

Service Contract - Front End & Back End

CRUD

These CRUD functions applies to all the endpoint in the data base and unless otherwise specified all the CRUD functions will behave as described below with the exception of the response and request body which will change based on the data

Create

1. API Endpoint:

- a. Endpoint: `/api/{table}/create`
- b. Method: POST

2. Request Payload:

- a. Content-Type: application/json
- b. Body:

```
{
  "attribute": "value",
  "attribute": value,
  ...
}
```

3. Response:

- a. HTTP Status Code:
 - i. Created: 200
 - ii. Error: 4xx/5xx
- b. Body:
 - i. Success

```
{
  "message" : "Appliance created successfully"
}
```

1. Error

```
{
  "error": "error message"
}
```

Read all

1. API Endpoint:

a. Endpoint: `/api/{table}/all`

b. Method: GET

2. Response

a. HTTP Status Code:

i. Success: 200

ii. Error: 4xx/5xx

b. Content-Type: application/json

c. Body:

i. Success

```
{
  "{tableName}s": [
    {
      "attribute": "value",
      "attribute": "value",
      "attribute": value,
      ...
    },
    {
      "attribute": "value",
      "attribute": "value",
      "attribute": value,
      ...
    },
    ...
  ]
}
```

1. Error

```
{
  "error": "error message"
}
```

Read

1. API Endpoint:
 - a. Endpoint: `/api/{table}/{table}id`
 - b. Method: GET
2. Request Parameters:
 - a. `{table}Id`: The unique identifier of the table id to be read. This parameter should be included in the endpoint URL.
3. Response:
 - a. HTTP Status Code:
 - i. Success: 200
 - ii. Error: 4xx/5xx
 - b. Content-Type: application/json
 - c. Body:
 - i. Success:

```
{
  "{table}Id": value,
  "attribute": "value"
  "value" : value
  ...
}
```

1. Error:

```
{
  "error": "error message"
}
```

Update

1. API Endpoint:

a. Endpoint: `/api/{table}/{table}Id`

b. Method: Patch

2. Request Parameters:

a. `{table}Id`: The unique identifier of the record to be updated. This parameter should be included in the endpoint URL.

b. Request Payload:

i. Content-Type: application/json

ii. Body

```
{
  "attribute": "value",
  "attribute": "value",
  "attribute": value,
  ...
}
```

3. Response:

a. HTTP Status Code:

i. Success: 200

ii. Error: 4xx/5xx

b. Content-Type: application/json

c. Body:

i. Success

```
{
  "message": "{record} updated successfully."
}
```

1. Error

```
{
  "error": "Failed to update {record}"
}
```

Delete

1. API Endpoint:

a. Endpoint: `api/{table}/{table}Id`

b. Method: DELETE

2. Request Parameters:

a. `{table}Id`: The unique identifier of the record to be deleted. This parameter should be included in the endpoint URL

3. Response:

a. HTTP Status Code:

i. Success: 200

ii. Error: 4xx/5xx

b. Content-Type: application/json

c. Body:

i. Success

```
{
  "message": "{record} deleted successfully."
}
```

1. Error

```
{
  "error": "error message"
}
```

Endpoints

Appliances

- Attributes:
 - applianceId
 - type

- powerUsage
- API endpoint:
 - Endpoint: `/api/appliance/`

Basic Calculation

- Attributes:
 - basicCalculationId
 - systemId
 - dayLightHours
 - location
 - batteryLife
 - dateCreated
- API endpoint:
 - Endpoint: `/api/basicCalculation/`

Image

- Attributes:
 - imageId
 - trainingDataId
 - image
- API endpoint:
 - Endpoint: `/api/image/`

Key

- Attributes:
 - keyId
 - owner
 - key
 - remainingCalls

- suspended
- API endpoint:
 - Endpoint: `/api/key/`

Report

- Attributes:
 - reportId
 - reportName
 - userId
 - basicCalculationId
 - solarScore
 - runningtime
 - dateCreated
- API endpoint:
 - Endpoint: `/api/report/`

Report Appliance

- Attributes:
 - reportId
 - applianceId
- API endpoint:
 - Endpoint: `/api/reportAppliance/`

System

- Attributes:
 - systemId
 - inverterOutput
 - numberOfPanels
 - batterySize

- numberOfBatteries
- solarInput
- API endpoint:
 - Endpoint: `/api/system/`

Training Data

- Attributes:
 - trainingData
 - solarIrradiation
- API endpoint:
 - Endpoint: `/api/trainingData/`

Users - alternate CRUD

- Attributes:
 - email
 - password
 - userRole
 - dateCreated
 - lastLoggedIn

Authentication

Registration

1. API Endpoint
 - a. Endpoint: `/api/auth/register`
 - b. Method: *POST*
2. Request Payload
 - a. Content-Type: application/json

```
{  
  "email": "john@example.com",
```



```
"password": "password123_hashed",  
"userRole" : 0  
}
```

3. Response:

a. HTTP Status Code

i. Success: 201

ii. Error: 4xx/5xx

b. Content-Type: application/json

c. Body:

i. Success

```
{  
  "message": "User registered successfully."  
}
```

1. Error

```
{  
  "error": "Invalid input data.",  
  "details": "Email already exists."  
}
```

Login

1. API Endpoint:

a. Endpoint: `/api/auth/login`

b. Method: POST

2. Request Payload

a. Content-Type: application/json

```
{  
  "email": "john@example.com",  
  "password": "password123"  
}
```

3. Response:

a. HTTP Status Code:

i. Success: 200

ii. Error: 4xx/5xx

b. Content-Type: application/json

c. Body:

i. Success:

```
{
  "accessToken": "<access_token>",
  "expiresIn": "<expiration_time>"
}
```

1. Error

```
{
  "error": "Invalid credentials.",
  "details": "Email or password is incorrect."
}
```

Check Email

1. API Endpoint:

a. Endpoint: `/api/auth/checkemail`

b. Method: GET

2. Request Payload

a. Content-Type: application/json

```
{
  "email": "john@example.com",
}
```

3. Response:

a. HTTP Status Code:

i. Success: 200

- ii. Error: 4xx/5xx
- b. Content-Type: application/json
- c. Body:
 - i. Success:

```
{
  "accessToken": "<access_token>",
  "expiresIn": "<expiration_time>"
}
```

1. Error

```
{
  "error": "Unauthorized.",
  "details": "Email already exists"
}
```

Update Last Logged In

- 1. API Endpoint:
 - a. Endpoint: `/api/auth/lastLoggedIn`
 - b. Method: Patch
- 2. Request Payload
 - a. Content-Type: application/json
- 3. Response:
 - a. HTTP Status Code:
 - i. Success: 200
 - ii. Error: 4xx/5xx
 - b. Content-Type: application/json
 - c. Body:
 - i. Success:

```
{
  "message": "User Last logged in field updated successfully"
}
```

1. Error

```
{
  "error": "error message"
}
```

CRUD users

Get all users

1. API Endpoint:

a. Endpoint: `/api/users/all`

b. Method: GET

2. Request Parameters: None

3. Response:

a. HTTP Status Code:

i. Success: 200

ii. Error: 4xx/5xx

b. Content-Type: application/json

c. Body:

i. Success:

```
{
  "users": [
    {
      "userId": 1,
      "email": "john@example.com"
      "password" : "asdasdads"
      "userRole" : 0
      "dateCreated": "2008-11-11 13:23:44"
    },
    {
      "userId": 2,
      "email": "jane@example.com"
      "password" : "asdasdads"
      "userRole" : 1,
      "dateCreated": "2008-11-11 13:23:44"
    }
  ]
}
```

```
    },  
    ...  
  ]  
}
```

1. Error:

```
{  
  "error": "Failed to retrieve users.",  
  "details": "Database connection error."  
}
```

Get user

1. API Endpoint:

- a. Endpoint: `/api/users/:userid`
- b. Method: GET

2. Request Parameters:

- a. `userId`: The unique identifier of the user to be deleted. This parameter should be included in the endpoint URL

3. Response:

- a. HTTP Status Code:
 - i. Success: 200
 - ii. Error: 4xx/5xx
- b. Content-Type: application/json
- c. Body:
 - i. Success:

```
{  
  "userId": 1,  
  "email": "john@example.com"  
  "password" : "asdasdads"  
  "userRole" : 0  
  "dateCreated": "2008-11-11 13:23:44",  
  "lastLoggedIn": "2008-11-11 13:23:45",  
}
```

1. Error:

```
{
  "error": "Not Found.",
  "details": "User does not exist"
}
```

Delete User

1. API Endpoint:

- a. Endpoint: `api/users/:userId`
- b. Method: DELETE

2. Request Parameters:

- a. `userId`: The unique identifier of the user to be deleted. This parameter should be included in the endpoint URL

3. Response:

- a. HTTP Status Code:
 - i. Success: 200
 - ii. Error: 4xx/5xx
- b. Content-Type: application/json
- c. Body:
 - i. Success

```
{
  "message": "User deleted successfully."
}
```

1. Error

```
{
  "error": "Failed to delete user.",
  "details": "User with the specified ID not found."
}
```

Update User

1. API Endpoint:

a. Endpoint: `/api/users/:userId`

b. Method: PUT

2. Request Parameters:

a. `userId`: The unique identifier of the user to be updated. This parameter should be included in the endpoint URL.

b. Request Payload:

i. Content-Type: application/json

ii. Body

```
{
  "email": "john@example.com",
  "password": "newpassword123"
  "userRole"? : 0
}
```

3. Response:

a. HTTP Status Code:

i. Success: 200

ii. Error: 4xx/5xx

b. Content-Type: application/json

c. Body:

i. Success

```
{
  "message": "User updated successfully."
}
```

1. Error

```
{
  "error": "Failed to update user.",
  "details": "User with the specified ID not found."
}
```