

# Immersive Quiz for Spanish Learners

A Manuscript

Submitted to

the Department of Computer Science

and the Faculty of the

University of Wisconsin–La Crosse

La Crosse, Wisconsin

by

**Austin Klum**

in Partial Fulfillment of the

Requirements for the Degree of

**Master of Software Engineering**

May, 2022

# Immersive Quiz for Spanish Learners

By Austin Klum

We recommend acceptance of this manuscript in partial fulfillment of this candidate's requirements for the degree of Master of Software Engineering in Computer Science. The candidate has completed the oral examination requirement of the capstone project for the degree.

---

Prof. Kasi Periyasamy  
Examination Committee Chairperson

---

Date

---

Prof. Steven Senger  
Examination Committee Member

---

Date

---

Prof. Kenny Hunt  
Examination Committee Member

---

Date

# Abstract

Austin Klum, J., “Immersive Quiz for Spanish Learners,” Master of Software Engineering, May 2022, (Elliot Forbes, Ph.D.).

This manuscript describes the development of a quiz creation tool combined with a virtual reality component to provide an immersive quiz taking experience for Spanish learners. The quizzes also have an orienteering course aspect as well, where each quiz is comprised of multiple timed locations where all questions must be completed correctly before continuing onto the next location.

## Acknowledgments

I would like to express my thanks to the Department of Computer Science at the University of Wisconsin–La Crosse for providing the learning materials and computing environment for my project.

# Table of Contents

Abstract . . . . .	i
Acknowledgments . . . . .	ii
List of Tables . . . . .	v
List of Figures . . . . .	vi
List of Code Listings . . . . .	vii
Glossary . . . . .	viii
1. Introduction . . . . .	1
1.1. Overview . . . . .	1
2. Requirements . . . . .	2
2.1. Overview . . . . .	2
2.2. Development Methodology . . . . .	2
2.3. Course Creator . . . . .	2
2.3.1. Users . . . . .	3
2.3.2. Create Course Flow . . . . .	3
2.4. Virtual Reality Orienteering . . . . .	3
2.4.1. Users . . . . .	4
2.4.2. Virtual Reality Orienteering Flow . . . . .	4
3. Design . . . . .	5
3.1. Overview . . . . .	5
3.2. Technologies . . . . .	5
3.2.1. Course Creator . . . . .	5
3.2.2. Virtual Reality Orienteering . . . . .	6
3.3. Class Diagrams . . . . .	7
3.3.1. Course Creator . . . . .	7
3.3.2. Virtual Reality Orienteering . . . . .	7
3.4. Database . . . . .	7
3.4.1. Database Schema . . . . .	8
3.5. Communication between Programs . . . . .	8
4. Implementation . . . . .	9
4.1. Overview . . . . .	9
4.2. Course Creator . . . . .	9
4.2.1. Authorization . . . . .	9
4.2.2. Courses . . . . .	9
4.2.3. Locations . . . . .	11
4.2.4. Questions . . . . .	11
4.2.5. Answers . . . . .	11
4.2.6. Course Results . . . . .	11
4.2.7. Web API . . . . .	11
4.3. Virtual Reality Orienteering . . . . .	11
4.3.1. Authorization . . . . .	11
4.3.2. Displaying Course . . . . .	11
4.3.3. Tracking and Submitting Score Results . . . . .	11
5. Testing . . . . .	12

5.1.	Overview . . . . .	12
5.2.	Point 1 . . . . .	12
5.3.	Point 2 . . . . .	12
5.4.	Point 3 . . . . .	12
6.	Conclusion . . . . .	13
6.1.	Overview . . . . .	13
6.2.	Point 1 . . . . .	13
6.3.	Point 2 . . . . .	13
	Bibliography . . . . .	14

# List of Tables

## List of Figures

1	Course Creator Use Case Diagram . . . . .	3
2	Virutal Reality Orienteering Use Case Diagram . . . . .	4



# List of Code Listings

1	New Script in Unity . . . . .	6
2	Securing Controllers using Filter on User Role . . . . .	9
3	Course Model . . . . .	9
4	Course Details Razor Page . . . . .	10

# **Glossary**

## **ASP.NET Core**

Lorem ipsum

## **ASP.NET Core MVC**

Lorem ipsum

## **Unity**

Lorem ipsum

## **Unity XR**

Lorem ipsum

## **C#**

Lorem ipsum

## **LINQ**

Lorem ipsum

## **Entity Framework Core**

Lorem ipsum

## **SQL Server**

Lorem ipsum

## **Transparent Data Encryption (TDE)**

Lorem ipsum

## **Basic Authentication**

Lorem ipsum

## **JSON**

Lorem ipsum

## **HTTPS**

Lorem ipsum

## **REST**

Lorem ipsum

## **CSS**

Lorem ipsum

## **HTML**

Lorem ipsum

## **Bootstrap**

Lorem ipsum

## **Web API**

Lorem ipsum

## **Git**

Lorem ipsum

## **Dependency Injection**

Lorem ipsum

## **Razor View Engine**

Lorem ipsum

## **Visual Studio**

Lorem ipsum

## **Unity Editor**

Lorem ipsum

# 1. Introduction

## 1.1. Overview

The rise of globalism has prompted people of different cultures to increasingly work together and interact with one another. Thus, understanding other cultures and languages will become ever more important. Often times this can be hard to teach, especially in a classroom. Virtual reality can be used as a means to bridge the gap between real-world understanding and classroom knowledge. Virtual reality allows for a more immersive experience. A more immersive experience is a more effective way to engage students and promote learning.

In 2017-2018 there was an initial virtual reality project conducted by Claire Mitchell to take tours of Medellin, Colombia. This project was a success and discussions were made to expand on this initial success. In 2019, there was a grant proposal for development of a new project to further enhance experiential learning. As virtual reality is a vanguard area of development such resources don't exist yet and would require new development. The proposal also requested an orienteering component to be included. Orienteering is an activity where participants "navigate between checkpoints along an unfamiliar course" [1]. The primary purpose of adding an orienteering aspect is to add to the depth of cultural understanding, as orienteering requires the participants to have a more active role in the experience.

## **2. Requirements**

### **2.1. Overview**

The following section is on the requirements of the project and development methodology.

### **2.2. Development Methodology**

The stakeholders for this project is the project owner and sponsor, project advisor, and developer, Claire Mitchell, Elliot Forbes, and Austin Klum respectively. The end users are the students using the virtual reality tool for their learning and the instructors using the quiz management tool for assessment of student's comprehension of classroom material.

The chosen development methodology for a project is an early and influential decision made that alters the course of development. As this was a solo-developer, web, and virtual reality project with a busy project sponsor, the decision was made to make use of an iterative agile methodology. An iterative agile approach focuses on delivering value to the product in fast small increments, rather than all at once. This approach allows software developers to adjust, refine, and review the development process to better provide value and output. This approach also allows for earlier risk identification and the flexibility to easily correct course.

Traditional methodology follows the waterfall approach which "contains five phases of management, where each requires a deliverable from the previous phase to proceed" [2]. The waterfall method is more suited for projects that follow a linear path and is fixed and rigid. The project did not have this rigidity or clarity of output, hence a more Agile approach was taken. The developer also made use of a KanBan board to help keep organized.

### **2.3. Course Creator**

This tool allows the project sponsor and verified users to create orienteering courses for the students to complete. Each course is comprised of locations which has a corresponding photo to accompany the location. To help with immersion and to best utilize the capabilities of virtual reality, the uploaded photos must be 360 Photos, also known as a photo sphere. This type of photo allows for the virtual reality tool to wrap the image around the user making a sphere, such that the user is able to look around as if they were really at that location. The tool also allows for locations to be added, updated, or deleted. Each location has a list of questions. There is no limit to the number of questions per location. The tool allows for each question to be added, updated, or deleted. Each question has a list of answers which can be of one to six possibilities. The tool allows for each answer to be added, updated, or deleted.

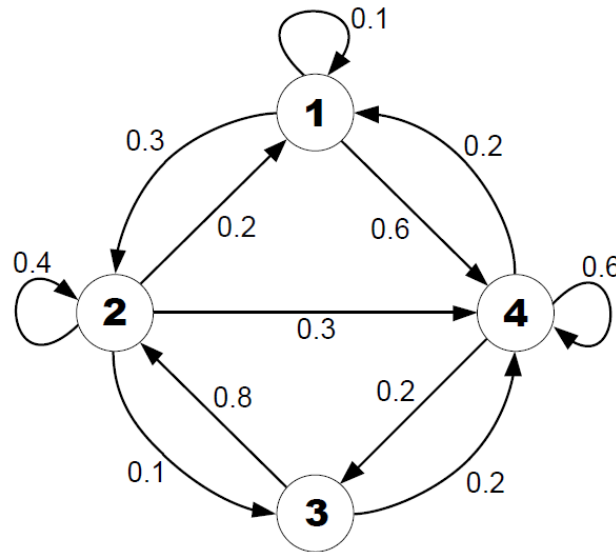
Once a course has been completed by a student, a verified user can view the results from the course main dashboard. This dashboard lists the student ID, point score, time score, and total score. As security is important, the database is encrypted and requires an authorized administrator database account to access the data outside of the tool.

For a user to be created, they must create an account with an email and password. Once the

account is created, the user is immediately able to login, but will be unable to access any of the course creation or course results pages, limited only to the homepage and user settings. Each verified user has the capability to approve other users. In the user settings page, there exists a link to verify users. The verify users page list all users and their status of verified or not verified. From here a verified user can verify or un-verify other users by selecting the corresponding checkbox and saving.

### 2.3.1. Users

Figure 1 explains the use case diagram.



**Figure 1.** Course Creator Use Case Diagram

### 2.3.2. Create Course Flow

This section will include photos and written steps on how to use the course creator.

## 2.4. Virtual Reality Orienteering

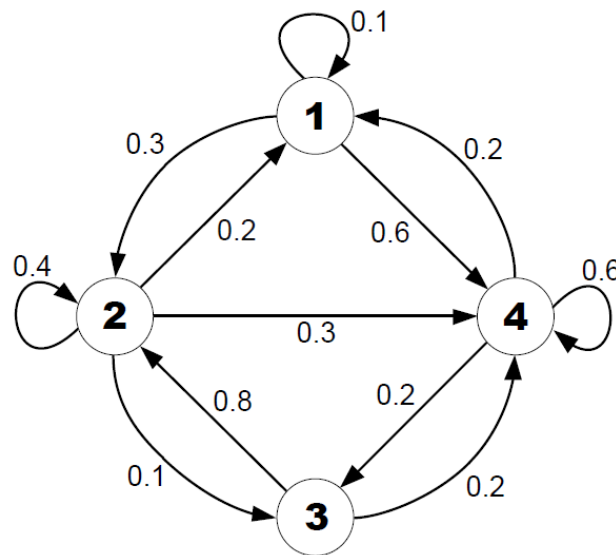
This tool is used by the students to complete the created orienteering courses that the project sponsor or verified user(s) created. Upon starting the program the user is prompted to enter their student ID using a "mallet" and tap virtual keyboard. Once the student has entered their student ID, they then select the course via a select list with a point and trigger arrow pointers buttons. After hitting the start button the student begins the course. The first location is loaded up and the student can look around the location. Upon touching the touch pad the first question and corresponding possible answers appears in the world view. As part of the orienteering spirit each question must be answered correctly to continue on to the next question. When a student answers incorrectly the button turns red and is disabled. When answered correctly the button flashes green and the next question and corresponding possible answers load. The student is graded on how quickly they came to the correct answer and, with the orienteering aspect in mind, the time to completely answer all the questions

in a location. Each question is worth one point and each location time is worth one point. The score point is awarded divided by the number of attempts. This means that a question with four possible answers the following outcomes are possible: answered correctly has one point awarded, one incorrect attempt has .75 points awarded, two incorrect attempts has .5 points awarded, and three incorrect attempts has .25 points awarded. For the time aspect of the grade, the student has 100 seconds to complete the location, where each second is worth .01 points. This means if the student took 30 seconds to answer the questions and complete the location, they are awarded with .70 points. A time of 45 seconds is awarded .65 points.

Upon answering all the questions for a location, the next location is loaded and the timer is reset. Once all locations have been completed a game over screen appears stating the student's point score, time score, total score, and what the maximum points awarded could be.

#### 2.4.1. Users

Figure 2 explains the use case diagram.



**Figure 2.** Virtual Reality Orienteering Use Case Diagram

#### 2.4.2. Virtual Reality Orienteering Flow

This section will include photos and written steps on the experience of the student completing a course.

## 3. Design

### 3.1. Overview

This section discusses the design of the project including technologies used, classes, database schema, and user interface.

### 3.2. Technologies

This project uses a combination of ASP.NET Core MVC and Unity XR. The Course Creator was developed with the former and the Virtual Reality Orienteering was developed with the latter.

#### 3.2.1. Course Creator

ASP.NET Core MVC “is a lightweight, open source, highly testable presentation framework optimized for use with ASP.NET Core.” [3] ASP.NET Core is the underlying framework that enables development. ASP.NET Core “is a cross-platform, high-performance, open-source framework for building modern, cloud-enabled, Internet-connected apps.” [4].

The MVC in ASP.NET Core MVC stands for Model-View-Controller. MVC is an architectural pattern which separates an application into three main components: Models, Views, and Controllers. This separation helps achieve a “separation of concerns”, which asserts “that software should be separated based on the kinds of work it performs” [5]. Models represent the data structure, independent of the user interface. Models are responsible for data, logic, and rules of the application. Views represent the user interface and information. Views are responsible for presenting content with minimal logic. Controllers represent the logic and actions for models and views. Controllers are responsible for responding to user input and performing operations. In summary, models are what it is, views are for what it looks like, and controllers are for how it behaves.

ASP.NET Core MVC was chosen because of the developer’s prior experience and the extensive functionalities that the framework provides. ASP.NET Core MVC is open-source and multi-platform, supporting Windows, macOS, and Linux out of the box. ASP.NET Core MVC provides routing which is a useful URL-mapping component. Routing provides for easy link generation without regard to the actual file structure. ASP.NET Core MVC also provides model binding on requests. This makes incoming and outgoing requests easy to process or generate without further processing. Model validation is also built-in using data annotation attributes. These are pre-built or custom attributes within the model that validate on the fly rather than requiring explicit checking. This allows for guaranteeing the state of the model before further processing. Dependency injection is also supported, which is a key feature for building the controllers. Dependency Injection (DI) is a software design pattern, “which is a technique for achieving Inversion of Control (IoC) between classes and their dependencies.” [6] A dependency is an object than another object depends on. When a class depends on another class, future changes become problematic. Dependency Injection



solves this by using an interface to abstract the dependency, registering the dependency in a service container, and then injects the dependency when needed. ASP.NET Core MVC provides filters which can be placed on controllers so that all actions must meet this filter. Oftentimes filters are added for exception handling or authorization. This way instead of each action requiring authorization, one can require the entire controller with all actions to be authorized. ASP.NET Core MVC is also a great platform for building Web APIs. HTTP content-negotiation with common data formats such as JSON or XML is already supported. The Razor view engine is another key advantage for ASP.NET Core MVC. Razor view engine is a compact and easy template markup language used for defining views with embedded C# code. Razor can be used to dynamically generate web content with a mix of server side and client side code. Tag Helpers are also used with Razor to facilitate creating and rendering HTML. Tag Helpers bind to certain HTML elements and vastly improves their use cases. ASP.NET Core MVC also provides a database object-relational mapper (O/RM) tool called Entity Framework Core. Entity Framework Core provides a “code first” experience to create database entity models by writing code. This cuts down on the boilerplate code required for making database connections. The database can be accessed and queried using LINQ, Language-Integrated Query, a set of technologies based on the integration of query capabilities directly into the C# language. Entity Framework Core makes committing changes to the database simple with automatic change tracking.

### 3.2.2. Virtual Reality Orienteering

Unity is a cross-platform game engine developed by Unity Technologies with the goal to provide developers with the tools to make game development simple. Unity supports a vast variety of platforms and user experiences. These tools extend to desktop, mobile, console, and virtual reality with support for 2D, 3D, and other experiences. It’s also used in other areas outside of game development such as film, engineering, architecture, and automotive modeling. This makes Unity a popular choice, not only for the developer of this project, but for the world at large. “In the fourth quarter of 2021, Unity had, on average, 3.9 billion monthly active end users who consumed content created or operated with its solutions.” [7]

The basic components of a game developed in Unity are GameObjects, Assets, Scenes, and Scripts. Every object in a game is a GameObject. Assets are reusable items that can be used throughout the game. They can be of any file type that Unity supports, such as a 3D model, audio file, image. Oftentimes these assets come from outside Unity, created by the developer or other developers which are found in the Asset Store. Scenes contain the objects of the game. Oftentimes these are split into logical groupings such as main menu, individual levels, or the environment. Scripts are what controls the behavior of the GameObjects. Without scripts the game would be static and have no interaction or logic. Unity supports C# natively and is the standard used for scripting. A new script has two functions, `Start()` and `Update()`. `Start()` is called once by Unity before gameplay begins which is used to setup initial configurations. `Update()` is called once per frame update for the GameObject. This is used to handle anything that needs to be done over time in the gameplay, such as movement, triggering actions, or responding to user input.

#### Listing 1. New Script in Unity

```
1 using UnityEngine;
2 using System.Collections;
3
4 public class NewBehaviourScript : MonoBehaviour {
5
6     // Use this for initialization
7     void Start()
8     {
9
10    }
11
12    // Update is called once per frame
13    void Update()
14    {
15
16    }
17 }
```

---

Unity XR Interaction Toolkit is a cross-platform plugin used for virtual, mixed, and augmented reality. XR Interaction Toolkit provides easy built-in functionality to select, grab, throw, rotate objects within a VR scene. This also extends to the UI interactions and the haptic feedback that comes with it. The ability to look around and move within the workspace is also provided with this plugin. As this project uses a HTC Vive headset, OpenXR was the targeted development platform. OpenXR is an open standard that targets a wide array of virtual reality devices.

### 3.3. Class Diagrams

#### 3.3.1. Course Creator

#### 3.3.2. Virtual Reality Orienteering

### 3.4. Database

The database uses Microsoft's SQL Server. SQL Server is the de facto used for .NET projects. SQL Server is a relation database management system which manages and stores information. The standard tool for working with SQL Server is SQL Server Management Studio which makes database changes and transactions easy. Transparent Data Encryption (TDE) encrypts data files at rest. This means that any data files stored are encrypted preventing any malicious attempts to read the database.

### 3.4.1. Database Schema

## 3.5. Communication between Programs

This project comprises of two separate programs which must communicate with each other, and do so securely. The Course Creator has a RESTful Web API which requires Basic Authentication to make calls to. Basic Authentication requires a username and password over an HTTP connection. Using just an HTTP connection is not secure though, as the username and password are sent over in plaintext. This means any malicious user can sniff the network and easily obtain the credentials. To counter this the project requires a secure connection using HTTPS, with the ‘S’ meaning secure. An HTTPS connection encrypts any data sent over it; thereby securing the Basic Authentication credentials. The RESTful Web API provides predefined endpoints for authorized users to use with REST calls. REST stands for REpresentational State Transfer and is a architectural standard for communication on the web. REST is stateless, meaning that the server does not need to know about what state the client is or vice versa. This allows for communication without needing to know the previous messages. A REST request usually is comprised of an HTTP verb, header, URI path, and an optional body containing data. The four basic HTTP verbs are GET, POST, PUT, and DELETE. GET retrieves data or a specific resource. POST creates new data or a new resource. PUT updates data or a specific resource. Finally, DELETE removes data or a specific resource. The header is used for carrying pertinent information about the request being made. This includes the authorization, cookies, caching, and other logistical information for fulfilling the request. The REST requests consume or return data in JSON which is a standard data-interchange format. JSON is easy for humans to read and simple for computers to parse and generate.

## 4. Implementation

### 4.1. Overview

This section goes into details on the implementation of the Course Creator and Virtual Reality Orienteering programs. Key examples and general explanations are presented, to help the reader understand the code without rehashing the entire project.

### 4.2. Course Creator

The Course Creator's primary responsibilities is creating courses and viewing results of those courses. Courses are comprised of locations, questions, and answers; each course has many locations, each location has many questions, and each question has many answers.

#### 4.2.1. Authorization

The Course Creator uses ASP.NET Core Identity to secure the application. This tool makes creating the registration and login pages simple, and has built-in security for securely storing passwords and other personal user data. ASP.NET Core Identity also makes use of user roles to ensure the user is authorized to do certain actions. The verified user role is an example of this in the project. Using a filter on the protected controllers is simple with Identity.

##### Listing 2. Securing Controllers using Filter on User Role

```
1 using Microsoft.AspNetCore.Authorization;
2
3 [Authorize(Roles = "Verified")]
4 public class HomeController : Controller
5 {
6     ...
7 }
```

---

#### 4.2.2. Courses

The course is the high level object with which the locations, questions, and answers become associated with. A course can be added, updated, or deleted. The `CoursesController` handles all of these actions. Each of these actions has a corresponding View Razor page. The basic course model looks like the following:

##### Listing 3. Course Model

```
1 public class Course
2 {
3     public int CourseId { get; set; }
4
5     [DisplayName("Name")]
6     public string Name { get; set; }
7
8 }
```

---

The create and edit pages are similar to the other pages for the locations, questions, and answers, so this section will not go in depth on them, but instead elaborate further on the details and delete pages. The details page displays not only the course model information, but also all information relating to the course. To make the related course information modularize and reusable, the developer made use of “View Components”. View Components render a chunk of HTML output within another markup’s file. This breaks up large markup files into smaller parts, reduces duplication of markup content, and provides an opportunity to use logic to control the rendered HTML. Each of the dependent classes on a course has a corresponding View Component which lists the data relating to the course. This can be seen in the course details page:

#### Listing 4. Course Details Razor Page

```
1  @model ImmersiveQuiz.Models.Course
2
3  @{
4  ViewData["Title"] = "Details";
5  }
6
7  <h1>Details</h1>
8
9  <div>
10     <h4>Course</h4>
11     <hr />
12     <dl class="row">
13         <dt class="col-sm-2">
14             @Html.DisplayNameFor(model => model.Name)
15         </dt>
16         <dd class="col-sm-10">
17             @Html.DisplayFor(model => model.Name)
18         </dd>
19     </dl>
20 </div>
21 <div class="mb-4">
22     <a class="badge large-badge bg-secondary text-light" asp-action="
23         Index">Back to List</a> |
24     <a class="badge large-badge bg-success text-light" asp-controller="
25         Scores" asp-action="Index" asp-route-id="@Model.CourseId">Scores
26         </a>
27 </div>
28 <div>
29     <a class="badge large-badge bg-success text-light" asp-action="
30         Create" asp-controller="Locations" asp-route-id="@Model.CourseId
31         ">Add Location</a> |
32     <a class="badge large-badge bg-primary text-light" asp-action="Edit
33         " asp-route-id="@Model.CourseId">Edit</a>
34     <a class="badge large-badge bg-danger text-light" asp-action="
35         Delete" asp-route-id="@Model.CourseId">Delete</a>
36 </div>
```

```
30 @await Component.InvokeAsync("LocationList", new { CourseId = Model.  
    CourseId.ToString(), search = "" })
```

---

#### **4.2.3. Locations**

#### **4.2.4. Questions**

#### **4.2.5. Answers**

#### **4.2.6. Course Results**

#### **4.2.7. Web API**

### **4.3. Virtual Reality Orienteering**

The Virtual Reality Orienteering primary responsibilities are to display the course and to keep track and submit the results. The Virtual Reality Orienteering program must have authorized communication with the Course Creator program.

#### **4.3.1. Authorization**

#### **4.3.2. Displaying Course**

#### **4.3.3. Tracking and Submitting Score Results**

## **5. Testing**

### **5.1. Overview**

This gives a brief overview of this section.

### **5.2. Point 1**

This subsection gives a great deal of precise description supporting point 1. For example,

### **5.3. Point 2**

This gives Point 2

### **5.4. Point 3**

This gives Point 3

## **6. Conclusion**

### **6.1. Overview**

This gives a brief overview of this section.

### **6.2. Point 1**

This subsection gives a great deal of precise description supporting point 1. For example,

### **6.3. Point 2**

This gives Point 2



An article [8]  
A book [9]  
A series [10]  
Someone's thesis [11]  
Some technical report [12]  
A collection [13]  
Visited website [14]  
Accepted for publication [15]  
Submitted for publication [16]  
Not published [17]  
Conversation [18]

# Bibliography

- [1] “Orienteering,” [Online]. Available: <https://www.merriam-webster.com/dictionary/orienteering/>.
- [2] L. Hoory, “What is waterfall methodology? here’s how it can help your project management strategy,” 25-Mar-2022. [Online]. Available: <https://www.forbes.com/advisor/business/what-is-waterfall-methodology/>.
- [3] “Overview of asp.net core mvc,” [Online]. Available: <https://docs.microsoft.com/en-us/aspnet/core/mvc/overview?view=aspnetcore-6.0>.
- [4] “Overview of asp.net core,” [Online]. Available: <https://docs.microsoft.com/en-us/aspnet/core/introduction-to-aspnet-core?view=aspnetcore-6.0>.
- [5] “Common design principles,” [Online]. Available: <https://docs.microsoft.com/en-us/dotnet/architecture/modern-web-apps-azure/architectural-principles#separation-of-concerns>.
- [6] “Dependency injection in asp.net core,” [Online]. Available: <https://docs.microsoft.com/en-us/aspnet/core/fundamentals/dependency-injection?view=aspnetcore-6.0>.
- [7] “Gaming poised to continue accelerated growth according to unity gaming report 2022,” [Online]. Available: <https://unity.com/our-company/newsroom/gaming-poised-continue-accelerated-growth-according-unity-gaming-report-2022>.
- [8] A. B. Cummings, D. Eftekhary, and F. G. House, “The accurate determination of college students’ coefficients of friction,” *Journal of Sketchy Physics*, vol. 13, no. 2, pp. 46–129, 2003.

- [9] I. J. Kuss, *On the Importance of Kissing Up to Your Boss*, 5th ed. Cambridge MA: Dilbert Books, 1995.
- [10] L. M. Napster, *Mathematical Theory of Efficient Piracy*, ser. Lecture Notes in Mathematics. New York NY: Springer Verlag, 1998, vol. 3204.
- [11] O. P. Qwerty, “History of the goofy layout of keyboards,” Ph.D. dissertation, Podunk IN, 1996.
- [12] R. Swearingen, “Morpholoty and syntax of british sailors’ english,” New York NY, Tech. Rep., 1985.
- [13] T. Upsilon, “Obscure greek letters and their meanings in mathematics and the sciences,” in *Proceedings of the seventh international trivia conference*, V. W. Xavier, Ed. Philadelphia PA: Last Resort Publishers, 1987, pp. 129–158.
- [14] J. Tetazoo, “A brief guide to recreational pyromania,” Available at <http://www.blowinglotsofweirdstuffup.com/guide.html> (2005/06/12).
- [15] J. Mentor, “Behavior of small animals on fire,” (in press).
- [16] —, “Behavior of small animals on fire,” 2012, unpublished Manuscript.
- [17] —, “Behavior of small animals on fire,” 2012, unpublished Manuscript.
- [18] S. Freud, Personal conversation, July 2012.