Software Architecture and Design

Final Project Group 10

Student 1: Joel Collins

CS 7319

Off Campus

Student 2: Bryan Putnam

CS 7319

Off Campus

Project Title: GameButler

Joel is the team leader who will be responsible for submitting the Project Proposal and Final Project Deliverables.

Project Overview:

1. Describe the main functions of your software system.

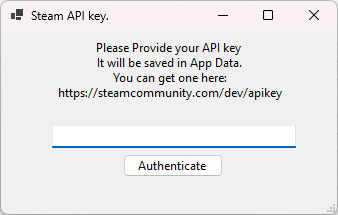
The Main Function of the program is to organize and display your video games. These video games would be organized via data provided from various API calls to external websites, namely, Steam (store.steampowered.com), via several API libraries (api.steampowered.com, partner.steam-api.com, steamcommunity.com, www.metacritic.com, and steamcharts.com). The API’s and websites would be queried to get: owned games lists, game title information, player base statistics, game critic scores, last official news posts, and also to authenticate and log in automatically. When clicking on the respective filter, it would organize the list of games, and clicking on the individual cell would display the HTML page that best pertains to the information you clicked on.

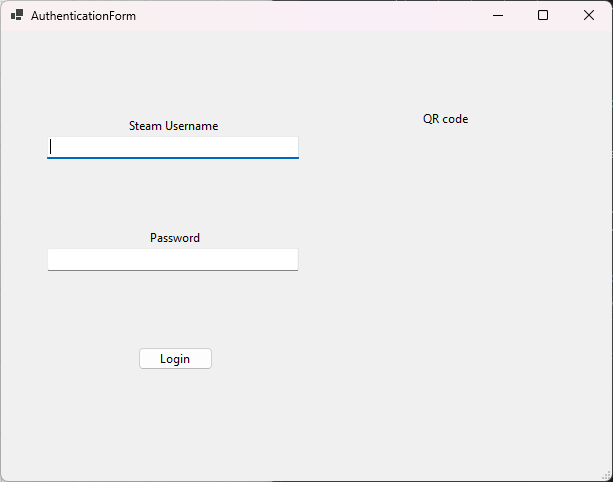
1. Describe the expected outputs or results of your software system.

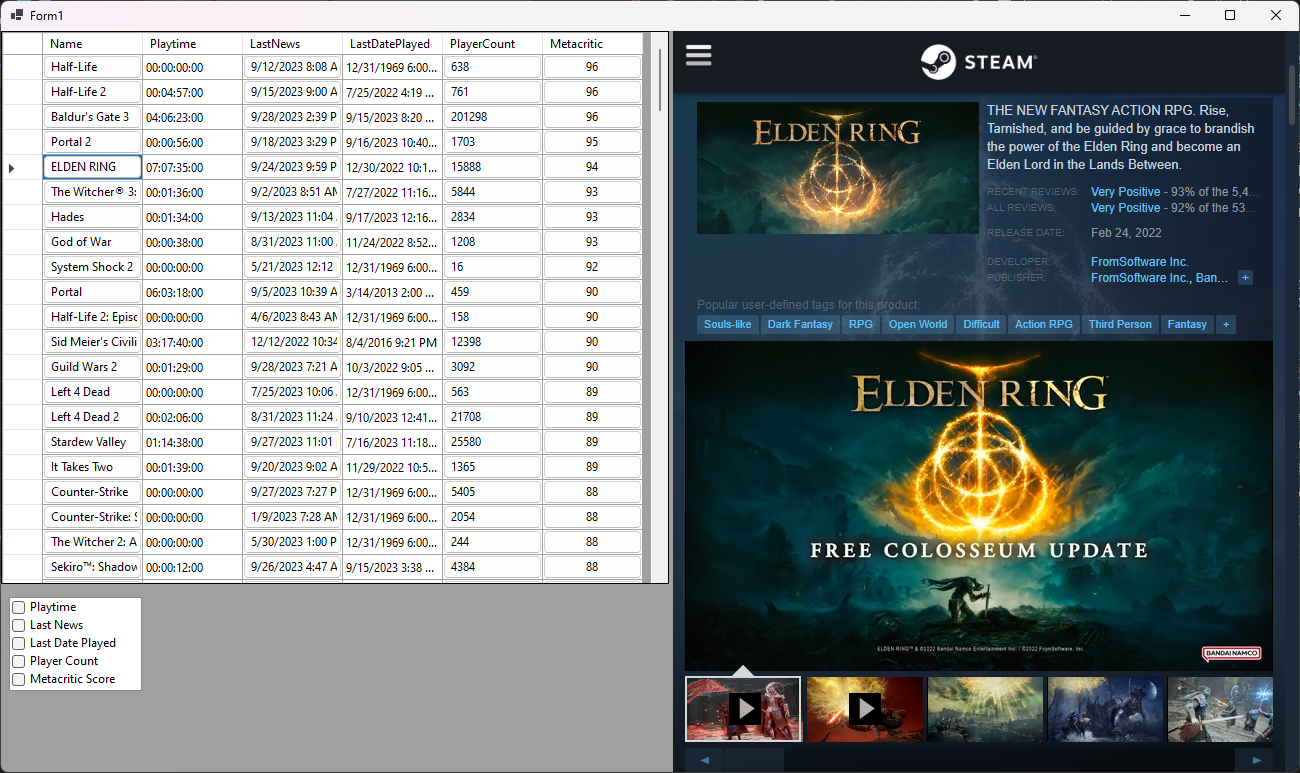
The results should be an interactive categorical table with filters that can be sorted to your specifications, with relevant table cells hyperlinked to relevant information. More features, like the filtering of columns themselves via the GUI are planned objectives, but uncommitted. We have several other features in mind.

1. Prototyping Interfaces:

Give snapshots of your project prototype.







Project Design:

1. Select two different architectural styles to design, compare, and evaluate.

We are planning on creating a class library written in C# to interface with the API’s. We will be coding in C# and using the framework .NET 6, a cross-platform coding standard that allows for multiple types of architectures and frameworks. With one of the Web-based frameworks, ASP.NET, Multiple architectural styles have been Included, like Model View Controller (MVC), which is actually an MVVM architecture, Server-only processing, and Client-Server processing. These are all renditions layer-based and component-driven architectures.

The two Architectural styles we will choose will both utilize the exact same class library for the API calls, which is layered and component-based, but also utilizes the architecture of the internet with cookies, HTTP GET, and POST commands via REST API style architecture. The Inputs of this layer will be Login/ Password information (encrypted) and method calls, and the output of this will be essentially a shared memory architecture, and Boolean variables indicating success.

The difference in the two different implementations of the program is that they will completely utilize different front-end implementation frameworks.

One will use a component-based layered architectural style done in Windows Forms, a native windows application, but with a side-loaded HTML viewer component called WebView2, which is a .net framework plugin based on Microsoft Edge. The main layers at the low-level implementation will be: a presentation layer, interface layer, a possible business layer (not currently utilized, but may be needed to meet uncommitted goals), and a link to the data-access layer. This final DAL layer will be in the Class library used in both architectures.

In this implementation there will be multiple presentation layers: one for authentication, one for login, and one for presentation of the game information itself. Behind each presentation layer item there will be an interface layer for the action routing of that specific Windows Forms based view. The uncommitted, but planned part of this application will include a chat feature with friends, however this implementation may need access to a payment only, C++ based, Partner level API, which I have recently applied for, but have not yet received the acceptance letter for access. This API is generally used in actual video games for speaking directly with steam servers to route messages and invites from friends in the Steam Network.

The other will utilize a complex web-based Client-Server, and Server-Server Architecture, with either REST-based MVC style architecture or purely REST-Based Controller architecture. In the Uncommitted example of this architecture, there will need to be Client processing written in C# but presented to the client as JavaScript code in order to hash and properly secure passwords for transmission across the internet to the Server, and then to the REST based API. But, because this code will be only transmitting locally to the server, this is an extended goal for hopes of actual deployment to a real web service.

This will have a similarly based GUI architecture, but follow a REST-like structure to interact with the other layers. This part of the implementation

We will refrain from using external packages, and not use any third-party frameworks or libraries to accomplish these goals. But, Because the C# language heavily relies on and strongly suggests coding standards which require the use of class structures, we will be coding the layers as classes but just as a nominal and organizational relation, not as a functional requirement. Some low-level objects that absolutely require class structure, will also be object-oriented, to organize them for data binding to UI elements.

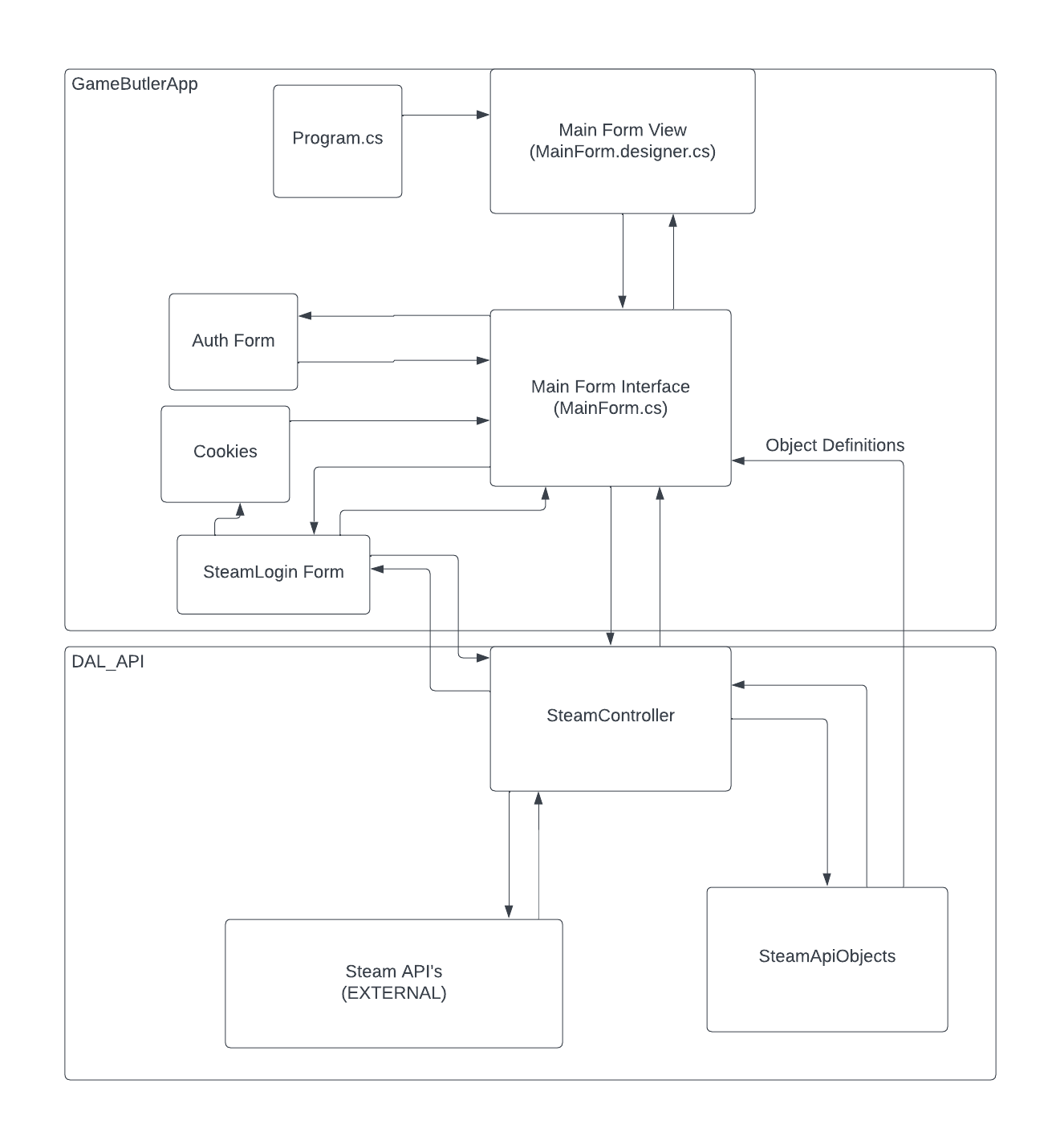
If this is a problem, it is possible, but entirely frowned-upon, to create the entire system as a method-only layered architecture, but I would argue against this for sake of coding standards, general practicality, framework required instances, like automatic data-binding with windows forms controls and the automatic parsing of JSON objects returned from HTTP GET requests. Many benefits of using C# would completely be reversed, and most of the time would be spent re-engineering libraries and namespaces which are readily available cross-platform. I couldn’t import system level libraries, and most of my time would be spent creating a JSON parser, which is not a good interest in time, and not practical. For example: I can parse a JSON object into a object-related field just by calling a system level command and providing it with a custom object that directly mirrors the JSON object I expect.

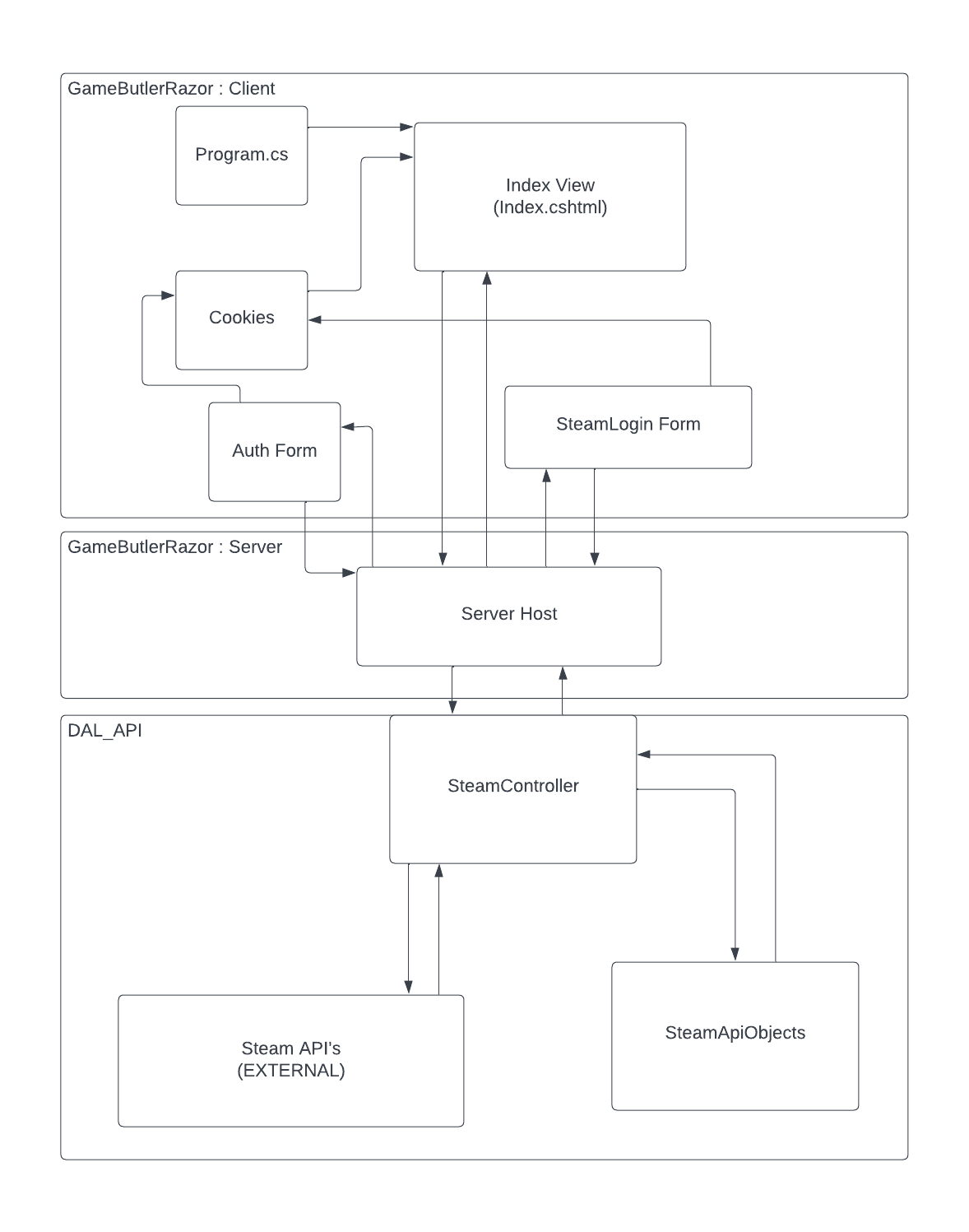
Yes, it is possible to not create a low-level object, manually parse the JSON object character by character, data-bind, rename, and re-size columns in C#. As trivial as this sounds, the complexity of implementation increases exponentially as you increase complexity of the data objects and increase the scope of where in the framework you levy such requirements. And as such, we feel we will learn more from focusing on large architectural constraints such as the utilization of multiple languages and interfaces, each using different architectures or connectors, user interface implementations, Security, and Modular coding across the web-stack and application-stack.

*2. Architecture Diagrams*

*[Draw two high-level architecture diagrams of your system, one for each architecture*

*design, showing the components and connectors in each archi.]*





3. Finally, choose the most suitable architecture to implement your project. Please

provide the rationale for your selection.

We will implement both of the architectures, because we are in a team. However, we feel that the web version we implement will be better because it is cross platform, and is natively built for the architecture.

Project Implementation Plan:

We will use C#

We will do the detailed system design together, each do a front-end implementation independently, but review each other every step of the way to ensure that we are coding to the correct standards and constraints.