CS 486C - Team Badgers
User Manual

Northern Arizona University

April 26, 2021

Sponsor: Dave Hopfensperger, Hanz Yeazel, and Glenn Austin

Team Mentor: Sambashiva Kethireddy

Members:
Abdulrahman Alamoudi
Tristan Marcus
Logan Ritter
Robel Tegegne
Yuanbo Xu

Overview:

The purpose of this document is to demonstrate how to install, operate and maintain our product. In this document we will provide instructions for installing the product, configuration it, and performing daily operations. We will also provide instructions for regular maintenance as well as troubleshooting some potential errors.

Table of Contents

1. Introduction

We are pleased that you have chosen Team Badgers to create an internal social media for your virtual employees!. The Skills/Knowledge Badging System is a powerful system that provides a way for employees working virtually to be recognized and rewarded for their work achievements that has been custom-designed to meet your needs. Some of the key highlights include:

- Interesting and responsive GUI

- Secure login / User Authentication

- Badge delivery system

- Badge printing

- Internal social currency - Kudos

    - Kudos sending

    - Kudos Spending

- Email signature integration (Via custom generator)

In order to operate our product an administrator would need to have a decent understanding in the following technologies:

- ReactJS

- MongoDB

- NodeJS (Express)

- Git/GitHub

The purpose of this user manual is to help you, the client, successfully install, administer, and maintain the Skills/Knowledge Badging System in your actual business context going forward. Our aim is to make sure that you are able to profit from our product for many years to come!

2. Installation

Note: The installation instructions assume you have the required technologies installed. That means having 'npm' and 'Node.JS' installed in order to run the commands for building the project. To install these tools visit and follow the instructions: https://www.npmjs.com/get-npm

Once given access to the private repository, the source code can be found here:

https://github.com/RobelDev/CS476-C-Badging-System.git

Upon obtaining the source file (CS476-C-Badging-System.zip) through git or simply downloading the source code from github's website, extract the file to your desired directory to begin installation.

Once extracted, open a terminal and navigate to the "CS476-C-Badging-System-main" directory. First install the dependencies for the back-end by typing 'npm install'. Upon successful installation of node modules, your console should display a similar message to the following:

Now install the front-end dependencies by navigating into the 'badge-system-frontend' directory and type 'npm install' once again.

To start the server navigate back into the "CS476-C-Badging-System-main" directory and type the following command 'npm run badgers'.

The server is now running and alongside this a new tab in your browser should open with the application. Installation is now complete.
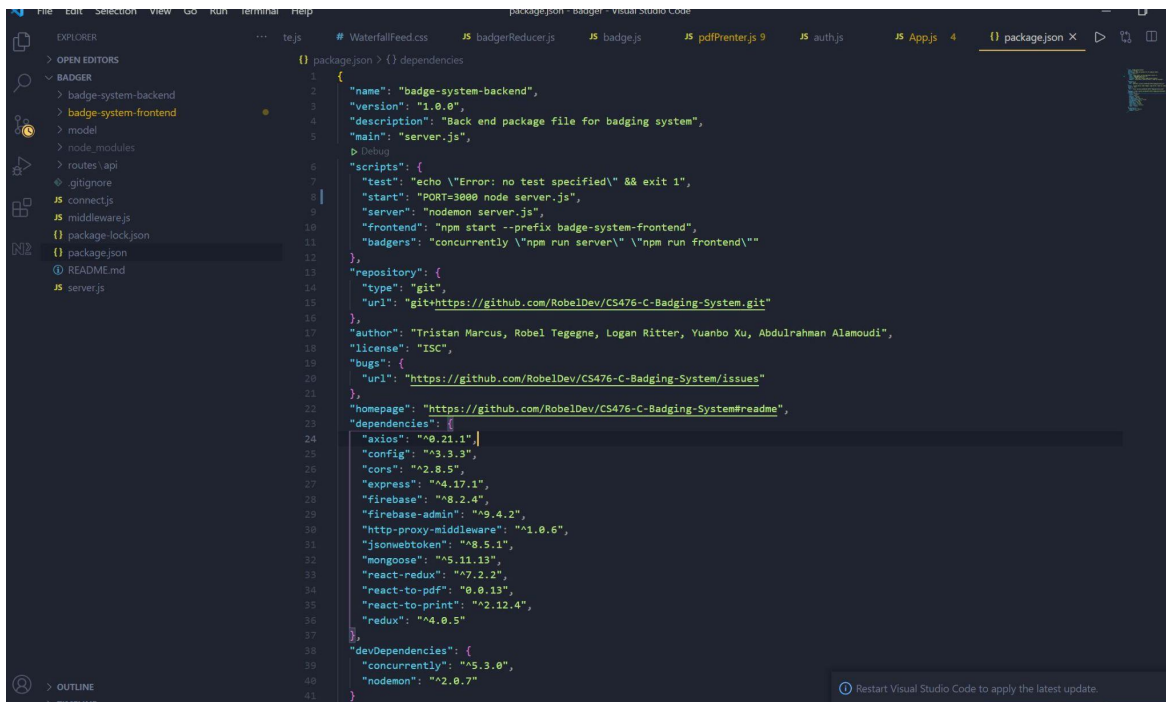
To change the host and port the application is running on, see the Configuration section.

3. Configuration and Daily Operations

After the installation process, users can run the server using the "npm run badgers" command on the root folder, however the host and port aren't configured. By default the port runs on 3000 and the IP is localhost.

To change the host and port the application is running on, users must navigate to the package.json file found in both the root folder "CS476-C-Badging-System-main" and the "badge-system-frontend" to make those changes. Open the file and edit the scripts "start" value changing port number to the desired port. If that port is already in use, React will increment the port by 1 until it finds a port that is open.

Upon successfully running the web application, the account authentication page will appear.

To gain access to the application, the user must first click the 'Sign Up' option.

Then type in a pre-existing email, password (a minimum of 8 characters), and password verification. Next click REGISTER to successfully create an account. Click Login to return back to the sign in form.

Proceed to type in the email and password combination from the account you had just registered.

Upon successfully logging in, you will be redirected to the homepage with access to the application and all of its features. Users now have access to all colleague badge transactions, personalized badge library and kudos bank.

Sending a Badge - to send a badge to a colleague or employee click the Send a Badge navigation button and fill out the required information such as the receiver email, a reason, an appropriate badge image and then click the Send Badge button.

Send Kudos - to give kudos to a colleague or employee click the Send Kudos navigation button and fill out the required information such as the receiver email, a reason, and the amount of kudos then click Send Kudos.
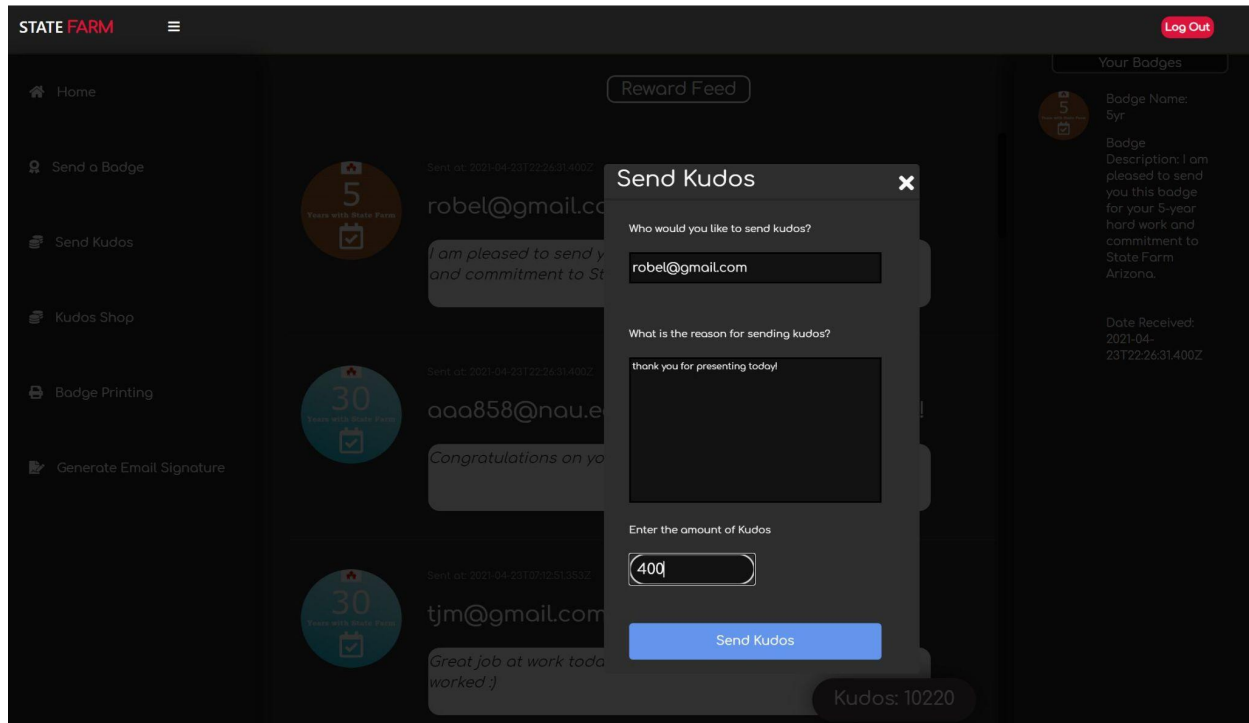
Keep in mind that users cannot send kudos to themselves nor can they send more kudos than their balance currently displays. This prevents negative balances and exploiting the "economy".

Kudos Shop - to spend kudos click the Kudos Shop navigation button and choose which product from the list to buy alongside the amount.
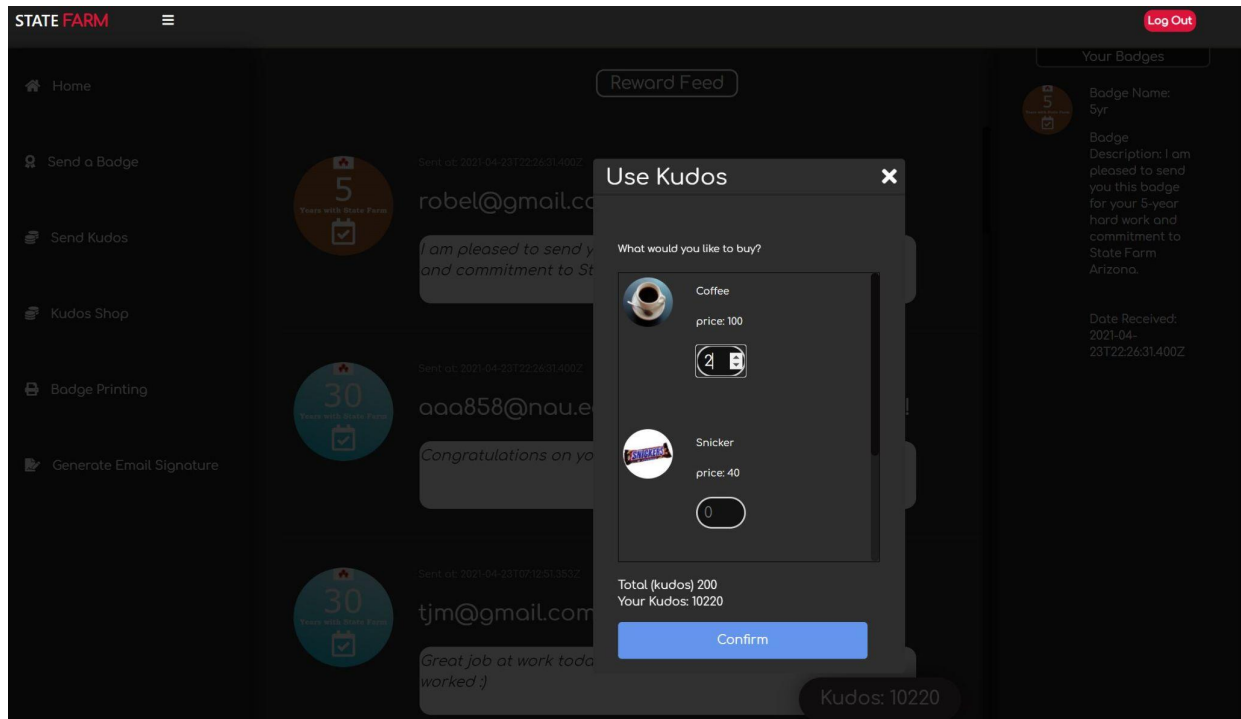
Keep in mind that users cannot send kudos to themselves nor can they send more kudos than their balance currently displays. This prevents negative balances and exploiting the "economy".

Print a Badge - to print a badge certificate, click download under the respective badge certificate. This will result in a PDF download to your desired directory on your computer. After opening the PDF users can navigate to the upper right hand corner and click the "print" icon to officially print their desired badge certificate.
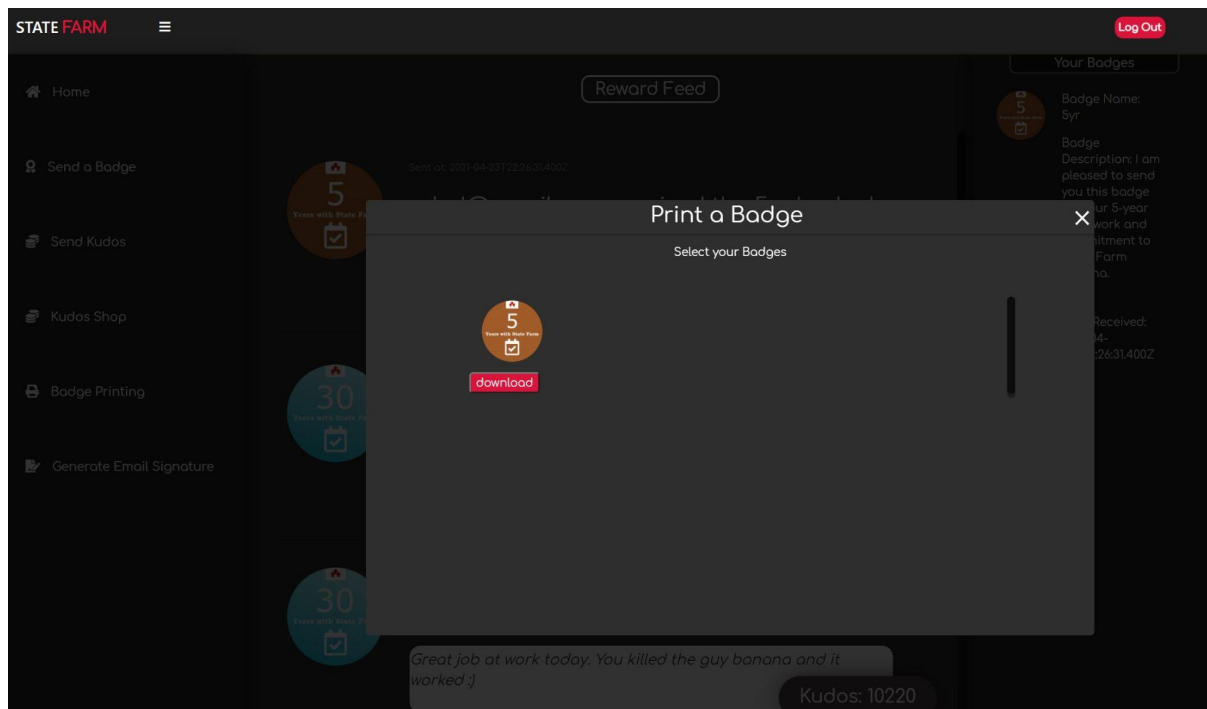
Email Signature - to generate an email signature first click the Email Signature navigation button and choose up to three badges and click next.
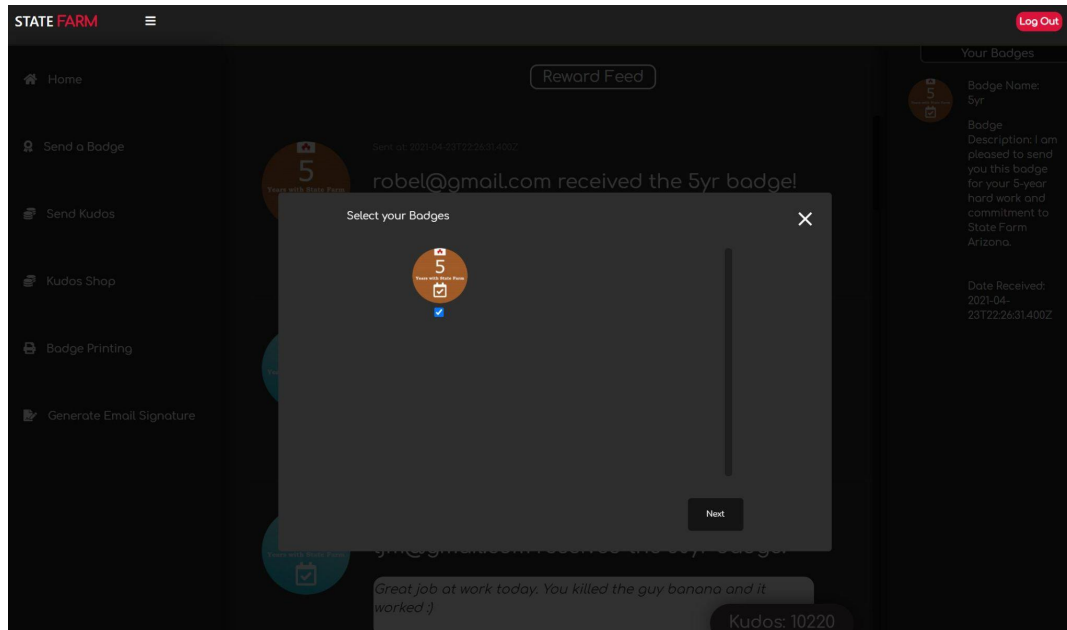
After that, fill out the displayed information such as the name, personal information, phone number, email, address. Users also have the option to choose a font style. Each one of these inputs are optional, making it entirely possible to just display the badges in your library.
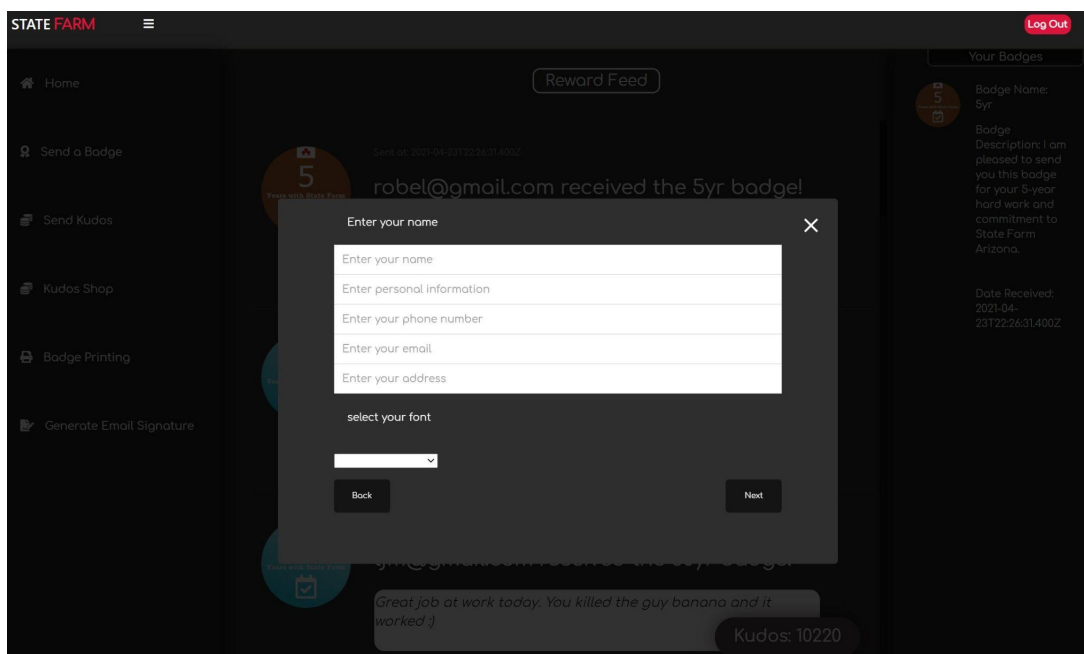
The email signature that is generated will be in PNG format. PNG images are lossless and do not lose quality and detail after image compression. Also PNG's have great transparency. To integrate this into your selected email service first navigate to your email settings and locate your signature. Once you are there copy the generated email signature to the bottom of your signature and save.

4. Maintenance

Team Badger's Skills and Knowledge Badging System was designed to be a single page application with minimal overhead and simple architecture allowing maintenance to be seamless.

Updating the system and its dependencies:

1. Navigate to the main directory and delete package-lock.json file.

```
.git
badge-system-frontend
model
routes
.gitignore
connect.js
middleware.js
package.json
package-lock.json
README.md
server.js
```
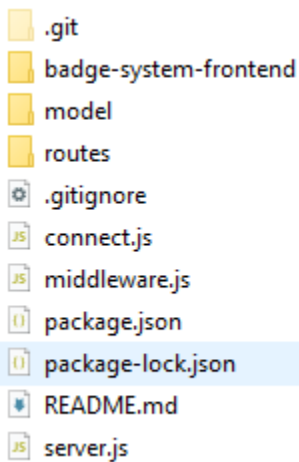
2. Open a terminal and navigate to the main directory.

```
15202@DESKTOP-NS89PA5 MINGW64 ~/Desktop/use mainu (master)
$ cd CS476-C-Badging-System/

15202@DESKTOP-NS89PA5 MINGW64 ~/Desktop/use mainu/CS476-C-Badging-System (main)
$ |
```

3. Type 'npm install' to update the back-end dependencies

```
15202@DESKTOP-NS89PA5 MINGW64 ~/Desktop/use mainu/CS476-C-Badging-System (main)
$ npm install|
```

4. Navigate to the 'badge-system-frontend' directory in your terminal and repeat these steps starting with deleting the package-lock.json file and typing 'npm install'. This will update the front-end dependencies.

5. Troubleshooting

When using the application, issues may arise that need troubleshooting. This section will go over possible problems and their respective solutions to combat any issues.

Problem

After entering the run command, the project reports errors or does not run as expected.

Solution

1. Check the package.json file under the top folder, look through "dependencies". Make sure you download the correct version of node modules.

```
.git
badge-system-frontend
model
node_modules
routes
.gitignore
connect
middleware
package
package-lock
README
server
```
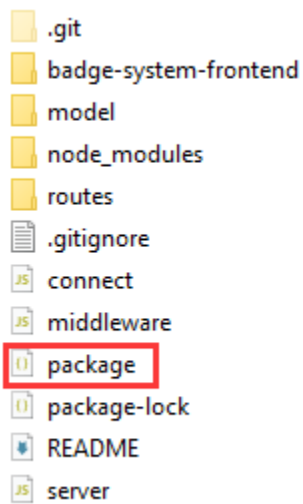
You can add versions after the name of the module to get a specific version

An example terminal command for express version 4.17.1:

```
$ npm install express@4.17.1.
```

```
"scripts": {
  "test": "echo \"Error: no test specified\" && exit 1",
  "start": "node server.js",
  "server": "nodemon server.js",
  "frontend": "npm start --prefix badge-system-frontend",
  "badgers": "concurrently \"npm run server\" \"npm run frontend\""
},
"repository": {
  "type": "git",          Robel Tegegne, 2 months ago • moved backend file out to badgers main file for
  "url": "git+https://github.com/RobelDev/CS476-C-Badging-System.git"
},
"author": "Tristan Marcus, Robel Tegegne, Logan Ritter, Yuanbo Xu, Abdulrahman Alamoudi",
"license": "ISC",
"bugs": {
  "url": "https://github.com/RobelDev/CS476-C-Badging-System/issues"
},
"homepage": "https://github.com/RobelDev/CS476-C-Badging-System#readme",
"dependencies": {
  "axios": "^0.21.1",
  "config": "^3.3.3",
  "cors": "^2.8.5",
  "express": "^4.17.1",
  "http-proxy-middleware": "^1.0.6",
  "jsonwebtoken": "^8.5.1",
  "mongoose": "^5.11.13",
},
"devDependencies": {
  "concurrently": "^5.3.0",
  "nodemon": "^2.0.7"
}
```

2. Check the package.json file under the frontend folder, look through "dependencies". Make sure you download the correct version of react plug-in.
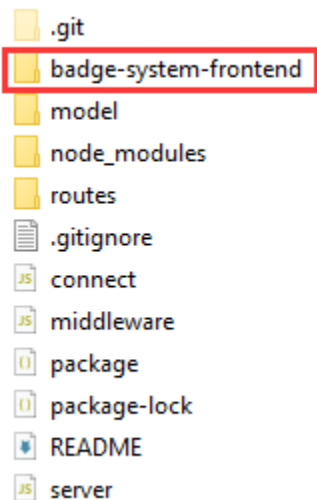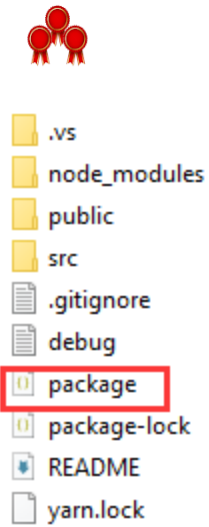
- .git
- badge-system-frontend
- model
- node_modules
- routes
- .gitignore
- connect
- middleware
- package
- package-lock
- README
- server

.vs
node_modules
public
src
.gitignore
debug
package
package-lock
README
yarn.lock

An example terminal command for express version 4.17.1:

$ npm install express@4.17.1.

```json
{
  "name": "badge-system-frontend",
  "version": "0.1.0",
  "private": true,
  "dependencies": {
    "@testing-library/jest-dom": "^5.11.6",
    "@testing-library/react": "^11.1.2",
    "@testing-library/user-event": "^12.2.2",
    "axios": "^0.21.1",
    "bootstrap": "^4.5.3",
    "eslint-plugin-react-hooks": "0.0.0-8e5adfbd7",
    "firebase": "^8.0.2",
    "jspdf": "^2.3.1",
    "postcss": "^7.0.35",
    "postcss-safe-parser": "^5.0.2",
    "react": "^17.0.2",
    "react-bootstrap": "^1.4.0",
    "react-dom": "^17.0.2",
    "react-firebase-hooks": "^2.2.0",
    "react-icons": "^4.2.0",
    "react-redux": "^7.2.2",
    "react-scripts": "^4.0.3",
    "react-spring": "^8.0.27",
    "react-to-pdf": "0.0.13",
    "reactjs-popup": "^2.0.4",
```

3. Open the setupProxy javascript file under the "src" folder, check if the domain name and port number are what you are using currently. Replace the content in the 'target' with yours.

```
Assets
components
context
App
App
index
setupProxy
```
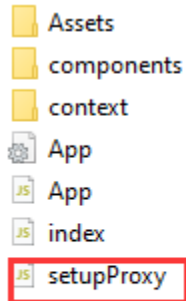
```javascript
const proxy = require('http-proxy-middleware')

module.exports = function (app) {
    app.use(proxy('/api',
        {
            target: 'http://localhost:8000',      You, 2 months ago • update
            changeOrigin: true,
            pathRewrite: {
                "^/api": ''
            }
        }))
}
```

Problem

Incomplete modal displays

Solution

1. Check that the window is fully scaled. If it is not, maximize the browser window.

2. Check your browser, our application may not display properly on some low-end browsers. You may have to use a high-end browser such as Chrome, Firefox, or Safari. The application was

mainly tested on Chrome and Firefox but other browsers such as Opera should not provide other issues as this development stack is supported by most mainstream browsers.

3. If you use a high-end browser and the window is fully scaled, the problem may be caused by the resolution of your monitor. Please check your device's display resolution. Try to use 1920 × 1080 resolution.

Problem

Unable to sign in.

Solution

1. Check if your password or account is wrong, try multiple times as our application does not limit how many times you try to sign in.

Problem

The data is not updated in real time or data is being unresponsive

Solution

1. Check if the network connection is stable and or disconnected. Make sure that the application is always connected to the database.

2. Check if you're giving the wrong badge or points to the wrong person. For example, if you try to give someone a badge that they already have.

3. Check the network requests and make sure that the server is not being overloaded.

4. Upgrade server hardware based on active users and server load.

6. Conclusion

Team Badgers is honored to have been able to work on the Skills/Knowledge Badging System as our capstone project. We hope our solution provides years of service to State Farm employees and allows them to feel recognized and rewarded for the work they do in the virtual workspace. As our team moves on to our professional careers we would be happy to answer any questions about our product. With that being said feel free to contact any of our team members.

Logan Ritter (Team Lead) - lkr57@nau.edu

Tristan Marcus (Development Architect) - tjm429@nau.edu

Robel Tegegne (Release Manager) - rk394@nau.edu

Yuanbo Xu (Coder) - yx59@nau.edu

Abdulrahman Alamoudi (Coder) - aaa858@nau.edu