CS 467
Winter 2019

Employe Portal: Mid Point Check

Ain Team Members:
Mel Drews: drewsm@oregonstate.edu
Ryan Wholey: wholeyr@oregonstate.edu
Vincent “Connor” Kelly: kellyvi@oregonstate.edu

Submission Contents

Included in the submission are:

- Mid Point Check document (pdf)
- Employee-awards-portal (code repo)
- Current deployment to a staging server at http://159.89.141.7:4000/login

The URL for the staging server above is the same as provided on the Canvas submission page.

Project Status

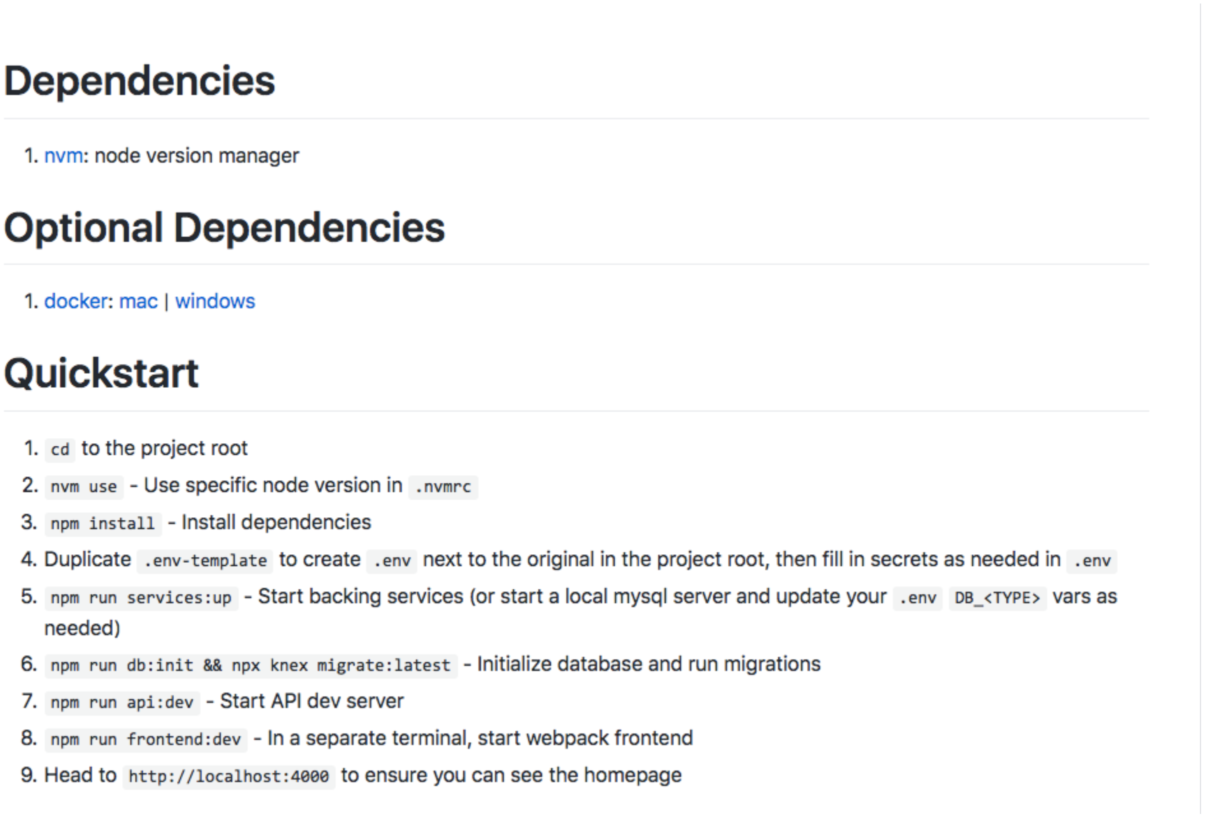
Our team has been making good progress on our project thus far. We have a basic front-end completed that handles user registration, login, password reset, and award creation / deletion. We have created a basic SQL database schema to handle user data, and we chose to utilize the Knex.js query builder to help streamline our db queries. Overall, our team has implemented the majority of the “base” user functionality of the site based on the project specifications. Additionally, a good deal of work has gone into “behind the scenes” functionality like continuous integration / testing / database integration that is not apparent to the end user but allows for our dev team to work together more efficiently. This will help serve us as the project development cycle progresses into testing / integration / deployment.

Moving forward, our team will focus on some of the other project specifications. We will build out the “admin” user frontend and corresponding backend code. Also, we will work on the certificate creation (via PDF) and how to build the appropriate LaTeX templating. Finally, we will incorporate the business intelligence reporting requirements for admin users. This additional frontend functionality will be enabled as more of the supporting elements of the server architecture are fleshed out.

User Instructions

All code and instructions can be found in the accompanying zip archive and also at the following Github link:

https://github.com/ryanwholey/employee-award-portal



Dependencies

1. nvm: node version manager

Optional Dependencies

1. docker: mac | windows

Quickstart

1. `cd` to the project root
2. `nvm use` - Use specific node version in `.nvmrc`
3. `npm install` - Install dependencies
4. Duplicate `.env-template` to create `.env` next to the original in the project root, then fill in secrets as needed in `.env`
5. `npm run services:up` - Start backing services (or start a local mysql server and update your `.env` `DB_<TYPE>` vars as needed)
6. `npm run db:init && npx knex migrate:latest` - Initialize database and run migrations
7. `npm run api:dev` - Start API dev server
8. `npm run frontend:dev` - In a separate terminal, start webpack frontend
9. Head to `http://localhost:4000` to ensure you can see the homepage

<u>Container Deployment</u>

In addition to the instructions for a development build contained in the project README file, functionality currently exists to package the application for containerized deployment. Dockerfile is provided in the code archive and git repository. The configuration has been tested successfully and is currently being used to support deployments to the staging server noted above.

Containerized deployment has also been tested and provides minimal functionality for deployment to a Google Compute Engine (GCE) instance. Our Project Plan specified the final deployment would be to Google App Engine and this is still anticipated, though a kubernetes deployment is another good option. A GCE deployment currently exists and is being tested for connectivity to a standalone CloudSQL instance over a Virtual Private Cloud network, to better approximate a real-world deployment scenario. Once this configuration is validated, we will have more of the pieces in place to move toward the target deployment model.

Current Frontend (User) Functionality

Create Award:

- Currently, when the application starts up you are directed to the Create Award page.
- Here, you are able to create a new employee award by entering the following data:
  - Award Type (drop down)
  - Recipient Name
  - Recipient Email
  - Message
- Once created, a new award will be visible below the award creation form.
- You are also able to remove any award created by using the “Remove” button.

Login:

- From the Create Award page, you should see a small button at the top left hand side of the screen called “Logout”. Clicking this button takes the user to our /login route.
- On this page, the user will find a simple login form to gain access to the webapp.
- There are also two other buttons on the page, title “New User” and “Forgot Password”.

New User:

- Clicking on the New User button will take the user to an account registration form.
- This form validates the user information and creates a new user entity in our backend.

Forgot Password:

- Clicking on the Forgot Password button takes the user to a new page that requests an email from the user to use for password reset/validation.

Current Server Functionality

As described in our original Project Planning document, the server presents a RESTful API. Not all server routes documented here comply fully with the principles of REST. This is due in part to the nature of the application and the need to support additional API functionality. Further modifications are anticipated as we work through later sprints to bring API calls into alignment with RESTful principles when feasible.

As we work toward the full functionality described in that planning document, currently implemented functionality includes:

POST /api/awards - Accepts json parameters in request body:

type - The type of award
creator - id number of a user in the users table
recipient - id number of a user in the users table
granted - a date in the format mm/dd/yyyy

The associated function is partially implemented.

GET /api/awards - Current status is that the route exists, but does not yet provide functionality

POST /login_tokens - Properly validates a user authentication attempt using a JSON Web Token (JWT).

POST /forgot_password - Partially functional to provide a basis for further development.

POST /reset_password - Partially functional to provide a basis for further development.

GET /award_types/:id - Partially functional to provide a basis for further development. The intended functionality is to return information about a specific award type.

GET /award_types - Partially functional to provide a basis for further development. The intended functionality is to return a list from a database search.

POST /award_types - Partially functional to provide a basis for further development. The intended functionality is to provide a RESTful way to create a new type of award.

PATCH /award_types/:id - Partially functional to provide a basis for further development. The intended functionality is to allow modification of an existing award type as specified by id.

DELETE /award_types/:id - Partially functional to provide a basis for further development. The intended functionality is to permit deletion of an existing award type as specified by id.

GET /regions/:id - Partially functional to provide a basis for further development. The intended functionality is to return information about a specific region.

GET /regions - Partially functional to provide a basis for further development. The intended functionality is to return a list of regions from a database query.

POST /regions - Partially functional to provide a basis for further development. The intended functionality is to provide a RESTful way to create a new region.

DELETE /regions/:id - Partially functional to provide a basis for further development. The intended functionality is to permit deletion of an existing region as specified by id.

GET /system/check - When complete, will return information about the system to an authorized user.

GET /system/dbconn - When complete, will return information about the application database to an authorized user.

POST /system/post - When complete, will permit an authorized user to modify system parameters.

GET /system/data - When complete will permit an authorized user to retrieve reporting data.

GET /users/:id - Partially functional to provide a basis for further development. The intended functionality is to return information about a specific user.

POST /users - Partially functional to provide a basis for further development. The intended functionality is to provide a RESTful way to add a new user, who can then become a recipient or creator of an award, or be granted administrator privileges..

GET /login - When complete, will interface with an end-user browser to present a login interface when appropriate.