



ScrumPoint

World Cup Tracker

TABLE OF CONTENTS

PAGE NO.	PAGE DESCRIPTION
1	Cover Page
2	Table of Contents
3	The Purpose of the Project
4	The Scope of the Project
5	Functional Requirements
6	Non-functional Requirements
7	External Interface Requirements
8	Systems and Architectures
10	Database Design
11	Frontend User Interface Design
17	Components Design

THE PURPOSE OF THE PROJECT

It's 2023 and that means another year of the Rugby World Cup. Four years ago, the Springboks made our country proud and won the tournament and are now back defending their title. With the inclusion of APIs and the advancement of technology within the sporting world, Team ScrumPoint has made it their goal to unite rugby fans to keep up to date with the current season of the Rugby World Cup.

ScrumPoint is a Rugby World Cup web-based Tracker that you will be able to view all events of the World Cup and see the scores of previous games and the current game. Users of the app can be in different leagues with their friends, and they will be able to make predictions for every match which will be scored depending on how close their prediction was to the actual score, they can also choose a player to predict the man of the match.

The system will essentially bring together rugby fans from all over the world to view their countries performance as well as the performances of the other countries within the tournament. This is ScrumPoint, the Rugby World Cup Tracker.

THE SCOPE OF THE PROJECT

The scope of this project will include aspects such as timeline, budget, and milestones.

Timeline

The project will run through nine weeks of planning, designing, implementing, and development.

Week 1: Brain storming and documentation.

Week 2: Finalizing documentation.

Week 3: Project start.

Week 4 – Week 8: Project development.

Week 9: Bug fixes.

Final Day: Presentation.

Budget

The project will consist of a free budget and make use of free plans with APIs, Azure storage systems, and azure subscriptions to host and service the web-applications and other functions within the system.

FUNCTIONAL REQUIREMENTS

1. Countries – The system shall display a set of all countries that are taking part in the tournament.
2. Upcoming Matches – The system shall display a list of upcoming matches.
3. Previous Matches - The system shall display a list of previous matches.
4. Current Match – The system shall display the current match taking place if a match is occurring at the time of view and will update the scores in real time.
5. Standings Pools – The system shall display a list of the standings, such as pools (groups) and countries taking part within each pool.
6. Pools Countries – The system shall display the played number of games, the number of wins and losses, the percentage of wins, and the points per country within a specific pool.
7. Standings Stages – The system shall display a visual representation of the stages of the tournament following the beginning of the tournament, including stages such as quarter finals, semi-finals, and the final match, all being updated as the tournament progresses.
8. Predication Game – The system shall allow users and viewers to compete in a prediction game that will accept predicted winning teams and predicted scores by those teams within matches. Points will be awarded by how accurate these predictions are.
9. Prediction Game Leaderboard – The system shall display a leaderboard with the top 10 users for the tournament based on their total prediction score.
10. Prediction Game User Progress – The system shall display the users current progress with their points based on their participation and predications within the tournament.

NON-FUNCTIONAL REQUIREMENTS

1. Performance – The system shall respond to user requests within X seconds for all functionalities and handle concurrent user sessions without significant performance degradation.
2. Scalability – The system shall be designed to scale horizontally to accommodate an increase in the number of users and data volume as the tournament progresses.
3. Availability – The system shall be available 24/7 with scheduled maintenance windows, if any, communicated to users in advance.
4. Reliability – The system shall have a backup and disaster recovery plan to ensure data integrity and availability in case of system failures.
5. Security – The system shall implement user authentication and authorization mechanisms to ensure that only authorized users can access certain functionalities.
6. Usability – The user interface shall follow best practices for usability and accessibility to ensure that it is user-friendly and accessible to individuals with disabilities.
7. Maintainability – The system's codebase shall be well-documented and adhere to coding standards to facilitate future maintenance and enhancements.
8. Backup and Recovery – Regular backups of the system's data shall be performed and, and a tested recovery process shall be in place to restore the system in case of data loss or corruption.
9. Compliance with Third-Party Services – The system shall adhere to the third-party API providers terms of use and service-level agreements.
10. Cloud Integration – The system shall implement as many features as possible using cloud services for extra functionality, performance, security, and resilience.

EXTERNAL INTERFACE REQUIREMENTS

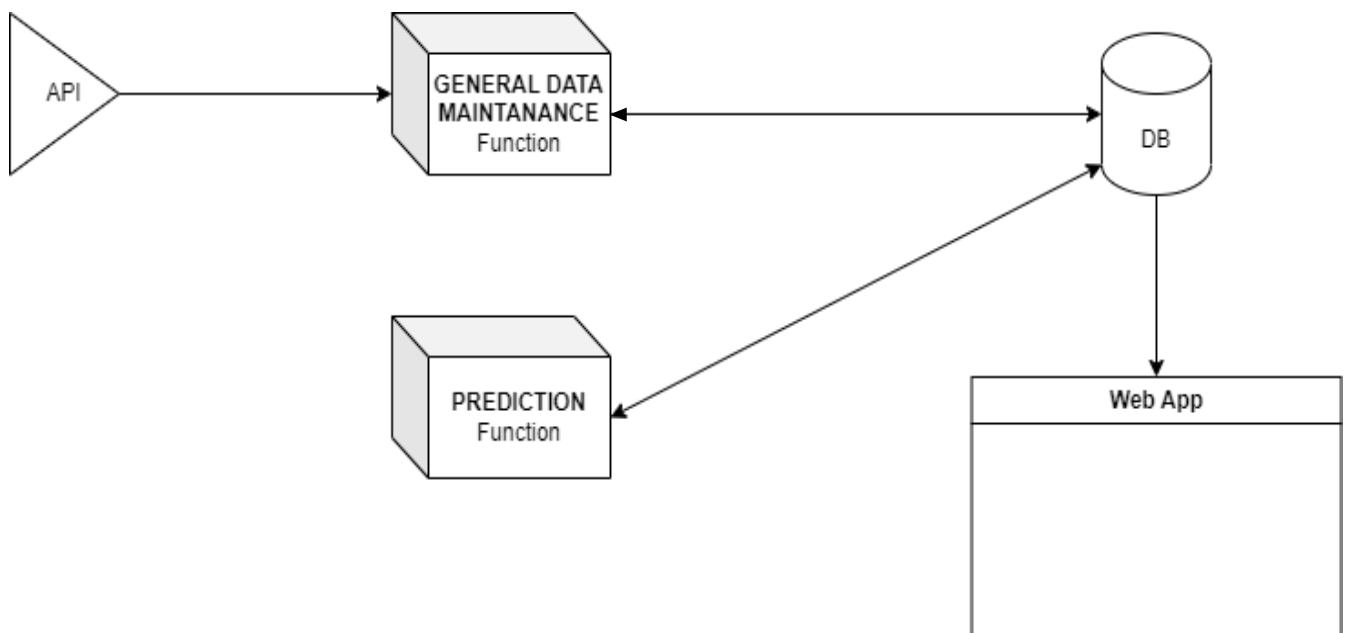
1. The system will use the RUGBY API for all information about the tournament and will adhere to the documentation and rights defined by the API SPORTS providers.
2. The system will use resources from Microsoft Azure for storage, backup, and other cloud supported services.

SYSTEMS AND ARCHITECTURES

For the complete system to work as intended, many different sub systems and architectures need to be used and work together coherently to create a seamless experience for the user. These are the core systems and architectures associated with ScrumPoint.

Below is a simplified diagram of the structure of the application and how services will be connected, and data will be transferred.

Below that is a table of the system, architecture, and software being used for each main functionality that the complete system will provide.

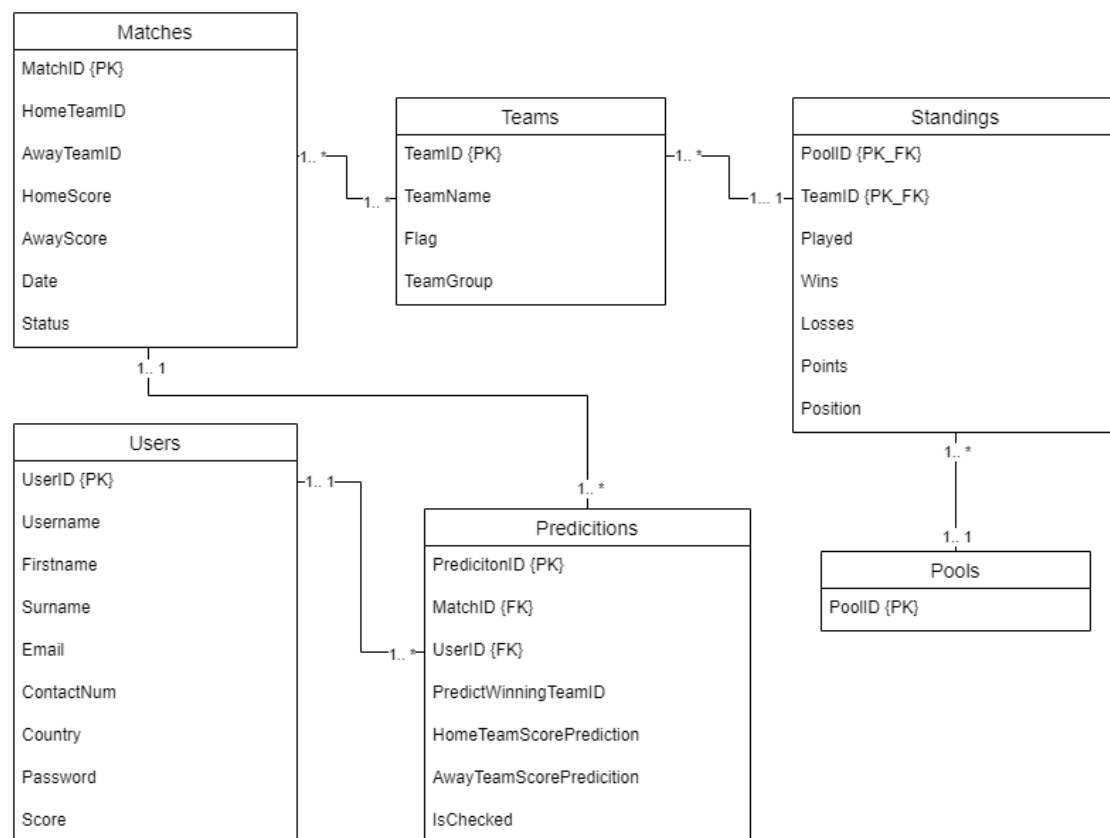


System	Architecture	Software
Entire Application	Monolithic / Microservice	N/A
Acquiring data for connected services	RESTful API	API RUGBY
Data storage	SQL database	SSMS
Cloud Storage for resilience and backup	Azure SQL database PaaS	Azure SQL Database connected to SSMS service
Website front-end	Static Front-End Website	Next.js with Visual Studio Code
General Data Maintenance Function	Serverless event driven	Azure Timer Trigger Function Application, Visual Studio
Prediction Function	Serverless event driven	Azure Queue Trigger Function Application, Visual Studio

DATABASE DESIGN

The SQL Server Management Studio ScrumPoint database will make use of the following UML ERD.

SCRUMPOINT ERD



FRONTEND USER INTERFACE DESIGN

The prototype design for each type of page within the website. Pages are subject to change; however, this is the main idea for the look and feel of ScrumPoint.

Sign-Up



The Sign-Up form is a white rounded rectangle centered on a background image of a rugby player running with the ball on a field at night. The form contains the title 'Sign-Up' in large bold text, the ScrumPoint logo (a football icon with the text 'ScrumPoint World Cup Tracker'), and three input fields for 'Email' (with placeholder 'email@gmail.com'), 'Phone' (with placeholder '+27 9999-9999'), and 'Password' (with placeholder 'P@ssw0rd'). An orange 'Sign-up' button is at the bottom right.

Sign-Up

Email
email@gmail.com

Phone
+27 9999-9999

Password
P@ssw0rd

Sign-up

Sign-In



The Sign-In form is a white rounded rectangle centered on the same background image as the Sign-Up form. It contains the title 'Sign-in' in large bold text, the ScrumPoint logo, and two input fields for 'Username or Email' (with placeholder 'email@gmail.com') and 'Password' (with placeholder 'P@ssw0rd'). Below the password field is a 'Remember Me' checkbox with an orange circle. An orange 'Sign-in' button is at the bottom right. A link 'Don't have an account Sign-up' is at the bottom. A green brushstroke graphic is on the left side of the form.

Sign-in

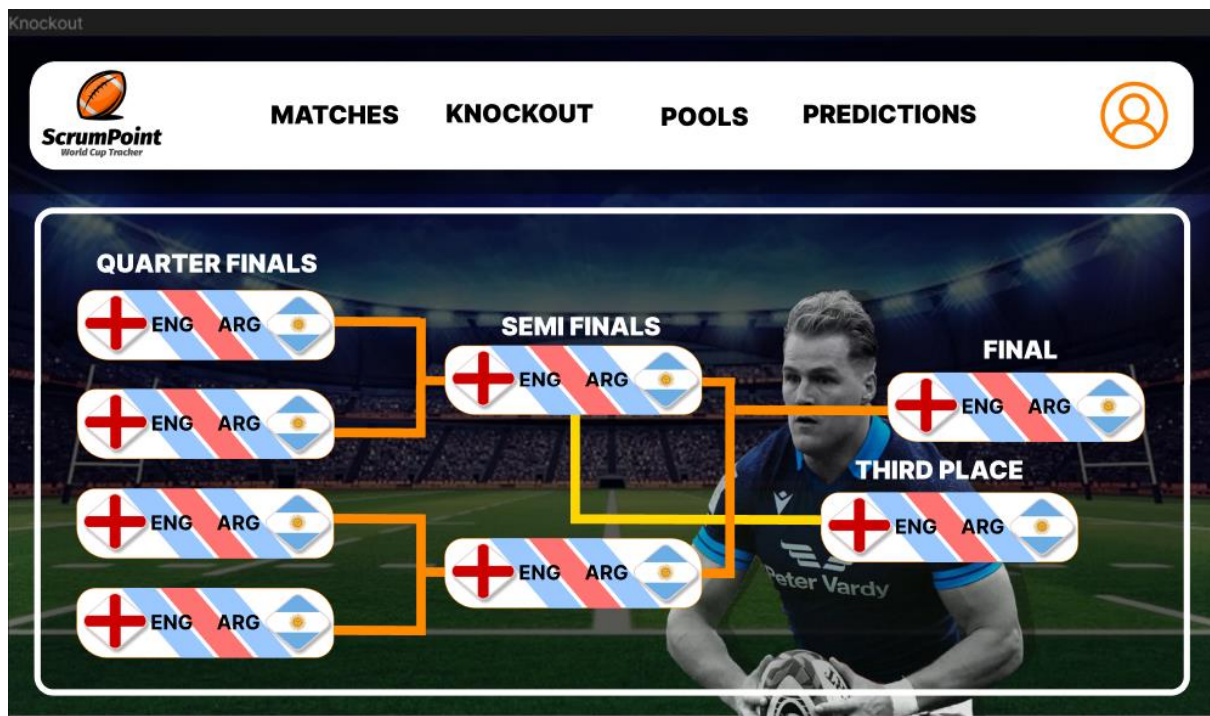
Username or Email
email@gmail.com

Password
P@ssw0rd

☐ Remember Me

Sign-in

Don't have an account [Sign-up](#)



Pools

ScrumPoint
World Cup Tracker

MATCHES KNOCKOUT POOLS PREDICTIONS






POOL A		POOL B		POOL C		POOL D	
COUNTRY				PLAYED	W	L	POINTS
	ENGLAND			4	4	0	18
	ENGLAND			4	4	0	18
	ENGLAND			4	4	0	18
	ENGLAND			4	4	0	18
	ENGLAND			4	4	0	18

Pools

ScrumPoint
World Cup Tracker

MATCHES KNOCKOUT POOLS PREDICTIONS

POOL A POOL B POOL C POOL D






COUNTRY	PLAYED	W	L	POINTS
 ENGLAND	4	4	0	18
 ENGLAND	4	4	0	18
 ENGLAND	4	4	0	18
 ENGLAND	4	4	0	18
 ENGLAND	4	4	0	18

Pools

ScrumPoint
World Cup Tracker

MATCHES KNOCKOUT POOLS PREDICTIONS

POOL A POOL B POOL C POOL D

COUNTRY	PLAYED	W	L	POINTS
 ENGLAND	4	4	0	18
 ENGLAND	4	4	0	18
 ENGLAND	4	4	0	18
 ENGLAND	4	4	0	18
 ENGLAND	4	4	0	18



MATCHES

KNOCKOUT

POOLS

PREDICTIONS



POOL A

POOL B

POOL C

POOL D

	COUNTRY	PLAYED	W	L	POINTS
	ENGLAND	4	4	0	18
	ENGLAND	4	4	0	18
	ENGLAND	4	4	0	18
	ENGLAND	4	4	0	18
	ENGLAND	4	4	0	18



MATCHES

KNOCKOUT

POOLS


PREDICTIONS




LEADER BOARD

	USER	CORRECT GUESSES	POINTS
1ST	username	5	18
2ND	username	5	18
3RD	username	5	18
4TH	username	5	18
5TH	username	5	18
6TH	username	5	18


Predictions




MATCHES KNOCKOUT POOLS **PREDICTIONS**



PREDICTIONS

 ENG

Saturday
14 October
17:00 pm

FRA 

☐

Predict Winner

☐

0

Predict Score

0

Save

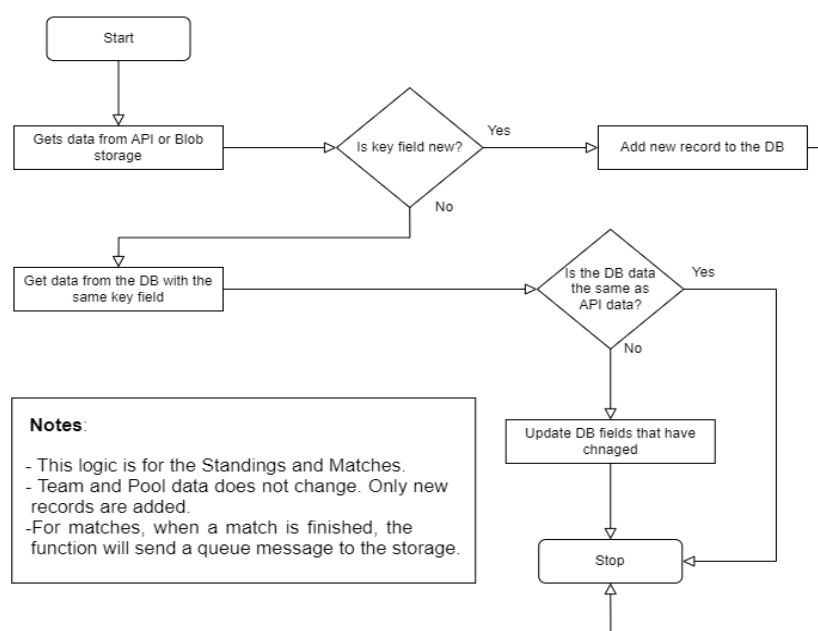
COMPONENTS DESIGN

General Maintenance

The General Maintenance logic is designed to periodically update and maintain data related to rugby teams, matches, pools, and standings. The process begins with the initialization of connection strings, where the application retrieves sensitive information like database credentials and API keys from secure storage, such as user secrets or environment variables. Subsequently, the Data Collection function is executed on a scheduled basis, triggered by a timer that runs every hour. Within this function, the application fetches the latest information from an external API regarding rugby teams and matches for a specified season. The obtained data is then processed, and relevant details are inserted or updated in the Azure SQL database. This includes actions such as inserting new teams, updating match results, and managing pool and standings data. Additionally, the system sends information about finished matches to a prediction queue, enhancing functionality beyond data storage. Overall, the General Maintenance process ensures that the application's database reflects the most current and accurate information available, supporting real-time insights and predictions for rugby events.

General Maintenance

Basic Logic



Prediction System

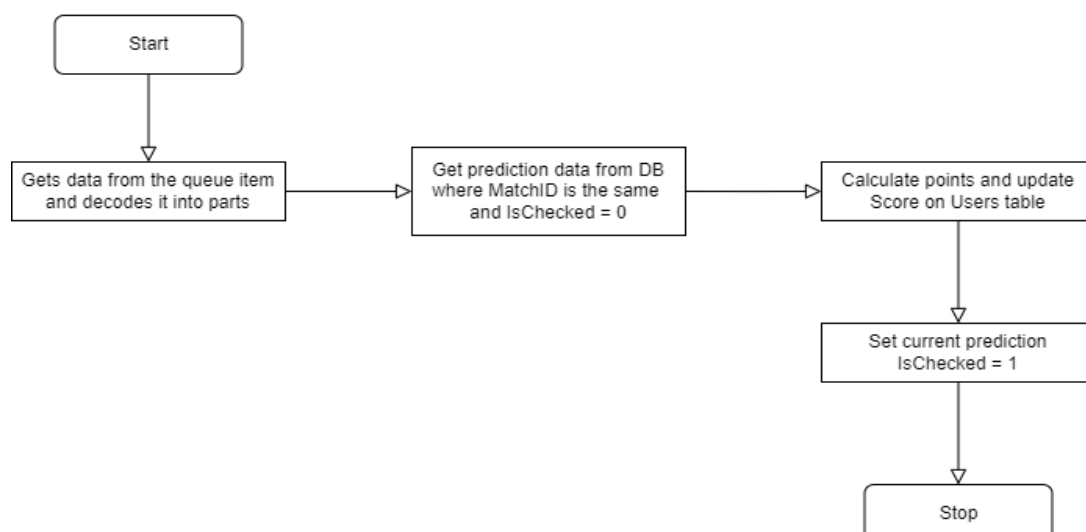
The Prediction System is designed to handle the asynchronous processing of prediction updates based on finished rugby matches. Triggered by messages in the Azure Storage Queue, the function extracts relevant information from the message, including the match ID, winning team, winning team's score, and losing team's score. It then connects to the Azure SQL Database using the provided connection string and retrieves predictions from the "Predictions" table that match the specified criteria.

For each valid prediction, the function calculates the user's points using a custom point calculation logic, considering factors such as the predicted and actual winning teams and their respective scores. The calculated points are then used to update the user's score in the "Users" table and mark the corresponding prediction as checked in the "Predictions" table, preventing redundant processing.

The function demonstrates robust database interaction, utilizing asynchronous commands to update user scores and prediction statuses. Additionally, it incorporates a well-defined point calculation method to fairly assess user predictions, providing an accurate reflection of their forecasting accuracy. This cohesive and efficient process ensures the seamless functioning of the Prediction System, contributing to a comprehensive and responsive website.

Prediction System

Basic Logic



OBJECTIVES, CHOICES, AND LIMITATIONS

For every effective website to work, objectives need to be met. These are the objectives that define the websites success:

Self-Healing and Redundancy / Resilience

Limitations: Due to the resource group of the system, using Varsity College's assigned group, a great self-healing method such as Application Insights was not granted as a permission to use on Azure's platform, thus we could not see our logs when our backend experienced errors, which could have resulted in longer periods to fix bugs and greater confusion with errors if we did not do anything to solve this issue.

Overcoming the limitations: To overcome the limitation found within this objective, the function applications were thoroughly tested locally making use of the logging system within the console app. These components were tested to guarantee that they worked so that when published to the cloud, no errors were encountered. Sometimes errors did arise, even on the cloud, but we then resorted to using database queries to see when the functions did work, and when they didn't, which helped us in debugging the problems experienced.

Choices: Zone redundant storage options have been used as a way to ensure that if components of the system break down at its main site, Azure will automatically route to another site in the zone to keep the system up and running without fail.

Unfortunately, due to Varsity College account permissions, only locally redundant storage has been chosen for the database and storage account static website hosted on the blob storage with the queues and blob containers, while the function apps do not use any redundant storage due to the free tier plan. This poses hardly an issue though, as the website is static, meaning it is less prone to attacks due to its client-side rendering without a server. Azure is incredibly well known for its robust systems and great component management, so the functions are reasonably safe with no redundancy, and if the database or storage account fail, there are still 2 more sites that they can be redirected to ensure continuous processing. The best plan of action in a business environment would obviously be spend slightly more on resources for more redundant sites.

Partitioning and Operations

Limitations: N/A

Choices: When it came to storing our data, the obvious choice was a database. The system makes use of an API for data collection and get request inputs, but with the free API subscribed plan, using this as a means for showing data to users would be a costly and inefficient method. Every button clicked to view a page would potentially mean another API request for data that only really updates on the weekend from games and standings changes. Pair these excessive get requests with 150000 users for the tournament final and the total costs becomes exponentially high. This is why the database was chosen, so that data can be stored and manually retrieved from there, along with user predictions be possible within the system. Instead of each user making each individual request, the backend system would make 1 request per table and update the data accordingly and letting users see only the relevant information in those tables. This allows less over get requests, and efficient scalability since getting faster responses during an intense game just means shortening the time between each trigger within the timer trigger function. When it comes to the standings of the tournament, the blob storage is being used while hosted on the cloud to act as an API due to a standings get request return data error. This allows us to make more use of the cloud services along with fixing the unforeseen API issue.

Managed Services and Data Storage

Limitations: The original plan was to host on either Azure blob storage Static Website or Azure Static Websites, however due to Next.js build issues and limited resource permissions from the college resource group, we could not host on these platforms.

Overcoming the limitations: To overcome the limitation found within this objective, we decided to host on Vercel which is the best way to host Next.js static websites online that is free. If we had more resources, we would buy a domain and connect to Cloudflare for security and load balancing, although static websites in general are more secure due to less server-side processing that can either be SQL injected or server side scripted.