Routes (API Endpoint Description)

API Endpoint URL	Method	Endpoint Description	Request Payload	Response Example
			(All the bold fields are required, and 400 will be responded with if not presented in the request body) (All the italic fields are passed as a URL params, means missing one of the fields will cause 404) (geom: please refer to GEOM data structure example)	(If there is no example included, it means the response is not important for this endpoint)

/api/evaporation	POST	fetch evapoartion data	• **fieldID**
			• startDate: will use current time if not presented
			• endDate: same as startDate
				`{`
				`  "data": [`
				`    {`
				`      "evaporation": 1.756682698859334`
				`      "date": "07-28-2023",`
				`      "field_id": "159e607d-5f68-4350-`
				`    }`
				`  ]`
				`}`

/api/farm	POST	get farm data with username	• **userName**	`{`
				`  "data": [`
				`    {`
				`      "farm_name": "DemoFarm",`
				`      "username": "demo"`
				`    }`
				`  ]`
				`}`

/api/farm/addFarm	POST	insert new farm	• **userName** • **farmName**	
/api/farm/:username/:farm	DELETE	delete a farm		

/api/	POST	get field information by username,
		
		if passed with fieldName, it will get the specified field if this field is linked with the presented username
			• **userName**
			• fieldName: if this entry is defined, fields with this name will be returned if there is any field with this fieldName is linked with the given userName
				`{`
				`  "data": [`
				`    {`
				`      "points": "{\"type\":\"Polygon\"`
				`      "field_name": "field 1",`
				`      "crop_type": null,`
				`      "soil_type": null,`
				`      "farm_name": "test farm",`
				`      "username": "demo",`
				`      "geom": "...",`
				`      "elevation": null,`
				`      "field_id": "c51e3b20-719b-44d4`
				`    }`
				`  ]`
				`}`

'/api/field/addField'	POST	Create new field	• **fieldName** • **userName** • **farmName** • **geom** • cropType • soilType	
'/api/field/:fieldId'	DELETE	Delete field, and delete all sensors installed in that field		

'/api/field/:fieldId'	GET	Get field data by field id		`{`
				`  "data": {`
				`    "points": "{\"type\":\"Polygon\",\`
				`    "field_name": "field 1",`
				`    "crop_type": null,`
				`    "soil_type": null,`
				`    "farm_name": "test farm",`
				`    "username": "demo",`
				`    "geom": "...",`
				`    "elevation": null,`
				`    "field_id": "c51e3b20-719b-44d4-8`
				`  }`
				`}`

'/api/sensorformula/:sensorId'	GET	get sensor formula by sensor id		`{`
				`  "data": {`
				`    "sensor_id": "...",`
				`    "formula": "x1*(val^x2)/100",`
				`    "formula_id": 516,`
				`    "parameter": "...",`
				`    "mode": "default",`
				`    "lowest_adt": 0,`
				`    "h_parameter": "...",`
				`    "h_formula": "..."`
				`  }`
				`}`
				
				`mode` is either `default` or `specific`

/api/sensorformula	GET	get all sensor formulas		`{`
				`  "data": [`
				`    {`
				`      "sensor_id": "...",`
				`      "formula": "x1*(val^x2)/100",`
				`      "formula_id": 516,`
				`      "parameter": "...",`
				`      "mode": "default",`
				`      "lowest_adt": 0,`
				`      "h_parameter": "...",`
				`      "h_formula": "..."`
				`    }`
				`  ]`
				`}`

/api/sensorformula	POST	create new sensor formula	• **sensor_id**
			• **formula:**
			◦ must include `val` in the formula string, 400 will be returned if not present
			◦ example: `x1*(val^x2)/100`
			• **parameter**
			◦ must in the form of `number, number, number...`
			◦ formula must have same amount of parameters
			◦ example: `"2285.7, -0.944"`
			• mode: set to specific if not defined
			• lowest_adt: set to 0 if not defined
			• h_parameter
			• h_formula
				`{`
				`  "data": "success"`
				`}`

/api/sensorformula/:formulaId	PUT	update formula by formula id	same structure with create formula	same above
/api/sensorformula/:formulaId	DELETE	delete formula by formula id		same above

'/api/gateway/'	POST	fetch gateway by username	**userName**	`{`
				`  "data": [`
				`    {`
				`      "gateway_id": "AquaTerraGatewayD`
				`      "username": "demo",`
				`      "geom": "..."`
				`    }`
				`  ]`
				`}`

'/api/gateway/new'	POST	create a new gateway	• **userName** • **gatewayId** • **geom**	
'/api/gateway/delete'	POST	delete a gateway	• **userName** • **gatewayId**	

'/api/gateway/field'	POST	get gateways within a field	• **fieldId**
			• **username**
				`{`
				`  "data": [`
				`    {`
				`      "points": "{\"type\":\"Point\",\`
				`      "gateway_id": "AquaTerraGatewayD`
				`    }`
				`  ]`
				`}`

'/api/gateway/setup'	POST	activate the gateway, allow	• **status:** boolean type data	`{`
				`  "data": "ok"`

		the gateway to start pairing sensors	• **gatewayIds**	`}`

'/api/gateway/sensors'	POST	use AWS IoT shadow to check paired sensors and store the sensor data into PostgreSQL database	• **gatewayIds**
			• **userName**
				```
				{
				  "data": [
				    {
				      "sensor_id": "AFA00000DEMO3",
				      "points": "{\"type\":\"Point\",\'
				    }
				  ]
				}
				```

'/api/moisture'	POST	get latest moisture data of a sensor	• **sensorID**
			• **fieldName**
			• **userName**
				```
				{
				  "data": [
				    {
				      "time": "2023-08-11T09:14:19.07
				      "latitude": -38.280853,
				      "longitude": 145.079919,
				      "humidity": null,
				      "temperature": 11.5,
				      "geom": "...",
				      "battery_vol": 4.03,
				      "cap50": 47.5,
				      "cap100": 46.9,
				      "cap150": 52.37,
				      "sensor_id": "AFA00000DEMO10",
				      "has_notified": false
				    }
				  ]
				}
				```

'/api/moisture'	POST	1. get latest moisture data of a sensor in within given dataRange
		2. get all moisturedata of all sensors from startDate to endDate
		3. get all moisturedata of all sensors within the given dataRange
			• sensorID: used for case 1
			• **fieldName**
			• **userName**
			• startDate: used for case 2, in the form of `new Date()`
			• endDate: same as startDate
			• dateRange: in milli seconds used in case 1 and 3, it equals to 14 days if not provided, i.e, the return value will contain record from last 14 days
				```
				{
				  "data": [
				    {
				      "time": "2023-08-11T09:14:19.07
				      "latitude": -38.280853,
				      "longitude": 145.079919,
				      "humidity": null,
				      "temperature": 11.5,
				      "geom": "...",
				      "battery_vol": 4.03,
				      "cap50": 47.5,
				      "cap100": 46.9,
				      "cap150": 52.37,
				      "sensor_id": "AFA00000DEMO10",
				      "has_notified": false
				    }
				  ]
				}
				```

'/api/moisture/prediction'	POST	get the prediction data of a given sensor, the data will contain prediction for every 2 hours within the future 3 days	**sensorID**	`{`
				`  "data": [`
				`    {`
				`      "name": "Fri Aug 11 2023 12:00:`
				`      "Temperature": 11.3,`
				`      "Layer_0": 30,`
				`      ...,`
				`      "Layer_19": 30`
				`    }`
				`  ]`
				`}`

api/moisture/past-prediction	POST	deprecated and not in use basically the same as the above except it returns all historical prediction record		

'/api/sensor/field'	POST	fetch sensors installed in the given field	**fieldId**	`{`
				`  "data": [`
				`    {`
				`      "sensor_id": "...",`
				`      "gateway_id": "...",`
				`      "field_id": "...",`
				`      "geom": "...",`
				`      "datetime": "2021-03-26T00:00:0`
				`      "is_active": true,`
				`      "has_notified": false,`
				`      "username": "demo",`
				`      "sleeping": 3,`
				`      "alias": null,`
				`      "points": "{\"type\":\"Point\",`
				`      "field_name": "Mornington"`
				`    }, ...`
				`  ]`
				`}`

'/api/sensor/'	POST	fetch sensors within the field not in use behaves basically the same with the above	**fieldName** **userName**	

'/api/sensor/:sensorId'	DELETE	delete sensor by sensor id and		`{`
				`  "data": [`

		also unpair the gateway from AWS IoT shadow		`    {`
				`      "gateway_id": ...,`
				`      "sensor_id": ...`
				`    }`
				`  ]`
				`}`

'/api/sensor/new'	POST	link the sensor to the given field with the provided position	• **sensorId** • **gatewayId** • **fieldId** • **geom**	

'/api/sensor/:sensorId?username=${username}&fieldId=${fieldId}'	GET	get sensor details with given sensor id	The following fields are passed as query parameters
			• **username**
			• **fieldId**
				`{`
				`  "sensor": {`
				`    "sensor_id": "test",`
				`    "gateway_id": null,`
				`    "field_id": "...",`
				`    "geom": "...",`
				`    "datetime": "2023-08-11T00:00:00.`
				`    "is_active": false,`
				`    "has_notified": false,`
				`    "username": "...",`
				`    "sleeping": 3600,`
				`    "alias": null,`
				`    "points": "{\"type\":\"Point\",\"`
				`  }`
				`}`

'/api/sensor/:sensorId'	PUT	update sensor details with given sensor id, update details with the provided fields in the body	• geom
			• isActive
			• sleeping
			• frequency
			• alias
				`{`
				`  "data": [`
				`    {`
				`      "sensor_id": "...",`
				`      "gateway_id": null,`
				`      "field_id": "...",`
				`      "geom": "...",`
				`      "datetime": "2023-08-11T00:00:0`
				`      "is_active": false,`
				`      "has_notified": false,`
				`      "username": "...",`
				`      "sleeping": 3600,`
				`      "alias": "...",`
				`      "st_asgeojson": "{\"type\":\"Po`
				`    }`
				`  ]`
				`}`

'/api/sensor/v2/new'	POST	install version 2 sensor (sensors that are not paired with gateway, instead it connect to AWS IoT Core directly	• **sensorId** • **fieldId** • **geom** • **username**	`{ data: "success" }`

		on the hardware side)		
'/api/user/'		All the user endpoints are not in use currently.		

'/api/zone'	POST	Get all the zone data,
		
		If `withSensor` is set to `true`, the return data will contains `sensors: [sensorIds]` that is geologically within the zone
			• **userName**
			• withSensor: boolean type, false by default
				`{`
				`  "data": [`
				`    {`
				`      "points": "{\"type\":\"Polygon\`
				`      "zonename": "testZone",`
				`      "fieldname": "Mornington",`
				`      "croptype": "Corn",`
				`      "soiltype_25": "Loam",`
				`      "soiltype_75": "Loam",`
				`      "soiltype_125": "Loam",`
				`      "geom": "...",`
				`      "username": "demo",`
				`      "farmname": "DemoFarm",`
				`      "wpoint_50": 7,`
				`      "wpoint_100": 7,`
				`      "wpoint_150": 7,`
				`      "fcapacity_50": 20,`
				`      "fcapacity_100": 20,`
				`      "fcapacity_150": 20,`
				`      "saturation_50": 30,`
				`      "saturation_100": 30,`
				`      "saturation_150": 30,`
				`      # this field will present when`
				`      "sensors": ["...","..."]`
				`    }`
				`  ]`
				`}`

'/api/zone/wpoint'	POST	fetchWPoints	• **fieldName**
			• **userName**
			• **sensorID**
				`{`
				`  "data": [`
				`    {`
				`      "wpoint_50": 7,`
				`      "wpoint_100": 7,`
				`      "wpoint_150": 7,`
				`      "fcapacity_50": 20,`
				`      "fcapacity_100": 20,`
				`      "fcapacity_150": 20,`
				`      "saturation_50": 30,`
				`      "saturation_100": 30,`
				`      "saturation_150": 30`
				`    }`
				`  ]`
				`}`

'/api/zone/deleteZone'	DELETE	delete the zone	• **userName** • **fieldName** • **zoneName**	

'/api/zone/addZone'	POST	create a new zone	• **userName** • **fieldName** • **zoneName** The following is also required but will not raise any error • farmName • geom • cropType soilType25 soilType75 soilType125 wPoint50 wPoint100 wPoint150 fCapacity50 fCapacity100 fCapacity150 saturation50 saturation100 saturation150	
'/api/zone/'	PUT	update zone details	same as above with following extra: **oldZoneName**	