

**1(a) API Endpoint:** Create Admin

Creates a new admin user with the provided details.

### **HTTP Request**

* **Method:** POST
* **Endpoint:** /admin

### **Request Body**

The request body must be sent as JSON and include the following fields:

|  |  |  |  |
| --- | --- | --- | --- |
| **Field** | **Type** | **Required** | **Description** |
| name | String | Yes | The name of the admin. |
| email | String | Yes | The email address of the admin. Must be unique. |
| password | String | Yes | The password for the admin. |
| phone | String | Yes | The phone number of the admin. |
| role | String | Yes | The role assigned to the admin (e.g., "admin"). |
| status | String | No | The status of the admin (e.g., "active"). Defaults to active. |

### **Responses**

* **201 Created**
  + **Description:** The admin was created successfully.
  + **Body:**

{  
 "data": {  
 "\_id": "string",  
 "name": "string",  
 "email": "string",  
 "phone": "string",  
 "role": "string",  
 "status": "string"  
 },  
 "message": "Admin created successfully"  
}

* **400 Bad Request**
  + **Description:** One or more required fields are missing.
  + **Body:**

{  
 "error": "Required fields are missing"  
}

### **Error Handling**

* If any required field is missing, a 400 Bad Request response is returned.
* If an admin or user with the provided email already exists, a 409 Conflict response is returned.
* If any other error occurs, a 500 Internal Server Error response is returned.

### **Notes**

* Passwords are hashed before being stored in the database.
* The endpoint ensures that the email provided for the admin is unique across both the Admin and User collections.

## **1(b) API Endpoint: User Login**

Authenticate and login a user.

### **Route**

POST api/login

### **Description**

This endpoint verifies user credentials (email and password) and returns a JWT token upon successful authentication.

### **Request Body**

|  |  |  |  |
| --- | --- | --- | --- |
| **Field** | **Type** | **Required** | **Description** |
| email | string | Yes | User's email address. |
| password | string | Yes | User's password. |

### **Responses**

* **200 OK**
  + Successfully authenticated.
  + Returns user information and JWT token.

{  
 "data": {  
 "\_id": "user\_id",  
 "name": "user\_name",  
 "email": "[user\_email@example.com](mailto:user_email@example.com)",  
 "phone": "user\_phone\_number",  
 "role": "user\_role",  
 "token": "jwt\_token"  
 },  
 "message": "Successfully logged in"  
}

* **400 Bad Request**
  + Invalid credentials or missing fields.
  + Returns error message.

{  
 "message": "The credentials you provided are incorrect, please try again."  
}

* **500 Internal Server Error**
  + Server encountered an unexpected condition.
  + Returns error message.

{  
 "error": "Something broke"  
}

### **Example**

#### **Request**

POST /login  
Content-Type: application/json  
  
{  
 "email": "[user\_email@example.com](mailto:user_email@example.com)",  
 "password": "user\_password"  
}

#### **Response** Content-Type: application/json { "data": { "\_id": "user\_id", "name": "user\_name", "email": "[user\_email@example.com](mailto:user_email@example.com)", "phone": "user\_phone\_number", "role": "user\_role", "token": "jwt\_token" }, "message": "Successfully logged in" }

### **Notes**

* Ensure the email and password fields are provided in the request body.
* Use the JWT token (token) for authenticated requests requiring user identification.

## **2 ) API Endpoint: Forgot Password**

Initiate the process to reset user password by sending a reset email.

### **Route**

POST api/forgotPassword

### **Description**

This endpoint generates a reset token, sends a reset email to the user's email address, and stores the token with an expiry timestamp for verification.

### **Request Body**

|  |  |  |  |
| --- | --- | --- | --- |
| **Field** | **Type** | **Required** | **Description** |
| email | string | Yes | User's email address. |

### **Responses**

* **200 OK**
  + Email successfully sent.
  + Returns success message.

{  
 "message": "Email sent successfully"  
}

* **500 Internal Server Error**
  + Server encountered an unexpected condition.
  + Returns error message.

{  
 "error": "Something broke"  
}

### **Example**

#### **Request**

POST api/forgotPassword  
Content-Type: application/json  
  
{  
 "email": "[user\_email@example.com](mailto:user_email@example.com)"  
}

#### **Response**

Content-Type: application/json  
  
{  
 "message": "Email sent successfully"  
}

### **Notes**

* Ensure the email field is provided in the request body.
* The reset token is generated and stored with an expiry time of 15 minutes (expiresAt).
* Implement the sendEmail function to send the reset email, passing resetToken and appropriate email template details (data, "forgot", "Reset Password").

## **3 ) API Endpoint: Reset Password**

Reset user password using a reset token.

### **Route**

POST api/resetPassword?token=<reset\_token>

### **Description**

This endpoint resets the user's password based on the provided reset token.

### **Query Parameters**

|  |  |  |  |
| --- | --- | --- | --- |
| **Parameter** | **Type** | **Required** | **Description** |
| token | string | Yes | Reset token received via email |

### **Request Body**

|  |  |  |  |
| --- | --- | --- | --- |
| **Field** | **Type** | **Required** | **Description** |
| password | string | Yes | New password for the user account |

### **Responses**

* **200 OK**
  + Password successfully updated.
  + Returns success message.

{  
 "message": "Password updated successfully"  
}

* **400 Bad Request**
  + Invalid or expired token.
  + Returns error message.

{  
 "message": "Token not found" or "Token is expired"  
}

* **500 Internal Server Error**
  + Server encountered an unexpected condition.
  + Returns error message.

{  
 "error": "Something broke"  
}

### **Example**

#### **Request**

POST api/resetPassword?token=reset\_token\_value  
Content-Type: application/json  
  
{  
 "password": "new\_password"  
}

#### **Response**

Content-Type: application/json  
  
{  
 "message": "Password updated successfully"  
}

### **Notes**

* Ensure the token is provided as a query parameter (?token=<reset\_token>).
* Provide the password field in the request body to set the new password.
* Use bcrypt to securely hash the new password before updating it in the database.
* Handle cases where the token is not found or has expired ("Token not found" or "Token is expired").

## **4 ) API Endpoint: User Profile**

Retrieve user profile information.

### **Route**

GET api/profile

### **Description**

This endpoint fetches and returns the profile information of the authenticated user.

### **Headers**

|  |  |  |  |
| --- | --- | --- | --- |
| **Header** | **Value** | **Required** | **Description** |
| Authorization | Bearer <JWT token> | Yes | JWT token for authentication |

### **Responses**

* **200 OK**
  + User profile successfully retrieved.
  + Returns user profile data.

json

Copy code

{  
 "data": {  
 "\_id": "user\_id",  
 "name": "user\_name",  
 "email": "[user\_email@example.com](mailto:user_email@example.com)",  
 "phone": "user\_phone\_number",  
 "role": "user\_role",  
 "division": "user\_division\_name"  
 }  
}

* **401 Unauthorized**
  + Missing or invalid JWT token.
  + Returns error message.

json

Copy code

{  
 "message": "Unauthorized access"  
}

* **404 Not Found**
  + User not found.
  + Returns error message.

json

Copy code

{  
 "message": "User not found"  
}

* **500 Internal Server Error**
  + Server encountered an unexpected condition.
  + Returns error message.

json

Copy code

{  
 "error": "Something broke"  
}

### **Example**

#### **Request**

GET api/profile  
Authorization: Bearer <JWT token>

#### **Response**

Content-Type: application/json  
  
{  
 "data": {  
 "\_id": "user\_id",  
 "name": "user\_name",  
 "email": "[user\_email@example.com](mailto:user_email@example.com)",  
 "phone": "user\_phone\_number",  
 "role": "user\_role",  
 "division": "user\_division\_name"  
 }  
}

### **Notes**

* Ensure the request includes a valid JWT token in the Authorization header.
* The endpoint retrieves and returns user profile details such as name, email, phone, role, and division.
* Handle cases where the user ID or associated role/division is not found with appropriate error messages.

## **5 ) API Endpoint: Create Division**

Create a new division.

### **Route**

POST api/addDivision

### **Description**

This endpoint allows an authenticated admin user to create a new division.

### **Headers**

|  |  |  |  |
| --- | --- | --- | --- |
| **Header** | **Value** | **Required** | **Description** |
| Authorization | Bearer <JWT token> | Yes | JWT token for admin authentication |

### **Request Body**

|  |  |  |  |
| --- | --- | --- | --- |
| **Field** | **Type** | **Required** | **Description** |
| divisionName | string | Yes | Name of the division to be created |
| status | Boolean | No | Status of the division (optional) |

### **Responses**

* **201 Created**
  + Division created successfully.
  + Returns newly created division data.

{  
 "data": {  
 "\_id": "division\_id",  
 "divisionName": "division\_name",  
 "status": "division\_status",  
 "adminId": "admin\_user\_id"  
 },  
 "message": "Division created successfully"  
}

* **400 Bad Request**
  + Missing required fields or validation error.
  + Returns error message.

{  
 "message": "Validation error: divisionName is required"  
}

* **500 Internal Server Error**
  + Server encountered an unexpected condition.
  + Returns error message.

{  
 "error": "Something broke"  
}

### **Example**

#### **Request**

POST api/addDivision  
Authorization: Bearer <JWT token>  
Content-Type: application/json  
  
{  
 "divisionName": "Marketing Division",  
 "status": "true"  
}

#### **Response**

Content-Type: application/json  
  
{  
 "data": {  
 "\_id": "division\_id",  
 "divisionName": "Marketing Division",  
 "status": "true",  
 "adminId": "admin\_user\_id"  
 },  
 "message": "Division created successfully"  
}

### **Notes**

* Ensure the request includes a valid JWT token in the Authorization header with admin privileges.
* The endpoint creates a division with the specified divisionName and optional status.
* Handle validation errors for missing divisionName or other required fields.
* Adjust the example values and error messages as per your implementation requirements.

## **6 ) API Endpoint: Get Divisions**

Retrieve divisions with pagination, sorting, and search functionality.

### **Route**

GET api/divisions

### **Description**

This endpoint retrieves a list of divisions with options for pagination, sorting, and search.

### **Headers**

|  |  |  |  |
| --- | --- | --- | --- |
| **Header** | **Value** | **Required** | **Description** |
| Authorization | Bearer <JWT token> | Yes | JWT token for authentication |

### **Query Parameters**

|  |  |  |  |
| --- | --- | --- | --- |
| **Parameter** | **Type** | **Required** | **Description** |
| page | integer | No | Page number (default: 1) |
| limit | integer | No | Number of items per page (default: 10) |
| order | string | No | Sorting order (ascending or descending) |
| sort | string | No | Field to sort by |
| search | string | No | Search keyword to filter divisions by divisionName |

### **Responses**

* **200 OK**
  + Divisions fetched successfully.
  + Returns paginated list of divisions.

{  
 "message": "Divisions fetched successfully",  
 "data": [  
 {  
 "\_id": "division\_id",  
 "divisionName": "division\_name",  
 "status": "division\_status",  
 "adminId": "admin\_user\_id",  
 "createdAt": "timestamp"  
 },  
 ...  
 ],  
 "totalCounts": 50  
}

* **500 Internal Server Error**
  + Server encountered an unexpected condition.
  + Returns error message.

{  
 "error": "Something broke"  
}

### **Example**

#### **Request**

GET api/divisions?page=1&limit=10&order=ascending&sort=divisionName&search=marketing  
Authorization: Bearer <JWT token>

#### **Response**

Content-Type: application/json  
  
{  
 "message": "Divisions fetched successfully",  
 "data": [  
 {  
 "\_id": "division\_id",  
 "divisionName": "Marketing Division",  
 "status": "Active",  
 "adminId": "admin\_user\_id",  
 "createdAt": "2024-07-10T12:00:00Z"  
 },  
 ...  
 ],  
 "totalCounts": 50  
}

### **Notes**

* Ensure the request includes a valid JWT token in the Authorization header.
* Pagination parameters (page and limit) control the number of divisions returned per page.
* Sorting can be applied by specifying order (ascending or descending) and sort (field name).
* Use search parameter to filter divisions by divisionName containing the specified keyword.
* Handle cases where no divisions match the search criteria or other errors with appropriate error messages.

## **7 ) API Endpoint: Update Division**

Update an existing division.

### **Route**

PUT api/updateDivision/:divisionId

### **Description**

This endpoint allows an authenticated admin user to update an existing division by its ID.

### **Headers**

|  |  |  |  |
| --- | --- | --- | --- |
| **Header** | **Value** | **Required** | **Description** |
| Authorization | Bearer <JWT token> | Yes | JWT token for admin authentication |

### **Path Parameters**

|  |  |  |  |
| --- | --- | --- | --- |
| **Parameter** | **Type** | **Required** | **Description** |
| divisionId | string | Yes | ID of the division to update |

### **Request Body**

|  |  |  |  |
| --- | --- | --- | --- |
| **Field** | **Type** | **Required** | **Description** |
| divisionName | string | No | New name for the division |
| status | Boolean | No | New status for the division |

### **Responses**

* **201 Created**
  + Division updated successfully.
  + Returns updated division data.

{  
 "data": {  
 "\_id": "division\_id",  
 "divisionName": "updated\_division\_name",  
 "status": "updated\_division\_status",  
 "adminId": "admin\_user\_id",  
 "createdAt": "timestamp"  
 },  
 "message": "Division updated successfully"  
}

* **404 Not Found**
  + Division not found.
  + Returns error message.

{  
 "message": "Division not found"  
}

* **500 Internal Server Error**
  + Server encountered an unexpected condition.
  + Returns error message.

{  
 "error": "Something broke"  
}

### **Example**

#### **Request**

PUT api/updateDivision/division\_id  
Authorization: Bearer <JWT token>  
Content-Type: application/json  
  
{  
 "divisionName": "Updated Marketing Division",  
 "status": "false"  
}

#### **Response**

Created  
Content-Type: application/json  
  
{  
 "data": {  
 "\_id": "division\_id",  
 "divisionName": "Updated Marketing Division",  
 "status": "Inactive",  
 "adminId": "admin\_user\_id",  
 "createdAt": "2024-07-10T12:00:00Z"  
 },  
 "message": "Division updated successfully"  
}

### **Notes**

* Ensure the request includes a valid JWT token in the Authorization header with admin privileges.
* Use the divisionId path parameter to specify which division to update.
* Update the division's divisionName and/or status as needed.
* Handle cases where the division ID is not found with appropriate error messages.

## **8 ) API Endpoint: Delete Division**

Delete an existing division.

### **Route**

bash

Copy code

DELETE api/deleteDivision/:divisionId

### **Description**

This endpoint allows an authenticated admin user to delete an existing division by its ID.

### **Headers**

|  |  |  |  |
| --- | --- | --- | --- |
| **Header** | **Value** | **Required** | **Description** |
| Authorization | Bearer <JWT token> | Yes | JWT token for admin authentication |

### **Path Parameters**

|  |  |  |  |
| --- | --- | --- | --- |
| **Parameter** | **Type** | **Required** | **Description** |
| divisionId | string | Yes | ID of the division to delete |

### **Responses**

* **201 Created**
  + Division deleted successfully.
  + Returns deleted division data.

{  
 "data": {  
 "\_id": "deleted\_division\_id",  
 "divisionName": "deleted\_division\_name",  
 "status": "deleted\_division\_status",  
 "adminId": "admin\_user\_id",  
 "createdAt": "timestamp"  
 },  
 "message": "Division deleted successfully"  
}

* **404 Not Found**
  + Division not found.
  + Returns error message.

{  
 "message": "Division not found"  
}

* **500 Internal Server Error**
  + Server encountered an unexpected condition.
  + Returns error message.

json

Copy code

{  
 "error": "Something broke"  
}

### **Example**

#### **Request**

DELETE api/deleteDivision/deleted\_division\_id  
Authorization: Bearer <JWT token>

#### **Response**

Content-Type: application/json  
  
{  
 "data": {  
 "\_id": "deleted\_division\_id",  
 "divisionName": "deleted\_division\_name",  
 "status": "deleted\_division\_status",  
 "adminId": "admin\_user\_id",  
 "createdAt": "2024-07-10T12:00:00Z"  
 },  
 "message": "Division deleted successfully"  
}

### **Notes**

* Ensure the request includes a valid JWT token in the Authorization header with admin privileges.
* Use the divisionId path parameter to specify which division to delete.
* Handle cases where the division ID is not found with appropriate error messages.

## **9) API Endpoint - Update Division Status**

Toggles the status of a specific division by its ID.

### **HTTP Request**

* **Method:** PATCH
* **Endpoint:** /updateDivisionStatus/:divisionId
* **Authentication:** Requires a valid JWT token.

### **Path Parameters**

|  |  |  |  |
| --- | --- | --- | --- |
| **Parameter** | **Type** | **Required** | **Description** |
| divisionId | String | Yes | The unique identifier of the division whose status is to be updated. |

### **Responses**

* **201 Created**
  + **Description:** The status of the division was successfully updated.
  + **Body:**

{  
 "data": {  
 "\_id": "string",  
 "name": "string",  
 "status": "boolean"  
 },  
 "message": "Status updated successfully"  
}

* **400 Bad Request**
  + **Description:** The divisionId parameter is missing.
  + **Body:**

{  
 "error": "Division ID is required"  
}

* **404 Not Found**
  + **Description:** The division with the provided ID was not found.
  + **Body:**

{  
 "error": "Division not found"  
}

* **500 Internal Server Error**
  + **Description:** An error occurred while updating the division's status.
  + **Body:**

{  
 "error": "Something broke"  
}

### **Error Handling**

* If divisionId is missing, a 400 Bad Request response is returned.
* If the division with the provided divisionId is not found, a 404 Not Found response is returned.
* If any other error occurs, a 500 Internal Server Error response is returned.

### **Notes**

* The endpoint requires authentication via a JWT token.
* The status field of the division is toggled (i.e., if it was true, it will be set to false and vice versa).

## **10) Api Endpoint - Add Issue Code**

Creates a new issue code associated with a specific component.

### **HTTP Request**

* **Method:** POST
* **Endpoint:** /addIssueCode
* **Authentication:** Requires a valid JWT token.

### **Request Body**

The request body must be sent as JSON and include the following fields:

|  |  |  |  |
| --- | --- | --- | --- |
| **Field** | **Type** | **Required** | **Description** |
| componentName | String | Yes | The name of the component for which the issue code is being created. |
| issueCode | String | Yes | The unique issue code associated with the component. |
| status | String | No | The status of the issue code (e.g., "active"). Defaults to active. |

### **Responses**

* **201 Created**
  + **Description:** The issue code was created successfully.
  + **Body:**

{  
 "data": {  
 "\_id": "string",  
 "componentName": "string",  
 "issueCode": "string",  
 "status": "string",  
 "adminId": "string"  
 },  
 "message": "IssueCode created successfully"  
}

* **400 Bad Request**
  + **Description:** One or more required fields are missing.
  + **Body:**

{  
 "error": "Required fields are missing"  
}

* **500 Internal Server Error**
  + **Description:** An error occurred while creating the issue code.
  + **Body:**

{  
 "error": "Something broke"  
}

### **Error Handling**

* If any required field is missing, a 400 Bad Request response is returned.
* If any other error occurs, a 500 Internal Server Error response is returned.

### **Notes**

* The adminId is automatically associated with the issue code based on the authenticated user's ID (extracted from the JWT token).
* The status field can be used to indicate whether the issue code is currently active or inactive.

## **11) API Endpoint - Get Issue Codes**

Fetches a paginated list of issue codes with optional sorting and search functionality.

### **HTTP Request**

* **Method:** GET
* **Endpoint:** /issueCodes
* **Authentication:** Requires a valid JWT token.

### **Query Parameters**

|  |  |  |  |
| --- | --- | --- | --- |
| **Parameter** | **Type** | **Required** | **Description** |
| page | Number | Yes | The page number to retrieve. |
| limit | Number | Yes | The number of items per page. |
| order | String | No | The order of sorting, either ascending or descending. Default is descending by createdAt. |
| sort | String | No | The field to sort by. Defaults to createdAt if not provided. |
| search | String | No | A search string to filter issue codes by componentName or issueCode. |

### **Responses**

* **200 OK**
  + **Description:** The issue codes were successfully retrieved.
  + **Body:**

{  
 "message": "IssueCodes fetched successfully",  
 "data": [  
 {  
 "\_id": "string",  
 "componentName": "string",  
 "issueCode": "string",  
 "status": "string",  
 "adminId": "string",  
 "createdAt": "string"  
 }  
 ],  
 "totalCount": "number",  
 "next": {  
 "page": "number",  
 "limit": "number"  
 },  
 "previous": {  
 "page": "number",  
 "limit": "number"  
 }  
}

* **400 Bad Request**
  + **Description:** Invalid query parameters were provided.
  + **Body:**

{  
 "error": "Invalid query parameters"  
}

* **500 Internal Server Error**
  + **Description:** An error occurred while fetching the issue codes.
  + **Body:**

{  
 "error": "Something broke"  
}

### **Error Handling**

* If invalid query parameters are provided, a 400 Bad Request response is returned.
* If any other error occurs, a 500 Internal Server Error response is returned.

### **Notes**

* The endpoint supports pagination, sorting, and search functionality.
* The search parameter filters results based on matching componentName or issueCode.
* The response includes pagination details in the next and previous fields if applicable.

## **12) API Endpoint - Update Issue Code**

Updates an existing issue code by its ID.

### **HTTP Request**

* **Method:** PUT
* **Endpoint:** /updateIssueCode/:issueCodeId
* **Authentication:** Requires a valid JWT token.

### **Path Parameters**

|  |  |  |  |
| --- | --- | --- | --- |
| **Parameter** | **Type** | **Required** | **Description** |
| issueCodeId | String | Yes | The unique identifier of the issue code to be updated. |

### **Request Body**

The request body must be sent as JSON and may include the following fields:

|  |  |  |  |
| --- | --- | --- | --- |
| **Field** | **Type** | **Required** | **Description** |
| componentName | String | No | The updated name of the component associated with the issue code. |
| issueCode | String | No | The updated issue code. |
| status | String | No | The updated status of the issue code (e.g., "active" or "inactive"). |

### **Responses**

* **201 Created**
  + **Description:** The issue code was successfully updated.
  + **Body:**

{  
 "data": {  
 "\_id": "string",  
 "componentName": "string",  
 "issueCode": "string",  
 "status": "string",  
 "adminId": "string"  
 },  
 "message": "IssueCode updated successfully"  
}

* **400 Bad Request**
  + **Description:** The issueCodeId parameter is missing.
  + **Body:**

{  
 "error": "Issue Code ID is required"  
}

* **404 Not Found**
  + **Description:** The issue code with the provided ID was not found.
  + **Body:**

json

Copy code

{  
 "error": "Issue Code not found"  
}

* **500 Internal Server Error**
  + **Description:** An error occurred while updating the issue code.
  + **Body:**

{  
 "error": "Something broke"  
}

### **Error Handling**

* If issueCodeId is missing, a 400 Bad Request response is returned.
* If the issue code with the provided issueCodeId is not found, a 404 Not Found response is returned.
* If any other error occurs, a 500 Internal Server Error response is returned.

### **Notes**

* The adminId is automatically updated based on the authenticated user's ID (extracted from the JWT token).
* Only the fields provided in the request body will be updated; fields left out will remain unchanged.

## **13) API Endpoint - Update Issue Code Status**

Toggles the status of an existing issue code by its ID.

### **HTTP Request**

* **Method:** PUT
* **Endpoint:** /updateIssueCodeStatus/:issueCodeId
* **Authentication:** Requires a valid JWT token.

### **Path Parameters**

|  |  |  |  |
| --- | --- | --- | --- |
| **Parameter** | **Type** | **Required** | **Description** |
| issueCodeId | String | Yes | The unique identifier of the issue code to be updated. |

### **Responses**

* **201 Created**
  + **Description:** The status of the issue code was successfully updated.
  + **Body:**

{  
 "data": {  
 "\_id": "string",  
 "componentName": "string",  
 "issueCode": "string",  
 "status": "boolean",  
 "adminId": "string"  
 },  
 "message": "Status updated successfully"  
}

* **400 Bad Request**
  + **Description:** The issueCodeId parameter is missing.
  + **Body:**

{  
 "error": "Issue Code ID is required"  
}

* **404 Not Found**
  + **Description:** The issue code with the provided ID was not found.
  + **Body:**

{  
 "error": "Issue Code not found"  
}

* **500 Internal Server Error**
  + **Description:** An error occurred while updating the status of the issue code.
  + **Body:**

{  
 "error": "Something broke"  
}

### **Error Handling**

* If issueCodeId is missing, a 400 Bad Request response is returned.
* If the issue code with the provided issueCodeId is not found, a 404 Not Found response is returned.
* If any other error occurs, a 500 Internal Server Error response is returned.

### **Notes**

* The status of the issue code is toggled between its current state (e.g., from true to false or vice versa).
* The endpoint only changes the status field; other fields remain unchanged.

## **14) API Endpoint - Delete Issue Code**

Deletes an existing issue code by its ID.

### **HTTP Request**

* **Method:** DELETE
* **Endpoint:** /deleteIssueCode/:issueCodeId
* **Authentication:** Requires a valid JWT token.

### **Path Parameters**

|  |  |  |  |
| --- | --- | --- | --- |
| **Parameter** | **Type** | **Required** | **Description** |
| issueCodeId | String | Yes | The unique identifier of the issue code to be deleted. |

### **Responses**

* **201 Created**
  + **Description:** The issue code was successfully deleted.
  + **Body:**

{  
 "data": {  
 "\_id": "string",  
 "componentName": "string",  
 "issueCode": "string",  
 "status": "boolean",  
 "adminId": "string"  
 },  
 "message": "IssueCode deleted successfully"  
}

* **400 Bad Request**
  + **Description:** The issueCodeId parameter is missing.
  + **Body:**

{  
 "error": "Issue Code ID is required"  
}

* **404 Not Found**
  + **Description:** The issue code with the provided ID was not found.
  + **Body:**

{  
 "error": "Issue Code not found"  
}

* **500 Internal Server Error**
  + **Description:** An error occurred while attempting to delete the issue code.
  + **Body:**

{  
 "error": "Something broke"  
}

### **Error Handling**

* If issueCodeId is missing, a 400 Bad Request response is returned.
* If the issue code with the provided issueCodeId is not found, a 404 Not Found response is returned.
* If any other error occurs, a 500 Internal Server Error response is returned.

### **Notes**

* The endpoint permanently deletes the issue code from the database.

## **15) API Endpoint - Create Location**

Creates a new location with the provided details.

### **HTTP Request**

* **Method:** POST
* **Endpoint:** /createLocation
* **Authentication:** Requires a valid JWT token.

### **Request Body**

|  |  |  |  |
| --- | --- | --- | --- |
| **Field** | **Type** | **Required** | **Description** |
| locationName | String | Yes | The name of the location to be created. |
| status | Boolean | No | The status of the location (active/inactive). |

### **Responses**

* **201 Created**
  + **Description:** The location was successfully created.
  + **Body:**

{  
 "data": {  
 "\_id": "string",  
 "locationName": "string",  
 "status": "boolean",  
 "adminId": "string"  
 },  
 "message": "Location created successfully"  
}

* **400 Bad Request**
  + **Description:** The locationName field is missing or invalid.
  + **Body:**

{  
 "error": "Location name is required"  
}

* **500 Internal Server Error**
  + **Description:** An error occurred while creating the location.
  + **Body:**

{  
 "error": "Something broke"  
}

### **Error Handling**

* If locationName is missing, a 400 Bad Request response is returned.
* If any other error occurs, a 500 Internal Server Error response is returned.

### **Notes**

* The adminId field is automatically populated from the authenticated user's ID.
* The status field is optional and can be set to true (active) or false (inactive). If not provided, it may default to a pre-defined value.

## **16) API Endpoint - Get Location**

Fetches a list of locations with support for pagination, sorting, and searching.

### **HTTP Request**

* **Method:** GET
* **Endpoint:** /locations
* **Authentication:** Requires a valid JWT token.

### **Query Parameters**

|  |  |  |  |
| --- | --- | --- | --- |
| **Parameter** | **Type** | **Required** | **Description** |
| page | Number | No | The page number for pagination. Defaults to 1 if not provided. |
| limit | Number | No | The number of locations per page. Defaults to 10 if not provided. |
| order | String | No | The sort order. Can be ascending or descending. Defaults to descending. |
| sort | String | No | The field to sort by (e.g., locationName). Defaults to createdAt. |
| search | String | No | The search term to filter locations by locationName. |

### **Responses**

* **200 OK**
  + **Description:** Locations fetched successfully.
  + **Body:**

{  
 "message": "Location fetched successfully",  
 "data": [  
 {  
 "\_id": "string",  
 "locationName": "string",  
 "status": "boolean",  
 "adminId": "string"  
 }  
 ],  
 "totalCounts": "number"  
}

* **500 Internal Server Error**
  + **Description:** An error occurred while fetching locations.
  + **Body:**

{  
 "error": "Something broke"  
}

### **Pagination**

* **next**: Contains the page number and limit for the next set of results if there are more pages.
* **previous**: Contains the page number and limit for the previous set of results if there are previous pages.

### **Sorting**

* Locations can be sorted by any field specified in the sort query parameter.
* The order query parameter determines the sorting direction.

### **Searching**

* The search parameter filters locations based on locationName.

### **Error Handling**

* If any internal error occurs, a 500 Internal Server Error response is returned.

### **Notes**

* The response includes a totalCounts field that indicates the total number of locations available.
* Pagination, sorting, and searching options provide flexibility in querying location data.

## **17) API Endpoint - Update Location**

Updates the details of an existing location by its ID.

### **HTTP Request**

* **Method:** PUT
* **Endpoint:** /updateLocation/:locationId
* **Authentication:** Requires a valid JWT token.

### **Path Parameters**

|  |  |  |  |
| --- | --- | --- | --- |
| **Parameter** | **Type** | **Required** | **Description** |
| locationId | String | Yes | The unique identifier of the location to be updated. |

### **Request Body**

|  |  |  |  |
| --- | --- | --- | --- |
| **Field** | **Type** | **Required** | **Description** |
| locationName | String | No | The new name of the location. |
| status | Boolean | No | The new status of the location (active/inactive). |

### **Responses**

* **200 OK**
  + **Description:** The location was successfully updated.
  + **Body:**

{  
 "data": {  
 "\_id": "string",  
 "locationName": "string",  
 "status": "boolean",  
 "adminId": "string"  
 },  
 "message": "Location updated successfully"  
}

* **400 Bad Request**
  + **Description:** The locationId parameter is missing.
  + **Body:**

{  
 "error": "Location ID is required"  
}

* **404 Not Found**
  + **Description:** The location with the provided locationId was not found.
  + **Body:**

{  
 "error": "Location not found"  
}

* **500 Internal Server Error**
  + **Description:** An error occurred while updating the location.
  + **Body:**

{  
 "error": "Something broke"  
}

### **Error Handling**

* If locationId is missing, a 400 Bad Request response is returned.
* If the location with the provided locationId is not found, a 404 Not Found response is returned.
* If any other error occurs, a 500 Internal Server Error response is returned.

### **Notes**

* The adminId field is automatically populated with the authenticated user's ID.
* Only the fields provided in the request body will be updated.

## **18) API Endpoint - Update Location Status**

Toggles the status of a location between active and inactive.

### **HTTP Request**

* **Method:** PATCH
* **Endpoint:** /updateLocationStatus/:locationId
* **Authentication:** Requires a valid JWT token.

### **Path Parameters**

|  |  |  |  |
| --- | --- | --- | --- |
| **Parameter** | **Type** | **Required** | **Description** |
| locationId | String | Yes | The unique identifier of the location whose status is to be updated. |

### **Responses**

* **201 Created**
  + **Description:** The location status was successfully updated.
  + **Body:**

{  
 "data": {  
 "\_id": "string",  
 "locationName": "string",  
 "status": "boolean",  
 "adminId": "string"  
 },  
 "message": "Status updated successfully"  
}

* **400 Bad Request**
  + **Description:** The locationId parameter is missing.
  + **Body:**

{  
 "error": "Location ID is required"  
}

* **404 Not Found**
  + **Description:** The location with the provided locationId was not found.
  + **Body:**

{  
 "error": "Location not found"  
}

* **500 Internal Server Error**
  + **Description:** An error occurred while updating the location status.
  + **Body:**

{  
 "error": "Something broke"  
}

### **Error Handling**

* If locationId is missing, a 400 Bad Request response is returned.
* If the location with the provided locationId is not found, a 404 Not Found response is returned.
* If any other error occurs, a 500 Internal Server Error response is returned.

### **Notes**

* The status is toggled between true (active) and false (inactive) with each request.
* The adminId field remains unchanged.

## **19 ) API Endpoint - Delete Location**

Deletes a specific location by its ID.

### **HTTP Request**

* **Method:** DELETE
* **Endpoint:** /deleteLocation/:locationId
* **Authentication:** Requires a valid JWT token.

### **Path Parameters**

|  |  |  |  |
| --- | --- | --- | --- |
| **Parameter** | **Type** | **Required** | **Description** |
| locationId | String | Yes | The unique identifier of the location to be deleted. |

### **Responses**

* **201 Created**
  + **Description:** The location was successfully deleted.
  + **Body:**

{  
 "data": {  
 "\_id": "string",  
 "locationName": "string",  
 "status": "boolean",  
 "adminId": "string"  
 },  
 "message": "Location deleted successfully"  
}

* **400 Bad Request**
  + **Description:** The locationId parameter is missing.
  + **Body:**

{  
 "error": "Location ID is required"  
}

* **404 Not Found**
  + **Description:** The location with the provided locationId was not found.
  + **Body:**

{  
 "error": "Location not found"  
}

* **500 Internal Server Error**
  + **Description:** An error occurred while deleting the location.
  + **Body:**

{  
 "error": "Something broke"  
}

### **Error Handling**

* If locationId is missing, a 400 Bad Request response is returned.
* If the location with the provided locationId is not found, a 404 Not Found response is returned.
* If any other error occurs, a 500 Internal Server Error response is returned.

### **Notes**

* Once a location is deleted, it cannot be recovered.
* Ensure that the locationId provided is valid and exists in the database.

## **20) API Endpoint - Create Role**

Creates a new role with specified permissions and status.

### **HTTP Request**

* **Method:** POST
* **Endpoint:** /addRole
* **Authentication:** Requires a valid JWT token.

### **Request Body**

|  |  |  |  |
| --- | --- | --- | --- |
| **Field** | **Type** | **Required** | **Description** |
| role | String | Yes | The name of the role to be created. |
| status | Boolean | No | The status of the role (e.g., active or inactive). |
| addMachines | Boolean | No | Permission to add machines. |
| viewMachines | Boolean | No | Permission to view machines. |

### **Responses**

* **201 Created**
  + **Description:** The role was successfully created.
  + **Body:**

{  
 "data": {  
 "\_id": "string",  
 "role": "string",  
 "status": "boolean",  
 "addMachines": "boolean",  
 "viewMachines": "boolean",  
 "adminId": "string"  
 },  
 "message": "Role created successfully"  
}

* **400 Bad Request**
  + **Description:** The role field is missing in the request body.
  + **Body:**

{  
 "error": "Role field is required"  
}

* **500 Internal Server Error**
  + **Description:** An error occurred while creating the role.
  + **Body:**

{  
 "error": "Something broke"  
}

### **Error Handling**

* If the role field is missing from the request body, a 400 Bad Request response is returned.
* If any other error occurs during the role creation process, a 500 Internal Server Error response is returned.

### **Notes**

* Ensure that the role provided is unique and does not conflict with existing roles.
* The adminId is automatically set based on the authenticated user making the request.

## **21) API Endpoint - Get Roles**

Fetches a list of roles with pagination, sorting, and search capabilities.

### **HTTP Request**

* **Method:** GET
* **Endpoint:** /roles
* **Authentication:** Requires a valid JWT token.

### **Query Parameters**

|  |  |  |
| --- | --- | --- |
| **Parameter** | **Type** | **Description** |
| page | Number | Page number for pagination. Default is 1. |
| limit | Number | Number of items per page. Default is 10. |
| order | String | Sort order, either ascending or descending. |
| sort | String | Field to sort by. Default is createdAt. |
| search | String | Search term to filter roles by name. |

### **Responses**

* **200 OK**
  + **Description:** Roles fetched successfully.
  + **Body:**

{  
 "message": "Roles Fetched Successfully",  
 "data": [  
 {  
 "\_id": "string",  
 "role": "string",  
 "status": "boolean",  
 "addMachines": "boolean",  
 "viewMachines": "boolean",  
 "adminId": "string"  
 }  
 ],  
 "totalCounts": "number"  
}

* **500 Internal Server Error**
  + **Description:** An error occurred while fetching the roles.
  + **Body:**

{  
 "error": "Something broke"  
}

### **Pagination**

* **next**: Includes the page and limit parameters for the next page if available.
* **previous**: Includes the page and limit parameters for the previous page if available.

### **Sorting**

* By default, roles are sorted by createdAt in descending order. You can specify sorting by a different field and order using the sort and order query parameters.

### **Search**

* Roles can be filtered based on the search query parameter, which performs a case-insensitive search on the role field.

### **Error Handling**

* If an error occurs while processing the request, a 500 Internal Server Error response is returned.

### **Notes**

* Ensure that the page and limit parameters are valid numbers to avoid unexpected behavior.
* The search term is applied to the role field and is case-insensitive.

## **22) API Endpoint - Update Role**

Updates an existing role with new values.

### **HTTP Request**

* **Method:** PUT
* **Endpoint:** /updateRole/:roleId
* **Authentication:** Requires a valid JWT token.

### **URL Parameters**

|  |  |  |
| --- | --- | --- |
| **Parameter** | **Type** | **Description** |
| roleId | String | ID of the role to update. |

### **Request Body**

|  |  |  |
| --- | --- | --- |
| **Field** | **Type** | **Description** |
| role | String | New role name. Optional. |
| status | Boolean | New status of the role. Optional. |
| addMachines | Boolean | Permission to add machines. Optional. |
| viewMachines | Boolean | Permission to view machines. Optional. |

### **Responses**

* **201 Created**
  + **Description:** Role updated successfully.
  + **Body:**

{  
 "data": {  
 "\_id": "string",  
 "role": "string",  
 "status": "boolean",  
 "addMachines": "boolean",  
 "viewMachines": "boolean",  
 "adminId": "string"  
 },  
 "message": "Role updated successfully"  
}

* **400 Bad Request**
  + **Description:** Invalid role ID or missing fields.
  + **Body:**

{  
 "error": "Invalid role ID or missing fields"  
}

* **404 Not Found**
  + **Description:** Role not found.
  + **Body:**

{  
 "error": "Role not found"  
}

* **500 Internal Server Error**
  + **Description:** An error occurred while updating the role.
  + **Body:**

{  
 "error": "Something broke"  
}

### **Error Handling**

* **Validation Errors:** The endpoint validates the presence of roleId and checks if the role exists. If any of these validations fail, appropriate error responses are returned.

### **Notes**

* Ensure that roleId is a valid ID and that the request body contains valid data.
* The adminId is automatically set to the ID of the currently authenticated user.

## **23 ) API Endpoint - Update Role Status**

Toggles the status of a specific role.

### **HTTP Request**

* **Method:** PATCH
* **Endpoint:** /updateRoleStatus/:roleId
* **Authentication:** Requires a valid JWT token.

### **URL Parameters**

|  |  |  |
| --- | --- | --- |
| **Parameter** | **Type** | **Description** |
| roleId | String | ID of the role whose status will be updated. |

### **Responses**

* **201 Created**
  + **Description:** Role status updated successfully.
  + **Body:**

{  
 "data": {  
 "\_id": "string",  
 "role": "string",  
 "status": "boolean",  
 "addMachines": "boolean",  
 "viewMachines": "boolean",  
 "adminId": "string"  
 },  
 "message": "Status updated successfully"  
}

* **400 Bad Request**
  + **Description:** Invalid role ID.
  + **Body:**

{  
 "error": "Invalid role ID"  
}

* **404 Not Found**
  + **Description:** Role not found.
  + **Body:**

{  
 "error": "Role not found"  
}

* **500 Internal Server Error**
  + **Description:** An error occurred while updating the role status.
  + **Body:**

{  
 "error": "Something broke"  
}

### **Error Handling**

* **Validation Errors:** The endpoint checks if roleId is provided and if the role exists. If any of these checks fail, appropriate error responses are returned.

### **Notes**

* This endpoint toggles the status field of the specified role between true and false.
* Ensure that roleId is valid and exists in the database.

## **24) API Endpoint - Delete Role**

Deletes a specific role by its ID.

### **HTTP Request**

* **Method:** DELETE
* **Endpoint:** /deleteRole/:roleId
* **Authentication:** Requires a valid JWT token.

### **URL Parameters**

|  |  |  |
| --- | --- | --- |
| **Parameter** | **Type** | **Description** |
| roleId | String | ID of the role to be deleted. |

### **Responses**

* **201 Created**
  + **Description:** Role deleted successfully.
  + **Body:**

{  
 "data": {  
 "\_id": "string",  
 "role": "string",  
 "status": "boolean",  
 "addMachines": "boolean",  
 "viewMachines": "boolean",  
 "adminId": "string"  
 },  
 "message": "Role deleted successfully"  
}

* **400 Bad Request**
  + **Description:** Invalid role ID.
  + **Body:**

{  
 "error": "Invalid role ID"  
}

* **404 Not Found**
  + **Description:** Role not found.
  + **Body:**

{  
 "error": "Role not found"  
}

* **500 Internal Server Error**
  + **Description:** An error occurred while deleting the role.
  + **Body:**

{  
 "error": "Something broke"  
}

### **Error Handling**

* **Validation Errors:** The endpoint verifies if roleId is provided and if the role exists. Errors will be returned if these checks fail.

### **Notes**

* Ensure that roleId is valid and exists in the database before sending a delete request.
* Deleting a role removes it permanently from the database.

## **Update Role Permissions**

Updates permissions for a specific role.

### **HTTP Request**

* **Method:** PATCH
* **Endpoint:** /permission/:roleId
* **Authentication:** Requires a valid JWT token.

### **24) API Endpoint - URL Parameters**

|  |  |  |
| --- | --- | --- |
| **Parameter** | **Type** | **Description** |
| roleId | String | ID of the role to update. |

### **Request Body**

* **Content Type:** application/json
* **Body:**

{  
 "permissionField1": "value",  
 "permissionField2": "value",  
 ...  
}

Replace permissionField1, permissionField2, etc., with actual permission fields to be updated.

### **Responses**

* **200 OK**
  + **Description:** Permissions updated successfully.
  + **Body:**

{  
 "message": "Permissions updated successfully",  
 "data": {  
 "\_id": "string",  
 "role": "string",  
 "status": "boolean",  
 "addMachines": "boolean",  
 "viewMachines": "boolean",  
 "adminId": "string",  
 "permissionField1": "value",  
 "permissionField2": "value"  
 }  
}

* **400 Bad Request**
  + **Description:** Failed to update permissions.
  + **Body:**

{  
 "message": "Failed to update permissions"  
}

* **404 Not Found**
  + **Description:** Role not found.
  + **Body:**

{  
 "error": "Role not found"  
}

* **500 Internal Server Error**
  + **Description:** An error occurred while updating permissions.
  + **Body:**

{  
 "error": "Something broke"  
}

### **Error Handling**

* **Validation Errors:** The endpoint verifies if roleId is provided and if the role exists. Errors will be returned if these checks fail.
* **Update Failure:** If the update operation fails, a 400 status is returned with a message indicating failure.

### **Notes**

* Ensure that roleId is valid and exists in the database before sending a request to update permissions.
* The request body should contain the specific permission fields to be updated.

## **25) API Endpoint - Create User**

Creates a new user in the system.

### **HTTP Request**

* **Method:** POST
* **Endpoint:** /create
* **Authentication:** Optional JWT token. If provided, the user ID from the token is used as the adminId.

### **Request Body**

* **Content Type:** application/json
* **Body:**

{  
 "name": "string",  
 "email": "string",  
 "password": "string",  
 "phone": "string",  
 "role": "string",  
 "division": "string",  
 "status": "boolean"  
}

* + name (string, required): The full name of the user.
  + email (string, required): The user's email address.
  + password (string, required): The user's password (plain text).
  + phone (string, required): The user's phone number.
  + role (string, optional): The role assigned to the user.
  + division (string, optional): The division to which the user belongs.
  + status (boolean, optional): The status of the user (active/inactive).

### **Responses**

* **201 Created**
  + **Description:** User created successfully.
  + **Body:**

{  
 "data": {  
 "\_id": "string",  
 "name": "string",  
 "email": "string",  
 "phone": "string",  
 "role": "string",  
 "division": "string",  
 "status": "boolean",  
 "adminId": "string",  
 "createdAt": "ISO8601 date",  
 "updatedAt": "ISO8601 date"  
 },  
 "message": "User created successfully"  
}

* **400 Bad Request**
  + **Description:** Missing required fields or validation errors.
  + **Body:**

{  
 "error": "Required fields are missing or invalid"  
}

* **409 Conflict**
  + **Description:** User with the provided email already exists.
  + **Body:**

{  
 "error": "User with this email already exists"  
}

* **500 Internal Server Error**
  + **Description:** An error occurred while creating the user.
  + **Body:**

{  
 "error": "Something broke"  
}

### **Error Handling**

* **Validation Errors:** The endpoint checks for the presence of required fields (name, email, password, phone). If any of these are missing, a 400 status code with an appropriate error message is returned.
* **Duplicate User:** If a user with the provided email already exists, a 409 status code is returned with a message indicating the conflict.
* **Server Errors:** Any server-side errors during user creation will return a 500 status code.

### **Notes**

* The password is hashed before being stored in the database for security.
* The optional JWT token can be used to associate the created user with the admin performing the action. If no token is provided, the adminId will be null.

## **26) API Endpoint - Get Users**

Fetches a paginated list of users with optional filtering, sorting, and search capabilities.

### **HTTP Request**

* **Method:** GET
* **Endpoint:** /users
* **Authentication:** JWT token required

### **Query Parameters**

* **page** (integer, optional): The page number to retrieve. Defaults to 1.
* **limit** (integer, optional): The number of users per page. Defaults to 10.
* **order** (string, optional): The sort order for results. Can be "ascending" or "descending". Defaults to "descending" by createdAt.
* **sort** (string, optional): The field by which to sort the results. Defaults to createdAt.
* **search** (string, optional): A search term to filter users by name, email, role, or division.

### **Responses**

* **200 OK**
  + **Description:** Users fetched successfully.
  + **Body:**

{  
 "message": "Users Fetched Successfully",  
 "data": [  
 {  
 "\_id": "string",  
 "name": "string",  
 "email": "string",  
 "phone": "string",  
 "role": {  
 "\_id": "string",  
 "role": "string"  
 },  
 "division": {  
 "\_id": "string",  
 "divisionName": "string"  
 },  
 "status": "boolean",  
 "createdAt": "ISO8601 date",  
 "updatedAt": "ISO8601 date"  
 }  
 ],  
 "totalCount": "integer"  
}

* **400 Bad Request**
  + **Description:** Invalid query parameters.
  + **Body:**

{  
 "error": "Invalid query parameters"  
}

* **500 Internal Server Error**
  + **Description:** An error occurred while fetching the users.
  + **Body:**

{  
 "error": "Something broke"  
}

### **Error Handling**

* **Invalid Query Parameters:** If query parameters are invalid, a 400 status code is returned with an error message.
* **Server Errors:** Any server-side errors during the user retrieval process will return a 500 status code.

### **Notes**

* **Pagination:** The page and limit parameters control pagination. startIndex and endIndex are used to determine which users to return.
* **Sorting:** The order and sort parameters determine the order of the results. By default, users are sorted by createdAt in descending order.
* **Searching:** The search parameter filters users based on name, email, role, or division. Role and division searches are resolved by first finding the relevant IDs and then filtering users based on these IDs.
* **Populating:** The role and division fields are populated with their respective details for each user.

## **27 ) API Endpoint - Get User Details**

Fetches detailed information about a specific user by their ID.

### **HTTP Request**

* **Method:** GET
* **Endpoint:** /user/:userId
* **Authentication:** JWT token required

### **URL Parameters**

* **userId** (string, required): The ID of the user whose details are to be fetched.

### **Responses**

* **200 OK**
  + **Description:** User details fetched successfully.
  + **Body:**

{  
 "message": "User fetched successfully",  
 "data": {  
 "\_id": "string",  
 "name": "string",  
 "email": "string",  
 "phone": "string",  
 "role": {  
 "\_id": "string",  
 "role": "string"  
 },  
 "division": {  
 "\_id": "string",  
 "divisionName": "string"  
 },  
 "status": "boolean",  
 "createdAt": "ISO8601 date",  
 "updatedAt": "ISO8601 date"  
 }  
}

* **400 Bad Request**
  + **Description:** Missing or invalid userId.
  + **Body:**

{  
 "error": "Invalid user ID"  
}

* **404 Not Found**
  + **Description:** User not found.
  + **Body:**

{  
 "error": "User not found"  
}

* **500 Internal Server Error**
  + **Description:** An error occurred while fetching the user details.
  + **Body:**

{  
 "error": "Something broke"  
}

### **Error Handling**

* **Invalid User ID:** If userId is missing or invalid, a 400 status code is returned with an error message.
* **User Not Found:** If no user is found with the given ID, a 404 status code is returned with an error message.
* **Server Errors:** Any server-side errors during the user retrieval process will return a 500 status code.

### **Notes**

* **Populating:** The role and division fields are populated with their respective details for the user.
* **ID Validation:** The validateId function is used to check if the userId is valid. If not, it returns a 400 status code.
* **User Existence:** The validateFound function is used to check if the user exists. If not, it returns a 404 status code.

## **28 ) API Endpoint - Update User**

Updates the details of a specific user by their ID.

### **HTTP Request**

* **Method:** PUT
* **Endpoint:** /update/:userId
* **Authentication:** JWT token required

### **URL Parameters**

* **userId** (string, required): The ID of the user to be updated.

### **Request Body**

* **name** (string, optional): The updated name of the user.
* **email** (string, optional): The updated email address of the user.
* **password** (string, optional): The updated password for the user (should be hashed before saving).
* **phone** (string, optional): The updated phone number of the user.
* **role** (string, optional): The updated role ID for the user.
* **division** (string, optional): The updated division ID for the user.
* **status** (boolean, optional): The updated status of the user.

### **Responses**

* **200 OK**
  + **Description:** User details updated successfully.
  + **Body:**

{  
 "message": "User updated successfully",  
 "data": {  
 "\_id": "string",  
 "name": "string",  
 "email": "string",  
 "phone": "string",  
 "role": {  
 "\_id": "string",  
 "role": "string"  
 },  
 "division": {  
 "\_id": "string",  
 "divisionName": "string"  
 },  
 "status": "boolean",  
 "createdAt": "ISO8601 date",  
 "updatedAt": "ISO8601 date"  
 }  
}

* **400 Bad Request**
  + **Description:** Missing or invalid userId, or required fields in the request body.
  + **Body:**

{  
 "error": "Invalid user ID or missing fields"  
}

* **404 Not Found**
  + **Description:** User not found.
  + **Body:**

{  
 "error": "User not found"  
}

* **500 Internal Server Error**
  + **Description:** An error occurred while updating the user details.
  + **Body:**

{  
 "error": "Something broke"  
}

### **Error Handling**

* **Invalid User ID:** If userId is missing or invalid, a 400 status code is returned with an error message.
* **User Not Found:** If no user is found with the given ID, a 404 status code is returned with an error message.
* **Server Errors:** Any server-side errors during the user update process will return a 500 status code.

### **Notes**

* **Password Hashing:** The password should be hashed before saving. Ensure you use a secure method for password hashing.
* **Admin ID:** The adminId field is updated to track the user who made the changes.
* **Validation:** The validateId function is used to check if the userId is valid. If not, it returns a 400 status code. The validateFound function checks if the user exists; if not, it returns a 404 status code.

## **29 ) API Endpoint - Update User Status**

Toggles the status of a specific user by their ID.

### **HTTP Request**

* **Method:** PATCH
* **Endpoint:** /updateStatus/:userId
* **Authentication:** JWT token required

### **URL Parameters**

* **userId** (string, required): The ID of the user whose status is to be updated.

### **Request Body**

* No request body is required for this endpoint.

### **Responses**

* **200 OK**
  + **Description:** User status updated successfully.
  + **Body:**

{  
 "message": "User status updated successfully",  
 "data": {  
 "\_id": "string",  
 "name": "string",  
 "email": "string",  
 "phone": "string",  
 "role": {  
 "\_id": "string",  
 "role": "string"  
 },  
 "division": {  
 "\_id": "string",  
 "divisionName": "string"  
 },  
 "status": "boolean",  
 "createdAt": "ISO8601 date",  
 "updatedAt": "ISO8601 date"  
 }  
}

* **400 Bad Request**
  + **Description:** Missing or invalid userId.
  + **Body:**

{  
 "error": "Invalid user ID"  
}

* **404 Not Found**
  + **Description:** User not found.
  + **Body:**

{  
 "error": "User not found"  
}

* **500 Internal Server Error**
  + **Description:** An error occurred while updating the user status.
  + **Body:**

{  
 "error": "Something broke"  
}

### **Error Handling**

* **Invalid User ID:** If userId is missing or invalid, a 400 status code is returned with an error message.
* **User Not Found:** If no user is found with the given ID, a 404 status code is returned with an error message.
* **Server Errors:** Any server-side errors during the status update process will return a 500 status code.

### **Notes**

* **Status Toggle:** The status is toggled between true and false. If the current status is true, it will be changed to false, and vice versa.
* **Validation:** The validateId function checks if the userId is valid. If not, it returns a 400 status code. The validateFound function checks if the user exists; if not, it returns a 404 status code.

## **30 ) API Endpoint - Delete User**

Deletes a specific user by their ID.

### **HTTP Request**

* **Method:** DELETE
* **Endpoint:** /deleteUser/:userId
* **Authentication:** JWT token required

### **URL Parameters**

* **userId** (string, required): The ID of the user to be deleted.

### **Request Body**

* No request body is required for this endpoint.

### **Responses**

* **200 OK**
  + **Description:** User deleted successfully.
  + **Body:**

{  
 "message": "User deleted successfully",  
 "data": {  
 "\_id": "string",  
 "name": "string",  
 "email": "string",  
 "phone": "string",  
 "role": {  
 "\_id": "string",  
 "role": "string"  
 },  
 "division": {  
 "\_id": "string",  
 "divisionName": "string"  
 },  
 "status": "boolean",  
 "createdAt": "ISO8601 date",  
 "updatedAt": "ISO8601 date"  
 }  
}

* **400 Bad Request**
  + **Description:** Missing or invalid userId.
  + **Body:**

{  
 "error": "Invalid user ID"  
}

* **404 Not Found**
  + **Description:** User not found.
  + **Body:**

{  
 "error": "User not found"  
}

* **500 Internal Server Error**
  + **Description:** An error occurred while deleting the user.
  + **Body:**

{  
 "error": "Something broke"  
}

### **Error Handling**

* **Invalid User ID:** If userId is missing or invalid, a 400 status code is returned with an error message.
* **User Not Found:** If no user is found with the given ID, a 404 status code is returned with an error message.
* **Server Errors:** Any server-side errors during the deletion process will return a 500 status code.

### **Notes**

* **Deletion Process:** The findByIdAndDelete function is used to find and delete the user by their ID.
* **Validation:** The validateId function checks if the userId is valid. If not, it returns a 400 status code. The validateFound function checks if the user exists; if not, it returns a 404 status code.

## **31 ) API Endpoint - Get Selected Roles**

Fetches a list of roles that are currently active.

### **HTTP Request**

* **Method:** GET
* **Endpoint:** /selectRole
* **Authentication:** JWT token required

### **Request Parameters**

* No query parameters are required.

### **Responses**

* **200 OK**
  + **Description:** Roles fetched successfully.
  + **Body:**

{  
 "message": "Roles fetched successfully",  
 "data": [  
 {  
 "\_id": "string",  
 "role": "string"  
 }  
 ]  
}

* **500 Internal Server Error**
  + **Description:** An error occurred while fetching roles.
  + **Body:**

{  
 "error": "Something broke"  
}

### **Error Handling**

* **Server Errors:** Any issues that occur while querying the roles will result in a 500 status code with an appropriate error message.

### **Notes**

* **Status Filter:** The query filters roles to include only those with status: true.
* **Field Selection:** Only the \_id and role fields are returned for each role.

## **32 ) API Endpoint - Get Selected Divisions**

Fetches a list of divisions that are currently active.

### **HTTP Request**

* **Method:** GET
* **Endpoint:** /selectDivision
* **Authentication:** JWT token required

### **Request Parameters**

* No query parameters are required.

### **Responses**

* **200 OK**
  + **Description:** Divisions fetched successfully.
  + **Body:**

{  
 "message": "Division fetched successfully",  
 "data": [  
 {  
 "\_id": "string",  
 "divisionName": "string"  
 }  
 ]  
}

* **500 Internal Server Error**
  + **Description:** An error occurred while fetching divisions.
  + **Body:**

{  
 "error": "Something broke"  
}

### **Error Handling**

* **Server Errors:** Any issues that occur while querying the divisions will result in a 500 status code with an appropriate error message.

### **Notes**

* **Status Filter:** The query filters divisions to include only those with status: true.
* **Field Selection:** Only the \_id and divisionName fields are returned for each division.

## **33 ) API Endpoint - Get Selected Locations**

Fetches a list of locations that are currently active.

### **HTTP Request**

* **Method:** GET
* **Endpoint:** /selectLocation
* **Authentication:** JWT token required

### **Request Parameters**

* No query parameters are required.

### **Responses**

* **200 OK**
  + **Description:** Locations fetched successfully.
  + **Body:**

{  
 "message": "Locations fetched successfully",  
 "data": [  
 {  
 "\_id": "string",  
 "locationName": "string"  
 }  
 ]  
}

* **500 Internal Server Error**
  + **Description:** An error occurred while fetching locations.
  + **Body:**

{  
 "error": "Something broke"  
}

### **Error Handling**

* **Server Errors:** Any issues that occur while querying the locations will result in a 500 status code with an appropriate error message.

### **Notes**

* **Status Filter:** The query filters locations to include only those with status: true.
* **Field Selection:** Only the \_id and locationName fields are returned for each location.

## **34 ) API Endpoint - Get Selected Zones**

Fetches a list of zones that are currently active.

### **HTTP Request**

* **Method:** GET
* **Endpoint:** /selectZone
* **Authentication:** JWT token required

### **Request Parameters**

* No query parameters are required.

### **Responses**

* **200 OK**
  + **Description:** Zones fetched successfully.
  + **Body:**

{  
 "message": "Zones fetched successfully",  
 "data": [  
 {  
 "\_id": "string",  
 "zoneName": "string"  
 }  
 ]  
}

* **500 Internal Server Error**
  + **Description:** An error occurred while fetching zones.
  + **Body:**

{  
 "error": "Something broke"  
}

### **Error Handling**

* **Server Errors:** Any issues that occur while querying the zones will result in a 500 status code with an appropriate error message.

### **Notes**

* **Status Filter:** The query filters zones to include only those with status: true.
* **Field Selection:** Only the \_id and zoneName fields are returned for each zone.

## **35 ) API Endpoint - Create Zone**

Creates a new zone with the specified details.

### **HTTP Request**

* **Method:** POST
* **Endpoint:** /addZone
* **Authentication:** JWT token required

### **Request Body**

* **Content-Type:** application/json
* **Body:**

{  
 "zoneName": "string",  
 "status": "boolean"  
}

* **Fields:**
  + zoneName (string, required): The name of the zone to be created.
  + status (boolean): The status of the zone, indicating whether it is active.

### **Responses**

* **201 Created**
  + **Description:** Zone created successfully.
  + **Body:**

{  
 "message": "Zone created successfully",  
 "data": {  
 "\_id": "string",  
 "zoneName": "string",  
 "status": "boolean",  
 "adminId": "string"  
 }  
}

* **500 Internal Server Error**
  + **Description:** An error occurred while creating the zone.
  + **Body:**

{  
 "error": "Something broke"  
}

### **Error Handling**

* **Validation Errors:** If the zoneName is missing, the server will respond with a 400 status code indicating that required fields are missing.
* **Server Errors:** Any issues during zone creation will result in a 500 status code with an appropriate error message.

### **Notes**

* **Admin ID:** The adminId field is automatically populated with the ID of the user making the request.
* **Required Fields:** Ensure that the zoneName is provided in the request body.

## **36 ) API Endpoint - Get Zones**

Fetches a list of zones with optional pagination, sorting, and searching.

### **HTTP Request**

* **Method:** GET
* **Endpoint:** /zones
* **Authentication:** JWT token required

### **Query Parameters**

* **page** (integer, optional): The page number for pagination (default: 1).
* **limit** (integer, optional): The number of records per page (default: 10).
* **order** (string, optional): The sort order. Possible values are "ascending" or "descending".
* **sort** (string, optional): The field to sort by.
* **search** (string, optional): The search term to filter zones by zoneName.

### **Responses**

* **200 OK**
  + **Description:** Zones fetched successfully.
  + **Body:**

{  
 "message": "Zone fetched successfully",  
 "data": [  
 {  
 "\_id": "string",  
 "zoneName": "string",  
 "status": "boolean",  
 "adminId": "string"  
 }  
 ],  
 "totalCount": "integer"  
}

* **500 Internal Server Error**
  + **Description:** An error occurred while fetching the zones.
  + **Body:**

{  
 "error": "Something broke"  
}

### **Error Handling**

* **Server Errors:** Any issues while fetching zones will result in a 500 status code with an appropriate error message.

### **Notes**

* **Pagination:** If there are more zones available than the limit per page, result.next will be included in the response with the details for the next page.
* **Sorting:** If the sort parameter is not provided, zones will be sorted by createdAt in descending order by default.
* **Searching:** If the search parameter is provided, the response will be filtered to include only those zones with zoneName matching the search term (case-insensitive).

## **37 ) API Endpoint - Update Zone**

Updates the details of an existing zone.

### **HTTP Request**

* **Method:** PUT
* **Endpoint:** /updateZone/:zoneId
* **Authentication:** JWT token required

### **URL Parameters**

* **zoneId** (string, required): The ID of the zone to be updated.

### **Request Body**

* **zoneName** (string, optional): The new name for the zone.
* **status** (boolean, optional): The new status of the zone (active/inactive).

### **Responses**

* **201 Created**
  + **Description:** Zone updated successfully.
  + **Body:**

{  
 "data": {  
 "\_id": "string",  
 "zoneName": "string",  
 "status": "boolean",  
 "adminId": "string"  
 },  
 "message": "Zone updated successfully"  
}

* **400 Bad Request**
  + **Description:** Invalid zoneId or other input validation errors.
  + **Body:**

{  
 "error": "Invalid ID or missing fields"  
}

* **404 Not Found**
  + **Description:** Zone not found.
  + **Body:**

{  
 "error": "Zone not found"  
}

* **500 Internal Server Error**
  + **Description:** An error occurred while updating the zone.
  + **Body:**

{  
 "error": "Something broke"  
}

### **Error Handling**

* **Invalid Zone ID:** If zoneId is not provided or is invalid, a 400 status code will be returned.
* **Zone Not Found:** If no zone with the given zoneId exists, a 404 status code will be returned.
* **Server Errors:** Any issues during the update process will result in a 500 status code with an appropriate error message.

### **Notes**

* **Admin ID:** The ID of the user performing the update is recorded as adminId in the zone document.
* **Partial Updates:** You can update only the fields you provide; any unspecified fields will remain unchanged.

## **38 ) API Endpoint - Update Zone Status**

Toggles the status of an existing zone between active and inactive.

### **HTTP Request**

* **Method:** PATCH
* **Endpoint:** /updateZoneStatus/:zoneId
* **Authentication:** JWT token required

### **URL Parameters**

* **zoneId** (string, required): The ID of the zone whose status is to be updated.

### **Request Body**

* No request body is required for this endpoint.

### **Responses**

* **201 Created**
  + **Description:** Status updated successfully.
  + **Body:**

{  
 "data": {  
 "\_id": "string",  
 "zoneName": "string",  
 "status": "boolean",  
 "adminId": "string"  
 },  
 "message": "Status updated successfully"  
}

* **400 Bad Request**
  + **Description:** Invalid zoneId.
  + **Body:**

{  
 "error": "Invalid ID"  
}

* **404 Not Found**
  + **Description:** Zone not found.
  + **Body:**

{  
 "error": "Zone not found"  
}

* **500 Internal Server Error**
  + **Description:** An error occurred while updating the status.
  + **Body:**

{  
 "error": "Something broke"  
}

### **Error Handling**

* **Invalid Zone ID:** If zoneId is not provided or is invalid, a 400 status code will be returned.
* **Zone Not Found:** If no zone with the given zoneId exists, a 404 status code will be returned.
* **Server Errors:** Any issues during the status update process will result in a 500 status code with an appropriate error message.

### **Notes**

* **Status Toggle:** The status field will be toggled between its current value (active/inactive).
* **No Request Body:** This endpoint does not require a request body; it only toggles the status based on the existing zoneId.

## **39 ) API Endpoint - Delete Zone**

Deletes an existing zone by its ID.

### **HTTP Request**

* **Method:** DELETE
* **Endpoint:** /deleteZone/:zoneId
* **Authentication:** JWT token required

### **URL Parameters**

* **zoneId** (string, required): The ID of the zone to be deleted.

### **Request Body**

* No request body is required for this endpoint.

### **Responses**

* **201 Created**
  + **Description:** Zone deleted successfully.
  + **Body:**

{  
 "data": {  
 "\_id": "string",  
 "zoneName": "string",  
 "status": "boolean",  
 "adminId": "string"  
 },  
 "message": "Zone deleted successfully"  
}

* **400 Bad Request**
  + **Description:** Invalid zoneId.
  + **Body:**

{  
 "error": "Invalid ID"  
}

* **404 Not Found**
  + **Description:** Zone not found.
  + **Body:**

{  
 "error": "Zone not found"  
}

* **500 Internal Server Error**
  + **Description:** An error occurred while deleting the zone.
  + **Body:**

{  
 "error": "Something broke"  
}

### **Error Handling**

* **Invalid Zone ID:** If zoneId is not provided or is invalid, a 400 status code will be returned.
* **Zone Not Found:** If no zone with the given zoneId exists, a 404 status code will be returned.
* **Server Errors:** Any issues during the deletion process will result in a 500 status code with an appropriate error message.

### **Notes**

* **No Request Body:** This endpoint does not require a request body; it simply deletes the zone with the specified zoneId.
* **Deletion Result:** If the zone is successfully deleted, the response will include the deleted zone’s data.