Part 1.

**Semester Three Full Stack JavaScript + Algorithms & Data Structures Sprint One**

**Project Description:**

Many full-stack applications also require a Command Line Interface (CLI) to aid in configuration and system administration. This Full Stack JavaScript project is to build a simple CLI for a web application that needs to confirm new user accounts. The project will use node.js to create the web applications directory (or folder) structure and to add the configuration and text files for the new application. The CLI will provide commands to view and alter the configuration of the application and to create tokens to be used to confirm a new user, which will have their data saved into a JSON file.

Next for the project will be functionality for searching through the JSON file to locate information from these tokens. To keep things organized this should be a separate app with its own commands for search queries based upon username, email, or phone. Your search queries should search for the JSON data using a doubly linked list data structure. Your search result should be able to return one or multiple items depending on criteria/matching. (i.e. searching for a portion of a name/number may return multiple items that have a match)

**Learning Outcomes:**

1. Proven ability to build a Command Line Interface (CLI) using Node.
2. Proven ability to organize node.js code into modules and to deploy a small full stack project.
3. Proven ability to confirm and create a directory (folder) structure and organize files into the structure.
4. Proven ability to read and write json and text files using the node fs common core module(s).
5. Proven ability to alter and update attributes of a json configuration file.
6. Proven ability to generate a consistently unique token based upon the Cyclic Redundancy Check (CRC) to hash a value which is used to confirm every new user.
7. Proven ability to store new user authentications to a json file.
8. Proven ability to work with data structures (doubly linked list)
9. Proven ability to call the same token generation module from both the command line and a simple html page hosted by a node http server.
10. Proven ability to log all the application actions to an events log file saved to disk.

**Project features as user stories:**

Roles:

* **End users** are hands-on customers who work directly with various software products and tools to deliver on their personal or business goals.
* **Helpdesk employees** are the go-to people for providing technical assistance and support related to computer systems, hardware, and software. They are responsible for answering queries and addressing system and end user issues in a timely and professional manner.
* **System administrators** are responsible for managing, troubleshooting, licensing, and updating hardware and software assets.

Page Break

Stories:

* As a system administrator I would like a command line interface (CLI) to initialize the application to build the required directory structure and add the default configuration and help files.
* As a system administrator I would like the CLI to provide a status for the initialization and configuration.
* As a system administrator I would like the CLI to provide a view of the current configuration file(s).
* As a system administrator I would like the CLI to provide the ability to update the configuration file with new values.
* As a system administrator I would like the CLI to provide the ability to reset the configuration file back to its original state.
* As a helpdesk employee I would like the CLI to provide the ability to generate a user token based on an end users username. This token would be the same as presented to the user via the end user self service web form.
* As a helpdesk employee I would like the CLI to provide the ability to search through tokens with specific search terms for user queries/issues. (i.e. username, email, phone)
* As a new end user I would like a web form that allows me to enter my username and after pressing submit receive a token I would use to confirm my new membership.
* (Note: this token should be sent to the new user via email or SMS. (At this time Keyin does **not** have either an outbound email or SMS server available for this development testing. We will stay with using the web form)

**Project Deliverables:**

A single zip file that contains the following;

1. All of the node.js files used to implement the project.
2. All other text or json files implemented by the project.
3. All dependent .json files that reference the project dependencies (ie. package.json)
4. As a bonus upload your project to GitHub and make sure every team member is pushing commits. (This is less for the course but will be a great piece for your portfolio when you are out searching for jobs!)

Note: do not include the node\_modules folder in the zip file.

**Key Points for Project:**

Your JSON file can follow the following structure for receiving and storing all the information:

let newToken = JSON.parse(`{

    "created": "1969-01-31 12:30:00",

    "username": "username",

    "email": "[user@example.com](mailto:user@example.com)",

    "phone": "5556597890",

    "token": "token",

    "expires": "1969-02-03 12:30:00"

    "confirmed": "tbd"

}`);

Your separate CLIs should follow a command flow similar to this:

myapp <command> <option> 

Usage:

myapp init --all creates the folder structure and config file   
myapp init --mk creates the folder structure   
myapp init --cat creates the config file with default settings   
myapp config --show displays a list of the current config settings   
myapp config --reset resets the config file with default settings   
myapp config --set sets a specific config setting

tokenapp <command> <option>

Usage:

tokenapp --count displays a count of the tokens created   
tokenapp --new <username> generates a token for a given username, saves tokens to the json file   
tokenapp --add p <username> <phone>   
tokenapp --add e <username> <email>   
tokenapp --search u <username> fetches a token for a given username   
tokenapp --search e <email> fetches a token for a given email   
tokenapp --search p <phone> fetches a token for a given phone number

Part 2.

**Semester Three Database Programming Sprint One**

**Project Description:**

Having the skills and knowledge to derive a database design from a business scenario is essential for the full stack software developer. In this project each team will select a business scenario from a collection provided by the instructor to build their database. The project includes a fully attributed Entity Relationship Diagram (ERD) brought to third normal form. The database design should then be used to create a database in PostgreSQL containing all the identified entities with all primary and foreign keys defined and implemented. Once deployed to PostgreSQL the database should be loaded with test data. SQL queries with JOINs should be written to exercise the database design against use cases identified from the written business description. Once all the project outcomes have been completed the database should be backed up and included with the database ER diagram and all the SQL queries for submission.

**Learning Outcomes:**

1. Proven ability to design a relational database from a written business scenario.
2. Proven ability to identify all the business entities from the business scenario.
3. Proven ability to fully attribute all entities with the fields (columns) to fulfill the requirements found in the business scenario.
4. Proven ability to create an Entity Relationship Diagram (ERD) modeling all the entities, attributes, and their relationships.
5. Proven ability to use a standard symbology to identify one-to-one, one-to-many, and many-to-many relationships.
6. Proven ability to model the entities and attributes to third normal form.
7. Proven ability to implement the database in PostgreSQL using diagramming tools and SQL CREATE statements
8. Proven ability to load a database using mock data generation and written INSERT and UPDATE SQL statements.
9. Proven ability to backup and restore a database from one computer to another.
10. Proven ability to confirm their database design by exercising the data model with SQL queries.

**Project Scenarios:**

As a team, choose one of the written business scenarios from the collection provided in the companion zip file. Choose two scenarios, your first and second choice. Message your chosen scenarios by number and title to **Nicholas Hodder** as soon as you can with your team number. No two teams will be doing the same scenarios, so first come first served scenario selection to each team. Keep in mind these scenarios may have incomplete or incorrect identification of entities.

**Project features as user stories:**

Roles:

* **Database Administrators** (DBA) will keep the database up and running smoothly 24/7. They are responsible for design, implementation, and optimization of database systems. Their goal is to provide a seamless flow of information throughout the company, considering both backend infrastructure and frontend accessibility for end-users.
* **Business User** means a person who uses the software services and/or equipment in the course of any trade or business activity.

Stories:

* As a DBA I would like to have an ER diagram to assess the database design for data quality, referential integrity, infrastructure requirements, and database performance.
* As a DBA I would like to have a development instance of the database with sample data to assess the creation scripts, table relationships, and data usability.
* As a business user I would like to have pre-written SQL queries that provide the resulting data sets that fulfill my reporting and business analysis requirements.

**Project Deliverables:**

A single zip file that contains the following.

1. The original document describing the chosen business scenario
2. A fully attributed ERD with all PK and FK relationships defined
3. A database backup including all database tables and loaded mock data saved as a tar file
4. SQL Queries that provide proof of normalization for basic business reporting queries.