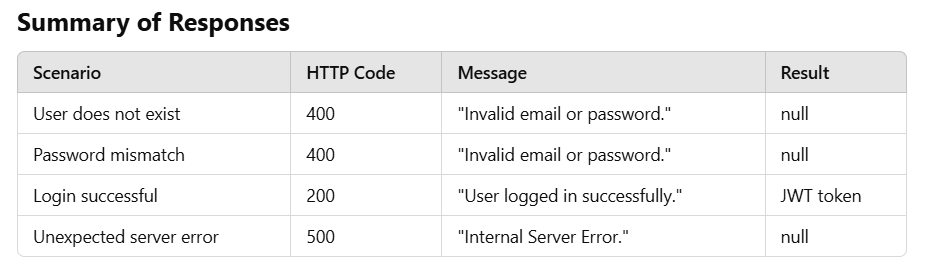
}  
Response :- {

"code": 200,

"message": "User logged in successfully.",

"result": "eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJpZCI6IjY3NjE4OGQwOGYyM2U4MzQ5MTlhY2UxZiIsInJvbGUiOjEsImlhdCI6MTczNDYxNjQzOCwiZXhwIjoxNzM0NjIwMDM4fQ.2aaVOSAd1P6c88omlgbHR33vyK8JzRq-IK64wckwp3U"

}

Api Flow :- The login controller begins by extracting the email and password from the request body. It then checks if a user with the provided email exists in the database. If no user is found, a 400 status code is returned with the message "Invalid email or password.". If a user exists, the provided password is compared with the stored hashed password using bcrypt. If the passwords do not match, the same 400 status code and message are returned. Upon successful validation, a JWT token is generated using the user details and sent back in the response with a 200 status code and the message "User logged in successfully.". If any unexpected error occurs during the process, a 500 status code is returned with the message "Internal Server Error.". This flow ensures secure handling of user credentials while providing clear responses for success or failure scenarios.  
  


Invalid Input Scenarios

Request :- {

"email": "superu",

"password": "1111"

}

Response :- {

"code": 400,

"message": "Validation failed.",

"result": [

"\"email\" must be a valid email",

"\"password\" length must be at least 6 characters long"

]

}

**2. Create User Api**

Curl :- curl --location 'http://localhost:8000/sms/user/create' \

--header 'Content-Type: application/json' \

--header 'Authorization: Bearer eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJpZCI6IjY3NjE4OGQwOGYyM2U4MzQ5MTlhY2UxZiIsInJvbGUiOjEsImlhdCI6MTczNDYxNjg4OCwiZXhwIjoxNzM0NjIwNDg4fQ.mZhjrTxbTWOv\_9jcskwRE6P2FwL5EJCU1BzAibNUEcc' \

--data-raw '{

"name": "School Admin 3",

"email": "schooladmin3@yopmail.com",

"role": 2,

"password" : "schooladmin@123"

}'

Request :- {

"name": "School Admin 3",

"email": "schooladmin3@yopmail.com",

"role": 2,

"password" : "schooladmin@123"

}

Response :- {

"code": 200,

"message": "User created successfully.",

"result": {

"name": "School Admin 3",

"email": "schooladmin3@yopmail.com",

"password": "$2a$10$es3vLk/zNLPFjiBur4q/UelI5OQw1jRHt5.up1hMJ3nV81H8GT47m",

"role": 2,

"schoolIds": [],

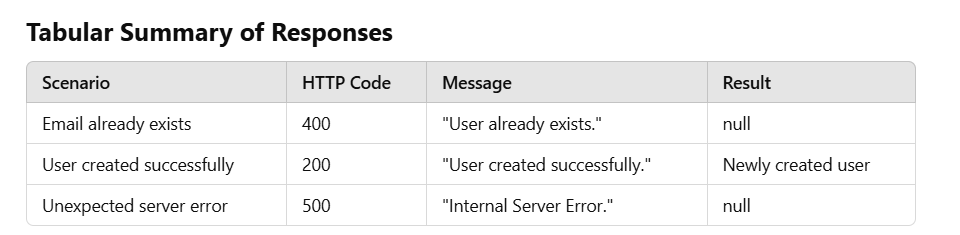
"\_id": "6764279463d05e3d0172839f",

"\_\_v": 0

}

}

Api Flow :- The create controller is responsible for registering a new user. It begins by extracting the user data from the request body and creating a new instance of the User model. The email field is then checked against the database to ensure uniqueness. If a user with the provided email already exists, a 400 status code is returned with the message "User already exists.". If the email is unique, the user data is saved to the database. Upon successful creation, the newly created user object is returned in the response with a 200 status code and the message "User created successfully.". If an unexpected error occurs during the process, a 500 status code is returned with the message "Internal Server Error.". This flow ensures new users are only created if their email is unique and provides appropriate feedback for success or failure scenarios.



Invalid Input Scenarios  
  
Request :- {

"name": "School Admin 3",

"email": "school.com",

"role": 3,

"password" : "@123"

}

Response :- {

"code": 400,

"message": "Validation failed.",

"result": [

"\"email\" must be a valid email",

"\"password\" should have a minimum length of 8"

]

}

**3. Get Users Api**

Curl :- curl --location 'http://localhost:8000/sms/user/getAllUsers' \

--header 'Authorization: Bearer eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJpZCI6IjY3NjE4OGQwOGYyM2U4MzQ5MTlhY2UxZiIsInJvbGUiOjEsImlhdCI6MTczNDYxNjg4OCwiZXhwIjoxNzM0NjIwNDg4fQ.mZhjrTxbTWOv\_9jcskwRE6P2FwL5EJCU1BzAibNUEcc'

Response :- {

"code": 200,

"message": "Users fetched successfully.",

"result": [

{

"\_id": "676188d08f23e834919ace1f",

"name": "Super User",

"email": "superuser@yopmail.com",

"password": "$2a$10$ibdL8It9eWVjl1HmM3QaFO78Z4xsh1AO2xZXd2xzbVKpPGByR56bC",

"role": 1,

"schoolIds": [],

"\_\_v": 0

},

{

"\_id": "6761e45afc30b799b4e75ba9",

"name": "School Admin 1",

"email": "schooladmin1@yopmail.com",

"password": "$2a$10$ftHIU8SbQqDCg6NsDdl0k.8QNWjEGE50vzJASTjOOw4HJWbI4.6ZK",

"role": 2,

"schoolIds": [

"676331bb9d5f7594b2aa4d90",

"676331c49d5f7594b2aa4d93"

],

"\_\_v": 7

},

{

"\_id": "6764279463d05e3d0172839f",

"name": "School Admin 3",

"email": "schooladmin3@yopmail.com",

"password": "$2a$10$es3vLk/zNLPFjiBur4q/UelI5OQw1jRHt5.up1hMJ3nV81H8GT47m",

"role": 2,

"schoolIds": [],

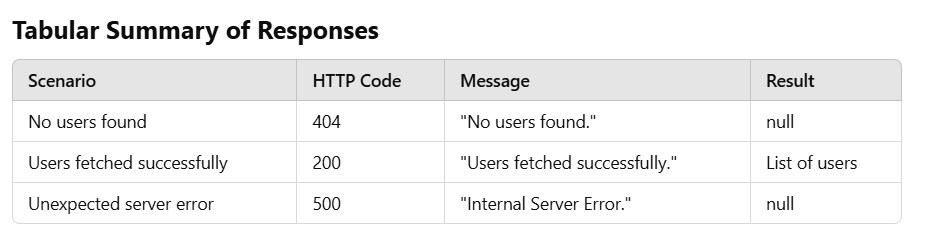
"\_\_v": 0

}

]

}

Api Flow :- The fetch controller is responsible for retrieving all users from the database. It begins by querying the User collection to fetch all user records. If no users are found, a 404 status code is returned with the message "No users found.". If users are found, the controller returns a 200 status code with the message "Users fetched successfully." and the list of users in the response. If an unexpected error occurs during the process, a 500 status code is returned with the message "Internal Server Error.". This flow ensures efficient user data retrieval and provides clear responses for success or failure scenarios.



**4. Update User Api**

Curl :- curl --location --request PUT 'http://localhost:8000/sms/user/update/6764279463d05e3d0172839f' \

--header 'Content-Type: application/json' \

--header 'Authorization: Bearer eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJpZCI6IjY3NjE4OGQwOGYyM2U4MzQ5MTlhY2UxZiIsInJvbGUiOjEsImlhdCI6MTczNDYxNjg4OCwiZXhwIjoxNzM0NjIwNDg4fQ.mZhjrTxbTWOv\_9jcskwRE6P2FwL5EJCU1BzAibNUEcc' \

--data-raw '{

"name": "School Admin 2",

"email": "schooladmin2@yopmail.com",

"password": "schooladmin@123",

"role": 2

}'

Request :- {

"name": "School Admin 2",

"email": "schooladmin2@yopmail.com",

"password": "schooladmin@123",

"role": 2}

Response :- {

"code": 200,

"message": "User updated successfully.",

"result": {

"\_id": "6764279463d05e3d0172839f",

"name": "School Admin 2",

"email": "schooladmin2@yopmail.com",

"password": "schooladmin@123",

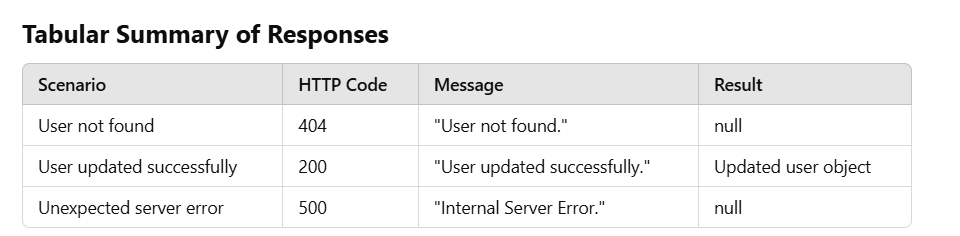
"role": 2,

"schoolIds": [],

"\_\_v": 0

}

}

Api Flow :- The update controller handles updating a user's details in the database. It begins by extracting the id parameter from the request URL and checks if a user with the provided ID exists in the database. If no user is found, a 404 status code is returned with the message "User not found.". If the user exists, the controller updates the user's details using the data provided in the request body and saves the changes. Upon successful update, it returns a 200 status code with the message "User updated successfully." and the updated user data. If any unexpected error occurs during the process, a 500 status code is returned with the message "Internal Server Error.". This flow ensures proper handling of user updates and provides clear feedback for success or failure scenarios.  
  


Invalid Scenario

Request :- {

"name": "School Admin 2",

"email": "schooladmin2@yopmail.com",

"password": "schooladmin@123"

}

Response :- {

"code": 400,

"message": "Validation failed.",

"result": [

"\"role\" is required"

]

}

**5. Delete User Api**

Curl :- curl --location --request DELETE 'http://localhost:8000/sms/user/delete/6764279463d05e3d0172839f' \

--header 'Authorization: Bearer eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJpZCI6IjY3NjE4OGQwOGYyM2U4MzQ5MTlhY2UxZiIsInJvbGUiOjEsImlhdCI6MTczNDYxNjg4OCwiZXhwIjoxNzM0NjIwNDg4fQ.mZhjrTxbTWOv\_9jcskwRE6P2FwL5EJCU1BzAibNUEcc'

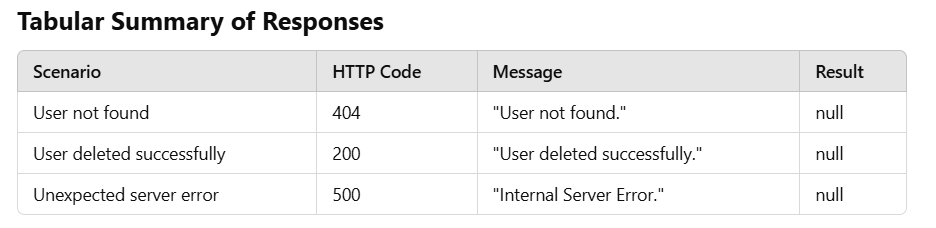
Response :- {

"code": 200,

"message": "User deleted successfully.",

"result": **null**

}

Api Flow :- The deleteUser controller is responsible for deleting a user from the database. It begins by extracting the id parameter from the request URL and checks if a user with the given ID exists in the database. If no user is found, a 404 status code is returned with the message "User not found.". If the user exists, the controller deletes the user using the provided ID. Upon successful deletion, it returns a 200 status code with the message "User deleted successfully.". If any unexpected error occurs during the process, a 500 status code is returned with the message "Internal Server Error.". This flow ensures secure and efficient deletion of user records while providing clear feedback for success or failure scenarios.  
  


Invalid Scenarios   
  
Response :- {

"code": 400,

"message": "Invalid parameters.",

"result": [

"\"id\" must be a valid 24-character hexadecimal string"

]

}

**6. Create School Api**

Curl :- curl --location 'http://localhost:8000/sms/school/create' \

--header 'Content-Type: application/json' \

--header 'Authorization: Bearer eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJpZCI6IjY3NjE4OGQwOGYyM2U4MzQ5MTlhY2UxZiIsInJvbGUiOjEsImlhdCI6MTczNDYxNjg4OCwiZXhwIjoxNzM0NjIwNDg4fQ.mZhjrTxbTWOv\_9jcskwRE6P2FwL5EJCU1BzAibNUEcc' \

--data '{

"name": "ABCD School",

"address": "Palestine"

}'

Request :- {

"name": "ABCD School",

"address": "Palestine"

}

Response :- {

"code": 200,

"message": "School created successfully.",

"result": {

"name": "ABCD School",

"address": "Palestine",

"classIds": [],

"\_id": "67642cce63d05e3d017283a7",

"createdBy": "676188d08f23e834919ace1f",

"createdAt": "2024-12-19T14:25:18.987Z",

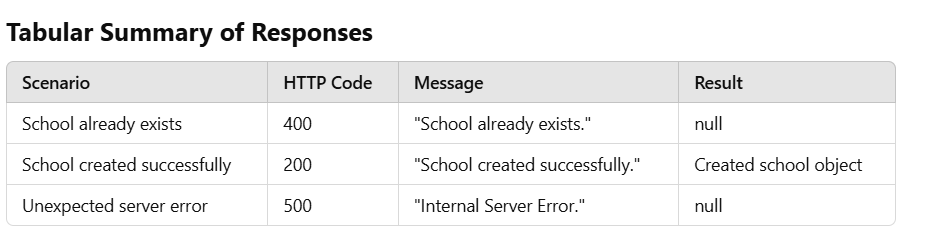
"updatedAt": "2024-12-19T14:25:18.987Z",

"\_\_v": 0

}

}

Api Flow :- The create controller is responsible for adding a new school to the database. It begins by creating a new School instance using the data provided in the request body and extracting the name field. The controller checks the database to determine if a school with the given name already exists. If such a school exists, a 400 status code is returned with the message "School already exists.". If no school with the same name exists, the createdBy field is set to the ID of the currently logged-in user (retrieved from req.user.id), and the new school is saved to the database. Upon successful creation, a 200 status code is returned with the message "School created successfully." and the details of the newly created school. If an unexpected error occurs during the process, a 500 status code is returned with the message "Internal Server Error.". This flow ensures the uniqueness of school names and provides clear feedback for both success and failure scenarios.



Invalid Scenario  
  
Request :- {

"name": "ABCD School"

}

Response :- {

"code": 400,

"message": "Validation failed.",

"result": [

"\"address\" is required"

]

}

**7. Get Schools Api**

Curl :- curl --location 'http://localhost:8000/sms/school/getSchools' \

--header 'Authorization: Bearer eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJpZCI6IjY3NjE4OGQwOGYyM2U4MzQ5MTlhY2UxZiIsInJvbGUiOjEsImlhdCI6MTczNDYxNjg4OCwiZXhwIjoxNzM0NjIwNDg4fQ.mZhjrTxbTWOv\_9jcskwRE6P2FwL5EJCU1BzAibNUEcc'

Response :- {

"code": 200,

"message": "Schools fetched successfully.",

"result": [

{

"\_id": "676331bb9d5f7594b2aa4d90",

"name": "St. Thomas Convent School",

"address": "Ujjain",

"classIds": [

"676332759d5f7594b2aa4d9b",

"6763327b9d5f7594b2aa4da0"

],

"createdBy": "676188d08f23e834919ace1f",

"createdAt": "2024-12-18T20:34:03.245Z",

"updatedAt": "2024-12-18T20:37:15.357Z",

"\_\_v": 2

},

{

"\_id": "676331c49d5f7594b2aa4d93",

"name": "Carmel Convent School",

"address": "Ujjain",

"classIds": [

"676332899d5f7594b2aa4da5"

],

"createdBy": "676188d08f23e834919ace1f",

"createdAt": "2024-12-18T20:34:12.600Z",

"updatedAt": "2024-12-18T20:37:29.464Z",

"\_\_v": 1

},

{

"\_id": "67642cce63d05e3d017283a7",

"name": "ABCD School",

"address": "Palestine",

"classIds": [],

"createdBy": "676188d08f23e834919ace1f",

"createdAt": "2024-12-19T14:25:18.987Z",

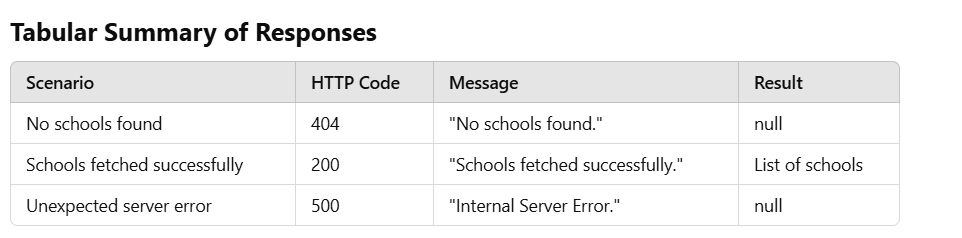
"updatedAt": "2024-12-19T14:25:18.987Z",

"\_\_v": 0

}

]

}

Api Flow :- The fetch controller is responsible for retrieving all schools from the database. It begins by querying the School collection to fetch all school records. If no schools are found, a 404 status code is returned with the message "No schools found.". If schools are found, the controller responds with a 200 status code, the message "Schools fetched successfully.", and the list of schools. If an unexpected error occurs during the process, a 500 status code is returned with the message "Internal Server Error.". This flow ensures efficient retrieval of school data and provides clear responses for both success and failure scenarios.  
  


**8. Get School By Id Api**

Curl :- curl --location 'http://localhost:8000/sms/school/getSchools/67642cce63d05e3d017283a7' \

--header 'Authorization: Bearer eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJpZCI6IjY3NjE4OGQwOGYyM2U4MzQ5MTlhY2UxZiIsInJvbGUiOjEsImlhdCI6MTczNDYyMDc5NywiZXhwIjoxNzM0NjI0Mzk3fQ.MekNqGuxu\_iy\_GC19dv67PSmrEwJy1YyrrDoVepkIcY'

Response :- {

"code": 200,

"message": "School fetched successfully.",

"result": {

"\_id": "67642cce63d05e3d017283a7",

"name": "ABCD School",

"address": "Palestine",

"classIds": [],

"createdBy": "676188d08f23e834919ace1f",

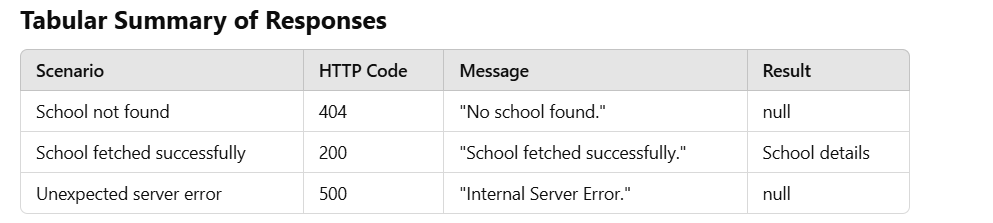
"createdAt": "2024-12-19T14:25:18.987Z",

"updatedAt": "2024-12-19T14:25:18.987Z",

"\_\_v": 0

}

}

Api Flow :- The fetchById controller is designed to retrieve a specific school by its ID from the database. It begins by extracting the id parameter from the request URL and querying the School collection for a record matching the given ID. If no school is found with the provided ID, a 404 status code is returned with the message "No school found.". If the school exists, the controller responds with a 200 status code, the message "School fetched successfully.", and the school details. If any unexpected error occurs during the process, a 500 status code is returned with the message "Internal Server Error.". This flow ensures precise and secure retrieval of individual school data, providing clear responses for both success and failure cases.  
  
  
Invalid Scenario  
Response :- {

"code": 400,

"message": "Invalid parameters.",

"result": [

"\"id\" must be a valid 24-character hexadecimal string"

]

}

**9. Update School By Id Api**

Curl :- curl --location --request PUT 'http://localhost:8000/sms/school/updateSchool/67642cce63d05e3d017283a7' \

--header 'Content-Type: application/json' \

--header 'Authorization: Bearer eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJpZCI6IjY3NjE4OGQwOGYyM2U4MzQ5MTlhY2UxZiIsInJvbGUiOjEsImlhdCI6MTczNDYyMDc5NywiZXhwIjoxNzM0NjI0Mzk3fQ.MekNqGuxu\_iy\_GC19dv67PSmrEwJy1YyrrDoVepkIcY' \

--data '{

"name": "ABCD School",

"address": "Gaza"

}'

Request :- {

"name": "ABCD School",

"address": "Gaza"

}

Response :- {

"code": 200,

"message": "School updatd successfully.",

"result": {

"\_id": "67642cce63d05e3d017283a7",

"name": "ABCD School",

"address": "Gaza",

"classIds": [],

"createdBy": "676188d08f23e834919ace1f",

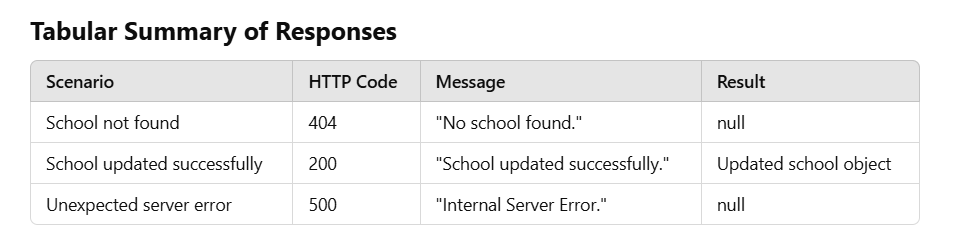
"createdAt": "2024-12-19T14:25:18.987Z",

"updatedAt": "2024-12-19T15:11:50.189Z",

"\_\_v": 0

}

}

Api Flow :- The update controller is responsible for updating the details of a specific school. It begins by extracting the id parameter from the request URL and querying the School collection to check if a school with the given ID exists. If no school is found, a 404 status code is returned with the message "No school found.". If the school exists, the name and address fields from the request body are used to update the corresponding fields in the school record. If a field is not provided in the request, the existing value is retained. The updated school record is saved back to the database using findByIdAndUpdate, and the response is sent with a 200 status code, the message "School updated successfully.", and the updated school details. If any unexpected error occurs during the process, a 500 status code is returned with the message "Internal Server Error.". This flow ensures efficient updates to school records while providing clear feedback for both success and failure scenarios.  
  
Invalid Scenarios

Request :- {

"name": "ABCD School"

}

Response :- {

"code": 400,

"message": "Validation failed.",

"result": [

"\"address\" is required"

]

}

**10. Delete School By Id Api**

Curl :- curl --location --request DELETE 'http://localhost:8000/sms/school/deleteSchool/67642cce63d05e3d017283a7' \

--header 'Authorization: Bearer eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJpZCI6IjY3NjE4OGQwOGYyM2U4MzQ5MTlhY2UxZiIsInJvbGUiOjEsImlhdCI6MTczNDYyMDc5NywiZXhwIjoxNzM0NjI0Mzk3fQ.MekNqGuxu\_iy\_GC19dv67PSmrEwJy1YyrrDoVepkIcY'

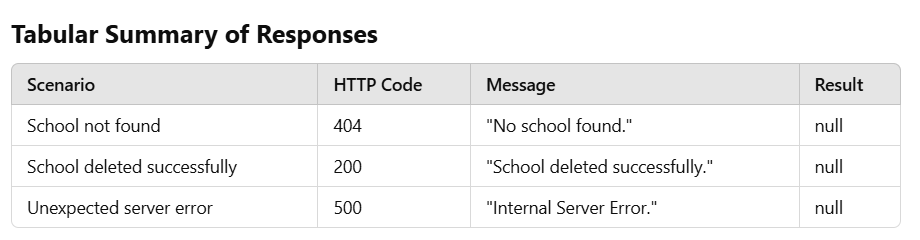
Response :- {

"code": 200,

"message": "School deleted successfully.",

"result": **null**

}

Api Flow :- The deleteSchool controller is responsible for deleting a school and all associated data from the database. It begins by extracting the school ID (id) from the request URL and checking if a school with the given ID exists in the database. If no school is found, a 404 status code is returned with the message "No school found.". If the school exists, the controller cascades the deletion process to remove related entities. First, it retrieves all users linked to the school by checking their schoolIds array. For each user, the school ID is removed from their schoolIds, and the user data is updated in the database. Next, it fetches all classes associated with the school and iterates through them to retrieve their students. Each student in the class is deleted, followed by the deletion of the class itself. Once all related users, classes, and students are removed, the school is deleted from the database. Upon successful completion, a 200 status code is returned with the message "School deleted successfully.". If any unexpected error occurs during the process, a 500 status code is returned with the message "Internal Server Error.". This flow ensures a complete and consistent removal of a school and its dependencies.  
  
  
Invalid Scenario  
  
Response :- {

"code": 400,

"message": "Invalid parameters.",

"result": [

"\"id\" must be a valid 24-character hexadecimal string"

]

}

**11. Create Class Api**

Curl :- curl --location 'http://localhost:8000/sms/class/create' \

--header 'Content-Type: application/json' \

--header 'Authorization: Bearer eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJpZCI6IjY3NjE4OGQwOGYyM2U4MzQ5MTlhY2UxZiIsInJvbGUiOjEsImlhdCI6MTczNDYyMDc5NywiZXhwIjoxNzM0NjI0Mzk3fQ.MekNqGuxu\_iy\_GC19dv67PSmrEwJy1YyrrDoVepkIcY' \

--data '{

"name": "KG1",

"capacity": 5,

"resources": [

"LED"

],

"schoolId": "676439813d6b1eff117479d2"

}'

Request :- {

"name": "KG1",

"capacity": 5,

"resources": [

"LED"

],

"schoolId": "676439813d6b1eff117479d2"

}

Response :- {

"code": 200,

"message": "Class created successfully.",

"result": {

"name": "KG1",

"capacity": 5,

"resources": [

"LED"

],

"studentIds": [],

"schoolId": "676439813d6b1eff117479d2",

"\_id": "6764399d3d6b1eff117479d5",

"createdAt": "2024-12-19T15:19:58.014Z",

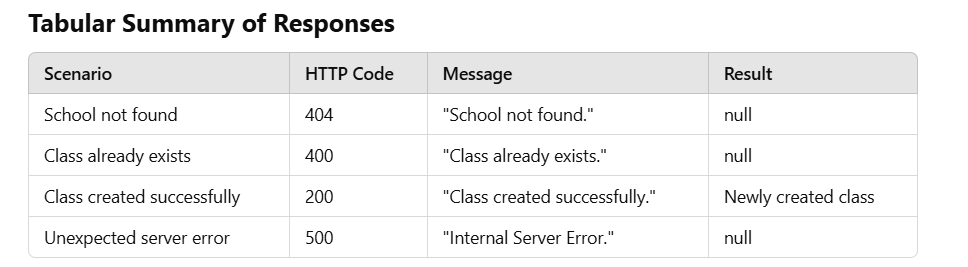
"updatedAt": "2024-12-19T15:19:58.014Z",

"\_\_v": 0

}

}

Api Flow :- The create controller handles the creation of a new class within a school. It begins by extracting the class data from the request body and initializing a new instance of the Class model. The controller verifies if the schoolId provided in the class data exists in the database by querying the School collection. If no school is found, a 404 status code is returned with the message "School not found.". If the school exists, it checks whether a class with the same name already exists for the given schoolId. If a matching class is found, a 400 status code is returned with the message "Class already exists.". If no duplicate is found, the class is saved to the database. Once saved, the new class's ID is added to the classIds array of the associated school, and the school record is updated. Upon successful creation, a 200 status code is returned with the message "Class created successfully." along with the newly created class details. If any unexpected error occurs during the process, a 500 status code is returned with the message "Internal Server Error.". This flow ensures the integrity of school-class relationships and provides clear feedback for success or failure cases.



Invalid Scenario

Request :- {

"name": "KG1",

"capacity": 5,

"resources": [

// "LED"

],

"schoolId": "676439813d6b1eff117479d2"

}

Response :- {

"code": 400,

"message": "Validation failed.",

"result": [

"\"resources\" must contain at least 1 items"

]

}

**12. Get Classes By School Id Api**

Curl :- curl --location 'http://localhost:8000/sms/class/fetchBySchoolId/676439813d6b1eff117479d2' \

--header 'Authorization: Bearer eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJpZCI6IjY3NjE4OGQwOGYyM2U4MzQ5MTlhY2UxZiIsInJvbGUiOjEsImlhdCI6MTczNDYyMDc5NywiZXhwIjoxNzM0NjI0Mzk3fQ.MekNqGuxu\_iy\_GC19dv67PSmrEwJy1YyrrDoVepkIcY'

Response :- {

"code": 200,

"message": "Classes fetched successfully.",

"result": [

{

"\_id": "6764399d3d6b1eff117479d5",

"name": "KG1",

"capacity": 5,

"resources": [

"LED"

],

"studentIds": [],

"schoolId": "676439813d6b1eff117479d2",

"createdAt": "2024-12-19T15:19:58.014Z",

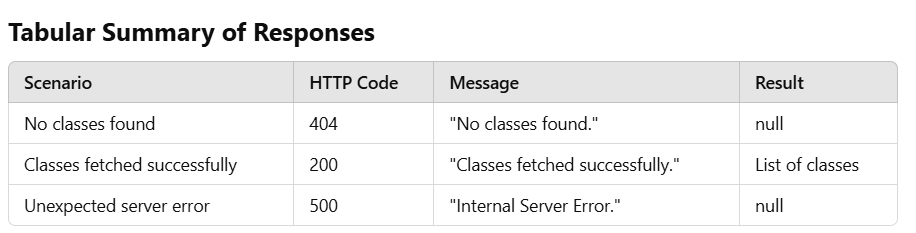
"updatedAt": "2024-12-19T15:19:58.014Z",

"\_\_v": 0

}

]

}

Api Flow :- The fetchBySchoolId controller retrieves all classes associated with a specific school. It begins by extracting the id parameter from the request URL, representing the school ID, and querying the Class collection to find all classes with the matching schoolId. If no classes are found, a 404 status code is returned with the message "No classes found.". If one or more classes are found, the controller responds with a 200 status code, the message "Classes fetched successfully.", and the list of classes. If an unexpected error occurs during the process, a 500 status code is returned with the message "Internal Server Error.". This flow ensures efficient retrieval of classes for a specific school, providing appropriate feedback for both success and failure scenarios.  
  
  
  
Invalid Scenarios  
  
Response :- {

"code": 400,

"message": "Invalid parameters.",

"result": [

"\"id\" must be a valid 24-character hexadecimal string"

]

}

**13. Get Classe By Id Api**

Curl :- curl --location 'http://localhost:8000/sms/class/fetchById/6764399d3d6b1eff117479d5' \

--header 'Authorization: Bearer eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJpZCI6IjY3NjE4OGQwOGYyM2U4MzQ5MTlhY2UxZiIsInJvbGUiOjEsImlhdCI6MTczNDYyMDc5NywiZXhwIjoxNzM0NjI0Mzk3fQ.MekNqGuxu\_iy\_GC19dv67PSmrEwJy1YyrrDoVepkIcY'

Response :- {

"code": 200,

"message": "Class fetched successfully.",

"result": {

"\_id": "6764399d3d6b1eff117479d5",

"name": "KG1",

"capacity": 5,

"resources": [

"LED"

],

"studentIds": [],

"schoolId": "676439813d6b1eff117479d2",

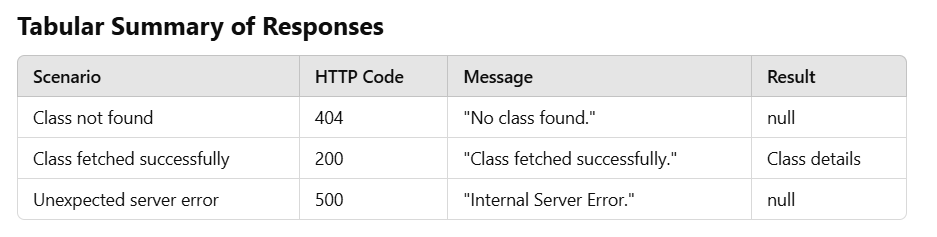
"createdAt": "2024-12-19T15:19:58.014Z",

"updatedAt": "2024-12-19T15:19:58.014Z",

"\_\_v": 0

}

}

Api Flow :- The fetchById controller is responsible for retrieving a specific class by its ID. It begins by extracting the id parameter from the request URL and querying the Class collection to find a class that matches the given ID. If no class is found, a 404 status code is returned with the message "No class found.". If a class is found, the controller responds with a 200 status code, the message "Class fetched successfully.", and the details of the class. If an unexpected error occurs during the process, a 500 status code is returned with the message "Internal Server Error.". This flow ensures precise and secure retrieval of individual class details, providing clear feedback for both success and failure cases.  
  
  
Invalid Scenario  
Response :- {

"code": 400,

"message": "Invalid parameters.",

"result": [

"\"id\" must be a valid 24-character hexadecimal string"

]

}

**14. Update Classe By Id Api**

Curl :- curl --location --request PUT 'http://localhost:8000/sms/class/update/6764399d3d6b1eff117479d5' \

--header 'Content-Type: application/json' \

--header 'Authorization: Bearer eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJpZCI6IjY3NjE4OGQwOGYyM2U4MzQ5MTlhY2UxZiIsInJvbGUiOjEsImlhdCI6MTczNDYyMDc5NywiZXhwIjoxNzM0NjI0Mzk3fQ.MekNqGuxu\_iy\_GC19dv67PSmrEwJy1YyrrDoVepkIcY' \

--data '{

"name": "KG1",

"capacity": 5,

"resources": [

"LED",

"BOOK"

],

"schoolId": "676439813d6b1eff117479d2"

}'

Request :- {

"name": "KG1",

"capacity": 5,

"resources": [

"LED",

"BOOK"

],

"schoolId": "676439813d6b1eff117479d2"

}

Response :- {

"code": 200,

"message": "Class updatd successfully.",

"result": {

"\_id": "6764399d3d6b1eff117479d5",

"name": "KG1",

"capacity": 5,

"resources": [

"LED",

"BOOK"

],

"studentIds": [],

"schoolId": "676439813d6b1eff117479d2",

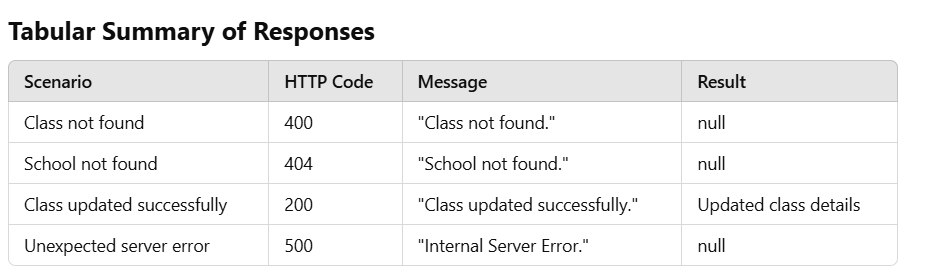
"createdAt": "2024-12-19T15:19:58.014Z",

"updatedAt": "2024-12-19T15:39:48.791Z",

"\_\_v": 0

}

}

Api Flow :- The update controller handles updating the details of a specific class in the database. It begins by extracting the class ID from the request URL and retrieving the class record from the Class collection. If no class is found with the given ID, a 400 status code is returned with the message "Class not found.". Next, it verifies whether the associated school (identified by the schoolId in the class record) exists in the School collection. If the school does not exist, a 404 status code is returned with the message "School not found.". If both the class and school are valid, the controller updates the class properties (name, schoolId, capacity, resources) with the new values provided in the request body, while retaining existing values for fields not included in the update. The updated class record is saved, and the response is returned with a 200 status code, the message "Class updated successfully.", and the updated class details. If an unexpected error occurs, a 500 status code is returned with the message "Internal Server Error.". This flow ensures efficient updates while maintaining data consistency between schools and their associated classes.  
  
  
  
Invalid Scenario  
  
Request :- {

"name": "KG1",

"capacity": 5,

"resources": [

// "LED",

// "BOOK"

],

"schoolId": "676439813d6b1eff117479d2"

}

Response :- {

"code": 400,

"message": "Validation failed.",

"result": [

"\"resources\" must contain at least 1 items"

]

}

**15. Delete Class By Id Api** Curl :- curl --location --request DELETE 'http://localhost:8000/sms/class/deleteClass/676332759d5f7594b2aa4d9b' \

--header 'Authorization: Bearer eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJpZCI6IjY3NjE4OGQwOGYyM2U4MzQ5MTlhY2UxZiIsInJvbGUiOjEsImlhdCI6MTczNDYyMDc5NywiZXhwIjoxNzM0NjI0Mzk3fQ.MekNqGuxu\_iy\_GC19dv67PSmrEwJy1YyrrDoVepkIcY'

Response :- {

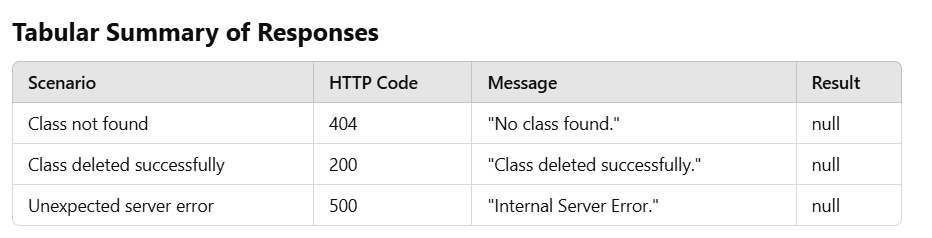
"code": 200,

"message": "Class deleted successfully.",

"result": **null**

}

Api Flow :- The deleteClass controller is responsible for deleting a specific class and all associated data, including references in the related school and student entities. It starts by extracting the class ID (id) from the request URL and querying the Class collection to check if the class exists. If no class is found, a 404 status code is returned with the message "No class found.". If the class exists, the controller retrieves the associated school from the School collection using the classIds field. The class ID is then removed from the school's classIds array, and the school record is updated. Next, all students belonging to the class are retrieved, and each student is deleted. Finally, the class itself is deleted from the Class collection. Upon successful deletion, a 200 status code is returned with the message "Class deleted successfully.". If any unexpected error occurs during the process, a 500 status code is returned with the message "Internal Server Error.". This flow ensures consistent and complete removal of the class and its dependencies while maintaining the integrity of related data.

  
Invalid Scenario  
  
Response :- {

"code": 400,

"message": "Invalid parameters.",

"result": [

"\"id\" must be a valid 24-character hexadecimal string"

]

}

**16. Create Student Api**

Curl :- curl --location 'http://localhost:8000/sms/student/create' \

--header 'Content-Type: application/json' \

--header 'Authorization: Bearer eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJpZCI6IjY3NjE4OGQwOGYyM2U4MzQ5MTlhY2UxZiIsInJvbGUiOjEsImlhdCI6MTczNDYyMDc5NywiZXhwIjoxNzM0NjI0Mzk3fQ.MekNqGuxu\_iy\_GC19dv67PSmrEwJy1YyrrDoVepkIcY' \

--data '{

"name" : "Moss",

"age" : 20,

"gender" : "Male",

"classId" : "6764399d3d6b1eff117479d5"

}'

Request :- {

"name" : "Moss",

"age" : 15,

"gender" : "Male",

"classId" : "6764399d3d6b1eff117479d5"

}

Response :- {

"code": 200,

"message": "Student created successfully.",

"result": {

"name": "Moss",

"age": 15,

"gender": "Male",

"classId": "6764399d3d6b1eff117479d5",

"\_id": "676440428d5386d6c12088f8",

"createdAt": "2024-12-19T15:48:18.529Z",

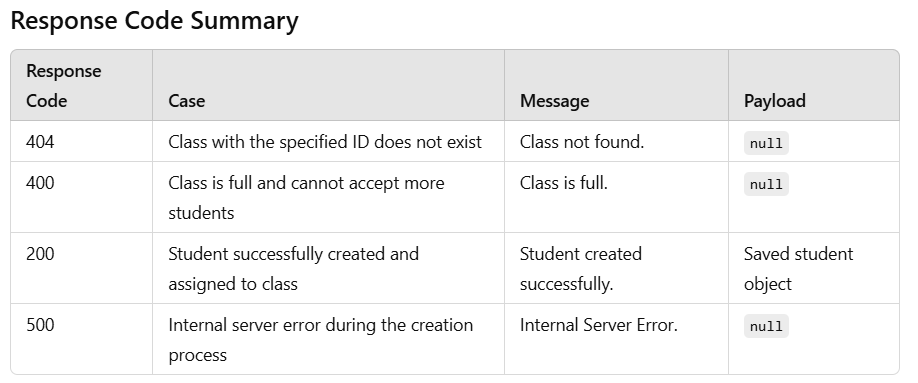
"updatedAt": "2024-12-19T15:48:18.529Z",

"\_\_v": 0

}

}

Api Flow :- The create method is designed to handle the creation of a new student and its assignment to a class. It begins by creating a new Student object from the data sent in the request body (req.body). The method then checks if the class specified by the student's classId exists by querying the Class collection with Class.findById(studentData.classId). If the class is not found, it returns a 404 status code with a message indicating that the class doesn't exist. If the class exists, it checks if the class has reached its maximum capacity by comparing the number of students already enrolled (classExist.studentIds.length) with the class capacity (classExist.capacity). If the class is full, it returns a 400 status code, indicating that the class cannot accept more students. If the class has space, the student data is saved to the database, and the student's ID is added to the studentIds array of the class. The updated class is then saved, and a successful response with a 200 status code and the saved student data is returned. In case of an error, a 500 status code is returned to indicate an internal server error.

  
Invalid Scenario  
  
Request :- {

"name" : "Moss",

"age" : 15,

"gender" : "Other",

"classId" : "6764399d3d6b1eff117479d5"

}

Response :- {

"code": 400,

"message": "Validation failed.",

"result": [

"\"gender\" must be one of [Male, Female]"

]

}

**17. Get Student By Class Id Api**

Curl :- curl --location 'http://localhost:8000/sms/student/fetchByClassId/6764399d3d6b1eff117479d5' \

--header 'Authorization: Bearer eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJpZCI6IjY3NjE4OGQwOGYyM2U4MzQ5MTlhY2UxZiIsInJvbGUiOjEsImlhdCI6MTczNDYyMDc5NywiZXhwIjoxNzM0NjI0Mzk3fQ.MekNqGuxu\_iy\_GC19dv67PSmrEwJy1YyrrDoVepkIcY'

Response :- {

"code": 200,

"message": "Students fetched successfully.",

"result": [

{

"\_id": "676440428d5386d6c12088f8",

"name": "Moss",

"age": 15,

"gender": "Male",

"classId": "6764399d3d6b1eff117479d5",

"createdAt": "2024-12-19T15:48:18.529Z",

"updatedAt": "2024-12-19T15:48:18.529Z",

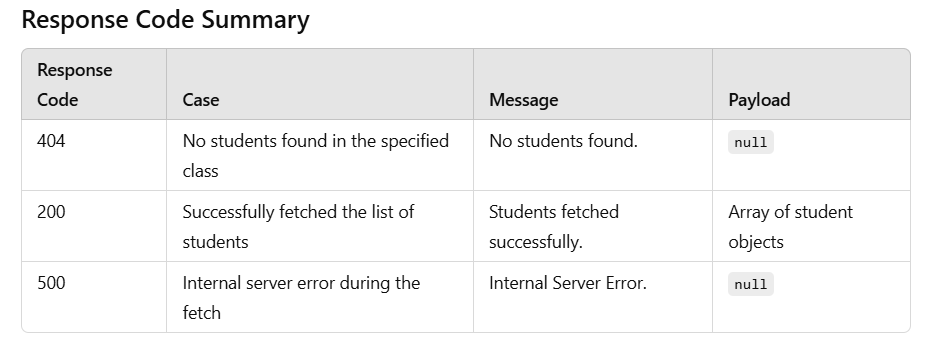
"\_\_v": 0

}

]

}

Api Flow :- The fetchByClassId method is designed to retrieve a list of students enrolled in a specific class based on the class ID provided in the request. The method begins by extracting the class ID from the request parameters using req.params.id. It then queries the Student collection to find all students with the specified classId using Student.find({ classId: req.params.id }). If no students are found for the given class, the method returns a 404 status code with a message indicating that no students were found, and the payload is set to null. If students are found, the method returns a 200 status code with a success message and includes the list of students in the response payload. In case of any error during the operation (such as a database issue), the method catches the error and returns a 500 status code indicating an internal server error, with a null payload.

  
Invalid Scenario

Response :- {

"code": 400,

"message": "Invalid parameters.",

"result": [

"\"id\" must be a valid 24-character hexadecimal string"

]

}

**18. Get Student By Student Id Api**

Curl :- curl --location 'http://localhost:8000/sms/student/fetchById/676440428d5386d6c12088f8' \

--header 'Authorization: Bearer eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJpZCI6IjY3NjE4OGQwOGYyM2U4MzQ5MTlhY2UxZiIsInJvbGUiOjEsImlhdCI6MTczNDYyMDc5NywiZXhwIjoxNzM0NjI0Mzk3fQ.MekNqGuxu\_iy\_GC19dv67PSmrEwJy1YyrrDoVepkIcY'

Response :- {

"code": 200,

"message": "Student fetched successfully.",

"result": {

"\_id": "676440428d5386d6c12088f8",

"name": "Moss",

"age": 15,

"gender": "Male",

"classId": "6764399d3d6b1eff117479d5",

"createdAt": "2024-12-19T15:48:18.529Z",

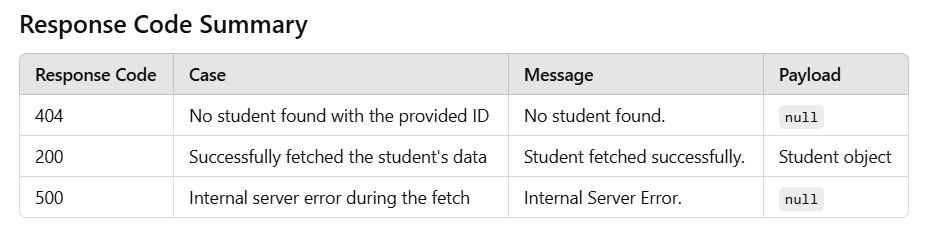
"updatedAt": "2024-12-19T15:48:18.529Z",

"\_\_v": 0

}

}

Api Flow :- The fetchById method is designed to retrieve a specific student's data based on the student ID provided in the request. The method begins by extracting the id parameter from the request's URL using req.params.id. It then queries the Student collection to find the student with the specified \_id using Student.findOne({ \_id: id }). If no student is found with the provided ID, the method returns a 404 status code with a message indicating that no student was found, and the payload is set to null. If the student is found, it responds with a 200 status code, signaling a successful retrieval, and includes the student data in the response payload. In case of any errors during the operation (such as a database issue), the method catches the error and returns a 500 status code with a message indicating an internal server error, and the payload is null.

  
Invalid Scenario

Response :- {

"code": 400,

"message": "Invalid parameters.",

"result": [

"\"id\" must be a valid 24-character hexadecimal string"

]

}

**19. Update Student By Id Api**

Curl :- curl --location --request PUT 'http://localhost:8000/sms/student/update/676440428d5386d6c12088f8' \

--header 'Content-Type: application/json' \

--header 'Authorization: Bearer eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJpZCI6IjY3NjE4OGQwOGYyM2U4MzQ5MTlhY2UxZiIsInJvbGUiOjEsImlhdCI6MTczNDYyNDQ1MSwiZXhwIjoxNzM0NjI4MDUxfQ.f09ZIa24UQRQP6EA97MDrJ-ISNtwDV3Y4D5DjWx4paQ' \

--data '{

"name" : "Ana",

"age" : 15,

"gender" : "Female",

"classId" : "6764399d3d6b1eff117479d5"

}'

Request :- {

"name" : "Ana",

"age" : 15,

"gender" : "Female",

"classId" : "6764399d3d6b1eff117479d5"

}

Response :- {

"code": 200,

"message": "Student updatd successfully.",

"result": {

"\_id": "676440428d5386d6c12088f8",

"name": "Ana",

"age": 15,

"gender": "Female",

"classId": "6764399d3d6b1eff117479d5",

"createdAt": "2024-12-19T15:48:18.529Z",

"updatedAt": "2024-12-19T16:14:22.841Z",

"\_\_v": 0

}

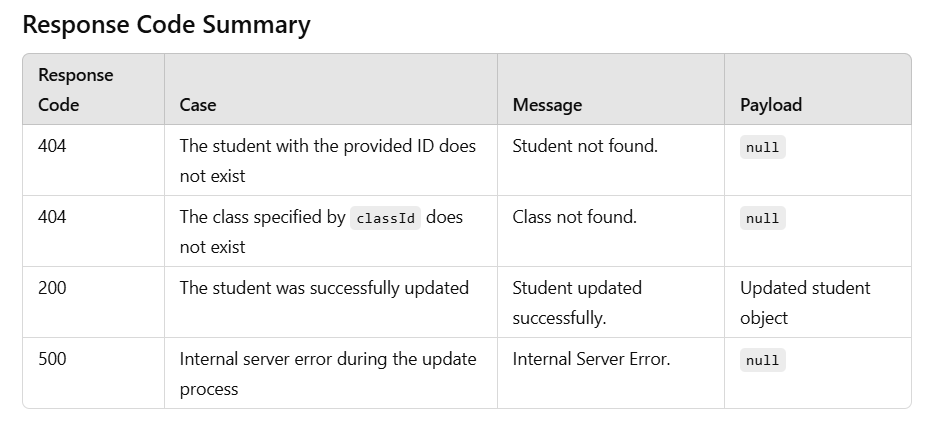
}

Api Flow :- The update method is designed to update the details of an existing student based on the student ID provided in the request. The method first creates a Student object from the request body (req.body). It then extracts the classId from the student data. The method begins by checking if a student with the provided ID exists using Student.findOne({\_id: req.params.id}). If the student is not found, it returns a 404 status code with a message indicating that the student was not found.

Next, it verifies if the class specified by the student's classId exists by querying the Class collection with Class.findById(studentData.classId). If the class doesn't exist, it returns a 404 status code with a message indicating that the class was not found.

If both the student and the class are found, the method proceeds to update the student's details. It assigns the new values from the request body (studentData) to the existing student's fields, only updating fields that have new values provided (using a fallback to the existing values if not provided). The method then performs the update operation using Student.findByIdAndUpdate() and sets the new: true option to ensure that the updated student data is returned.

Finally, it returns a 200 status code with a success message and the updated student data in the response. In case of an error during any step of the process, it catches the error and responds with a 500 status code indicating an internal server error, with a null payload.

  
Invalid Scenario  
  
Request :- {

"name" : "Ana",

"age" : 15,

"gender" : "Other",

"classId" : "6764399d3d6b1eff117479d5"

}

Response :- {

"code": 400,

"message": "Validation failed.",

"result": [

"\"gender\" must be one of [Male, Female]"

]

}

**20. Delete Student By Id Api**

Curl :- curl --location --request DELETE 'http://localhost:8000/sms/student/deleteStudent/676440428d5386d6c12088f8' \

--header 'Authorization: Bearer eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJpZCI6IjY3NjE4OGQwOGYyM2U4MzQ5MTlhY2UxZiIsInJvbGUiOjEsImlhdCI6MTczNDYyNDQ1MSwiZXhwIjoxNzM0NjI4MDUxfQ.f09ZIa24UQRQP6EA97MDrJ-ISNtwDV3Y4D5DjWx4paQ'

Response :- {

"code": 200,

"message": "Student deleted successfully.",

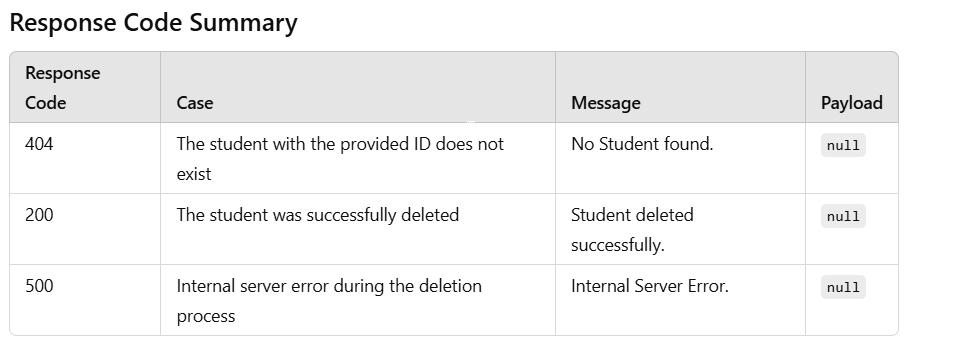
"result": **null**

}

Api Flow :- The deleteStudent method is designed to delete a student's record from the database and remove their reference from the associated class. The process begins by extracting the student ID from the request parameters (req.params.id). It then checks if the student exists by querying the Student collection using Student.findOne({ \_id: id }). If no student is found, the method returns a 404 status code with a message indicating that no student was found.

If the student exists, the method proceeds to find the corresponding class by querying the Class collection using Class.findOne({ \_id: studentData.classId }). It then removes the student's ID from the studentIds array of the associated class using the pull() method. After updating the class, the method saves the changes to the class document.

Next, the student's record is deleted using Student.findByIdAndDelete(id). Finally, the method responds with a 200 status code, indicating the successful deletion of the student, with a success message and a null payload. In case of any errors during the process (such as issues with database operations), the method catches the error and returns a 500 status code indicating an internal server error, with a null payload.

  
Invalid Scenario

Response :- {

"code": 400,

"message": "Invalid parameters.",

"result": [

"\"id\" must be a valid 24-character hexadecimal string"

]

}

**21. Map School Admin with Schools Api**

Curl :- curl --location 'http://localhost:8000/sms/map/schoolAdminWithSchools' \

--header 'Content-Type: application/json' \

--header 'Authorization: Bearer eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJpZCI6IjY3NjE4OGQwOGYyM2U4MzQ5MTlhY2UxZiIsInJvbGUiOjEsImlhdCI6MTczNDYyNDQ1MSwiZXhwIjoxNzM0NjI4MDUxfQ.f09ZIa24UQRQP6EA97MDrJ-ISNtwDV3Y4D5DjWx4paQ' \

--data '{

"schoolAdminId": "6761e45afc30b799b4e75ba9",

"schoolIds": [

"676439813d6b1eff117479d2"

]

}'

Request :- {

"schoolAdminId": "6761e45afc30b799b4e75ba9",

"schoolIds": [

"676439813d6b1eff117479d2"

]

}

Response :- {

"code": 200,

"message": "School admin updatd successfully with school ids.",

"result": {

"\_id": "6761e45afc30b799b4e75ba9",

"name": "School Admin 1",

"email": "schooladmin1@yopmail.com",

"password": "$2a$10$ftHIU8SbQqDCg6NsDdl0k.8QNWjEGE50vzJASTjOOw4HJWbI4.6ZK",

"role": 2,

"schoolIds": [

"676331bb9d5f7594b2aa4d90",

"676331c49d5f7594b2aa4d93",

"676439813d6b1eff117479d2"

],

"\_\_v": 8

}

}

Api Flow :- The allocateSchoolAdmin method is designed to allocate school IDs to a specific school admin by validating the provided data and ensuring that the admin has appropriate access. The process begins by extracting the schoolAdminId and schoolIds from the request body (req.body).

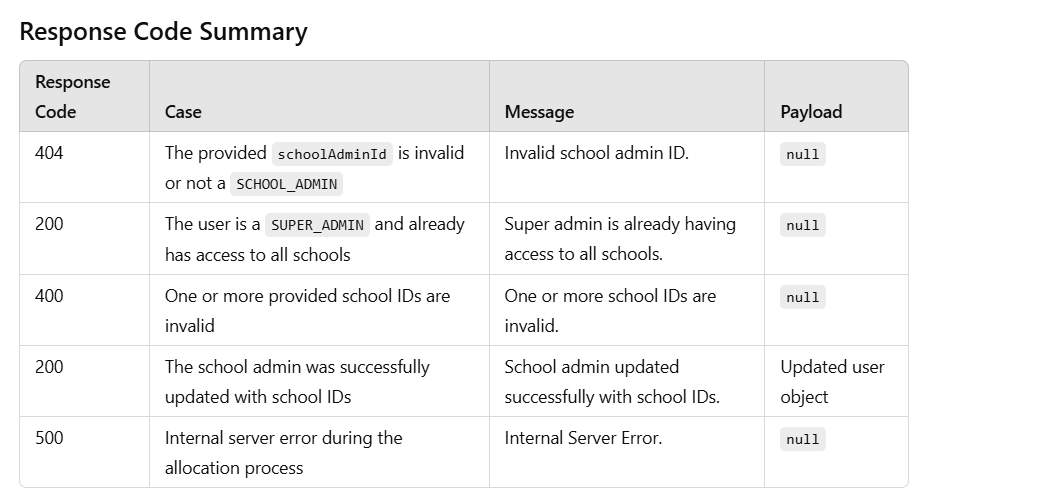
First, the method checks if a user exists with the provided schoolAdminId by querying the User collection using User.findById(schoolAdminId). If the user doesn't exist or if their role is not SCHOOL\_ADMIN, the method returns a 404 status code with a message indicating that the provided school admin ID is invalid.

If the user's role is SUPER\_ADMIN, which inherently has access to all schools, the method returns a 200 status code with a message indicating that a super admin already has access to all schools.

Next, the method validates the provided schoolIds by querying the School collection to check if all provided school IDs are valid. If any of the school IDs do not match a valid school, the method returns a 400 status code indicating that one or more school IDs are invalid.

If the school IDs are valid, the method proceeds to add the new school IDs to the schoolIds array of the user. To ensure no duplicates, the method creates a unique list of school IDs by combining the existing school IDs (if any) with the new ones, removing duplicates using new Set(). It then updates the user's schoolIds and saves the changes.

Finally, the method returns a 200 status code with a success message and the updated user data, indicating that the school admin has been successfully allocated with the specified school IDs. In case of any errors, a 500 status code is returned to indicate an internal server error.

  
Invalid Scenario  
  
Request :- {

"schoolAdminId": "6761e45afc30b799b4e75ba9",

"schoolIds": [

// "676439813d6b1eff117479d2"

]

}

Response :- {

"code": 400,

"message": "Validation failed.",

"result": [

"\"schoolIds\" does not contain 1 required value(s)",

"\"schoolIds\" must contain at least one item"

]

}

**22. Transfer Student Within School Api**

Curl :- curl --location 'http://localhost:8000/sms/map/transferWithinSchool' \

--header 'Content-Type: application/json' \

--header 'Authorization: Bearer eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJpZCI6IjY3NjE4OGQwOGYyM2U4MzQ5MTlhY2UxZiIsInJvbGUiOjEsImlhdCI6MTczNDYyNDQ1MSwiZXhwIjoxNzM0NjI4MDUxfQ.f09ZIa24UQRQP6EA97MDrJ-ISNtwDV3Y4D5DjWx4paQ' \

--data '{

"currentClassId" : "676332899d5f7594b2aa4da5",

"destinationClassId" : "6763327b9d5f7594b2aa4da0",

"studentId" : "6763334f9d5f7594b2aa4daf"

}'

Request :- {

"currentClassId" : "676332899d5f7594b2aa4da5",

"destinationClassId" : "6763327b9d5f7594b2aa4da0",

"studentId" : "6763334f9d5f7594b2aa4daf"

}

Response :- {

"code": 200,

"message": "Student transfered successfully.",

"result": **null**

}

Api Flow :- The transferWithinSchool method is designed to transfer a student from one class to another within the same school, ensuring that all necessary checks are performed before the transfer occurs. The process begins by extracting the currentClassId, destinationClassId, and studentId from the request body (req.body).

First, the method checks if the currentClassId and destinationClassId are the same. If they are, it returns a 400 status code indicating that the classes cannot be the same.

Next, it queries the Class collection to verify that both the current and destination classes exist. If either class is not found, the method returns a 404 status code with a message indicating that the class was not found.

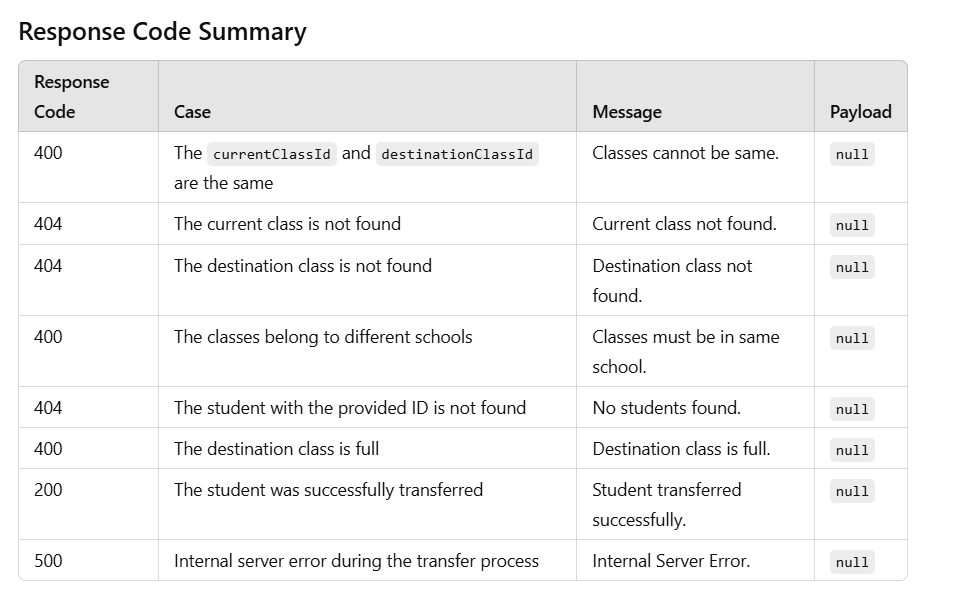
The method then ensures that both classes belong to the same school by comparing their schoolId properties. If the classes belong to different schools, it returns a 400 status code with a message indicating that the classes must be in the same school.

It then queries the Student collection to check if the student exists. If no student is found, it returns a 404 status code indicating that the student was not found.

Next, it checks if the destination class has reached its capacity by comparing the length of the studentIds array in the destination class with its capacity. If the class is full, the method returns a 400 status code indicating that the destination class is full.

If all the checks pass, the method proceeds to transfer the student. It adds the student's ID to the studentIds array of the destination class and removes the student's ID from the studentIds array of the current class. It also updates the student's classId to the new destinationClassId. The method then saves the changes to the student, destination class, and current class.

Finally, the method returns a 200 status code with a success message indicating that the student was successfully transferred. In case of any errors during the process, the method returns a 500 status code to indicate an internal server error.



Invalid Scenario  
  
Request :- {

"currentClassId" : "676332899d5f7594b2aa4da5",

// "destinationClassId" : "6763327b9d5f7594b2aa4da0",

"studentId" : "6763334f9d5f7594b2aa4daf"

}

Response :- {

"code": 400,

"message": "Validation failed.",

"result": [

"\"destinationClassId\" is required"

]

}

**23. Transfer Student Between School Api**

Curl :- curl --location 'http://localhost:8000/sms/map/transferBetweenSchool' \

--header 'Content-Type: application/json' \

--header 'Authorization: Bearer eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJpZCI6IjY3NjE4OGQwOGYyM2U4MzQ5MTlhY2UxZiIsInJvbGUiOjEsImlhdCI6MTczNDYyNDQ1MSwiZXhwIjoxNzM0NjI4MDUxfQ.f09ZIa24UQRQP6EA97MDrJ-ISNtwDV3Y4D5DjWx4paQ' \

--data '{

"currentClassId" : "6763327b9d5f7594b2aa4da0",

"destinationClassId" : "6764399d3d6b1eff117479d5",

"studentId" : "6763334f9d5f7594b2aa4daf"

}'

Request :- {

"currentClassId" : "6763327b9d5f7594b2aa4da0",

"destinationClassId" : "6764399d3d6b1eff117479d5",

"studentId" : "6763334f9d5f7594b2aa4daf"

}

Response :- {

"code": 200,

"message": "Student transfered successfully.",

"result": **null**

}

Api Flow :- The transferBetweenSchool method is designed to transfer a student from one class to another across different schools, ensuring that all necessary conditions are met before completing the transfer. The process begins by extracting the currentClassId, destinationClassId, and studentId from the request body (req.body).

First, the method checks if the currentClassId and destinationClassId are the same. If they are, it returns a 400 status code, indicating that the classes cannot be the same.

The method then queries the Class collection to verify that both the current and destination classes exist. If either class is not found, it returns a 404 status code with a message indicating that the respective class was not found.

Next, it checks whether both classes belong to different schools. If the schoolId of the current class is the same as the schoolId of the destination class, the method returns a 400 status code with a message indicating that the classes must be in different schools.

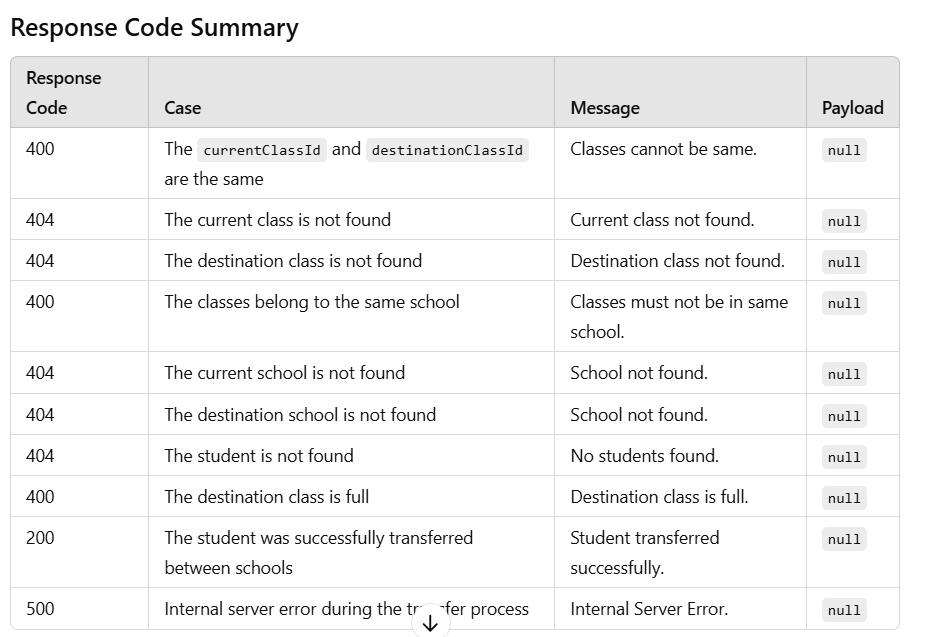
The method then queries the School collection to validate the existence of the schools associated with both classes. If either school is not found, it returns a 404 status code indicating that the school was not found.

The method then checks if the student exists by querying the Student collection using the provided studentId. If no student is found, it returns a 404 status code.

Next, it ensures that the destination class is not full by checking if the length of the studentIds array in the destination class exceeds the class's capacity. If the destination class is full, the method returns a 400 status code.

If all checks pass, the method proceeds to transfer the student. It removes the student's ID from the studentIds array of the current class and adds it to the studentIds array of the destination class. It also updates the student's classId to the destinationClassId. The method then saves the changes to the student, current class, and destination class.

Finally, the method returns a 200 status code with a success message indicating that the student has been successfully transferred between schools. In case of an internal error, the method returns a 500 status code indicating a server error.



Invalid Scenario  
  
Request :- {

// "currentClassId" : "6763327b9d5f7594b2aa4da0",

"destinationClassId" : "6764399d3d6b1eff117479d5",

"studentId" : "6763334f9d5f7594b2aa4daf"

}

Response :- {

"code": 400,

"message": "Validation failed.",

"result": [

"\"currentClassId\" is required"

]

}