CS39440 - GamePile

MAJOR PROJECT REPORT

Department of Computer Science,
Aberystwyth University

Last updated: **24th April, 2024**

v0.2 - DRAFT

Produced by:

Kal Sandbrook
kas143@aber.ac.uk
BSc in *Computer Science - G400 BSc*

Supervised by:

Dr. Edore Akpokodje
eta@aber.ac.uk
Lecturer in Computer Science

Declaration of originality

I confirm that:

- This submission is my own work, except where clearly indicated.
- I understand that there are severe penalties for Unacceptable Academic Practice, which can lead to loss of marks or even the withholding of a degree.
- I have read the regulations on Unacceptable Academic Practice from the University's Academic Registry (AR) and the relevant sections of the current Student Handbook of the Department of Computer Science.
- In submitting this work I understand and agree to abide by the University's regulations governing these issues.

Name: Kal Sandbrook
Date: 9th April 2024

Consent to share this work

By including my name below, I hereby agree to this project's report and technical work being made available to other students and academic staff of the Aberystwyth Computer Science Department.

Name: Kal Sandbrook
Date: 9th April 2024

Generative AI

No Generative AI tools have been used for this work.

Name: Kal Sandbrook
Date: 9th April 2024

Acknowledgements

TODO: Add Acknowledgements - Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magnam aliquam quaerat voluptatem. Ut enim aeque doleamus animo, cum corpore dolemus, fieri tamen permagna accessio potest, si aliquod aeternum et infinitum impendere malum nobis opinemur. Quod idem licet transferre in voluptatem, ut.

Abstract

TODO: Write the Abstract - Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magnam aliquam quaerat voluptatem. Ut enim aeque doleamus animo, cum corpore dolemus, fieri tamen permagna accessio potest, si aliquod aeternum et infinitum impendere malum nobis opinemur. Quod idem licet transferre in voluptatem, ut postea variari voluptas distinguique possit, augeri amplificarique non possit. At etiam Athenis, ut e patre audiebam facete et urbane Stoicos irridente, statua est in quo a nobis philosophia defensa et collaudata est, cum id, quod maxime placeat, facere possimus, omnis voluptas assumenda est, omnis dolor repellendus. Temporibus autem quibusdam et aut officiis debitis aut rerum necessitatibus saepe eveniet, ut et voluptates repudiandae sint et molestiae non recusandae. Itaque earum rerum defuturum, quas natura non depravata desiderat. Et quem ad me accedis, saluto: 'chaere,' inquam, 'Tite!' lictores, turma omnis chorusque: 'chaere, Tite!' hinc hostis mi Albucius, hinc inimicus. Sed iure Mucius. Ego autem mirari satis non queo unde hoc sit tam insolens domesticarum rerum fastidium. Non est omnino hic docendi locus; sed ita prorsus existimo, neque eum Torquatum, qui hoc primus cognomen invenerit, aut torquem illum hosti detraxisse, ut aliquam ex eo est consecutus? – Laudem et caritatem, quae sunt vitae sine metu degendae praesidia firmissima. – Filium morte multavit. – Si sine causa, nollem me ab eo delectari, quod ista Platonis, Aristoteli, Theophrasti orationis ornamenta neglexerit. Nam illud quidem physici, credere aliquid esse minimum, quod profecto numquam putavisset, si a Polyaeno, familiari suo, geometrica discere maluisset quam illum etiam ipsum dedocere. Sol Democrito magnus videtur, quippe homini erudito in geometriaque perfecto, huic pedalis fortasse; tantum enim esse omnino in nostris poetis aut inertissimae segnitiae est aut fastidii delicatissimi. Mihi quidem videtur, inermis ac nudus est. Tollit definitiones, nihil de dividendo ac partiendo docet, non quo.

Contents

1 - Background, Analysis & Process

1.1 - Background

1.1.1 - Aims

The aim of this project was to create a native desktop application that helps users to manage their backlogs (lists of games they want to play) and libraries (games they own). The application allows users to add games to the library and mark them as part of their backlog, in progress or completed. The application will also allow users to search through their games and filter them based on various attributes such as genre, platform or completion status.

Further to this, the application will also allow users to search for games to add to their library via the use of a Third-Party API. This will use a fuzzy search algorithm to allow users to search for games even if they are unsure of the exact name. The application will also allow users to view detailed information about the games in their library, such as the aforementioned attributes.

Some optional features that could be implemented include the ability to export a users game library graphically akin to an old-school forum signature, to facilitate sharing of game completion progress on forums and social media platforms and a recommendation system that suggests games to add to the users library based on their existing library and completion status.

1.1.2 - Research into Similar Tools

Initial research for this project involved investigation into similar tools that already exist. These tools were found by searching online for “game backlog manager” and “game library manager”. The most popular and recommended tools were found to be websites called “How Long To Beat”[1] and “Backloggery”[2]. These websites allow users to track their game completion progress and backlog, but they are web-based and do not offer a native desktop application, they are also limited to the library of games that they have in their database, not allowing for users to easily add their own games.

The research into these tools proved useful as it allowed for the mapping out of features that are essential for a backlog manager. In How Long To Beats case, the ability to track completion progress and the ability to search for games in a database were key features. Backloggery brought to light the idea of a graphical representation of a users library, which could be a unique feature to implement in this project. However, the design of the website was found to be very outdated and not visually appealing, with the sites design not having majorly changed since 2007, which could be a key area for improvement in this project, along with the performance improvements that come with a native application.

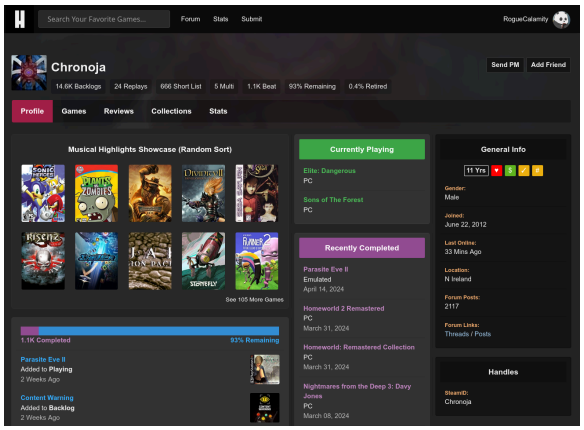

Figure 1: A screenshot of a profile on How Long To Beat.

Figure 2: A screenshot of the front page of The Backloggery.

Another technology that was investigated during development was the “Video Game Preservation Platform” Lutris[3] - however, Lutris is more focused on game installation and management, as opposed to tracking completion and a backlog. This tool was useful for understanding how an application could be used to launch games, which could be a feature to implement in this project. The design of Lutris was also taken into account, as it contains a modern design running on a native ui framework (GTK[4]).

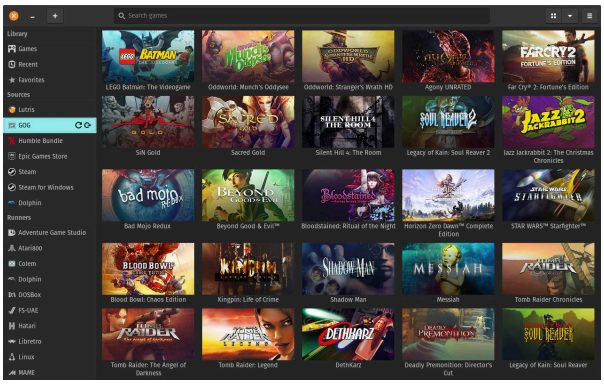

Figure 3: A screenshot of the Lutris application.

1.1.3 - Motivation

The motivation to undertake this project comes from a personal interest in the topic, as efficient tracking of game libraries and backlogs is not something readily available in the market, at least, not in a native desktop application. As existing tools are mainly web-based, they do not offer the same level of performance and integration that a native application could offer, such as being able to launch games directly from the application or being able to use the application offline.

This laid out some of the priorities for the project, such as performance improvements over existing tools, a modern and visually appealing design and the ability to be able to use the application without an internet connection. To this end, investigation into what technologies could be used to achieve these goals was undertaken.

1.1.4 - Research of UI Design

Research into UI design was conducted as a part of this project to ensure that the application was user-friendly and visually appealing. This research involved examining modern desktop applications, notably those in the KDE ecosystem, as well as inspecting the design guidelines for the Qt Framework [5] (The "4Cs": Consistency, Continuity, Context and Complementary being considered throughout design iterations.) and the KDE Human Interface Guidelines[6] (which can be summarised as "Simple by default, Powerful when needed.")

These principles promote reusing design patterns from other applications so that users can easily understand how to use the application, and that complex tasks should feel simple to the user. This research was used to inform the design of the application during development, ensuring a intuitive and user-friendly experience.

1.1.5 - Options for Third-Party API

For the API module of the project, a third-party API was required to allow users to fetch game data from the internet. Ideally, this API would allow for searching for games by name and return detailed information about the game - at the very least, the release date, genre, platform and a description of the game.

This API would also need to be free to use, and preferably not involve a complicated authentication process to access the data (such as having to sign up for an API key). Appropriate Licensing is also a consideration, as the API will be used in an educational capacity.

Two initial options were IGDB[7] (Internet Game Database) and the Steam Web API. The IGDB API was ruled out as it requires the user to have a Twitch account in order to sign up for an API key - which would require functionality that is out of scope for this project, requiring the application to sign up with the *Twitch Developers* system.

The Steam Web API does not require any authentication to access the data, and was found to be a good option for this project. The API does not support searching, but a list of games can be obtained and searched through locally. Steam Web API also provides information of a requisite level of detail for this project.

1018 WORDS

1.2 - Analysis

Whilst this project may seem simple on the surface, there are a number of considerations to be made. Firstly, the scope of the project had to be defined. The project was split into two main components - the main application and the API module. The main application would be responsible for managing the users library and backlog, whilst the API module would be designed to fetch game data from the internet. If the social features and recommendation system were to be implemented, they would be a part of the main application - although likely in seperate modules.

1.2.1 - Identification of Requirements and Objectives

The requirements for this project were identified through the research conducted in the background section. As a significant portion of the other tools investigated were web-based, there are considerations this project had to make which are not relevant for web-based apps. The key requirements for the project were as follows:

- The application must allow users to add games to their library and mark them as part of their backlog, in progress or completed.
- The application must allow users to search through their games and filter them based on various attributes.
- The application must allow users to search for games to add to their library via the use of a Third-Party API.
- The application must allow users to view detailed information about the games in their library, such as the aforementioned attributes.
- The application must be visually appealing and user-friendly, following modern design principles.
- The application must be performant, able to stand up even with a very large library of games.

These Requirements are discussed in more detail in Section 2.

1.2.2 - Choice of Technologies

The choice of language for this project was a matter of consideration up until the beginning of development. The two main languages considered were C++ and Python, and were the two languages used in the development of this project.

C++ was chosen for the main application due to its unrivalled performance, high suitability for Object-Oriented problems and, via use of the Qt[8] platform, a native and robust UI framework. Python was chosen for the API module due to its excellent networking libraries and ease of use, along with its libraries for fuzzy string matching.

The Qt Framework also boasts an extensive range of documentation and tutorials available online, making it an attractive choice for this project. It also has cross-platform capabilities, allowing the application to be shipped on the *Windows*, *MacOS* and *Linux* operating systems.*

Another advantage of using Python for the API module is that it allows for easy packaging of the module into an executable, which can be used standalone of the main program, allowing for easy testing and debugging of the module. This was achieved using the pyinstaller[9] library.

Finally, the method of persistent data storage had to be considered. Due to the nature of the data being stored, a relational database was chosen as the method of storage, in particular an

*However, the project was developed with a focus on Linux compatibility, with support for other operating systems coming second.

SQLite database, due to its lightweight nature and ease of use. In a bigger project, a more robust database system such as PostgreSQL could be considered.

1.2.3 - Alternative Approaches

Many other languages were considered for this project, such as an entirely Python-based solution. However, Python does not have the same level of performance as C++, and would not fulfil the performance objectives of the project. Python also does not have an object-oriented system that is as robust as C++, which would make the project harder to maintain and increase the difficulty of debugging the application.

Another language that was considered for this project was the increasingly popular Rust language. Rust is known for its performance and rigorous safety requirements, which would have made it a good choice for a project such as this. However, a significant caveat to using Rust for this project is the lack of a mature UI tooling ecosystem. Whilst there are bindings (bindings being a way to use a library from another language) for Qt in Rust, this would essentially involve doing the majority of the work in C++ regardless, which would defeat the purpose of using Rust in the first place.

Java and C# were also considered, but were ruled out - Java due to its performance and C# due to the impracticality of using it on Linux systems. Java also has an absence of modern UI tooling, with Swing and JavaFX being the only real options, both of which are outdated and not visually appealing. QtJambi was briefly investigated as a potential solution, but there was found to be a lack of documentation and community support for the library.

780 WORDS

1.3 - Process

When planning development, a Kanban framework was used to manage the project. The use of a Kanban Board is a common practice in software development, and one of the few practices that can be used for a solo project.

The Kanban Board was used to manage the project by decomposing the project down into smaller tasks and assigning them to one of three (although there are four columns in total) columns on the board. The columns were “To Do”, “In Progress” and “Done”. The “To Do” column contained all tasks that needed to be completed, with cards sorted by priority; the “In Progress” column contained tasks that were currently being worked on; and the “Done” column contained tasks that had been completed. A WIP Limit was set so only three tasks could be deemed to be “In Progress” at any one time, to prevent jobs from being left unfinished.

Using the “GitHub Projects” feature to host the Kanban Board, the project was able to be effectively managed. Using this feature allowed issues to be linked to cards on the board, which allowed for easy tracking of progress and completion of tasks.

When Functional Requirements were identified, they were added as issues to the GitHub repository and linked to cards on the Kanban board, serving as small milestones. Cards were also labelled with the type of task they were such as “Feature”, “Bug” or “Documentation”.

Certain cards were linked to larger milestones, such as the Mid-Project Demonstration, clearly showing what features I wanted to have in place by that point. This allowed for a clear plan in regards to the timing of the project, and what features were to be implemented.

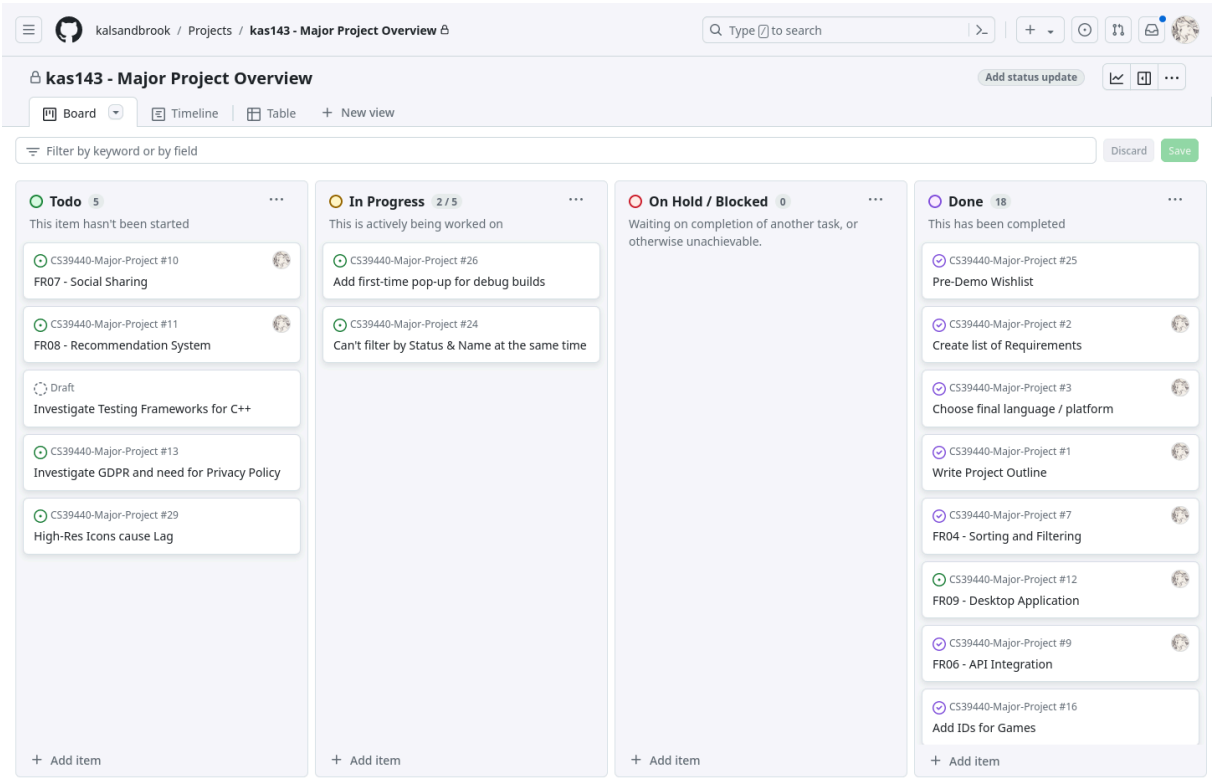

Figure 4: A screenshot of the GitHub Projects board used for this project.

A weekly log has been kept throughout the project to document progress and any issues that arose. Each week was broken up into the objectives for the week, the tasks completed, the challenges faced and the plans for the upcoming week. This log was used primarily as a starting point for the weekly meetings with the project supervisor, it also served as a good way to keep track of progress and the pace of development.

2 - Requirements

The requirements for this project, identified through research detailed in the background section, underpinned the majority of the development process, dictating the features to be implemented and key design decisions. Requirements were split into two main categories: Functional Requirements and Non-Functional Requirements.

Functional Requirements are the features that the application must have, and coinciding with the Kanban methodology, serve as small milestones for the project. Non-Functional Requirements are the critical aspects of the application that are not directly related to the features of the application, such as performance and usability, these were considered throughout development. This project also has a set of Optional Functional Requirements, which are features that could be implemented if time allowed.

Requirement IDs are used to reference requirements throughout the document, and are formatted as "FRXX" for Functional Requirements, "NFRXX" for Non-Functional Requirements and "OFRXX" for Optional Functional Requirements.

2.1 - Functional Requirements

FR01 - Game Management

Users will be able to perform CRUD operations on games in their library. This includes adding games to the library, removing games from the library, updating the details of a game and viewing the details of a game. Games will have attributes such as title, genre, platform, release date, completion status and a description.

FR02 - Backlog Management

Users will be able to mark games in their library as part of their backlog. These games will be able to be displayed as part of a seperate list.

FR03 - Progress Tracking

The application will allow for games to have their progress tracked, with statuses such as “In Progress”, “Completed”, “Not Started” and “Abandoned”. This will allow users to easily see which games they have completed and which they have yet to start.

FR04 - Sorting and Filtering

Users will be able to sort and filter their games based on information about the game itself (such as name, release date or genre) or meta-information about the game, such as completion status or any user-defined tags.

FR05 - Manual Game Entry

The program will allow for users to add games to their library manually, specifying data themselves, rather then fetching it via an API, for games that are not in the API’s database and in the event that an internet connection is not available. This will also aid in development and testing before the API module is complete.

FR06 - API Integration

Users will be able to use a third-party API to automatically fetch game information based on a given name. This will aid usability by increasing the speed at which games can be added to the library, and will allow for more detailed information to be displayed about the game. Users will be potentially be able to pick from multiple third-party APIs, if available.

FR07 - Desktop Application

The application will be a native desktop application, allowing for good performance and offline use. This will also allow for integration with the users system, such as being able to launch games directly from the application.

OFR01 - Social Sharing (Graphical Library Export)

The application will allow users to export a graphical representation of their library, akin to an old-school forum signature. This will allow users to share the games they have completed over a period of time, and will be a unique feature of the application. A visual representation will allow for users to share their progress online, withut ties to any particular platform. This is an optional feature.

OFR02 - Recommendation System

The application will be able to suggest games to add to the users library based on their existing library and completion status. This will allow users to easily find new games to play. The implementation of this feature is unlikely, due to the complexity of recommendation systems and the time constraints of the project. This is an optional feature.

2.2 - Non-Functional Requirements

NFR01 - Usability

The application must have an intuitive, easy-to-use interface that is able to be navigated by users with limited technical knowledge. The application should also be visually appealing, following modern design principles, including the KDE Human Interface Guidelines[6].

NFR02 - Performance

The application will be performant, responding to user interaction promptly and being able to handle game data efficiently. This is an important need, especially the considering the very large volume of data the application could be dealing with.

NFR03 - Reliability

The application make sure to store data in a robust format, taking measures to tackle potential data loss or corruption. In the event of data loss, the application should be able to recover. It is also important that the application is able to handle errors gracefully, providing useful error messages to the user and avoiding crashes where possible.

NFR04 - Compatibility

Whilst the application is being developed with the Linux Operating System in mind, it should be able to easily be made to run on Windows or potentially MacOS. In order to achieve this, the application should avoid using features specific to a particular operating system and use platform-agnostic code where possible.

NFR05 - Maintainability

The codebase of the application should be well-documented and well-structured, to aid in future development and maintenance. Features should be designed in a modular manner, with the addition of new futures being kept in mind. Code should be kept in a state where somebody else could pick up the project and understand it.

NFR06 - Interoperability

As the application is using and integrating with third-party APIs, it is important that the application is able to handle changes to the API gracefully. The application should be able to handle changes to the API without crashing, and should be able to provide useful error messages to the user in the event of an API failure. Relevant standards and protocols should be adhered to.

NFR07 - Localization

Whilst the application will not be translated into multiple languages at this time, the application should be designed in a way where translation is possible in future. A theoretical translator should be able to easily translate the application into another language without a significant knowledge of programming.

3 - Design

4 - Implementation

4.1 - Class Diagram

5 - Testing

6 - Evaluation

Bibliography

[1] "How Long To Beat." Accessed: Apr. 16, 2024. [Online]. Available: https://howlongtobeat.com/

One of the investigated similar applications that inspired the creation of this application. HLTB is a website that provides information on how long it takes to beat a video game, along with providing functionality to track games and create a backlog.

[2] "The Backloggery." Accessed: Apr. 16, 2024. [Online]. Available: https://backloggery.com/

Another similar application that was investigated. The Backloggery is a website that allows users to track their video game collection and progress.

[3] "Lutris - Open Gaming Platform." Accessed: Apr. 16, 2024. [Online]. Available: https://lutris.net/

The Lutris website, which provides an open gaming platform for Linux. It was investigated as a similar application to the one being developed.

[4] "The GTK Project - A free and open-source cross-platform widget toolkit." Accessed: Apr. 16, 2024. [Online]. Available: https://www.gtk.org/

The GTK Project website, which provides information on the GTK toolkit used for developing graphical user interfaces. It was not used in the development of the application, but used by other applications that were investigated.

[5] "How the 4Cs of UX design benefit your software development." Accessed: Apr. 16, 2024. [Online]. Available: https://www.qt.io/4cs-of-ux-design

An article that discusses the 4Cs of UX design and how they can benefit software development.

[6] "KDE Human Interface Guidelines." Accessed: Apr. 16, 2024. [Online]. Available: https://develop.kde.org/hig/

The KDE Human Interface Guidelines, which provide guidance on designing user interfaces for KDE applications.

[7] "IGDB API Documentation." Accessed: Feb. 07, 2024. [Online]. Available: https://api-docs.igdb.com/

One of the APIs considered for the application. It was not used in the final implementation, but was investigated during the planning phase.

[8] "Qt." Accessed: Apr. 16, 2024. [Online]. Available: https://www.qt.io/

The development framework used for the application. This allows the creation of native applications across multiple platforms with ease.

[9] "PyInstaller." Accessed: Apr. 16, 2024. [Online]. Available: https://pyinstaller.org/en/stable/

The tool used to package the application into a standalone executable.

[10] "Requests - HTTP for Humans." Accessed: Apr. 16, 2024. [Online]. Available: https://docs.python-requests.org/en/master/

The python library that was used to make HTTP requests to the API.

[11] "TheFuzz - Fuzzy String Matching in Python." Accessed: Apr. 16, 2024. [Online]. Available: https://github.com/seatgeek/thefuzz

The python library that was used to perform fuzzy string matching.

7 Appendices

Appendix A – Third-Party Code and Libraries

Qt [8]

The Qt Framework (particularly Qt Widgets) was used for the majority of development for this project. It provides a wide variety of features - notably its Qt Widgets, Qt SQL & Qt Concurrent modules.

For academic purposes, Qt Community Edition (the edition used for development) follows the *GNU Lesser General Public License* ("(L)GPL"), a copy of which can be accessed here: https://www.gnu.org/licenses/lgpl-3.0.txt

No tools such as Qt Creator or Qt Designer were used in development.

Python Libraries

For the API module of the project, Python was used due to its excellent range of libraries. The following libraries were used:

- *Requests* [10]
  - Used for making HTTP requests to the API.
- *TheFuzz* [11]
  - Used for fuzzy string matching, particularly using the jaro-winkler algorithm.
- *PyInstaller* [9]
  - Used for packaging the API module into an executable.