|  |
| --- |
|  |
| Question 1  Project |
| |  |  |  | | --- | --- | --- | | Ben Royans | Date ###### | Programming III | |

TABLE OF CONTENTS

[Description 1](#_Toc39748159)

[Introduction 1](#_Toc39748160)

[Description 1](#_Toc39748161)

[Specifics 2](#_Toc39748162)

[Dynamic data structures 2](#_Toc39748163)

[Hashing Techniques 2](#_Toc39748164)

[Sorting Algorithm 2](#_Toc39748165)

[Search Technique(s) 2](#_Toc39748166)

[Third Party Libraries 2](#_Toc39748167)

[Json.NET 2](#_Toc39748168)

[Source/Version Control 3](#_Toc39748169)

[Coding Standards 3](#_Toc39748170)

[Testing 3](#_Toc39748171)

[GUI Mock-Up 4](#_Toc39748172)

# Description

## Introduction

RetroAchievements.org is a website dedicated to bringing the achievement functionality of current generation video games to older console games of yester-year.

Achievements didn’t become a common game implementation until around 2005 when Microsoft’s Xbox 360 introduced the Gamerscore system. Valve’s Steam and Sony’s Playstation 3 also adopted this mechanic in the following years and is now considered a staple feature of modern gaming. RetroAchievements.org aims to bring this functionality to consoles ranging back to the Atari and Commodore 64 era.

RetroAchievements.org provides an API from which a registered user can generate queries for various pieces of data.

## Description

For this project I propose to develop an application with a GUI interface to browse the data from the RetroAchievement servers from within a Windows environment, outside of a web-browser.

The application will have login in functionality, which implements saving the users progress data. Once the user is logged in, progress data can be displayed to the user.

User data includes information about progress of achievements in particular games, on particular consoles.

# Specifics

## Dynamic data structures

Doubly Linked Lists will be used in this project as the dynamic data structure for chaining similar elements together.

## Hashing Techniques

Hashing techniques will be used to store the credentials of users, such as their username and password. Passwords stored will also be salted first to increase data security.

## Sorting Algorithm

The Merge sort algorithm will handle the sorting of elements in this project. Merge sort has been selected for its efficiency and reliability.

The Merge sort algorithm differs from the Selection and Bubble sort algorithms as it does not repetitively iterate the elements incrementally makes changes to the list. Instead it breaks down the elements, and then re-merges then in a sorted fashion.

## Search Technique(s)

This project will employ the use of both binary Search and linear search (Sequential) algorithms in its methods. A binary search will be implemented for searching medium to large numbers of elements, and a linear search will be implemented for small numbers of elements.

## Third Party Libraries

### Json.NET

#### Description

Json.NET is a 3rd party library created by Newtonsoft as a framework for handling JSON files in .NET. It provides the following functionalities:

* Create, parse, query and modify JSONs.
* Object serialization/deserialization.

#### Documentation

The documentation and other resources for this library can be found at <https://www.newtonsoft.com/json>.

## Source/Version Control



Source and version control will be handled by a GitHub repository. GitHub provides all the necessary tools for managing source code for this project.

## Coding Standards

The coding standards will be followed as the documentation outlined on the Microsoft website. The standards can be found at the following URL:

<https://docs.microsoft.com/en-us/dotnet/csharp/programming-guide/inside-a-program/coding-conventions>

## Testing

Unit testing will be used on this project to ensure the validity of the methods developed in the project.

Use Case testing will also be carried out during the final phases to ensure the functionality are performing as intended.

## GUI Mock-Up

