API Documentation

Base URL: https://rapid-320.vm.duke.edu:8443/

Basic Format

API endpoints always start with /api
Authentication related: /api/auth/…
User related: /api/user/…
Custom fields: /api/customField/…
Logging: /api/log/…
Cart: /api/cart/…
Item: /api/item/…
Requests: /api/request/…
Tags: /api/tag/...

To get an API key, first login to the web interface, then hit the /api/user/apiKey/get endpoint with a browser. Then, put the API key as “x-key” field in GET/POST header.

A tutorial with screenshots and reference videos can be found at the bottom of the document.

Login and OAuth:

Login with credentials
POST /api/auth/login
Parameters:
Sample JSON:

```
{
  "username": "test1",
  "password": "12345"
}
```

This method allows the user to be logged in on the server side; after logging in, the server will return a token that can be used for authentication.

Login with OAuth
GET /api/auth/oauth

Parameters: None

This API redirects the user to login via OAuth to use their NetID. A browser support is needed for this method

Create an Admin account (Can only be used once)
/api/auth/createAdmin

Parameters: None

Output:
Sample JSON:


```
{
  "username": "admin",
  "name": "Super User",
  "password": "admin"
}
```


This method allows the creation of the administrative user.

User:

Get current user info, if logged in (require login)
GET /api/user

Parameters: None

Sample output:
Not logged in:
{"status":403,"message":"Not Authorized"}
Logged in:


```
{
  "username": "admin",
  "name": "",
  "netId": "",
  "email": "",
  "status": "admin",
  "active": true,
  "apiKey":
"0c0817ecf9799c7dffc288b9ae92bab80c9696dd25969c145df0729daa7b50781370926e31767
96bc4f187e37bc808d505f2fdedf9671bf43d7db67f66adbfd6"
}
```

Get the API key for the current user (require login)
GET /api/user/apiKey/get

Parameters: None

This method returns the API key for the current logged-in user.

Revoke the API key for the current user (require login)
GET /api/user/apiKey/revoke

Parameters: None

This method revokes the API key for the current user.

Get the user info whose primary key in the database is :id (require admin or manager)
GET /api/user/:id/info

Parameters: None

This method returns a JSON file that represents the user in the database with the id matching :id.

View a list of all users (require admin or manager)
GET /api/user/show

Parameters: None

This method returns a JSON file that includes a list of all users in the database.

Add a user (require admin or manager)
POST /api/user/add

Parameters:
Sample JSON:


```
{
  "username": "test",
  "name": "Test User",
  "password": "12345",
  "netId": "",
  "email": "test@example.com",
  "status": "user",
```

```
  "active": true
}
```

This method allows an admin or a manager to create a user. Note that the fields username and password are required.

Update a user (require admin or manager)
POST /api/user/update

Parameters:

Sample JSON:

```
{
  "_id": < _id of the user, can get from /api/user/show, or leave empty to update the current user>,
  "username": "test1",
  "name": "Test User",
  "password": "12345",
  "netId": "",
  "email": "test@example.com",
  "status": "user",
  "active": true
}
```

This method allows an admin or a manager to create a user. Note that the id field is required to locate the user in the database. Username and password fields are again required.

Delete a user (require admin or manager)
POST /api/user/del
Sample JSON:

```
{
  "_id": < _id of the user, can get from /api/user/show>
}
```

This method allows an admin or a manager to create a user. Note that the id field is required to locate the user in the database.

Allow a user to subscribe(require admin or manager)
POST /api/user/del
Sample JSON:

```
{
  "_id": < _id of the user, can get from /api/user/show>
}
```

This method allows an admin or a manager to subscribe to the email serv. If the user is not an admin or a manager, this method will fail.

Custom Fields:

Show all custom fields:
GET /api/customField/show

Parameters: None

This method shows all the created custom fields; it returns them in a JSON file.

Add (requires admin or manager):
POST /api/customField/add

Parameters: Sample JSON:

```
{
  "name": “location”,
  "type": "short text",
  "private": "true",
  "req": "false",
  }
```

This is the method that is used to add a new custom field into the program. Both name and type fields are the only fields that are required. Type can be one of “short text”, “long text”, “int” or “float”. Only administrators can set a field to be private.

Update:
POST /api/customField/update

Parameters: Sample JSON:

```
{
  "name": “location”,
  "type": "short text",
  "private": "true",
```

```
  "req": "false",
  }
```

This method can be used to update a custom field. The name field is required to update

Delete:
POST /api/customField/del

Parameters:
JSON fields:
Parameters: Sample JSON:

```
 {
  "name": “location”,
  }
```

This method allows custom fields to be deleted. Since the method uses the name to locate the tag, the “name” field is required in the JSON.

Logging
Show a list of logs
GET /api/log/show?limit=20

Parameters: None

If limit param is supplied, the # of log entries will be subject to that param. Otherwise, the default limit is 20. The list is always sorted by descending time.

Filter a list of logs
GET or POST /api/log/filter

Sample JSON:

```
 {
  "limit": 40,
  "init_user": "Mike", (optional, regex match on name field of users)
  "rec_user": "Jason", (optional, regex match on name field of users)
  "item": "computer chip",  (optional, regex match on name field of items)
 }
```

This method returns the list of logs associated with the search queries. Limit field is required due to the potential high volume of the log data.

Cart:

All following APIs will return a JSON of all current items in the cart

Show all items in the cart

GET /api/cart/show

Parameters: None

This method allows the users to see their own cart at a given time.

Add an item to the cart

POST /api/cart/add

Parameters:

Sample JSON:

```
{
   "item ": "_id of an item (can get from /api/item/show ",
   "quantity": 100
 }
```

This method allows users to add items to their carts.

Delete an item from the cart

/api/cart/delete

Parameters:

Sample JSON:

```
{
   "item ": "_id of an item (can get from /api/item/show ",
 }
```

This method allows users to delete items to their carts.

Update the quantity of an item to the cart

/api/cart/update

Parameters:

Sample JSON:

```
{
   "item ": "_id of an item (can get from /api/item/show ",
   "quantity": 100
 }
```

Update works the same way as add. If the item is not currently in the cart, it will be added. If the item is already in the cart, its quantity will be updated.

Item

Show a list of items (require login)
GET /api/item/show?limit=20

Parameters: None

This shows a list of all items. If limit param is supplied, the # of items will be subject to that param. Otherwise, the default limit is 20.

Add an item (require admin or manager)
POST /api/item/add

Sample JSON:

```
{
   "name": "i7 Chip",
   "fields": {  // a list of custom fields
     "location": "room1",
     "note": "note"
   },
   "tags": ["chip","latest"], // an array of tags
   "description": "latest intel chip",
   "model": "i7 7700k",
   "quantity": 100
 }
```

Allows creation of an item. The name field, fields that are required, and quantity are required fields.

Delete an item (require admin or manager)
POST /api/item/del

Parameters:

Sample JSON:

```
{
   "name": "i7 Chip",
 }
OR
{
   "_id": "58ab62ceca32af82ffdef58b", (can get from /api/item/show)
 }
```

This method allows an admin or a manager to delete an item stored in the database. Note that

Update an item (require admin or manager)
POST /api/item/update

Parameters:

Sample JSON:

```
{
   "name": "i7 Chip",
   "fields": {  // a list of custom fields
     "location": "room1",
     "note": "note"
   },
   "tags": ["chip","latest"], // an array of tags
   "description": "latest intel chip",
   "model": "i7 7700k",
   "quantity": 100
 }
```

This method allows an admin or a manager to update an item by sending a formatted JSON. The only required field is the “name” field that needs to match the name exactly. (Otherwise, expect a ‘Not Found’ error)

Requests

Show a list of requests, ordered by descending date (Require login)
GET /api/request/show

Parameters: None

This method allows the user to login, and see the requests associated with him.

Add a request from cart (Require login)
POST /api/request/add

Parameters:
Sample JSON:
```
{
   "note": "latest intel chip"
}
```

This method allows a request to be created by the user. Note that it automatically associates the current cart of the user and tags along the note to send the server a request.

Close a request (Require login)
POST /api/request/close

Parameters:

Sample JSON:
```
{
  "_id": < _id of the user, can get from /api/request/show, or leave empty to update the current user>,
}
```

This method allows controlling the request submitted by the user. It requires the ID of the user.

Delete a request (Require admin/manager)
POST /api/request/del

Parameters:

Sample JSON:
```
{
    "_id": < _id of the user, can get from /api/request/show, or leave empty to update the current user>,

}
```

This method allows controlling the request submitted by the user. It requires the ID of the user.

Update a request (Require admin/manager for approval)
POST /api/request/update

Parameters:

Sample JSON:
{
"_id": < _id of the user, can get from /api/request/show, or leave empty to update the current user>,
"status": "approved"
"note": "latest intel chip"
}

This method allows updating the request submitted by the user..

Tags

Show a list of all tags. (Req login)
GET /api/tag/show

Parameters: None

This method returns a full list of tags for the user; note that there is no pagination implemented in this method due to number of tags assumed to be bounded.

Add a tag. (Require admin/manager)
POST /api/tag/add

Parameters:
Sample JSON:
{
"name": "cheap"

}

Delete a tag. (Require admin/manager)
POST /api/tag/del

Sample JSON:
{

```
    "name”: "cheap"

}
```

Emails

Delete a tag. (Require admin/manager)
GET /api/email/show

Parameters: None

This method shows the current state of the emails.In particular, it allows the user to see all the scheduled emails in a JSON format.

Update an email. (Require admin/manager)
POST /api/email/update

Parameters:

Sample JSON:

```
{
    "_id": < _id of the email, can get from /api/email/show
    "subject": “Hi”
    "subjectTag": ”TEST”
    "body": ”This is my test email”
    "dates": ”2017-03-20”
}
```

This method allows for updating an email associated with the given ID. This method also sets the subjectTag that is to be sent by the server.

Add an email. (Require admin/manager)
POST /api/email/add

Parameters:

Sample JSON:

```
{
    "_id": < _id of the email, can get from /api/email/show
    "subject": “Hi”
    "body": ”This is my test email”
    "dates": ”2017-03-20”
}
```

This method allows for scheduling an email to be sent.

Bulk Import

Import all items. (Require admin/manager)

POST /api/items/addAll

Parameters:

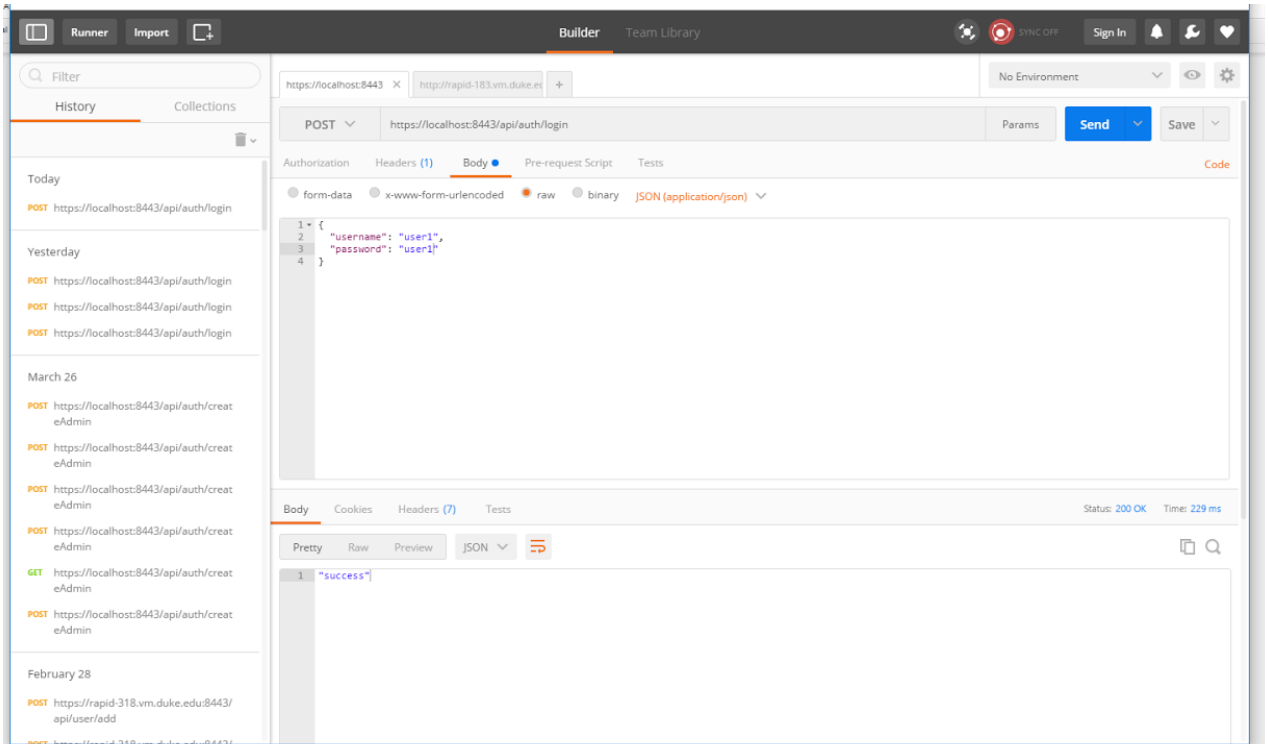
Sample JSON:

```
[{
    "name": “resistors”
    "quantity": “10”
    "quantity_available": ”10”
},
{
    "name": “resistors 10k”
    "quantity": “10”
    "quantity_available": ”10”
    "description": ”small”
}]
```

This method allows for bulk importing data. In particular, it requires a JSON file to be inputted, which is a list of all items that are to be added to the system.

Postman Tutorial and Reference Videos

For testing our API, we suggest using Postman. Postman is a Chrome plugin that allows sending HTTPS requests and analyzing the output clearly. Here is the sample interface:



In the above dropdown next to the address bar, we can select the type of request that is going to be sent to the server. For this project, all the requests are either POST or GET requests. Next, in the address bar, type the base URI + the API address. Note that all of our API functions by sending JSON files to the backend server, so if it is a method that requires an input, go to the Body tab right under the address bar, and select “raw” right below it. For your input type, select JSON. Next, format a raw JSON file. (These can be copied and pasted from this document). Then, simply pressing “Send” right next to the address bar will execute the request and show the response from the API.

If any of this is confusing, the user is encouraged to follow tutorial videos here:

https://www.getpostman.com/docs/

In particular, the user is encouraged to watch until the end of “Sending Requests and Viewing Responses”, and it should not take that long.

Any errors/issues can be communicated to our team via our GitHub page:

https://github.com/anikard/ECE_inventory_system