The Unified MakerSpace

Aru Bhoop

Apr 13, 2021

CONTENTS:

CHAPTER

ONE

OVERVIEW

1.1 Introduction

This is the documentation for the MakerSpace API. It is divided into sections based on functionality. Each page documents for each endpoint, the endpoint url, the required permissions to access, the parameters, and a brief description. The *models* page has specifications for all the objects that will be exchanged by the api.

Note: All dates should be encoded in epoch time (seconds since the start of January 1st, 1970).

1.2 Terminology

1.2.1 Entities

There are two entities that will be using the MakerSpace: *Users* and *Visitors*.

Users

Users are individuals who will be using the dashboard. This includes maintainers and administrators. This enables them, based on their permissions to create tasks, resolve tasks, see visitor statistics, view machine information, and perform administrative tasks.

Visitors

Visitors are students who will be visiting the MakerSpace. Creating an account will at a minimum grant them access to the MakerSpace, but could facilitate deeper interaction in the future (e.g. uploading personal works).

Note: One person may have both a *User* and *Visitor* account with the same email address. The account type may not be changed after creation.

1.3 Example Usage

1.3.1 User

Joe is a maintainer who wants to be able to view and resolve tasks created by his manager, Ivan. He finds the registration page and marks the checkbox that indicates he works at the MakerSpace. He submits the registration form and is prompted for a user token on the next page. He asks Ivan, who generates a new, temporary token from his profile page. Joe enters the token, and his registration is complete. He is redirected to the login page, where he can now use his new credentials.

1.3.2 Visitor

Beck's friends constantly tell her about how amazing the MakerSpace is, so she finally makes plans to visit it next week. Before going, she completes the Canvas tutorial, where she is prompted to create an account with the MakerSpace. She is redirected to the login page, where she enters her email address and creates a password. Completing the form, she is asked for a few more details on the next page, such as her name, major, degree type, and the hardware id of her TigerCard. She honestly enters this information and she is asked to validate her email address on the next page. Upon validating her email address, she finishes the authentication process and can now visit the MakerSpace.

CHAPTER

TWO

DESIGN

2.1 Errors

Errors can propagate to the backend in a few ways:

- Client errors that bypass frontend validation (e.g. user attempts to sign up with an existing email)
- Invalid requests made by the frontend,
- Invalid requests made by an external client (e.g. a bad actor).

Regardless of the intention of bad requests, the backend should handle them robustly and return a clear response. This documentation does not provide an exhaustive list of possible errors, as that would be very difficult and unwieldy.

Instead, implementers are strongly encouraged to perform as many backend checks as possible (even those that duplicate frontend validation), and return semantically meaningful responses. Error responses should have a code, an error description, and message. See the *Models.Error* class for more information.

2.1.1 Example Error

```
response_body = {
    'code': 400,
    'error': "EMAIL_IN_USE",
    'message': 'There is already an account '
               'with this email address.'
}
```

Warning: Do not provide sensitive information in error responses. The front-end may display messages exactly as provided.

CHAPTER

THREE

CONTENTS

3.1 Administrative

`admin.`**`generate_user_token`**(*auth_token: str*)

Generates a random token to create an account for a new user.

Endpoint	/api/admin
Request Type	POST
Access	MANAGER

Parameters **`auth_token`** (`str, required`) – Token to verify user credentials.

Returns

- *Success*
- **code** (*int*) – Return code.
- **user_token** (*str*) – N-digit token used to create a user.

`admin.`**`reset_password`**(*email: str*)

Sends a password reset email.

Endpoint	/api/admin
Request Type	PATCH
Access	ANY

Parameters **`email`** (`str, required`) – The email of the user to reset the password for.

Returns

- *Success*
- **code** (*int*) – Return code.
- **message** (*str*)

3.2 Authentication

The MakerSpace staff are able to create, delete, and manage users with varying permissions, such as administrators, managers, and visitors.

`auth.`**`change_password`**`(`*auth_token: str*, *user_id: str*, *password: str*, *new_password: str*`)`

Changes a user's password.

Note: This function uses the same endpoint as *login*. If *new_password* is specified, then *change_password* is called.

Endpoint	/api/users
Request Type	POST
Access	MAINTAINER

Parameters

- **`auth_token`** (*`str, required`*) – Token to verify user credentials.
- **`user_id`** (*`str, required`*) – The user_id of the user.
- **`password`** (*`str, required`*) – The password of the user.
- **`new_password`** (*`str, required`*) – The new password of the user.

Returns

- *Success*
- **code** (*int*) – Return Code
- **message** (*str*) – Response Message

`auth.`**`create_user`**`(`*email: str*, *password: str*, *first_name: str*, *last_name: str*, *user_token: str*`)`

Creates a new user in the Cognito Pool.

Endpoint	/api/users
Request Type	PUT
Access	ALL

Parameters

- **`first_name`** (*`str, required`*) – The first name of the user.
- **`last_name`** (*`str, required`*) – The last name of the user.
- **`email`** (*`str, required`*) – The email of the user.
- **`password`** (*`str, required`*) – The password of the user.
- **`user_token`** (*`str, required`*) – Token that is generated by an existing administrator that becomes the new user's id.

Returns

- *Success*
- **code** (*int*) – Return Code

- **user** (*models.User*) – The newly created user.
- **auth_token** (*str*) – New authentication token.
- *EmailInUse*
- **code** (*int*) – Return Code
- **message** (*str*) – Response Message
- *InvalidToken*
- **code** (*int*) – Return Code
- **message** (*str*) – Response Message

`auth.`**`delete_user`**(*auth_token: str*, *user_id: str*)

Deletes a user specified by their email.

Endpoint	/api/users
Request Type	DELETE
Access	MAINTAINER

Parameters

- **`auth_token`** (*`str, required`*) – Token to verify user credentials.
- **`user_id`** (*`str, required`*) – The id of the user.

Returns

- *Success*
- **code** (*int*) – Return Code
- **message** (*str*) – Response Message
- *InsufficientPermissions*
- **code** (*int*) – Return Code
- **message** (*str*) – Response Message

`auth.`**`get_users`**(*auth_token: str, user_ids: [<class 'str'>]*)

Gets all the users with their permissions. If called by maintainer, only retrieve names, otherwise get all information.

Endpoint	/api/users
Request Type	GET
Access	MAINTAINER

Parameters

- **`auth_token`** (*`str, required`*) – Token to verify user credentials.
- **`user_ids`** (*`[str], optional`*) – If specified, get only data for users specified by user_id.

Returns

- *Success*
- **code** (*int*) – Return Code

- **users** (*[models.User] | [str]*) – List of returned users or names. Must be in same order of request.

`auth.`**`login`**(*email: str*, *password: str*)

Logs in an existing user.

Endpoint	/api/users
Request Type	POST
Access	ALL

Parameters

- **`email`** (`str, required`) – The email of the user.
- **`password`** (`str, required`) – The password of the user.

Returns

- *Success*
- **code** (*int*) – Return Code
- **user** (*models.User*) – The existing user.
- **auth_token** (*str*) – New authentication token.
- *LoginFailure*
- **code** (*int*) – Return Code
- **message** (*str*) – Response Message

`auth.`**`update_user`**(*auth_token: str*, *user_id: str*, *user:* models.User)

Updates a user's attributes (permissions, card id, name, etc.)

Endpoint	/api/users
Request Type	PATCH
Access	MAINTAINER

Parameters

- **`auth_token`** (`str, required`) – Token to verify user credentials.
- **`user_id`** (`str, required`) – The id of the user to update attributes for.
- **`user`** – The new user object with changed attributes.

Returns

- *Success*
- **code** (*int*) – Return Code
- **message** (*str*) – Response Message
- *InsufficientPermissions*
- **code** (*int*) – Return Code
- **message** (*str*) – Response Message

3.3 Tasks

The MakerSpace staff is able to manage maintenance tasks so that equipment can be repaired in a timely manner.

`tasks.`**`create_task`**(*auth_token: str*, *task:* models.Task)

Creates a new task.`task` may not have valid *status* or *task_id*.

Endpoint	/api/tasks
Request Type	POST
Access	MAINTAINER

Parameters

- **`auth_token`** (*`str, required`*) – Token to verify user credentials.
- **`task`** (`models.Task`, *`required`*) – The task to create.

Returns

- *Success*
- **code** (*int*) – Return Code
- **task_id** (*str*) – Task_id of newly created task.

`tasks.`**`get_tasks`**(*auth_token: str*)

Gets all tasks.

Endpoint	/api/tasks
Request Type	GET
Access	MAINTAINER

Parameters **`auth_token`** (*`str, required`*) – Token to verify user credentials.

Returns

- *Success*
- **code** (*int*) – Return Code
- **tasks** (*[models.Task]*) – List of tasks.

`tasks.`**`resolve_task`**(*auth_token: str*, *task_id: str*)

Resolves or completes a task.

Notes

This does **not** delete the task from the database. It only marks it as *resolved*.

Endpoint	/api/tasks
Request Type	DELETE
Access	MAINTAINER

Parameters

- **`auth_token`** (*`str, required`*) – Token to verify user credentials.

- **`task_id`** (*`str, required`*) – The id of the task to be deleted.

Returns

- *Success*
- **code** (*int*) – Return Code
- **message** (*str*)

`tasks.`**`update_task`**(*auth_token: str*, *task: models.Task*)

Updates a task.

Endpoint	/api/tasks
Request Type	PATCH
Access	MAINTAINER

Parameters

- **`auth_token`** (*`str, required`*) – Token to verify user credentials.
- **`task`** (*`models.Task, required`*) – The task to be updated.

Returns

- *Success*
- **code** (*int*) – Return Code
- **message** (*str*)

3.4 Machines

The MakerSpace team can quickly get an overview of the status of all MakerSpace equipment.

`machines.`**`delete_machine`**(*auth_token: str*, *machine_name: str*)

Deletes a machine and any associated tasks.

Endpoint	/api/machines
Request Type	DELETE
Access	MAINTAINER

Parameters

- **`auth_token`** (*`str, required`*) – Token to verify user credentials.
- **`machine_name`** (*`str, required`*) – Name of the machine to be deleted.

Returns

- *Success*
- **code** (*int*) – Return Code
- **message** (*str*)

`machines.`**`get_machines_status`**(*auth_token: str*, *start_time: int*, *end_time: int*)

Gets the status for all machines within the given timeframe. Dates are expressed in epoch time.

Notes

If accessed with a public *auth_token*, returns only non-sensitive machine information.

Deprecated since version 5.9.1: Key is now *statuses* instead of *machines*.

Endpoint	/api/machines
Request Type	POST
Access	ANY, MANAGER

Parameters

- **`auth_token`** (*`str, required`*) – Token to verify user credentials.
- **`start_time`** (*`int, required`*) – The start date (inclusive) of the time frame.
- **`end_time`** (*`int, optional`*) – The end date (inclusive) of the time frame. If not specified, all requests until *now* are fetched.

Returns

- *Success*
- **code** (*int*) – Return Code
- **statuses** (*{ str : [(int, int), …], … }*) – Dictionary mapping the names of each machine to list of tuples indicating the start and end times of when the machine was not working. Tuples should be chronological. (see *app.py* for an example response).

3.5 Visitors

The MakerSpace is visited by students and faculty from all walks of life. The API enables the staff to see who exactly is visiting the MakerSpace, if they are a new member, and when.

`visitors.`**`create_visitor`**(*auth_token: str*, *visitor:* models.Visitor, *hardware_id: str*)

Creates a new visitor.

Endpoint	/api/visitors
Request Type	PUT
Access	ALL

Parameters

- **`auth_token`** (*`str, required`*) – Token to verify user credentials.
- **`visitor`** (`Visitor`) – Visitor information.
- **`hardware_id`** (*`str, required`*) – Unique hardware id of the user’s TigerCard.

Returns

- *Success*
- **code** (*str*) – Return code.
- **message** (*str*)

`visitors.`**`get_visitors`**(*auth_token: str*, *start_date: int*, *end_date: int*)

Gets all MakerSpace visitors within a given timeframe.

Endpoint	/api/visitors
Request Type	POST
Access	MANAGER

Parameters

- **`auth_token`** (`str, required`) – Token to verify user credentials.
- **`start_date`** (`str, required`) – The start date (inclusive) of the time frame.
- **`end_date`** (`str, optional`) – The end date (inclusive) of the time frame. If not specified, "today" is assumed.

Note: Endpoint returns type *visits*, not *visitors*.

Returns

- *Success*
- **code** (*int*) – Return Code
- **visitors** (*[model.Visit]*)

3.6 Models

These are the various objects, or "models" that the api uses. Each class should be represented as a javascript object, with the parameters as keys.

`class` `models.`**`Error`**(*code: int*, *error: str*, *message: str*)

General error class. *code* should be from 400 - 405. *error* should be a short snake case description of the error. Message should be a short message that can be displayed to the end user.

`class` `models.`**`Machine`**(*machine_name: str*, *machine_state: int*)

Represents a machine. *machine_state* can be either "1" (Working) or "O" (Not Working).

Depending on the implementation, this amy not be needed.

`class` `models.`**`Permission`**(*policy_id: str, policy_name: str, access: [<class 'models.Resource'>]*)

Represents rules regarding resources. Each user is associated with zero or more permissions, which specify access control for one or more resources. If a user does not have a certain permission, access is denied.

`class` `models.`**`Request`**(*requester_name: str*, *description: str*, *request_id: str*)

Represents a maintenance request to the MakerSpace. *request_id* is not passed in when creating requests.

`class` `models.`**`Resource`**(*resource_id: str*, *resource_name: str*, *can_read: bool*, *can_delete: bool*, *can_write: bool*)

Represents a resource. Currently valid resource names are "tasks", "visitors", "machines", and "administrative".

`class` `models.`**`Task`**(*task_id: str, task_name: str, description: str, assigned_to: str, date_created: int, date_resolved: int, tags: [<class 'str'>], status: int*)

Represents a task object. *status* must be either be 2 (Completed), 1 (In-Progress), or 0 (Not Started). *task_name* is a brief description of the task. *assigned_to* is the user_id of the user to who the task is assigned to. *tags* are a list of strings associated with the task (e.g. metadata). The first tag is the machine name or '*' if the task is not associated with any particular machine.

Note: Tags are not case-sensitive.

class models.**User**(*user_id: str, first_name: str, last_name: str, assigned_tasks: [<class 'str'>], permissions: [<class 'models.Permission'>]*)

Represents a registered user of the MakerSpace. Users are those who will be using the dashboard (not visitors). *assigned_tasks* is a list of task_ids.

class models.**Visit**(*visit_id: str*, *visitor_id: str*, *first_visit: bool*, *date_visited: int*, *sign_in_time: int*, *sign_out_time: int*)

Represents a visit to the MakerSpace, and whether it is the first visit. *date_visited* is equal to *sign_in_time* and may be deprecated in the future.

class models.**Visitor**(*visitor_id: str*, *first_name: str*, *last_name: str*, *major: str*, *degree: str*)

Represents a visitor to the MakerSpace.

3.7 Example Server

3.7.1 Example Data

Users

```
users: [

  {
    user_id: "1",
    first_name: "Joe",
    last_name: "Goldberg",
    assigned_tasks: [ "2", "1" ],
    permissions: [ "3" ]
  },
  {
    user_id: "2",
    first_name: "Guinevere",
    last_name: "Beck",
    assigned_tasks: [ "3" ],
    permissions: [ "3" ]
  },
  {
    user_id: "3",
    first_name: "Peach",
    last_name: "Salinger",
    assigned_tasks: [ ],
    permissions: [ "1" ]
  }

]
```

Tasks

```
tasks: [
  {
    task_id: "2",
    task_name: 'Replace 3D-Printer Cartridge',
    tags: ["3D Printer" , "Priority 1"],
    date_created: 5034590345,
    date_resolved: 4503403534,
    description: "N/A",
    assigned_to: '1',
    status: 2,
  },
  {
    task_id: "1",
    task_name: 'Install new CNC-Router',
    tags: ["CNC-Router"],
    date_created: 5034590345,
    date_resolved: 4503403534,
    description: "N/A",
    assigned_to: '2',
    status: 1,
  },
  {
    task_id: "3",
    task_name: 'Update the Sign-in System',
    tags: ["*", "Priority 2"],
    date_created: 5034590345,
    date_resolved: 4503403534,
    description: "N/A",
    assigned_to: '3',
    status: 0,
  }
]
```

Visitors

```
{
  "visits": [
    {
      "visit_id": "1",
      "visitor_id": "2",
      "date_visited": 234923432,
      "first_visit": 1
    },
    {
      "visit_id": "2",
      "visitor_id": "2",
      "date_visited": 2342343,
      "first_visit": 0
    }
  ]
}


# todo visitors -> visits
```

Visitors

```
visitors: [
  {
    visitor_id: "1",
    major: "Computer Science",
    degree: "Masters",
    first_name: "Love",
    last_name: "Quinn"
  },
  {
    visitor_id: "2",
    major: "Chemistry",
    degree: "Phd",
    first_name: "Delilah",
    last_name: "Alves"

  },
  {
    visitor_id: "3",
    major: "Literature",
    degree: "B.S",
    first_name: "Candace",
    last_name: "Stone"
  },
  {
    visitor_id: "3",
    major: "Chemistry",
    degree: "B.A",
    first_name: "Delilah",
    last_name: "Alves"
  }

]
```

Permissions

```
permissions: [
  {
    policy_id: "1",
    policy_name: "Maintainer_Full_Access",
    access: [
      { resource_id: "11"
          resource_name: "tasks"
          can_read: True
          can_delete: True
          can_write: True },
      { resource_id: "13"
          resource_name: "visitors"
          can_read: True
          can_delete: True
          can_write: True
      },
      { resource_id: "14"
          resource_name: "machines"
          can_read: True
```

(continues on next page)

(continued from previous page)

```
          can_delete: True
          can_write: True
      },
      { resource_id: "12"
          resource_name: "administrative"
          can_read: True
          can_delete: True
          can_write: True }
    ]
  },
  {
    policy_id: "2",
    policy_name: "MakerSpace_Visitor",
    access: [
      { resource_id: "14"
          resource_name: "machines"
          can_read: True
          can_delete: False
          can_write: False
      }]
  },
  {
    policy_id: "3",
    policy_name: "Maintainer_Regular",
    access: [
      { resource_id: "11"
          resource_name: "tasks"
          can_read: True
          can_delete: True
          can_write: True },
      { resource_id: "13"
          resource_name: "visitors"
          can_read: True
          can_delete: True
          can_write: True
      },
      { resource_id: "14"
          resource_name: "machines"
          can_read: True
          can_delete: True
          can_write: True
      }
    ]
  }


]
```

3.7.2 Example Responses

getMachinesStatus()

```
{
  "machines": {
    "Screwdriver":[
      [
          645098691,
          337039019
      ],
      [
          816336861,
          735436609
      ],
      [
          167945226,
          791069814
      ],
      [
          167142686,
          970382312
      ],
      [
          1054543838,
          863355688
      ],
      [
          864623274,
          1073589720
      ]
    ],
    "Hammer":[
      [
          661219839,
          328338763
      ],
      [
          221386429,
          233623103
      ],
      [
          813464246,
          476337567
      ],
      [
          931925865,
          810296037
      ],
      [
          473900771,
          346032672
      ],
      [
          516839184,
          1015728462
      ],
```

(continues on next page)

(continued from previous page)

```
      [
          342347102,
          628262298
      ],
      [
          578178746,
          568130312
      ]
    ],
    "Chain Saw":[
      [
          759384625,
          679329741
      ],
      [
          743820996,
          228323020
      ],
      [
          692340287,
          797733798
      ],
      [
          173341347,
          185458089
      ],
      [
          833450873,
          967374851
      ],
      [
          161012215,
          402735103
      ],
      [
          1013551265,
          752190793
      ],
      [
          33995523,
          185592580
      ]
    ],
    "Rocket Fuel":[
      [
          703479396,
          428707785
      ],
      [
          223904213,
          793377268
      ],
      [
          230433762,
          1008333624
      ],
      [
```

(continues on next page)

(continued from previous page)

```
            39204281,
            422309968
        ],
        [
            923247367,
            812794625
        ],
        [
            843156104,
            847226907
        ]
      ]
    }
}
```

3.7.3 Implementation

```
"""
app.py

Simple example server for the MakerSpace
website. The purpose is two-fold: to provide
mock data for the site and to test if requests
are valid.

Note
-----
For development, the default test server is
'http://localhost:5000'. This can be configured
in Angular's `environment` settings.

"""
import os
import sys

import dotenv
from base64 import b64encode
from flask import Flask, session, request
from flask_cors import CORS

from utils import fetch_data
from utils import fetch_response

# to enable importing of distant modules
sys.path.append("../../api")

# load environment vars
dotenv.load_dotenv()

# flask configuration
app = Flask(__name__)
CORS(app)
app.config['SECRET_KEY'] = b64encode(os.urandom(16)).decode('utf-8')
auth_token = app.config['SECRET_KEY']
```

(continues on next page)

(continued from previous page)

```
# load users before start
users = fetch_data('users')


"""
Admin
"""



@app.route('/api/admin', methods=['POST'])
def generate_user_token():
    # implemented
    return dict(code=200, user_token=app.config['SECRET_KEY'])



@app.route('/api/admin', methods=['PATCH'])
def reset_password():
    return dict(code=200, message="Password reset email sent.")

"""
Users
"""

@app.route('/api/users', methods=['POST'])
def change_password():
    if 'new_password' in request.form:
        return login()
    return dict(code=200, message="Password changed successfully.")


@app.route('/api/users', methods=['PUT'])
def create_user():
    return dict(code=200, user=users[0], auth_token=auth_token)


@app.route('/api/users', methods=['DELETE'])
def delete_user():
    return dict(code=200, message="User has been successfully deleted.")


@app.route('/api/users', methods=['GET'])
def get_users():
    return dict(code=200, users=users)


@app.route('/api/users', methods=['POST'])
def login():
    return dict(code=200, user=users[0], auth_token=auth_token)


@app.route('/api/users', methods=['PATCH'])
def update_user():
    return dict(code=200, message="User has been updated.")
```

(continues on next page)

(continued from previous page)

```python
"""
Tasks
"""

@app.route('/api/tasks', methods=['POST'])
def create_task():
    return dict(code=200, task_id="RANDOM_TASK_ID")


@app.route('/api/tasks', methods=['GET'])
def get_tasks():
    return dict(code=200, tasks=fetch_data('tasks'))


@app.route('/api/tasks', methods=['DELETE'])
def resolve_task():
    return dict(code=200, message="Resolved task successfully.")


@app.route('/api/tasks', methods=['UPDATE'])
def update_task():
    return dict(code=200, message="Task updated successfully.")


"""
Machines
"""

@app.route('/api/machines', methods=['DELETE'])
def delete_machine():
    return dict(code=200, message="Machine deleted successfully.")


@app.route('/api/machines', methods=['POST'])
def get_machines_status():
    return dict(code=200, machines=fetch_response('machines'))


"""
Visitors
"""

@app.route('/api/visitors', methods=['PUT'])
def create_visitor():
    return dict(code=200, message="Visitor successfully created.")


@app.route('/api/visitors', methods=['POST'])
def get_visitors():
    return dict(code=200, visitors=fetch_data('visits'))
```

PYTHON MODULE INDEX

INDEX