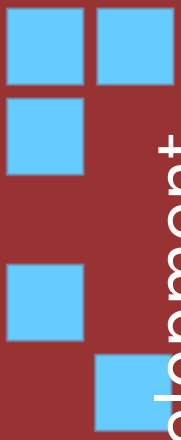

B.Sc. (Hons) in Software Development

Ollscoil Teicneolaíochta an Atlantaigh

Atlantic Technological University

Biometric Data Analysis in Digital Game Scenario

By
Rodrigo Almeida

January 23, 2024

Minor Dissertation

Department of Computer Science & Applied Physics,
School of Science & Computing,
Atlantic Technological University (ATU), Galway.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

PUBG: PUBG, short for PlayerUnknown's Battlegrounds, is a popular online multiplayer battle royale game developed and published by PUBG Corporation, a subsidiary of Bluehole Studio. In PUBG, up to 100 players parachute onto an island and scavenge for weapons and equipment to kill others while avoiding getting killed themselves. The game features a shrinking safe area to force players into close encounters, promoting tactical gameplay and intense firefights. The last player or team standing wins the game. PUBG became widely popular upon its release in 2017 and is available on various platforms, including PCs, consoles, and mobile devices1.[1].

Since its launch in 2017, PUBG has become incredibly popular, with more than 280M active players [2]. People of all ages and from all over the world enjoy playing this game. In PUBG, players need good hand-eye coordination and quick reflexes to compete well against others. The game offers a variety of weapons, and players can customize their controls to fit their preferences. This project builds upon the research done by university students, focusing on collecting biometric data to improve player performance in digital games [1].

Previous Project

The original project, titled **"Biometric Data Collection for Performance Optimization in a Digital Game Scenario"**, aimed to find out if a player's body data could help them play a video game better. The researchers set up a practice game similar to PUBG: Battlegrounds, with the same weapons and controls in a Unity Desktop `Application`. Biometric data was supplied by a `Smart Watch`. The goal was to see if there was a connection between how players performed in the game and their body data.

In our third-year project, our team was given the challenge of enhancing the visualization of data from both the test games and the Polar API. To achieve this,

we developed a chart API, enabling users to conveniently view all the results on a single page. The chart API was specifically created to present the data collected during the initial research in a coherent and user-friendly chart format, integrating various user data into one unified visual representation.

Project Objective

The aim of this research project is to overcome the limitations mentioned in previous projects and complete the future development goals of both projects. These include:

- Chart API Integration
  Integration of developed Chart API into the test application.
- Offline Data Storage
  Provision will be made for offline temporary file storage to improve the overall reliability of the whole system
- PUBG API
  Further research on new developments in the PUBG API for better user experience.

And eventually, seek to answer the following research questions:

- Can the user's current physical condition as indicated by their Biometric data, have any direct relationship with their performance in such a gaming scenario?
- Can Biometric and test data help suggest the most suitable settings for different game scenarios?

Chapter 2

Methodology

Describe the way you went about your project. Was your approach to the problem valid? You need to discuss both your software development methodology and your research methodology.

Chapter 3

Technology Review

3.1 Executive Dashboard

When it came to developing the Executive Dashboard for data analysis it was crucial to select the tools that could transform data into easily comprehensible information. After conducting research and exploring alternatives Angular and Chart.js were chosen for specific reasons.

3.1.1 Angular

Angular is based on TypeScript and is an open-source framework, and it's a great tool for building client-side web applications. It can be used for single-page applications (SPAs) or for enterprise-level solutions.

Main Concepts: Components, Modules and Services

Angular is composed of three foundational blocks: Components, Modules and services.

Components

Components are like Lego blocks of the application. Each component holds a portion of the user interface and its behaviour. Components serve as the bridge between the application data and what the user experiences on the screen.[3]

Modules

In Angular modules serve as containers that group related components, directives, and services together that can be combined with other modules. It plays an im-

portant role in improving maintainability and re-usability, key concepts of Angular development.[4]

Services

Angular services use typescript classes with injectable decorators. The decorator tells angular that the class is a service and can be injected into other components that need that service.[5]

3.1.2 Angular architecture

Angular uses the Model-View-Controller (MVC) pattern, with a variation known as Model-View-ViewModel (MVVM). The controller is responsible for the interaction between the model and the view.

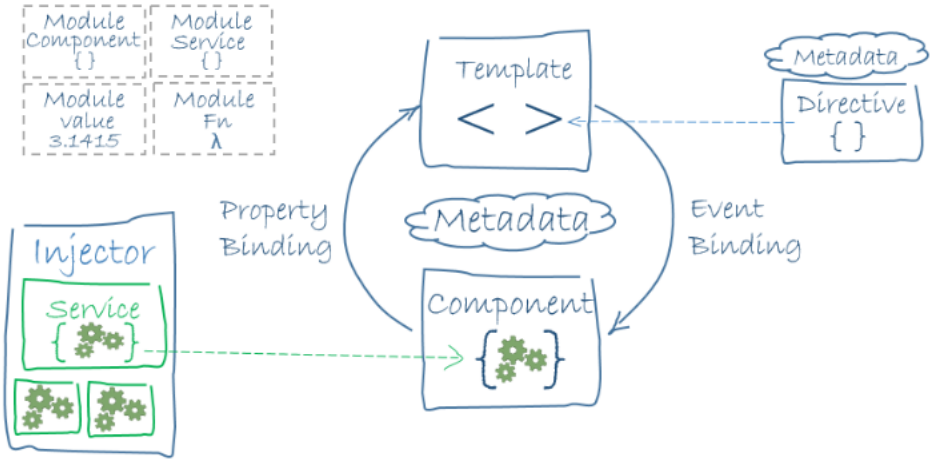

Figure 3.1: Angular application architecture

3.1.3 Advantages of using Angular

Component-Based Architecture

As mentioned earlier in discussions Angular organisess its functionalities into components. These components have the ability to communicate with each other enabling updates to sections without affecting the rest of the application.

Mobile-Friendly Approach

Angular incorporates techniques such, as lazy-loading, which means loading parts of the application (like images) only when they are needed. This ensures that users do not experience long waiting times.

Two-Way Data Binding

With Angular two way data binding data can seamlessly flow between the component and the view allowing for synchronization.

Asynchronous Programming

By utilizing programming executes code in a non-sequential manner and employs multi-threading to enhance performance. This speeds up operations and prevents system freezes, providing users with a seamless experience.

Single-Page Applications

Angular creates a dynamic single-page application which can be navigated without page reloads, improving the user experience with better user interaction and engagement.

Code Re-usability

The component-based architecture of Angular promotes the re-usability of UI components saving development time.

Dependency Injection

With dependency injection in place, Angular allows for the creation of objects that rely on other objects. This improves modularity and efficiency, within the app.

Angular Material

Angular's documentation offers a range of built user interface components and modules that adhere to Google's Material Design principles. This greatly facilitates the developer's work, simplifying the design process and enabling application development.

Angular CLI

Angular command line interface gives the developer the ability to generate Angular projects, modules, services, and components with a single command, this not only saves time but also reduces configuration errors, it gives the developer the freedom to dive into creative aspects of the project, focusing on innovation and functionality rather than getting bogged down by initial setup complexities.

Angular command line interface gives the developer the ability to generate Angular projects, modules, services, and components with a single command, this not only saves time but also reduces configuration errors, it gives the developer the freedom to dive into creative aspects of the project, focusing on innovation and functionality rather than getting bogged down by initial setup complexities.

3.2 Data Visualization

Data visualization is the process of translating large volumes of data into visual representations. It is a powerful tool that allows for the identification of patterns and trends that would otherwise be difficult to identify. It is a key component of data analysis and is used to communicate information clearly and efficiently.

3.2.1 Chart.js

Chart.js is robust, lightweight and open source javascript library that provides features for creating visually appealing charts and graphs that can be embedded into a web page. It is a great tool for data visualization and is easy to use. It is based on HTML5 canvas and is responsive, meaning that the charts will adapt to the size of the container element. It is also compatible with all modern browsers and is supported by a large community of developers. [6] It can be compared like a the artist's toolkit for developers, providing a flexible API to create a variety of chart types, such as line charts, radar charts, pie charts and many more. It also provides a range of customization options, allowing developers to create unique and visually appealing charts. [6]

Table 3.2.1 is showing some of the features of Chart.js[6]:

Feature	Description
Easy to use	Aesthetically charts can be created without the need of extensive configuration. The library follows a declarative approach allowing the developer to define the data and settings of the chart in a single object.
Responsive	Charts generated by Chart.js are responsive, adapting to different screen sizes and devices, ensuring that the visualization remains accessible and readable across various platforms.
Customization	Chart.js provides a high degree of customization. Colors, fonts, and other visual elements can be customized to create unique and visually appealing charts.
Interactivity	Chart.js provides built-in support for tooltips and animation, allowing the user to explore data points, adding a layer of engagement to the project.
Cross-browser compatibility	Chart.js is compatible with all modern browsers, including Chrome, Firefox, Safari, Edge, and Internet Explorer 11.

Table 3.1: Chart.js Features

Chart.js is a reliable and effective tool for data visualization, it is also easy to use and provides a range of customization options. It is also supported by a large community of developers, which is a great advantage.

3.2.2 D3.js

D3.js or Data-Driven Documents is a JavaScript library for manipulating documents based on data. It has the ability ti bind data to the DOM (Document Obect Model) and creates interactive and dynamic visualization. It provides a lower level and more granular approach to data visualization, giving the developer more control over the visualization process.[7]

Table 3.2.2 is showing some of the features of D3.js[7]:

Feature	Description
Data-Driven	D3.js is data-driven, meaning that it can bind data directly to HTML or SVG elements. This type of control is advantageous for creating highly customized visualizations.
Modular	D3.js is modular, composed of many small modules that can be used independently or combined together to create a custom visualization.
Extensible	D3.js is extensible, meaning that it can be extended to support any type of visualization.
Community	D3.js is supported by a large community of developers, which is a great advantage.

Table 3.2: D3.js Features

D3.js is a powerful tool for data visualization, it provides a high degree of customization and control over the visualization process. However, it is more complex and requires more time to learn and implement. It is also not supported by all browsers, which is a disadvantage.

In conclusion, while D3.js is a powerful tool for data visualization, it is more complex and requires more time to learn and implement. It is also not supported by all browsers, which is a disadvantage. Considering the scope of the project and the time constraints, Chart.js was chosen as the tool for data visualization. It is easy to use and provides a high degree of customization, allowing for the creation of unique and visually appealing charts. It is also supported by a large community of developers, which is a great advantage.

3.3 AI module - Data analysis

Ninety per cent of the world's data was generated in just the last two years. In the early 2000s, the amount of data being generated exploded exponentially with the use of the internet, social media, and various other technologies. Organizations found themselves facing a massive volume of data that was very hard to process. To address the challenge, the concept of Big data emerged. Big data refers to extremely large and complex data sets that are difficult to process using traditional methods. [8]

3.3.1 Hadoop

As the volume of data grew rapidly across the globe, organizations needed a way to manage it to gain valuable insights. That's where Hadoop came in. In 2006,

a team of engineers from Yahoo developed Hadoop inspired by Google's MapReduce. Hadoop has introduced a new way of handling data called distributed processing. Now, to process all data it wouldn't use single machines anymore, instead, Hadoop uses multiple computers, allowing large amounts of data to be processed way faster than before. Hadoop has mainly two main components: HDFS (Hadoop Distributed File System) and MapReduce. HDFS stores the into multiple computers, and MapReduce is responsible for processing the data in parallel allowing the organizations to store and process large amounts of data. Hadoop was a great advance in big data processing, however, it had a few limitations. One of the biggest problems was that it relied on storing data on disk. This slowed down data processing because every time a job ran it had to save its data to disk, read it back, process it, and save it back to disk Another problem is that Hadoop processes data only in batches. This means a new job couldn't be submitted before the other job ended. With Hadoop, storing and processing large amounts of data became possible, even with some limitations it was an important step in big data processing.[9]

3.3.2 Apache Spark

As described Hadoop has a few limitations, there was a need to process all this data faster and in real-time. And here is where Apache Spark comes into action. In 2009, researchers at the University of California developed Apache Spark as a research project. At this point, RDD (Resilient Distributed Dataset) was introduced and deemed to be a very powerful concept.[10] RDD is the backbone of Apache Spark, storing the data in memory for faster access and processing. It will not read and write the data from the disk, instead, Spark processes the entire data just in memory. The meaning of memory here is the RAM (Random Access Memory) stored inside the computer. This in-memory data processing of data makes Spark much faster than Hadoop. Spark also gives the ability to write code in various programming languages such as JAVA, Scala, Python and R, along with an optimized engine that supports general execution graphs.

Main components and Features

The Apache Spark ecosystem consists of the following main components[11]:

- Spark SQL: Formerly known as Shark. Spark SQL is a distributed framework that works with structured and semi-structured data. It facilitates analytical and interactive applications for both streaming and historical data which can be accessed from various sources such as JSON, Parquet and Hive table.[11]

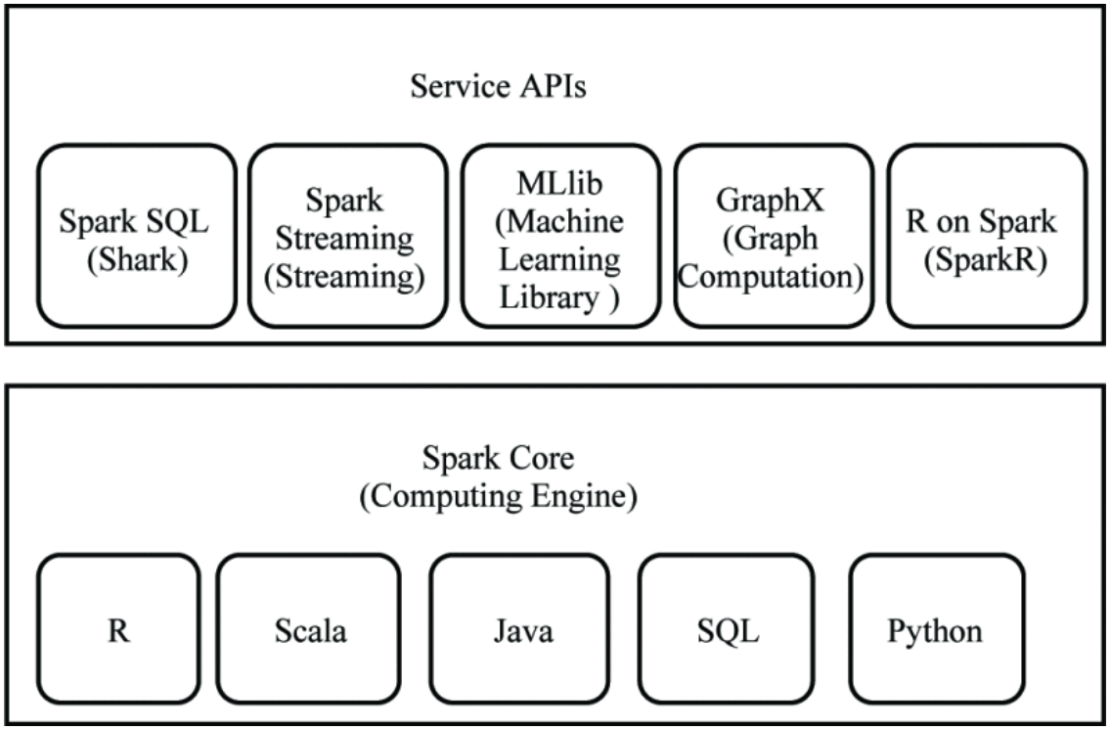

Figure 3.2: Apache Spark Ecosystem

- Spark Streaming: Allows to process real-time data. Is a scalable fault-tolerant streaming processing system that supports both batch and streaming workloads. Spark streaming enhances the fast scheduling capability of Apache Spark by inserting data into mini-batches. An operation known as transformation is then applied to those mini-batches that can be easily obtained from live streams and data sources such as Twitter, Apache Kafka, IoT sensors, and Amazon Kinesis.[11]

- Spark Core: One of the most important parts of the Spark ecosystem is called Spark Core. It helps process data across multiple computers and ensures everything works efficiently and smoothly. Various functionalities of Apache Spark are built on top of the Spark core. It provides a vast range of APIs as well as applications for programming languages such as Scala, Java, and Python APIs to facilitate the ease of development. In-memory computation is implemented in Spark core in order to deliver speed and solve the issue of MapReduce.

- MLlib: It delivers high-quality algorithms with high speed and makes ma-

chine learning easy to use and scale. Several machine learning algorithms such as regression, classification, clustering, and linear algebra are present. It also provides a library for lower-level machine learning primitives like the generic gradient descent optimization algorithm. It also provides other functions such as model evaluation and data import. It can be used in Java, Scala, and Python. [11]

With all these components working together, Apache Spark became a powerful tool for processing and analysing Big Data. Spark manages and coordinates the execution of tasks on data across a cluster of computers.

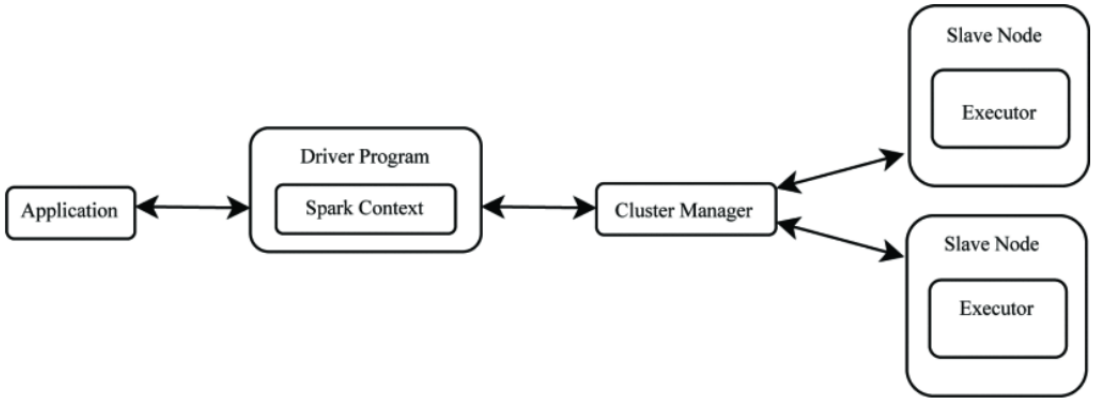

Figure 3.3: Apache Spark Architecture

Figure 3.3 shows the Spark Architecture. It consists of a master node which has a driver program that is responsible for calling the main program of an application, this driver could be the Spark shell or the application, written in Scala or Java, for example. It is basically the entry point and is responsible for creating Spark context, which behaves like a gateway to all of the functionalities of Apache Spark. It allows for communication and coordination with other nodes within the cluster, known as Slave nodes. Within the slave node, there are many numbers of executors which act as workers. Whenever a job is initiated, the driver process will make sure it goes through the Apache application properly. It analyzes the work that needs to be done, divides it into smaller tasks and assigns it to the executors. The driver process is the heart of the Apache Spark application and executors are the real workers, which execute the work assigned by the driver and report back the progress and results of the computation.[11]

Due to the powerful capabilities of distributed computing, large volumes of data can be processed quickly and efficiently. For all these reasons, it was deemed

to be the perfect technology to be used in this project. As the project evolves and more data is gathered from our research volunteers, Spark's performance scales with the size of the data, processing with high speed.

3.3.3 Node JS

NodeJS is an open-source, cross-platform, JavaScript runtime environment that executes JavaScript code outside of a browser. It is built on Chrome's V8 JavaScript engine and is used to build fast and scalable network applications. It is event-driven and non-blocking, meaning that it is capable of handling concurrent operations without blocking the execution of other operations. This makes it ideal for data-intensive real-time applications that run across distributed devices.[12] It has solidified its position as a cornerstone of modern web development, with a large community of developers and a vast ecosystem of open-source libraries. It is an asynchronous event-driven JavaScript runtime environment that is designed to build scalable network applications. It is lightweight and efficient, making it a great choice for data-intensive real-time applications.[13] One of the primary advantages of NodeJS is that it uses a single-threaded event loop, which allows it to handle concurrent operations without blocking the execution of other operations. However, NodeJS is not without its criticisms. Critics often point out the callback hell, a situation where the code becomes unreadable due to the excessive use of callbacks. Although it has been largely mitigated by the introduction of Promisses and async/await syntax, it remanins a valid criticism. [14] Another strength of NodeJS is its vast ecosystem of open-source libraries. The Node Package Manager (NPM) is the largest ecosystem of open-source libraries in the world, with over 1 million packages. [15] This wealthy of modules and libraries greatly facilitates the development process, allowing developers to focus on the creative aspects of the project, rather than reinventing the wheel. NodeJS also benefits from its single programming language across both the server and client sides. This allows for the sharing of code and data between the server and the client, which is a great advantage, reducing the learning curve associated with NodeJS development. [13] Performance wise NodeJS is very fast, it is built on Chrome's V8 JavaScript engine, which is a high-performance JavaScript engine. It is also lightweight and efficient, making it a great choice for data-intensive real-time applications. [13] Considering these noumerous advantages, NodeJS was chosen as the technology for the project. It is lightweight and efficient, making it a great choice for data-intensive real-time applications. It also benefits from its vast ecosystem of open-source libraries, which greatly facilitates the development process, allowing us to focus on the creative aspects of the project, rather than reinventing the wheel. It is also fast and scalable, which is a great advantage.

3.3.4 PM2

PM2, or Process Manager 2, is a production process manager for NodeJS applications. It is a feature-rich tool that provides a wide range of features, including monitoring, load balancing, and error handling. It is designed to simplify the deployment and management of NodeJS applications in production environments and to keep applications alive forever, restart them without downtime and simplify common system administration tasks. The standout feature of PM2 is its ability to keep processes running in the background indefinitely. This is particular important for web applications that requires constant availability. PM2 automatically resurrects crashed applications, ensuring that system glitches do not result in prolonged downtime.[16] This automatic restart capability is a safeguard against potentially costly application crashes that could otherwise lead to a poor user experience or even lost revenue. [16] PM2 also provides a built-in load balancer that can distribute incoming requests across multiple instances of the application. This improves performance and scalability, allowing the application to handle more requests, evenly distributing traffic across the instances, which can be particularly beneficial when running on multi-core systems. [17] However, PM2 is not one-size-fits-all solution. It is designed for NodeJS applications, and is not suitable for applications written in other languages. It is also not suitable for applications that require a high degree of customization. [16] Despite the considerations, PM2's benefits have outweighed its limitations, and it was chosen as the technology for the project. It is a feature-rich tool that provides a wide range of features, including monitoring, load balancing, and error handling. It is designed to simplify the deployment and management of NodeJS applications in production environments and to keep applications alive forever, restart them without downtime and simplify common system administration tasks. Its installation and setup process is straightforward, requiring minimal configuration. It is also lightweight and efficient, making it a great choice for this project. [13]

3.3.5 Docker

Docker has emerged as a revolutionary tool for software development. It is an open-source platform that allows developers to build, ship, and run applications in containers. It is a lightweight alternative to virtual machines, allowing for the isolation of applications and their dependencies into containers. Containers encapsulate the application's code, libraries, and dependencies in a single object, ensuring that the application runs reliably and consistently across different environments. Due to its simplicity and portability, Docker has become the de facto standard for containerization. It is supported by all major cloud providers, including Amazon Web Services, Microsoft Azure, and Google Cloud Platform. Its

lightweight nature compared to traditional virtual machines makes it ideal for cloud computing. Containers share the machine's OS kernel, and do not require an OS per application, which greatly reduces the memory footprint and improves performance. [18]

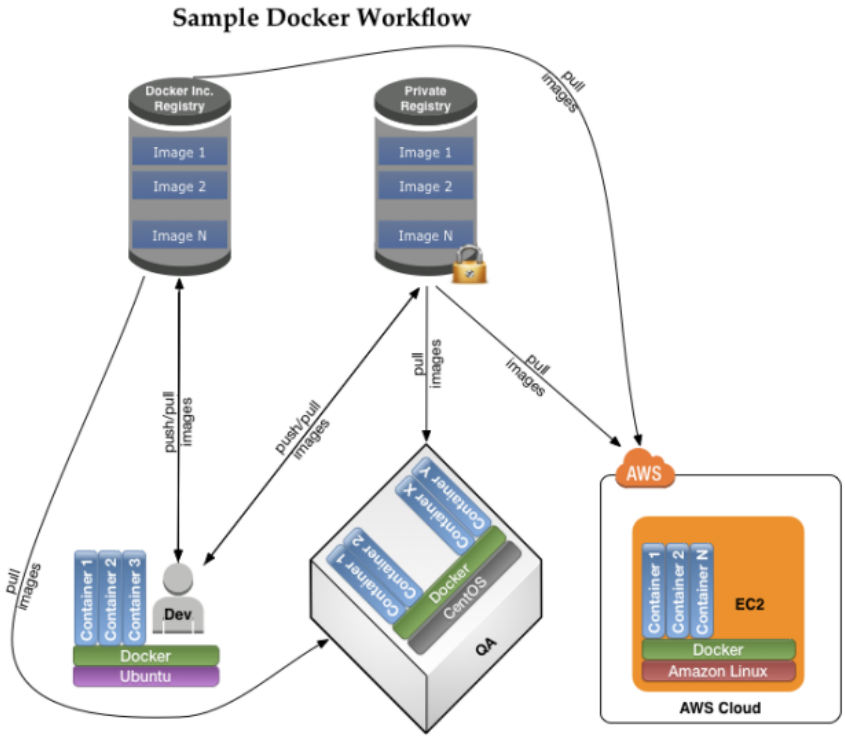

Figure 3.4: Development workflow with Docker

However, Docker's container model is not without challenges. In the context of this project, the average size of a single Docker image exceeding 400MB presents a concern regarding bandwidth utilization. Continous build and deployment processes involving large container images can consume substantial network resources, leading to potential cost overruns. [18] Another challenge is the security of Docker containers. Docker containers share the same kernel, which means that a vulnerability in the kernel can affect all containers. [19] While Docker's layering and image caching mechanisms can mitigate some of the network overhead, it is still a concern. In conclusion, Docker is a powerful tool that offers numerous benefits for application deployment, from development to production. However, it is not without its challenges. The large size of Docker images can lead to bandwidth utilization issues, and the shared kernel model can lead to security concerns. Considering the scope of the project, time constraints and budget, Docker wasn't chosen as the technology for the project as the cost would be too high.

3.3.6 NGINX

3.3.7 MySQL

3.3.8 Express JS

3.3.9 TensorFlow

3.3.10 AWS Services

Chapter 4

System Design

Provide a detailed explanation of the overall system architecture [**?**], i.e. the HOW of the project. Use UML, system architecture diagrams, screenshots, code snippets and algorithms to illustrate your design.

Legacy Architecture

The entire system architecture, as designed in the previous projects, is visualized in Figure 4.1. This system consists of three modules that interact with both an external API and user devices. These modules are detailed as follows:

- Test Application
  A Unity-designed desktop application on a Windows Platform where users can play tests for different game scenarios.
- Node Js Web Application
  An Express Node JS Web application deployed on Amazon EC2 Virtual machine, serving as a callback endpoint for OAuth 2.0 Authentication for the Polar Flow API.
- Firestore Data Storage
  Permanent storage medium for user's test and Biometric data.

User Devices:

- Smart Watch
- Smart Phone

The actions in the Figure Figure 4.1 can be summarized as follows:

- sync: passing of biometric data to a smartphone from a Polar watch.

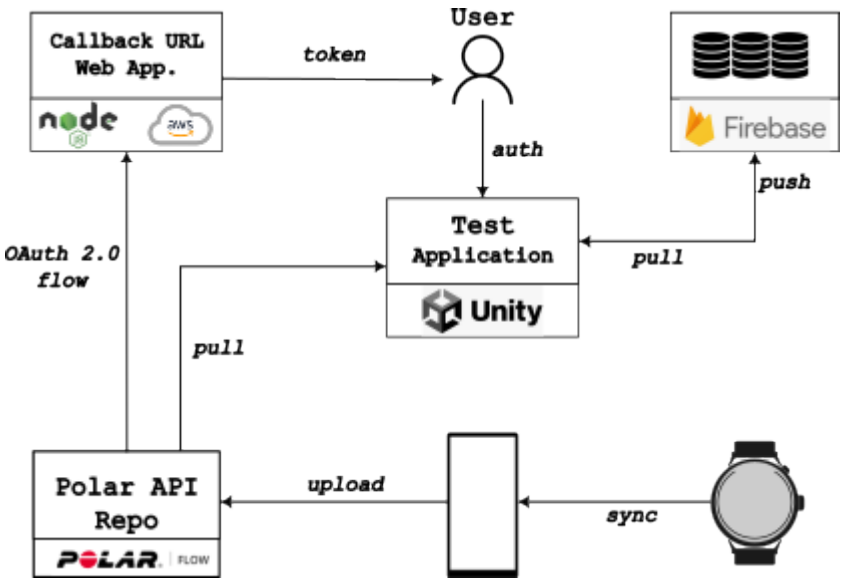

Figure 4.1: System Architecture

- upload: uploading of biometric data to Polar API repository.
- OAuth 2.0 flow: Multi step OAuth 2.0 authentication protocol.
- token: getting verification token from the Callback URL by the user.
- pull (Polar API - Test Application): retrieving biometric data from Polar API by the Test Application.
- auth: The user supplies a token to the Test Application as a final step of authentication.
- pull (Firebase - Test Application): retrieving user test/biometric data for rendering by Test Application.
- push: saving user's test/biometric data to a Datastore.

Proposed Architecture

The final system architecture will include new modules to facilitate data analytics using AI models. The new system calls for a relational database replica of the Firebase storage to ensure data consistency, predictability and structures suitable for relevant statistical analysis. A.I models as a mini-service that performs operations with supplied data and returns the result of the analysis. A Web Application that functions as a Controller that interfaces the data and A.I modules, Executive

Dashboard that renders user data and results from the A.I models and any other information about the system.
Figure 4.2 shows the proposed architectural layout for the System.

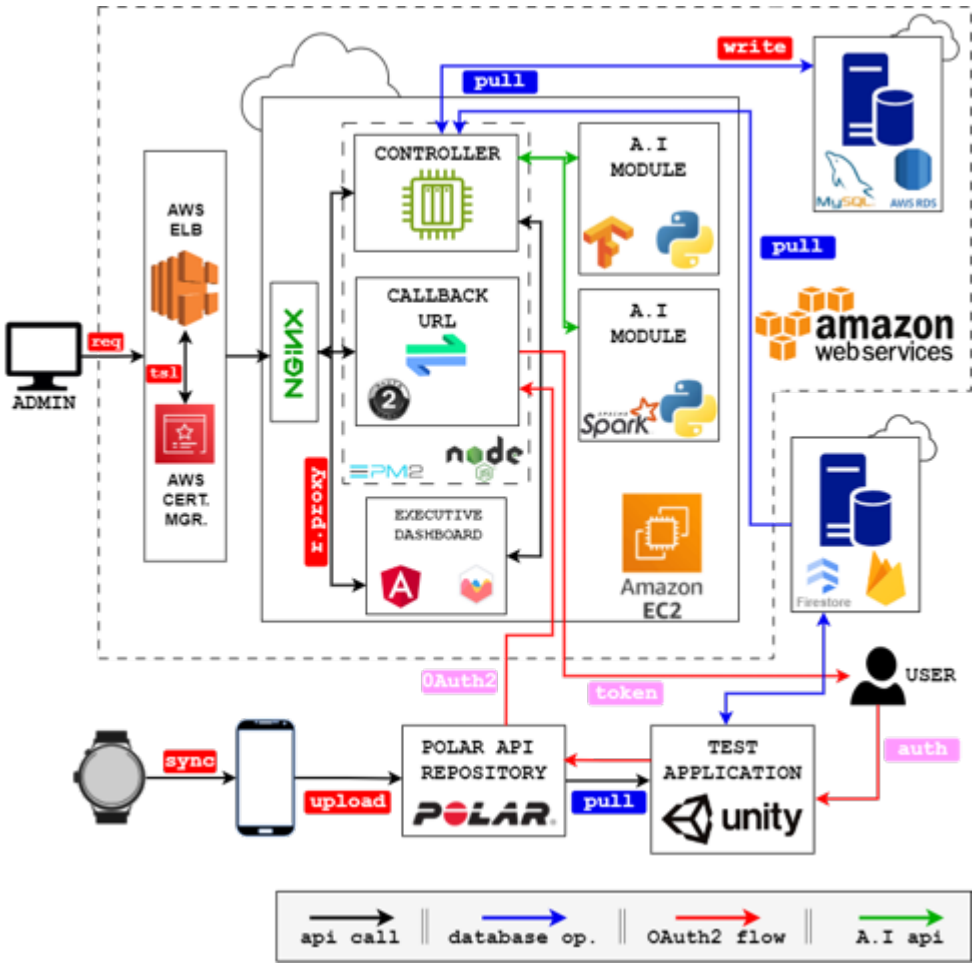

Figure 4.2: Web Application Architecture

Additional modules have been added to the web application namely back end 'Controller' & front end 'Executive Dashboard', and a 'Relational Database'. The functions of these modules are explained below:

- Controller: This module is responsible for coordinating data fetch from the Relational Database and making sure that the Relational Database is always in sync with the Firebase Datastore. It is also responsible for filtering and structuring data being fed into the A.I. Modules and when necessary using the output for an A.I module as input for another A.I module.

- Executive Dashboard: This module provides a Graphical User Interface that displays the working dataset, and offers interfaces to filter, tune and if needed bias the data being sent to the A.I Modules for analysis. This module will developed using the Angular Framework with integration of D3.js or Chart.js for displaying relevant charts. The choice for Angular Framework over other solutions namely ReactJS and Vue is because of the more structured Angular Framework which is more suited for collaborative work.

- A.I Modules: These modules are responsible for running analysis on data. The modules accept input from the 'Controller' and output a comprehensive summary of the results of the analysis done to the 'Executive Dashboard' module. For this purpose, TensorFlow and Apache Spark were chosen for the following reasons:

- Capabilities: Apache Spark is capable of performing linear and logical regression[20] which are common tools used for this class of problem. TensorFlow also has the same capabilities and can perform regression neural network models[21].

- Accessibility: Both tools are open-source and available to the general public[22].

- Python API interface: Both tools provide a Python API. This eases deployment as Python is platform-independent and relatively easy setup in a Ubuntu Virtual machine.

- Large Community: Both have a large community of contributors. Documentation and tutorials are vastly available on the internet.

- Relational Database: Data for the analysis will be sourced directly from this Database which is a structured replica of the FireStore Database as obtained from the Legacy System. The rationale for this Module is to provide a flexible means of retrieving data in various forms and shapes as may be needed for analysis and also overcome the high complexities involved in FireStore for compound queries.

- Chart API: Is responsible for displaying test results and biometric data to the users. This already developed module will be integrated into the 'Test Application' and undergo some modification to be able to display custom data as requirements change.

Chapter 5

System Evaluation

Evaluate your project against the objectives set out in the introduction. This chapter should present results if applicable and discuss the strengths and weaknesses of your system. This is a clear opportunity for you to demonstrate your critical thinking in relation to the project.

5.1 Working with Tables

Table 5.1 can be referenced with the label given to the table, i.e. **ref{table:HexToBin}**. Note that LaTeXwill place the table wherever it deems fit. Don't bother trying to change where a table or figure is placed until your document is ready for final layout.

Hexadecimal to Binary					
Hex	Binary 2	Hex	Binary	Hex	Binary
1	00000001	B	00001011	15	00010101
2	00000010	C	00001100	16	00010110
3	00000011	D	00001101	17	00010111
4	00000100	E	00001110	18	00011000
5	00000101	F	00001111	19	00011001
6	00000110	10	00010000	1A	00011010
7	00000111	11	00010001	1B	00011011
8	00001000	12	00010010	1C	00011100
9	00001001	13	00010011	1D	00011101
A	00001010	14	00010100	1E	00011110

Table 5.1: Conversion from Hexadecimal to Binary

Chapter 6

Conclusion

Briefly summarise your context and objectives. Remind the reader about the overall rationale and goals of the project. Highlight your findings from the System Evaluation chapter.

Bibliography

[1] https://en.wikipedia.org/wiki/PUBG:_Battlegrounds. Wikipedia, "PlayerUnknown's Battlegrounds.", Accessed: 12 October 2023.

[2] https://activeplayer.io/pubg/. Active Players, Accessed: 13 October 2023.

[3] Angular Components.

[4] Angular Modules.

[5] Angular Services.

[6] Helder Da Rocha. *Learn Chart. js: Create interactive visualizations for the web with chart. js 2*. Packt Publishing Ltd, 2019.

[7] Nick Qi Zhu. *Data visualization with D3. js cookbook*. Packt Publishing Ltd, 2013.

[8] News. *Significance*, 9(4):2–3, 08 2012.

[9] Ivanilton Polato, Reginaldo Ré, Alfredo Goldman, and Fabio Kon. A comprehensive view of hadoop research—a systematic literature review. *Journal of Network and Computer Applications*, 46:1–25, 2014.

[10] Salman Salloum, Ruslan Dautov, Xiaojun Chen, Patrick Xiaogang Peng, and Joshua Zhexue Huang. Big data analytics on apache spark. *International Journal of Data Science and Analytics*, 1:145–164, 2016.

[11] Eman Shaikh, Iman Mohiuddin, Yasmeen Alufaisan, and Irum Nahvi. Apache spark: A big data processing engine. In *2019 2nd IEEE Middle East and North Africa COMMunications Conference (MENACOMM)*, pages 1–6, 2019.

[12] https://nodejs.org/en/. Node.js, Accessed: 23 January 2024.

[13] Stefan Tilkov and Steve Vinoski. Node.js: Using javascript to build high-performance network programs. *IEEE Internet Computing*, 14(6):80–83, 2010.

[14] Mike Cantelon, Marc Harter, TJ Holowaychuk, and Nathan Rajlich. *Node. js in Action*. Manning Greenwich, 2014.

[15] Inc. (2021) npm. npm public registry, 2021.

[16] David Gonzalez. *Developing Microservices with node. js*. Packt Publishing Birmingham, UK, 2016.

[17] Olivier Aumage, Gabriel Antoniu, Luc Bougé, Vincent Danjean, and Raymond Namyst. Getting started with pm2. *LIP, ENS-Lyon*, 2001.

[18] Dirk Merkel et al. Docker: lightweight linux containers for consistent development and deployment. *Linux j*, 239(2):2, 2014.

[19] Inc. Docker. Docker hub, 2021.

[20] `https://spark.apache.org/docs/latest/ml-classification-regression.html`. Apache Spark, Accessed: 14 October 2023.

[21] Jeff Heaton, Steven McElwee, James Fraley, and James Cannady. Early stabilizing feature importance for tensorflow deep neural networks. In *2017 International Joint Conference on Neural Networks (IJCNN)*, pages 4618–4624, 2017.

[22] `https://www.databricks.com/blog/2016/01/25/deep-learning-with-apache-spark-and-tensorflow.html`. Apache Spark, Accessed: 11 October 2023.