Career Tracker

Programmer Manual

William Ntumba, Scott Shrout, Anthony Freitas

Table of Contents

1. Vision Statement

In 2019, Team Blue took on the task of creating a data management solution for the school of Natural Sciences at IUS. Our goal was to develop a simple, intuitive program that could assist the department with the collection and handling of student alumni data. With the completion of Career Tracker, Team Blue believes it has accomplished its goal.

2. Introduction

Career Tracker is a web-based database application, designed specifically for the NATS department of IUS. To gain access to the Career Tracker application, all potential users will need to be provided with the Career Tracker login name and password. Once the application is running and authentication is confirmed, users will then be able to access the main page and view the alumni database. From here users will be able to:

- View student records (Name, Address, Phone number, Employments, & Graduate Schools)
- Edit all information in the student records
- Import student data from a provided graduation report
- Export student data into a report

This Programmer Manual will detail the components and tools used in the construction of this application, as well as provide step-by-step instructions for how to install a new copy of Career Tracker onto a local machine.

3. Component Overview

The application is comprised of the following components:

3.1 Authentication

The Authentication component validates credentials supplied by a user and establishes a session with the client in order to allow for consumption of application resources. This component utilizes Bcrypt hashing in order to compare the submitted password with the password hash that has been stored.

3.2 Records

3.2.1 Alumni

The Alumni component allows a user to search for, view, add, and maintain records related to alumni of the School of Natural Sciences. The records include contact information as well as data related to an alumnus' employer(s) and graduate schools(s) attended. It also allows a user to maintain notes related to an alumnus record.

3.2.2 Employers
The Employers component allows a user to search for, view, add, and maintain records related to an employer. The employers found in this component are available for selection as an employer of an alumnus within the Alumni component. The records include contact information as well as notes related to the employer.

3.2.3 Graduate Schools
The Graduate Schools component that allows a user to search for, view, add, and maintain records related to an educational institution that offers graduate programs. The schools found in this component are available for selection as a graduate school attended by an alumnus within the Alumni component. The records include contact information as well as notes related to the graduate school.

3.3 Imports/Exports
The Imports/Exports component allows a user to import data into the application using a previously agreed upon spreadsheet layout which consists of alumni data received from the Office of the Registrar. Spreadsheets submitted for import processing are parsed by a server-side library within the Node JS framework and data is loaded accordingly into the backend database. In addition, this component allows a user to request and receive output from the database based upon specified criteria and returned in a spreadsheet format.

4. Tool Overview

The following tools/languages were used in the construction of Career Tracker:

Angular – an app-design and development framework used for creating the front end of the application.

Node JS – a JavaScript runtime environment used for running the back end of the application.

MySQL – a relational database management system used for storing the alumni, employer, graduate school, and login data.

JavaScript – a scripting language used for developing web applications.

TypeScript – a superset of the JavaScript language that is used in the creation of Angular components.

Visual Studio Code – a source-code editor used for the development and debugging of the application.

5. Project Repository

The following contains a list of all files relevant to the application and its development. Files can be located by clicking the hyperlink associated with the Name column.

5.1 Software

Name	Description
Career Tracker – Distribution	A distribution containing all files required for running the application aside from those related to pre-requisite applications available in .zip and .tar.gz formats
Career Tracker – Source Files	All source files for the application

5.2 Test Cases

Name	Description
Test Plan	A document outlining the expected testing activites
Testing Results	A document detailing the results of functional testing
Automated Test Cases	A collection of automated test cases executed using the Karma test runner for Angular
Test Plan Report	A document detailing the overall testing effort and associated results

5.3 Documentation

Name	Description
Programmer Manual	The current document
User Manual	A document detailing usage of the application for an end user

5.4 Test Platform

The following unit testing applications were used when developing this application:

Karma – a test runner intended for testing Javascript-based applications within a web browser. This is the primary unit testing application used with Angular. Unit tests for Angular are composed using Typescript and executed within a shell; results are displayed in a web browser.

Mocha – a framework used for testing Javascript within the Node JS framework. Unit tests are composed using Javascript and executed and results displayed within a shell.

5.5 Unit Testing Scripts

Name	Description
Alumni – API	A Mocha unit test script for the alumni API comprised of a "dummy" implementation of the API using most of the same code in order to evaluate the major execution paths.
Alumni - Validator	A Karma unit test script comprised of unit tests associated with the alumni data validator functionality.
Employers - API	A Mocha unit test script for the alumni API comprised of a "dummy" implementation of the API using most of the same code in order to evaluate the major execution paths.
Employers - Validator	A Karma unit test script comprised of unit tests associated with the alumni data validator functionality.
Graduate Schools - API	A Mocha unit test script for the alumni API comprised of a "dummy" implementation of the API using most of the same code in order to evaluate the major execution paths.
Graduate Schools - Validator	A Karma unit test script comprised of unit tests associated with the alumni data validator functionality.

6. New Installation (No Existing Data) Instructions

The following instructions are intended for installation of a new application instance in which there is no existing application data.

6.1 Pre-requisites

In order to install this application, the following applications must be installed on the web server:

Application	Recommended Version	Installation Support Links
Node JS	12.16.1 or later	https://nodejs.org/en/download/
MySQL	8.0.19 or later	https://dev.mysql.com/downloads/mysql/ https://dev.mysql.com/doc/refman/8.0/en/installing.html

Note: Required Node libraries, including Express, Express-Session, BcryptJS, and MySQL (connector) are included in this application's distribution.

6.2 Installing Career Tracker

1.) Download the application's distribution (contained within a .zip file or .tar.gz file) from the following link:

https://github.com/ntkwilliam/CareerTracker/tree/master/src/distribution

2.) Extract all files from the .zip or .tar.gz file into the directory where the application will be hosted. For details on how to extract a .zip or .tar.gz file, reference the applicable link:
Uncompressing Zip Files (Windows)
Extracting .tar.gz Files (Linux)

3.) Within the distribution, locate the /database-setup folder and the new-installation.sql file. This file can be used within Mysql Workbench or via the Mysql command line to create the database and the related schema. For more details about executing a .sql script in MySQL, reference the following links:
 https://dev.mysql.com/doc/refman/8.0/en/mysql-batch-commands.html
https://dev.mysql.com/doc/workbench/en/wb-admin-export-import-management.html
(section "Data Import/Restore" ; note that the "Import from self-contained file" option should be used)
Note: A user must have CREATE DATABASE privileges within MySQL to successfully execute this script.

4.) Create a user database user with the following object rights for the database: SELECT, INSERT, UPDATE, DELETE.

This can typically be done by executing the following commands within MySQL, substituting desired values as needed:

```
CREATE USER '[username]' IDENTIFIED BY 'test123' PASSWORD EXPIRE
NEVER;
GRANT SELECT,DELETE,UPDATE,INSERT ON [database name].* TO
'[username]'
```

5.) Within the root application directory, open the server.js file and locate the following line:


```
const APPLICATION_PORT = 8080;
```

Update the port number shown as necessary to the port on which the application should listen for HTTP requests.

Unless the application is hosted on a dedicated server, it is generally necessary to re-direct requests received on the standard HTTP port (80) to the application port configured above. Configuration related to port forwarding is beyond the scope of this document.

6.) Also within the server.js file, locate the following lines:


```
const DB_HOSTNAME = 'localhost';
const DB_PORT = 3306;
const DB_DATABASE_NAME = 'careertracker'
const DB_USERNAME = 'careertracker'
const DB_PASSWORD = …
```

Update these values as necessary to correspond with the applicable hostname, port, database name, username, and password values.

7.) Launch the application by using the following command by executing one of the following command within the application's root directory:
Linux:

```
nohup node server.js >>application.log 2>&1
```

Windows:

```
node server.js >>application.log 2>&1
```

8.) Access the application by navigating via a web browser to the hostname and port corresponding with the application.

Optional:
It is generally a good practice to routinely validate that the Node JS Express application is still up and running as it is possible that a fatal error could occasionally disrupt the application. This can be automated by implementing a cron job that executes at a particular interval to check if the process is running and restart it as needed. This functionality is also present within some process monitoring applications. The implementation and configuration of cron scripts or other related applications is beyond the scope of this document.

7. New Installation (Existing Data) Instructions
The following instructions are intended for installation of a new application instance in which there is existing data that must be transferred.

7.1 Pre-requisites

In order to install this application, the following applications must be installed on the web server:

Application	Recommended Version	Installation Support Links
Node JS	12.16.1 or later	https://nodejs.org/en/download/
MySQL	8.0.19 or later	https://dev.mysql.com/downloads/mysql/ https://dev.mysql.com/doc/refman/8.0/en/installing.html

Note: Required Node libraries, including Express, Express-Session, BcryptJS, and MySQL (connector) are included in this application's distribution.

7.2 Exporting Existing Data from MySQL

If the existing MySQL instance is operating on a **Windows** Server:

1.) Using the command prompt or Powershell, navigate to the MySQL Server bin folder. This is typically located in the following path:

C:\Program Files\MySQL\MySQL Server [version]\bin

2.) Execute the following command, substituting the desired backup output location, hostname, database name, and user name:

```
mysqldump -e -u [username] -p -h [hostname] [database name] > [file
location]
```

Please note that the default database name for this application is "careertracker" and that the file location should end with a .sql file.

Example:

```
mysqldump -e -u jsmith -p -h localhost careertracker >
C:\Users\jsmith\backup\careertracker.sql
```

Note: The user must have SELECT permissions for all objects within the database in order to use this function.

If the existing MySQL instance is operating on a **Linux** Server:

In the shell, execute the following command, substituting the desired backup output location, hostname, database name, and user name:

```
mysqldump -e -u [username] -p -h [hostname] [database name] > [file
location]
```

Please note that the default database name for this application is "careertracker" and that the file location should end with a .sql file.

Example:

```
mysqldump -e -u jsmith -p -h localhost careertracker >
careertracker.sql
```

Note: The user must have SELECT permissions for all objects within the database in order to use this function.

6.2 Installing Career Tracker

1.) Download the application's distribution (contained within a .zip file or .tar.gz file) from the following link:

https://github.com/ntkwilliam/CareerTracker/tree/master/src/distribution

2.) Extract all files from the .zip or .tar.gz file into the directory where the application will be hosted. For details on how to extract a .zip or .tar.gz file, reference the applicable link:
Uncompressing Zip Files (Windows)
Extracting .tar.gz Files (Linux)

3.) Use the existing .sql file created in the prior sessions to import existing data into the new MySQL instance. For more details about executing a .sql script in MySQL, reference the following links:
 https://dev.mysql.com/doc/refman/8.0/en/mysql-batch-commands.html
https://dev.mysql.com/doc/workbench/en/wb-admin-export-import-management.html
(section "Data Import/Restore" ; note that the "Import from self-contained file" option should be used)
Note: A user must have CREATE DATABASE and CREATE USER privileges within MySQL to successfully execute this script.

4.) Create a user database user with the following object rights for the database: SELECT, INSERT, UPDATE, DELETE.

This can typically be done by executing the following commands within MySQL, substituting desired values as needed:

```
CREATE USER '[username]' IDENTIFIED BY 'test123' PASSWORD EXPIRE
NEVER;
GRANT SELECT,DELETE,UPDATE,INSERT ON [database name].* TO
'[username]'
```

5.) Within the root application directory, open the server.js file and locate the following line:

```
const APPLICATION_PORT = 8080;
```

Update the port number shown as necessary to the port on which the application should listen for HTTP requests.

Unless the application is hosted on a dedicated server, it is generally necessary to re-direct requests received on the standard HTTP port (80) to the application port configured above. Configuration related to port forwarding is beyond the scope of this document.

6.) Also within the server.js file, locate the following lines:

```
const DB_HOSTNAME = 'localhost';
const DB_PORT = 3306;
const DB_DATABASE_NAME = 'careertracker'
const DB_USERNAME = 'careertracker'
const DB_PASSWORD = …
```

Update these values as necessary to correspond with the applicable hostname, port, database name, username, and password values.

7.) Launch the application by using the following command by executing one of the following command within the application's root directory:
Linux:

```
nohup node server.js >>application.log 2>&1
```

Windows:

```
node server.js >>application.log 2>&1
```

8.) Access the application by navigating via a web browser to the hostname and port corresponding with the application.

Optional:
It is generally a good practice to routinely validate that the Node JS Express application is still up and running as it is possible that a fatal error could occasionally disrupt the application. This can be automated by implementing a cron job that executes at a particular interval to check if the process is running and restart it as needed. This functionality is also present within some process monitoring applications. The implementation and configuration of cron scripts or other related applications is beyond the scope of this document.

8. Potential Future Enhancements

The following are potential enhancements that may have been pursued if we had additional time in this project:

- Implementation of a process to automatically update employer data related to alumni if possible by utilizing web APIs or by data scraping. For example, the process could locate information on an alumnus' LinkedIn profile.
- Implementation of additional export/reporting options.
- Implementation of functionality that could potentially allow an alumnus to maintain their contact information via an online form or for the application to receive updates from an existing application that receives this data.

9. Key Personnel Information

William Ntumba – Development Team, Team Leader

Scott Shrout – Development Team

Anthony Freitas – Development Team

Dana Hope – Project Sponsor

10. Individual Contributions

Scott Shrout

3. Component Overview

5. Project Repository

6. Installation for New Install

7. Installation for New Platform

Anthony Freitas

1. Vision Statement

2. Introduction

4. Tool Overview