Department of Computer Science & Applied Physics
Atlantic Technological University
Higher Diplomain Science in Software Development

Professional Portfolio

Edivagner Ribeiro
Student Number: G00411275
email: G00411275@gmit.ie

In this document I summarizes several software development projects in my portfolio, each with a brief description. The projects include text classification, multi-threaded text indexing, web-based chat applications, JUnit testing, sorting algorithm benchmarking, a wine e-commerce website, soccer player and quote reader mobile application, a weather application, and a dental clinic database project. Each project involves a different set of technologies, programming languages, and tools.

This portfolio showcases my skills, achievements, and growth in these fields, highlighting my ability to apply innovative problem-solving techniques to complex challenges. This portfolio also demonstrates my capacity to work collaboratively and independently, think critically, and deliver exceptional results.

1. **Java**:
    1. Text Classification using N-Gram
    2. Text Indexing using Multi-threaded
    3. Web-based chat application
    4. JUnit test - InsuranceProgram
    5. Sorting Algorithms benchmark
2. **Web**:
    1. web project - invictus Wine
3. **Mobile-WebApp**:
    1. Soccer Player App- Ionic
    2. WeatherApp - Angular
4. **DBMS**:
    1. Database Project - Dental Clinic
5. **Bash**:
    1. Simple Calculator in Bash
6. **Automation and Data Analysis**:
    1. IBM data science (Coursera) - Python
    2. Data Analysis Model - equivalent-circuit for capacitive device - Matlab
    3. Automation system Hydrogen Energy studies - LabVIEW
    4. Automation system Hydrogen Degradation studies - LabVIEW

Repositories Summary

1.1. Text Classification using N-Gram

The nGram program has been designed and implemented by the developer to facilitate text analysis and generate n-grams. A simple command-line interface is utilized, where the user inputs the directory address, number of characters for the n-gram, and output file name. Two methods are offered for building the n-gram table: Linear-search and Hash Table. The openAddressingIndex method has been developed to handle collision in the hash table using the Linear Probing method. The program is built in the BuildTable class, which contains methods to add n-grams to the table, print the table on the screen, and calculate relative frequency using the RelativeFrequencies method.

1.2. Text Indexing using Multi-threaded

A menu-driven program is utilized to implement Multi-threaded Indexing. Three files, including a text file for parsing, a dictionary, and a stop word list are required to run the program. The correlation report can be generated after entering these files. Preview mode virtual threads are used for processing and parsing the text. The Runner Class is utilized to run the application, which starts the APIMenu Class. The menu displayed by the APIMenu Class has six options, including specifying the text file, configuring the dictionary, configuring common words, specifying the output file, executing the BuildWordIndex class to generate the report, and quitting the program.

1.3. Web-based chat application

The web-based chat application was implemented using the Java Socket API and a thread pool was utilized to handle multiple user connections. A specific socket was bound to a designated port number on the server, which waits and listens for client connection requests. Both the server and client interfaces featured a menu-driven, command-line user interface, allowing for the checking and configuration of open ports to establish server-client connections via socket. The Server class was implemented as Runnable, using a thread pool to start the server and handle multiple client connections. The ConnectionsHandler class facilitated the exchange of messages via socket between clients. Upon entering a valid port for the chat server, the user received a message to input their name and could then engage in chat with other clients. In the case of an invalid port, the user received a message that communication could not be established and the application closed.

1.4. JUnit test – InsuranceProgram

The JUnit automated tests that were developed for this project have the capability to test the Java application Old_InsuranceProgram.java, which calculates the car insurance premium based on age. The code has been refactored into well-defined classes to enable JUnit5 testing. A set of tests has been created using annotations such as @BeforeAll, @BeforeEach, @Test, @ParameterizedTest, @Timeout, @AfterAll, @AfterEach, and two different types of exceptions. The automated test suite set for the InsuranceProgram conforms to the convention with the respective folder containing the tests.

1.5. Sorting Algorithms benchmark

The aim of this project is to explore the significance of sorting algorithms in minimizing computational expenses associated with processing large amounts of data. The study introduces the concepts of time and space complexity and how they serve to distinguish and categorize diverse algorithms. The project conducts a comparative analysis of five sorting algorithms, namely Bubble Sort, Selection Sort, Quicksort, Counting Sort, and Radix Sort, using them as a case study. The study concludes that the selection of an algorithm always hinges on the specific application and the attributes of the dataset that requires sorting.

2.1. web project - invictus Wine

For this web project, a business e-commerce website was created to showcase the principles of HTML5, CSS, and JavaScript. The website, Invictus Wine, requires customers to enter their login details, which are then validated before receiving a summary of their order. The login feature was implemented using JavaScript. A slideshow carousel, which displays a different image each time the page is loaded, was included on the banner. An object array in JavaScript was used to display product images and information. In the original project, a database was connected using PHP/XAMPP. However, the functional webpage, which is deployed on Github, has replaced the database with a JSON file.

3.1. Soccer Player App – Ionic

A Soccer Player & Quote Reader web app was built in this project. The Ionic mobile application was used to write the app that reads soccer player data, flag data, and quotes from internet resources. The complete list of players available in the database is displayed by the app if no age group is selected. If only one of the values is presented, the search is returned according to the age limits of the search. Functions have been implemented in the settings page to check the list of country IDs and another function to check the search history. On the home page, the player's table can be sorted by first and last name. When a valid search is performed, however, an appropriate message is displayed on the screen instead of the table if no results are returned.

3.2. WeatherApp - Angular

This is an Angular full stack weather app with Angular CLI 14.2.7 and we use a public API from RapidAPI.COM to get live weather forecast values for the searched city. When we run a new search, we get the update for the current weather conditions and the background image 'is updated with the temperature variation with a javascript function.

4.1. Database Project - Dental Clinic

A dental practice database was designed and implemented by utilizing the entity-relationship model and SQL scripting. The business rules were analyzed and translated into a relational schema, and stored procedures were developed to ensure data integrity and automate business processes such as adding new patient and appointments, generating bills, and updating patient records. The

system enables patients to request appointments, and make payments. The database also supports referrals to specialists and the tracking of treatment histories for each patient.

5.1. Simple Calculator in Bash

In this project, a program was developed to generate math tables based on user input. A selection menu was presented to the user, allowing them to choose between generating math tables or quitting the program. The program displayed a table of math operators and prompted the user to enter a math operator symbol and an operand. A while loop was used to generate the math table based on the user's input. The code performed addition, subtraction, multiplication, division, or exponentiation on the operand and the numbers from 1 to 15. If the user entered an invalid operator symbol or operand, the program prompted the user to try again. When the user selected "Quit," the program ended and printed a closing message to the terminal.

6.1. IBM data science (Coursera) – Python

The Coursera program consisted of nine courses covering topics such as Python programming, data analysis, data visualization, and machine learning. A solid foundation in statistics, data manipulation, and data cleaning was gained through the coursework.

6.2. Data Analysis Model - equivalent-circuit for capacitive device – Matlab

A set of MATLAB functions was developed and utilized to study and develop an equivalent circuit model. The objective of the model was to obtain values of physical parameters of volumetric variation and porosity of a metal hydride sample when it is subjected to variations in hydrogen pressure.

6.3. Automation system Hydrogen Energy studies – LabVIEW

An automated control system was developed for the study of metal hydrides under varying hydrogen pressure conditions. The automation of the system was carried out using Labview and Python programming languages.

6.4. Automation system Hydrogen Degradation studies – LabVIEW

A control interface was developed for the automation of metal hydride aging tests with hydrogen. The first prototype was created using Arduino, and later on, dedicated electronics and a LabVIEW control interface were employed to produce the final and definitive version. In this version, all pressure, temperature, and isothermal control protocols were implemented.