Durham Foodbank

Release 1.0.0

Group 12

Mar 30, 2020

CONTENTS

CHAPTER 1

INTRODUCTION

To get started with this code, first clone the git repository at https://github.com/samrobbins85/software-engineering-12 using

```
git clone https://github.com/samrobbins85/software-engineering-12.git
```

Then move into the directory created

Then run

```
npm install
```

And run

```
npm start
```

to start the server

1.1 System Structure

The whole system composes of three primary components: the MongoDB Database, the Express Server (which I will also refer to as the back-end server) and any number of clients running the front-end program.

The MongoDB database is where the data describing the contents and structure of the warehouse will be stored so that it can be accessed and edited by warehouse employees. We chose to use MongoDB as it is both easily extensible and flexible. Details on the exact structure of the database can be found in the Installation section.

The express server acts as a middle-man between the clients and the MongoDB and provides a number of useful API functions to interface with the database. It provides error handling between the client and server, and between the server and the database, ensuring that the whole system remains robust and to ensure consistency. Details on how to install and deploy the express server are also in the Installation section.

1.2 Back-end Functions

In this section I will give a brief overview of all back-end functions in `routes/stockTake.js`

1.2.1 addTray

When provided with a tray object and a MongoDB database object, this function will add the contents of the tray object to the database.

1.2.2 editTray

When provided with a tray object and a MongoDB database object, this function will update the contents of an existing tray object in the database, provided it exists.

1.2.3 removeTray

When provided with the location of a tray object and a MongoDB database object, this function will delete the tray from the database.

1.2.4 switchTray

When provided with the location of two different tray objects and a MongoDB database object, this function with swap the positions of two trays in the database.

1.2.5 addTrayMany

When provided with an array of tray objects and a MongoDB database object, this function will add all the tray objects to the database in a single command. This should be used if many trays are added at once, as adding them one-by-one will take an unacceptable amount of time.

1.2.6 editTrayMany

When provided with an array of tray objects and a MongoDB database object, this function will edit all the tray objects that are in the database in a single command. This should be used if many trays are edited at once, as editing them one-by-one will take an unacceptable amount of time.

1.2.7 removeTrayMany

When provided with an array of tray locations and a MongoDB database object, this function will remove all the trays at the positions specified. This should be used if many trays are deleted at once, as deleting them one-by-one will take an unacceptable amount of time.

1.2.8 getAllCategory

When provided with a category and a MongoDB database object, this function will return all trays with a matching category. This can be used to quickly obtain the locations of items of a specific type.

1.2.9 getNextNExpiring

When provided with a number N, an optional category argument and a MongoDB database object, this function will return the next N expiring trays. If a category is specified, it will return the next N expiring in that category only. This can be useful if we wish to find the items closest to expiring so that we can focus on taking them from the warehouse first.

1.2.10 getZones

This function, when provided with a MongoDB database object, simply returns all zones in the warehouse.

1.2.11 addZone

When provided with a zone object and a MongoDB database object, the zone will be added to the database. This can be used in the initial creation of the warehouse and to add zones temporarily when there is a need to meet a higher than normal capacity.

1.2.12 editZone

When provided with a zone object and a MongoDB database object, the zone in the database will be edited, provided that it does exist.

1.2.13 removeZone

When provided with a zone location and a MongoDB database object, the zone will be removed from the database.

1.2.14 addBay

When provided with a bay object and a MongoDB database object, the bay will be added to the database.

1.2.15 editBay

When provided with a bay object and a MongoDB database object, the bay will be edited, provided it does exist.

1.2.16 removeBay

When provided with the location of the bay and a MongoDB database object, the bay will be removed from the database.

1.2.17 getTraysInBay

When provided with the location of a bay and a MongoDB database object, the function will return a list of all tray objects inside that bay, provided the bay exists. This will be used to display bay contents in the front-end application.

1.2.18 getBaysInZone

When provided with the location of a zone and a MongoDB database object, the function will return a list of all the bay objects inside that zone, provided the zone exists.

1.2.19 moveTray

When provided with two tray locations and a MongoDB database object, the function will move the tray from one location to another provided a tray exists in the start location and does not exist in the end position.

1.2.20 mongoUpdate

This function takes a request body and a method code. It will first connect to the MongoDB database to get the database object. Then, using the method code, it will pass the request body to one of the functions described above. It is then responsible for handling errors that occur and returning the result to the front-end.

CHAPTER

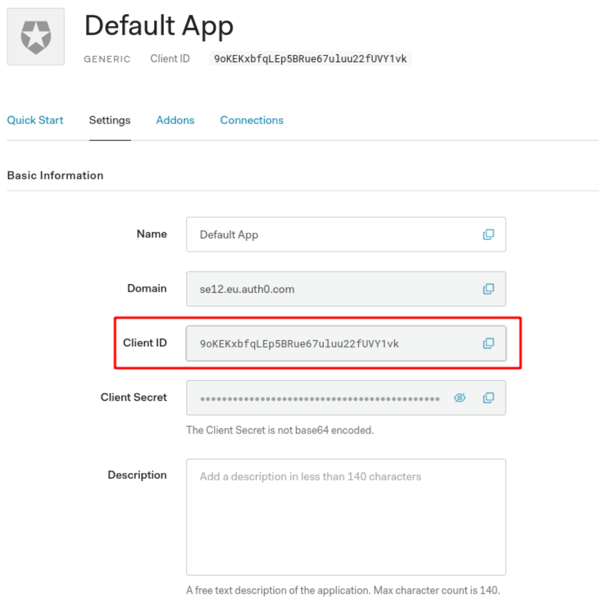
2 INSTALLATION

2.1 Authentication

Authentication for the website is handled through Auth0 the free tier allows for up to 7000 users, which should be more than ample.

The process of transferring the account to one that you own is very simple:

1. Sign up for Auth0 here
2. Under the applications tab on the dashboard create a new Generic application
3. Copy the client ID from the application, as pictured below



Client ID 9oKEKxbfqLEp5BRue67uluu22fUVY1vk

Basic Information

Name	Default App
Domain	se12.eu.auth0.com
Client ID	9oKEKxbfqLEp5BRue67uluu22fUVY1vk
Client Secret	••
	The Client Secret is not base64 encoded.
Description	Add a description in less than 140 characters
	A free text description of the application. Max character count is 140.

4. Paste this value under `clientID` in `auth_config.json`, which is located in the `src` folder of the `frontend` server
5. Also change the domain in the `auth_config.json` file, this domain can also be found in the application, as shown below
6. Disable signups by going to the connections tab of the dashboard, clicking on `Username-Password-Authentication`, then turning on the disable signups option as shown below

This is done as otherwise any user could come and sign up for the system without permission

2.2 Backend

In our handover there is a folder named `backend`, in this is stored the server that manages the requests made in the website. It is built using Node.js and Express and can be started with `npm install` followed by `npm start`, the server will then be hosted on port **3000**

When both the front end have been started, requests can be made between them ## Deployment Our recommendation for deploying this is on heroku as it has a good free tier and has suitable pricing plans for expanding if usage increases.

It is also simple to set up:

1. Create an account here
2. Click the new button on your dashboard to create a new application
3. Give it a name and set the region to Europe
4. Follow the on screen instructions to get your code deployed

2.3 Database

For this application to run it also needs a database set up, for this we have chosen MongoDB.

2.3.1 Deployment

Our recommendation for deployment is to use MongoDB Atlas this has a free tier but allows for scaling up to a paid plan if your requirements exceed the limits of the free tier.

To get started sign up here then choose any of the cloud providers, and select a local cluster with a free tier

Once this is done, a particular structure of the database will need to be established if you don't want to change the code

1. Create a database named `foodbank`
2. Create 4 collections, `bays`, `dummy`, `food` and `zones` in that database

If you do want to change the names of the database or collections, please consult the expansion section of this user guide.

2.4 Frontend

In our handover there is a folder named `frontend`, in this is stored the user interface with the program. It is a react app, and can be started with the command `npm install` followed by `npm start`, you will then be taken to a browser window with the application loaded. ## Deployment The code can be deployed using any static hosting, our recommendation is ZEIT Now as they have a very generous free tier and it is very easy to get set up. There are two main ways of getting set up

2.4.1 Git Repository

If you don't know what this is, then proceed to the other option.

For continuous development, this is a better option as any changes you make will be automatically deployed to the website.

This can be done in 3 steps

1. Add the files to a GitHub, GitLab or Bitbucket repository
2. Create an account here
3. Link together the repository and ZEIT by clicking `Import Project`

2.4.2 Command line

1. Install npm using the instructions here
2. Create a ZEIT account here
3. Install the Now CLI using the instructions here
4. The code can then be deployed with the command `now`, as documented here