**FullStack JavaScript – Sprint 1 – Documentation**

Elliott Butt, Joseph Flores, Zachary Hulan

June 24, 2024

The team is working on a Full Stack JavaScript project to build a simple CLI for managing a web application. This CLI, created with Node.js, will help confirm new user accounts by setting up the app's directory, adding configuration files, and generating confirmation tokens. It will also allow users to view and change app settings easily.

**DIRECTORY STRUCTURE**

A *help* folder contains all necessary application text files. *Logs* will contain all log files for initialization, configuration and token requests. *Config.json* and *tokens.json* are saved in *json* folder. *Server.js* and other *js* files are saved under *scripts* folder and *ejs* files under *views* folder.

**PROJECT DELIVERABLES BASED ON USER STORIES PROVIDED**

**As a System Administrator:**

The required directory structure was created. Initialization using `app init --all` or `--mk` will display the current settings and status of directories and files created. Current configuration settings are available using `app config --show`. System administrators can update the configuration using `app config --set`. Additionally, configuration attributes can be added using `app config --add`.

**As a Helpdesk Employee:**

A user token can be generated based on an end user’s username, which is the same token presented to the user via the end-user self-service form. Adding and updating records can be done, with the ability to change both the email address and phone number. There is a search option where the helpdesk employee can check for a record by providing the username, email, or phone number.

**As a New End User:**

The user interface will have tabs for creating a new token, viewing the existing number of tokens, updating the email address, and updating the phone number. The user will be able to enter the username, and a token will be generated to confirm membership. The expiry date will automatically be set to three days after token creation. If the user chooses to edit the email or phone number, the system will automatically check if the token is expired and will assign a new token if it is.

**PROJECT DELIVERABLES BASED ON RUBRIC PROVIDED**

**Build a Command Line Interface (CLI) using Node.**

A publicly available GitHub repository has been created, and the URL is provided. The code structure is easy to understand. A well-designed CLI solution has been submitted. The CLI includes a help option that displays various commands and options. Help is also available for initialization, configuration, and token inquiries.

**Read and write JSON and text files using the Node fs common core module(s).**

The CLI checks if files are available and creates them if requested. Help is presented either when called or when an incorrect command is entered. The creation of a JSON configuration file is available, with options to either add or update attributes. Similarly, users can add or update records in the user.json file, specifically the email and phone.

**Generate a consistently unique token based upon the Cyclic Redundancy Check (CRC) to hash a value, which is used to confirm a new user.**

There is an option to add a new user, wherein the CLI will assign an expiry date to the token, which is three days later. Similarly, the CLI checks the token expiration when updating the records. If it is expired, the CLI will assign a new token and prompt the user with a message indicating that the current token has expired and a new expiration date will be assigned to the token.

**Call the same token generation module from both the command line and a simple HTML page hosted by a Node HTTP server.**

Requesting a new token from either the command line or the HTML page uses the same token generation module and logs information on both the HTML page and terminal. Other features include viewing the number of existing tokens, updating the email, and updating the phone number. Each of these uses the same module.

**Log all the application actions to an events log file saved to disk.**

The CLI generates two log files (one for configuration events and one for initialization events) for each day where the events are recorded and available for review. The format for the config log events is:

YYYYMMDD HH:MM:SS <dir or file> <success or failure> <log message> <system gen code>

Similarly, the format for initialization log events is:

YYYYMMDD HH:MM:SS <set or reset> <success or failure> <log message> <system gen code>

**The features of GitHub were used to manage the sprint project until DONE.**

GitHub organization was set up, and story cards reflecting all issues (user stories) are organized. Branching, merging, and pull requests are present and used. Members have at least one story card assigned and completed, and all features requested in the sprint documents are present.