

# Video Game Emulation for the Consumer

**Samuel Mocock**

978082

May 2022

## **Abstract**

The excitement and nostalgia associated with video games played during childhood is, to many people, an unmatched experience. Video game emulation is becoming increasingly popular as shown by large companies, such as Xbox and Nintendo, investing resources into making their classic games playable. With so many options for emulation now available, it becomes tricky for users to choose the one that best fits their criteria. Due to this, I created a mobile application that uses an easily accessible questionnaire in order to present the user with the option that might best suit them, whilst also providing them with the opportunity to learn about other possible options for emulation. Overall, this project has been relatively successful, and this body of writing explores the process in further detail.

Submitted to Swansea University in fulfilment  
of the requirements for the Degree of Bachelor of Science



**Swansea University**  
**Prifysgol Abertawe**

Department of Computer Science  
Swansea University

# Declaration

This work has not been previously accepted in substance for any degree and is not being concurrently submitted in candidature for any degree.

Signed: Samuel Moccock

Date: 03/05/2022

# Statement 1

This thesis is the result of my own investigations, except where otherwise stated. Other sources are acknowledged by footnotes giving explicit references. A bibliography is appended.

Signed: Samuel Moccock

Date: 03/05/2022

# Statement 2

I hereby give my consent for my thesis, if accepted, to be made available for photocopying and inter-library loan, and for the title and summary to be made available to outside organisations.

Signed: Samuel Moccock

Date: 03/05/2022

## Table of Contents

1	Introduction .....	3
1.1	Motivation.....	3
1.2	Aims.....	3
2	Background .....	4
2.1	History of Video Game Consoles, Backward Compatibility, and Console-Based Emulation ..	4
2.1.1	Console Generations .....	4
2.2.2	Atari.....	5
2.2.3	Nintendo .....	5
2.2.4	PlayStation .....	7
2.2.5	Xbox.....	8
2.2	Unlicensed Emulators .....	9
2.3	Emulator Specification Analysis .....	10
3	Project Management .....	11
3.1	Timeline Analysis.....	11
3.2	Project Aims Analysis .....	11
3.3	Risk Analysis .....	12
4	Implementation .....	13
4.1	Stage One.....	13
4.2	Stage Two.....	13
4.3	Stage Three .....	13
4.4	Stage Four .....	14
4.5	Stage Five .....	16
5	Results .....	16
6	Conclusion.....	18
	Bibliography .....	20

# 1 Introduction

## 1.1 Motivation

The video game industry is one that is rapidly expanding with thousands of games being released each year across multiple platforms. The number of games that are actually worth playing is up for debate, but there are certainly many existing video games that are worth the average consumer's time. The technology world moves quickly, and older consoles are left to degrade over time whilst newer editions are released. Old video game cartridges and disks become obsolete as they fail to work on most newer consoles, rendering the game(s) unplayable unless the compatible console has been retained (and still works).

The preservation of our video game heritage is important as many consumers can find great value in older games. This could be due to feelings of nostalgia, interest in how older video games helped shape those of today, or even because the game itself is still inherently enjoyable.

Another reason preservation is important is that it can help to inspire video game developers into making even more enjoyable games.

There are 3 main ways of playing classic video games. The first way is by playing the original game on the original console. This can be difficult unless the player already owns the console as they are sometimes rare to find and expensive to obtain. Even if the player does own the console, it may not work properly as hardware tends to degrade over time.

The second way is by playing a ported version or a "remaster" of the original game. These are usually consumer-friendly as they tend to release on the newest and most popular platforms but eventually these platforms become outdated as newer ones are released.

The third way is by using an emulator. This can be either a hardware or software-based emulator. Over the years, different developers have used different techniques in order to achieve an emulation experience that is enjoyable for the user. Nintendo, Xbox, and PlayStation are all currently using different types of services to let consumers play their old games. On top of this, there are 3<sup>rd</sup>-party emulators available to download that are not console-specific, such as MAME.

My goal was to create a mobile application that could help users find the best way for them specifically to enjoy classic video games. This is so I could help people get more enjoyment out of a popular hobby and also make consumers more aware of the different options when it comes to playing classic video games. I created the application using Android Studio and the programming language Kotlin. Throughout the next few sections, we will see how I was able to achieve my main goal and what methods were used.

## 1.2 Aims

More specifically, the aims of the project are as follows:

- To identify key properties of various video game emulators to aid in the creation of a mobile algorithm.
- To create an algorithm that will inform users on the best way(s) for them to play classic video games based on their responses. This means that the output should show the user what video game emulator is best for them.
- To create a bug-free application using Android Studio, Kotlin, and XML.
- To create a form that enables users to answer questions.
- To create a section in the app that allows users to view other potential emulators.

However, I must note that my initial goal was to collect data from this application and to use it to make an informed decision on what techniques make a good video game emulator. I changed from this goal as I wanted to focus on giving the user a good experience and the most accurate output possible.

## 2 Background

To fully understand the need for emulation, and therefore the need for this application, we need to have knowledge of how video game consoles have progressed over time and how certain consoles have or have not utilised backward compatibility.

### 2.1 History of Video Game Consoles, Backward Compatibility, and Console-Based Emulation

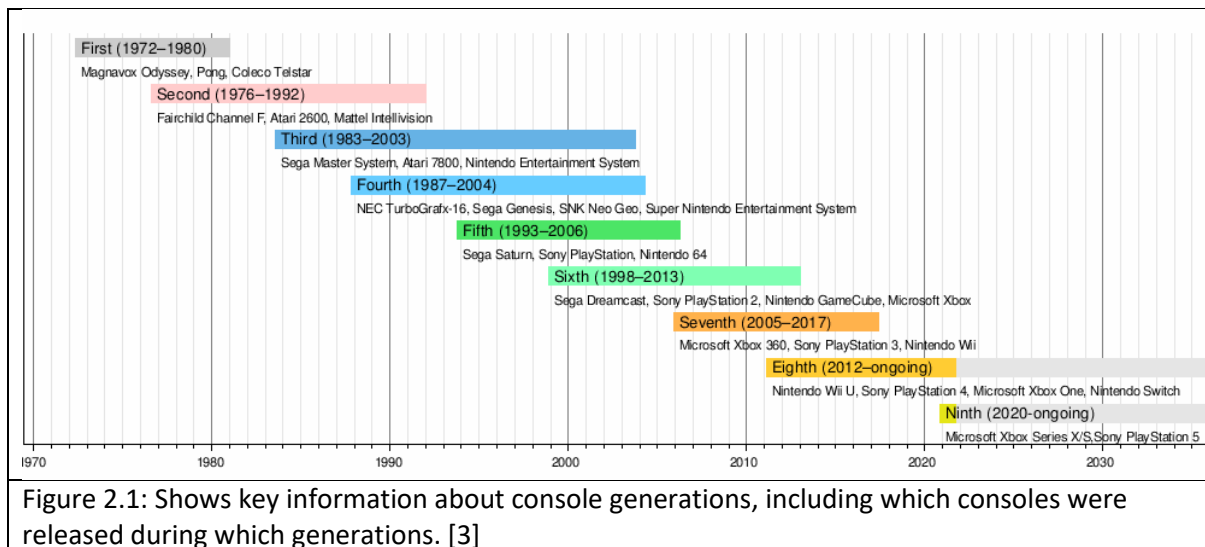
When looking at the history of home video game consoles, it is important for our understanding that we either group them into generation, manufacturer, or both. Consoles tend to be grouped by generation so that it is easier to make comparisons between consoles that release in a similar span of years, and often have similar technical specifications or power [1]. From this history, described throughout the next few pages, we should be able to better understand the importance of video game emulation, and thus, the why my application is useful.

#### 2.1.1 Console Generations

Here is a quick summary of the console generations:

- The first generation started in 1972 with the Magnavox Odyssey and most consoles were dedicated and lacked colour [2].
- The second generation started in 1976 and introduced the microprocessor, leading to more colourful games with better audio. It also saw a decrease in the number of dedicated consoles with game cartridges becoming the standard format. The generation ended with the video game crash of 1983 [2].
- The third generation saw the revival of the industry, mainly thanks to the release of the Nintendo Entertainment System in 1983 (released in North America in 1985). This was known as the “8-bit generation” [3].
- The fourth generation started in 1987 and saw consoles upgrade to 16-bit microprocessors. The first popular non-dedicated handheld console, the Game Boy, was released during this time [4].
- The fifth generation started in 1993 and despite being known as the “32-bit” generation, it saw the release of the 64-bit console, the Nintendo 64 [3]. Sony also released their first console, the PlayStation during this generation. CD-ROMs also came into use as a storage medium during this generation [2].
- The sixth generation began in 1998 and saw a jump to 128-bits [3]. Microsoft released their first console, the Xbox, during this period [5]. The Xbox helped to popularise online gaming.
- The seventh generation began in 2005 and saw an increased adoption of internet services [2]. The Nintendo Wii was released with a key feature being motion controls [6].
- The eighth generation started in 2012 and continued the trend of increased power. Nintendo released both the Wii U and the Nintendo Switch during this time. The latter was far more successful. Several “classic editions” of retro consoles were released [7].

- The ninth generation started in 2020 and has so far has seen the release of two consoles (with variations) [3]. These are the PlayStation 5 and the Xbox Series X.



### 2.2.2 Atari

Atari's release of Home Pong in 1975 was their first foray into the home console market [8]. The only playable game was *Pong* and it came out towards the end of the first generation. Two years later (and in the second generation) Atari released the Atari 2600 which sold over 30 million units [9] and helped Atari dominate the market before the video game crash of 1983. Its successor, the Atari 5200, was released in 1982 but was discontinued two years later.

Released in 1986, Atari's third generation console was the 7800 and it was one of the first-ever video game consoles to feature backward compatibility [10]. This meant that the 7800 could run almost all Atari 2600 games using their cartridges. This helped to make Atari 2600 games playable for longer whilst also creating more of an incentive to upgrade from the 2600. The main issue for Atari was that by this point, Nintendo had already released the Nintendo Entertainment System (NES).

Atari only released two more consoles after the 7800, finishing with the Jaguar in 1993. Since then, several dedicated consoles (which act as emulators) have been released in an effort to preserve Atari games and also profit off of the retro games market. This is probably the best way to play Atari games nowadays unless the consumer wishes to play a game that is not on one of these systems. In this case, their only other options are either purchasing an original console or using an unlicensed emulator.

### 2.2.3 Nintendo

The first video game console released by Nintendo was the Color TV-Game in 1977 [11]. This was a dedicated console which meant that each one was only able to play one or more built-in games (such as *Pong*). It also meant that there was no opportunity for it to be compatible with later consoles as there were no cartridges or discs to use. It was very similar to the first generation of consoles except that it made use of colour.

In 1982, Nintendo released the Game & Watch. This was a series of handheld video games to be played on dedicated devices [12]. Due to the dedicated nature of the devices, it made it impossible for future consoles to be backward compatible with the games in the series.

During the third generation, Nintendo released the Family Computer (widely known as the Famicom) in Japan in 1983. Two years later it was redesigned and released in North America as the Nintendo Entertainment System (NES). The platform saw the releases of several games that would go on to become massive franchises for Nintendo, such as *Super Mario Bros.* and *The Legend of Zelda*. The console used cartridges so this finally gave Nintendo games the opportunity to be compatible with future platforms, but it was never utilised by Nintendo.

Nintendo released their first proper handheld video game console in 1989 and dubbed it the Game Boy. It was their final 8-bit device and made use of cartridges to store games on, just like the NES [4].

The release of the Super Nintendo Entertainment System (SNES) in 1990 saw Nintendo jump from the 8-bit era and into the 16-bit era. Although the console itself had no backward compatibility feature built-in, a peripheral called the Super Game Boy allowed Game Boy games to be played on the 16-bit platform [13]. Despite this, I would not say that backward compatibility was a feature of the NES due to the two consoles being regarded as being from the same generation (the fourth) [2].

The Nintendo 64 (N64) was released in 1996 during the fifth generation. Despite using cartridges like its two predecessors, it featured no backward compatibility and no future device was compatible with games of the console.

Nintendo's next two handhelds, the Game Boy Color (GBC) and the Game Boy Advance (GBA) were both fully backward compatible with their predecessors. This was great for both systems as it added many more games to their libraries without the games having to be developed. It was great for players as well as it meant that their old games were still relevant and more joy could be had by playing them. Unfortunately, the same could not be said for their older home consoles.

In 2001, Nintendo released the GameCube and it was their first home console to make use of optical discs for storing video games. It had no backward compatibility features but similarly to the SNES, a peripheral was released so that GBA, GBC and original Game Boy games could be played on the console [14].

I'm going to skip over the Nintendo DS and Nintendo 3DS family of consoles as they both follow the same trend as the Game Boy family in terms of being backward compatible (although different versions are more compatible than others). One key thing to note is that they used game cards as a storage medium.

Initially launched in 2006, the Nintendo Wii finally saw Nintendo implement backward compatibility on a home console. The Wii essentially had a GameCube built into the system, meaning all GameCube games could be played on the new console [15]. The Wii even had slots for both GameCube controllers and memory cards. A key part of the compatibility was the fact that both consoles used optical discs to store games. In addition to this, the Wii introduced the Virtual Console. This was an emulation service that enable users to play games from the NES, SNES and N64 [16]. Although not all the games from the three platforms were available on the Virtual Console, this was a huge step in the right direction for Nintendo, as it showed they were now committed to the preservation of their own catalogue of video games. It also showed that there was a market for these older games and that many people were still interested in playing them. The Virtual Console went on to feature on both the 3DS and the Wii U, the latter of which was fully backward compatible with Wii games. The 3DS and Wii U versions of the Virtual Console added extra features, with the key one being save states. This meant that classic games that were previously very punishing, as they had no option to save, became more accessible to everyone who wanted to play them to their completion. Throughout the Virtual Console's lifespan, more games and platforms were added, and

several third-party consoles became available for emulation, such as the Sega Master System and the Commodore 64. The main issue with the emulator was that Nintendo tended to charge hefty prices for games that were decades old.

In 2017, Nintendo released a console that was both handheld and playable as a home console – the Nintendo Switch [17]. The Switch uses game cards and is not backward compatible in any way. This was a surprising departure for Nintendo but also not surprising when considering the nature of the device. However, Nintendo would go on to release an emulator for NES games as part of a subscription service (Nintendo Switch Online) in late 2018 [18]. This emulator could play a library of 20 games at launch. Since then, the library has been expanded, and emulators for the SNES, N64 and Sega Genesis have been added to the service. All the games available are chosen by Nintendo and are all included as part of the subscription. This means that subscribers don't actually own these games as once their subscription ends, they no longer have access. These emulators make use of the save states feature from the Virtual Console, whilst also employing a "rewind" feature that allows players to virtually go back in time in case they make a mistake [19]. This is very useful for more challenging games, and it can help stop players from becoming frustrated during difficult sections. Once again, the feature was introduced to make classic games more accessible to players of a lower skill level and in turn, more enjoyable for them.

Before NES games were playable on the Switch, Nintendo released the NES Classic Edition (also known as the NES Classic Mini) in 2016. This was a dedicated home video game console that contained a library of thirty NES games [20]. The console is essentially an emulator that is completely dedicated to running NES games. It made use of features from the Virtual Console and introduced the rewind feature that would later be used on the Switch. The main upside of this console compared to playing on the Switch, is that once you buy the console, you own all the games on it. The following year, the SNES Classic Edition was released. This had all the same features as its predecessor and had a library of 21 SNES games, including an unreleased game called *Star Fox 2* [21]. Both consoles were packaged with controllers that felt and looked like the originals, helping to give the player the feeling of playing on the original console.

Other than on the original hardware, these consoles are the most faithful way to legally play the games that are on them, whilst also improving the experience with added features. The only downside to these consoles, is that no games can be added to their libraries.

From this brief history of Nintendo consoles, it is clear that the company recognises the value of preserving their old titles. They have made increasingly more effort to make their classic titles playable and purchasable by consumers. There is a limit to this however, as there are still many games from past generations that are unplayable on their newer platforms. Nintendo have made a great effort to make their "classics" available but seem to have completely forgotten about a large percentage of titles that have been released on their platforms. This might be because they don't think it is worth making less popular games playable as interest in them would be lower. Alternatively, it could be because they want more focus on Nintendo franchises that are still running, such as *Super Mario*, *Metroid*, and *The Legend of Zelda*.

## 2.2.4 PlayStation

The PlayStation (now commonly known as the PS1) was originally released in 1994 by Sony Computer Entertainment. The console made use of compact discs to store its games. It was released during the fifth generation of consoles and became the first console to sell over 100 million units [22].



The PlayStation's successor, the PlayStation 2 (PS2) came out in 2000 and featured full backward compatibility. This gave it a leg up over its biggest competitor, the Nintendo GameCube, as it meant that PS1 owners had even more of a reason to stick with the PlayStation line. It also meant that it had a much larger library of games than its competition. Not only were PS1 games compatible with the PS2, but the controllers were as well.

Initially released in December 2004, Sony released their first handheld video game console, the PlayStation Portable (PSP). Strangely for a handheld device, the PSP used an optical disc format to store video games on. This was the Universal Media Disc and movies for the console were also released in this format [23]. Its successor, the PlayStation Vita, was released in 2011. The unique nature of the PSP's game storage format meant that physical games were not compatible with the Vita. However, many PSP games were released digitally on the PlayStation Store and the Vita was backward compatible with every PSP game on there.

In 2006, Sony released the PlayStation 3 (PS3). The system was fully backward compatible but later revisions of the model removed that feature, perhaps hinting at Sony's reluctance to make older games playable on the PlayStation 4 (PS4). The PS3 also had a very complex system architecture which made it difficult to develop games for [24].

The PS4 was released seven years after the PS3 with no backward compatibility whatsoever. The complex architecture of the PS3 likely played a part in this. Eventually, PlayStation 2 games became purchasable through the PlayStation Store. These were playable through emulation. Sony went on to launch a subscription service called PlayStation Now which allowed users to stream PS3 games on the PS4. Sony created new hardware to avoid the difficulty of replicating the PS3 through emulation. This hardware was essentially eight miniature PS3s strapped into a single motherboard [25]. This gave it the necessary power to stream games over the internet. More games have been added over time, including games for the PS2 and PS4.

Sony released the PS5 in 2020 and it featured nearly full backward compatibility for PS4 games. The two consoles have a similar architecture which made the feature more easily achievable. Sony made the switch from PS5 to PS4 even easier as they allowed players to synchronize save data for their games by use of cloud storage [26]. Unfortunately, games from older consoles are not compatible but PlayStation Now functions just as well on the PS5.

In 2018, Sony tried to cash-in on one of their older consoles by releasing the PlayStation Classic. This was a miniaturised dedicated console that contained a library of twenty games. Like the NES and SNES Classic editions, the PlayStation Classic used an emulator to run its games. However, many viewed the console as having a weak library considering the number of great games that were released on the console [27].

Sony have an odd relationship with backward compatibility on their consoles. Initially, they seemed committed to having all their games carry across to the next generation before complications with the PS3 led to no backward compatibility on the PS4. They've clearly made efforts since the console's launch to make many older titles playable on both the PS4 (and now the PS5), but they have still left a large portion of their library unplayable on their newer machines.

### 2.2.5 Xbox

The sixth generation of consoles saw Microsoft enter the arena with the Xbox. It was released as a home video game console in 2001. Video games were stored on discs, as it was now the common format after being popularised by the PlayStation.

Four years later, Microsoft released the Xbox 360. Due to very different system architectures in the two consoles, no backward compatibility was available through the hardware. Instead, Microsoft utilised emulation to make some Xbox games playable on the 360. The library grew for two years until 2007. For the emulated games to work, an official hard drive was needed. Until 2007, the original Xbox discs were needed for all backward compatible games, before certain titles became available digitally. This was through the Xbox Originals program.

The 360's successor, the Xbox One, was released in 2013. The console had no backward compatibility at launch and Microsoft had no intentions to develop the feature for the console at any point in the future [28]. Things started to change once Phil Spencer became the head of Xbox in 2014. At E3 2015, Microsoft announced plans for 100 Xbox 360 titles to become playable on the Xbox One by the end of the year [29]. Since then, hundreds of more Xbox 360 games have been made compatible, along with many original Xbox games. Xbox One's backward compatibility uses emulation for both Xbox 360 and Xbox games. Network and multiplayer features are also supported.

Microsoft's latest releases, the Xbox Series X and Xbox Series S, are both fully compatible with Xbox One games. In addition to this, they are compatible with every Xbox and Xbox 360 game that was playable on their predecessor. Selected games have also been enhanced for the new consoles, with upgraded resolutions and framerates. Even games that haven't been enhanced tend to run and look better on the ninth generation consoles, helped in part by machine learning [30]. Like the PS5, cloud saves are utilised so that players' game data can be transferred across the devices.

Outside of the console space, Microsoft released Project xCloud (now known as Xbox Cloud Gaming) in 2020. This is a service that allows users to play selected games, from across all Xbox consoles, on their mobile devices. Xbox Cloud Gaming is included as a feature of Xbox Game Pass Ultimate, a subscription service. Since its launch, more games have been added and many games have been given touch control support, meaning that no controller is needed.

Despite not initially putting much stock or effort into backward compatibility and emulation, Microsoft have grown to be one of the biggest advocates for the feature. I think they recognise that many gamers and consumers enjoy having a big library to choose from, as it means they will always be able to find something worth playing. I also believe that Microsoft know they can use their gaming history as a marketing tool by making consumers feel nostalgic. In spite of all this, the vast majority of Xbox and Xbox 360 games are still unplayable on any console released after them.

## 2.2 Unlicensed Emulators

The most versatile emulators and the ones that give you access to the biggest library of games are those that are unlicensed. The main issue with these emulators is that you often need to download video game ROMs or ISOs in order to play the games, and these are usually illegal. ROMs are copies of game cartridge data and ISOs are copies of optical media.

Emulators are usually developed by reverse-engineering the hardware of a video game console [31]. They can be developed into three different forms. They can either be software-based (this is what I will mainly be focusing on), hardware-based, or a mixture of hardware and software. Popular software-based emulators include MAME (for arcade games), Dolphin (for GameCube and Wii games), and PPSSPP (for PSP games). Technology has now reached a point where many video game emulators can run on mobile devices as well as PC devices. Most emulator developers employ special techniques (as part of the reverse engineering process) to avoid breaking the law.

The real legal issues begin when a user has to obtain a game to play. Video games are copyrighted and many retro games are still copyrighted to this day. ROM files and ISO images infringe on these

copyrights. In 2019, Nintendo sued a video game ROM site and was awarded \$2.1 million in damages [32]. The site, called RomUniverse, was also shut down. This wasn't the first time that Nintendo cracked down on ROM sellers and it probably won't be the last.

As many games from the past are unavailable to play by disk on the consoles of today, the only easy way to play these games is by use of emulation. This applies further to games that might have had all their physical copies lost or destroyed. So unless the user is willing to engage in some piracy, for which they'll most likely never face any repercussions, then many games that they may want to play will immediately become unavailable to them.

### 2.3 Emulator Specification Analysis

In the application, the user interaction comes from answering questions. These answers are then used to help form the output. That output contains information about the emulator that the algorithm decides is best for them. By analysing the attributes of each of our chosen emulators, we can make a decision on how each answer will affect the chances of each emulator becoming the chosen output. In the figure below, I have included the most vital data from the analysis. It lists each of the possible emulators that could be received as output, what consoles they support, what platforms they run on, and more.

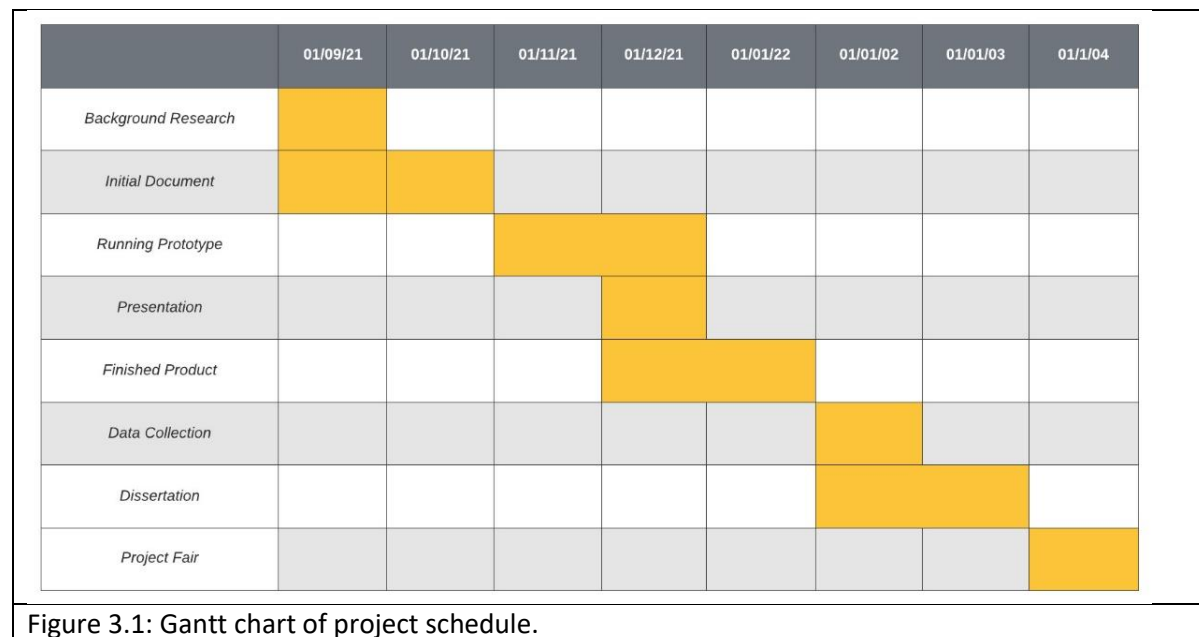
	A	B	C	D	E	F
1	Emulator	Console(s)	Platform(s)	Price	Easy to use	Library
2	Dolphin	GameCube, Wii	Windows, macOS, Linux, Android	Free	Yes	Expandable
3	DeSmuME	DS	Windows, Linux, Android	Free	Yes	Expandable
4	Kega Fusion	Sega Genesis, SG-1000, SC-3000, SF-7000, Master System, Game Gear, Sega CD, and 32X	Windows, macOS, Linux	Free	Yes	Expandable
5	MAME	Arcade, Atari	Windows, macOS, Linux	Free	No	Expandable
6	Mednafen	PS1, Sega Saturn, GBA, Genesis, NES, SNES	Windows, Linux	Free	No	Expandable
7	Mesen	NES, Famicom, Famicom Disk System	Windows, Linux	Free	Yes	Expandable
8	OpenEmu	DS, GB, GBC, GBA, Atari, Famicom Disk System, Game Gear, NeoGeo Pocket, NES, N64, SNES, GameCube, Sega, PS1, PSP	macOS	Free	Yes	Expandable
9	RetroArch	DS, 3DS, GB, GBC, GBA, Atari, Famicom Disk System, Game Gear, NeoGeo Pocket, NES, N64, SNES, GameCube, Wii, Sega, PS1, PS2, PSP	Windows, macOS, Linux, iOS, Android	Free	No	Expandable
10	PCSX2	PS2	Windows, macOS, Linux	Free	Yes	Expandable
11	PPSSPP	PSP	Windows, macOS, Linux, iOS, Android	Free	Yes	Expandable
12	Mupen64Plus	N64	Windows, macOS, Linux, Android	Free	Yes	Expandable
13	Redream	Dreamcast	Windows, macOS, Linux, Android	Free, \$6 for pre	Yes	Expandable
14	RetroPie	Atari, Dreamcast, GB, GBC, GBA, GameCube, Game Gear, Sega Genesis, Arcade, N64, NDS, NES, PS1, PS2, PSP, SNES, Wii	Raspberry Pi	Free	Yes but no i	Expandable
15	Snes9x	SNES	Windows, macOS, Linux, Android	Free	Yes	Expandable
16	VisualBoyAdv	GB, GBC, GBA	Windows, macOS, Linux	Free	Yes	Expandable
17	Nintendo Switc	NES, SNES, N64, Sega Mega Drive	Nintendo Switch	£1.50-3.49 pm	Yes	Fixed (but gar
18	NES Classic	NES	Dedicated	~£70 used	Yes	Fixed
19	SNES Classic	SNES	Dedicated	~£100 used	Yes	Fixed
20	Playstation Cla	PS1	Dedicated	~£80 new, £40	Yes	Fixed
21	Xbox BackCom	Xbox, Xbox 360	Xbox One, Xbox Series X/S	£100+	Yes	Expandable
22	Cemu	Wii U	Windows	Free	Yes	Expandable
23	PS BackCompat	PS1, PS2, PS3	PS4, PS5	£150+	Yes	Fixed

Figure 2.2: Shows the key details about each possible emulator.

### 3 Project Management

#### 3.1 Timeline Analysis

Below is my initial Gantt chart for the project timeline. To clarify, the task should have been finished by the start of the next unhighlighted block (e.g. background research should have been finished by the 1<sup>st</sup> of October. The first four activities were all completed on time. However, the final product was not completed until the 1<sup>st</sup> of April. This meant that the data collection never took place, but this is more than fine as my project aims changed and so I did not need to complete this task. Despite this, I still finished the final product later than I would have liked. Looking at my Gantt chart critically, it was not detailed enough and some sections became inaccurate once I had changed my project aims.



#### 3.2 Project Aims Analysis

Below, I have created a table containing the original project aims that were outlined in the initial document, along with reasoning for why I have or haven't completed them. Overall, I believe that I have achieved what I wanted to achieve, but not what I initially set out to do.

Aim	Completed?	Why? / Why Not?
Present users with a form that outputs questions. These questions will include topics such as cost, ease of use, game library, extra features and portability.	Mostly	My application presents users with a form that contains questions. All of the topics mentioned are covered, apart from extra features. This is because, in my emulator specification analysis, I found that most of the emulators have the same extra features, and so I did not believe it was worth asking. I wanted all my questions to have an affect on the output, and this question would not have done.
Show the users videos of games running on different emulators.	No	Part of my original plan was to collect data and part of that data collection entailed asking users which emulator they thought looked better. After the data collection

		element was taken out of the project, this was no longer an aim to be fulfilled.
Receive input from the form and store the data. The input could be received in numerous ways, including text and checkboxes.	Mostly	My application does receive input from the form in numerous ways, such as through radio buttons and checkboxes. However, The data is not actually stored because, as I have said previously, the data no longer needed to be collected. It is still used for the output though.
Use an algorithm to determine the best output based on the user's input. This will let the user know the best option(s) for them based on their answers.	Yes	This is where the functionality of the program comes from. It may not present the objectively best option for that specific user, but it will give the best option based off of the algorithm used.
Figure 3.2: Shows a breakdown of the initial project aims, outlined in the initial document.		

### 3.3 Risk Analysis

Despite this being a relatively low-risk project, there were still some risks that need to be addressed. The figure below shows the risks that were identified before the project began, whether they were encountered, and how they were dealt with and/or avoided in the first instance.

Risk	Was it Encountered?	How was it resolved?
Legal issues with obtaining ROMs and ISOs for gameplay videos.	No	I did not illegally download any ROMs as I did not need to record video.
Illness.	Yes	I was not ill for very long with relation to the timeframe needed to complete the project.
Not being able to spend as much time on the project as I would like.	Yes	I reduced the amount of "free time" I had so I could spend more time on my project, as well as my other studies.
Lack of other work similar to the algorithm I wanted to use.	Yes	I found it very difficult to find information about writing the kind of algorithm I wanted. So instead, I made sure my emulator specification analysis was thorough so my decisions could still be well informed. For the most part, the values given from the algorithm still had to be chosen based off my own opinion.
Bugs in code.	Yes	I made sure to do testing at every stage and for every time I added some functionality.

Figure 3.3: Shows an analysis of the risks involved in the project.

## 4 Implementation

### 4.1 Stage One

I started off the creation of my app by creating a simple prototype with a single activity. It had a single “question” and three radio buttons (contained inside one radio group) for input. It also had a “submit” button which, when pressed, would tell the user which answer had been selected. The idea of this prototype was to ensure that I was able to capture data using radio buttons, as they are useful for receiving answers from questions. I also wanted to prove that I could produce an output from the user’s input. This prototype did not take long to make as it was not complicated, but it was a useful starting point for the project.

### 4.2 Stage Two

The next step was designing the questions for the user. These had to include topics such as cost, platforms, portability, supported systems, legality, and ease of use. I believe all of these topics to be key factors in deciding what emulator to use. Most importantly, these questions had to break down what the user wanted into programmable parts. The questions did not have to be in their final state straight away, and I repeatedly modified the questions throughout the further stages of development.

One issue that arose in the development of my second version, was that the questions would run off the screen at the bottom. To fix this, I added a *ScrollView* with a *LinearLayout* as opposed to the default *ConstraintLayout* that I had been using previously. Adding this in enabled us to move (scroll) the screen so that all the questions could be both answered and read.

Another issue that I noticed was that having the radio buttons in a horizontal line did not leave enough room for text and/or more than about three buttons. Even though I wanted my buttons to be horizontal from my initial designs, I had to change them to appear vertically so that the associated text became legible.

### 4.3 Stage Three

For the third stage, I added in a new type of question. These questions had the potential for multiple answers and so radio buttons were not suitable. I originally tried using chips as I knew you could have multiple selected at once inside a chip group. Unfortunately, I found it too difficult to capture data from them in a way that was usable, so I swapped to checkboxes which were much easier to obtain data from.

Next, I added some actual functionality to the application. I created integers to represent each possible emulator (output) and assigned them all the value of zero. I then used my emulator research to determine which answers would increase or decrease the value of each emulator. After this, I made it so when the “submit” button was pressed, it found which integer variable had the highest value and output both the value itself and the emulator it represented through an Android *Snackbar*.

Initially I had the emulator variables be incremented (or decremented) through the radio buttons’ respective handler functions. This meant that the variables changed whenever they were pressed, which led to values being changed even when they shouldn’t have been by the final answer. Alternatively, it would mean that some variables were incremented multiple times if the same answer was clicked multiple times. To amend this, I gave each question (that used radio buttons) an integer variable representing the answer and then added a new function for each of those questions. These functions used *when* statements to determine which variables were increased or

decreased. I also used my emulator specification analysis to group certain emulators together, in order to write more efficient code. For example, the function *incWindows()* increments all emulator variables that run on windows. Alternatively, *decWindows* decrements all emulator variables that run on windows. I also wrote the same style of functions for macOS, Linux, Android, and iOS.

Similar to the radio buttons, the checkboxes could cause multiple incrementations if pressed multiple times. This was an easier fix however, as I did not need to write extra functions for them to affect the variables in the way I intended them to. Instead, I just had to reverse the affects of them being clicked when they became unclicked (or unticked). For example, if *incWindows()* got called when clicked, then the *else* statement had to call *decWindows()*.

#### 4.4 Stage Four

In the fourth stage of development, I added a more complex algorithm which combined the answers from two different questions. One question was about how much the user was willing to spend, and the other question asked what platforms and devices the user had available to them. These two questions are related because if the user is not willing to spend any money, and they do not have, for example, a macOS device, then the emulators that run on macOS should be a lot less likely to appear as the output. The main complexity comes from the crossover between platforms. For example, most of the emulators that run on macOS also run on Windows. The issue here is that you can't, in this situation, call *decMac()* as this would also decrement some of the variables representing the emulators that run on Windows. I also would not be able to write specific *if-else* statements for every possible combination of platforms as this would create a huge amount of code.

From Windows, macOS, Linux, Android, and iOS, there are 31 different possible branches which would then need unique code. Each branch would affect multiple emulator variables, and this would be extremely time consuming. To avoid this, I analysed the relationships between the platforms. Relating only to the emulators selected for this project, there was a very clear hierarchy that I could work with for four of the platforms. All of iOS emulators were on Android, all of the Android emulators were on Linux, and all of the Linux emulators were on Windows. In addition to this, all the iOS emulators were on macOS. However, it got more complicated when looking at the relationship between macOS and the other three platforms. There were three emulators on Windows that weren't on macOS, and there was one (openEmu) on macOS that wasn't on Windows. These numbers increased when comparing macOS to both Linux and Android. Unfortunately, this meant that there was no concise relationship to be found here. However, the hierarchical relationship between Windows, Linux, Android, and iOS meant there didn't have to be.



Windows	Linux	MacOS	Android	iOS	Other
Dolphin	Dolphin	Dolphin	Dolphin	RetroArch	Nintendo Switch Online
DeSmuME	DeSmuME	Kega Fusion	DeSmuME	PPSSPP	NES Classic
Kega Fusion	Kega Fusion	MAME	RetroArch		SNES Classic
MAME	MAME	OpenEmu	PPSSPP		Playstation Classic
Mednafen	Mednafen	RetroArch	Mupen64Plus		Xbox BackCompat
Mesen	Mesen	PCSX2	Redream		PS BackCompat
RetroArch	RetroArch	PPSSPP	Snes9x		
PCSX2	PCSX2	Mupen64Plus			
PPSSPP	PPSSPP	Redream			
Mupen64Plus	Mupen64Plus	Snes9x			
Redream	Redream	VisualBoyAdvance			
Snes9x	Snes9x				
VisualBoyAdvance	VisualBoyAdvance				
Cemu					
		iOS -> Android -> Linux -> Windows			
		iOS -> macOS			

Figure 4.1: Shows a spreadsheet containing all the emulators and the platforms they run on.

Due to the nature of the relationships, I decided to work backwards. This meant decrementing the variables of the emulators that the user didn't have, as opposed to incrementing the ones they did have. For example, if the user had an Android device and nothing else, then all emulators that were not on Android would be decremented. To do this, I created Boolean variables for all platforms (including Nintendo Switch, Xbox One, and PlayStation). Then when a platform's checkbox is ticked, their respective Boolean variable is set to true. Next, I wrote the *questionTwo()* function (which involves the question about cost) and a helper function, for the more complicated part. The main function contains a *when* statement and branches for each price range. The first four branches call the helper function because if the user selects one of those price ranges, then the five main platforms will be affected due to their prices.

The helper works by covering each platform and their combinations incrementally through nested *if-else* statements. It starts with macOS as that is not part of the hierarchy but still shares emulators with the other platforms. Inside this branch, it asks if Windows is owned as it is at the top of the hierarchy. However, this branch performs no action as all emulators on the five main platforms are covered just by the union of Windows and macOS. Therefore, no emulators on those platforms should be decremented as they are all obtainable with no extra cost. The first inner *else-if* asks if the user has a Linux device, as it is next in the hierarchy. Because this branch comes after the Windows branch, I know that the user does not have a Windows device. This means I can decrement the emulator variables(s) that are on Linux and macOS, but not on Windows. I do not have to decrement the variables representing emulators that run on either Android or macOS, as they are both below Linux in the hierarchy.

Essentially, this function is based on set theory. It works by finding the set of emulators that lie outside of the selected platforms and then decrements them. My working was as follows:

*Let W be the set of emulators that run on Windows.*  
*Let L be the set of emulators that run on Linux.*  
*Let A be the set of emulators that run on Android.*  
*Let I be the set of emulators that run on iOS.*  
*Let M be the set of emulators that run on macOS.*  
*Let D be the set of emulators that run on dedicated platforms.*



$$I \subseteq A \subseteq L \subseteq W$$

$$I \subseteq M$$

The Venn diagram below helps to show why I decremented the variables of the emulators that hadn't been selected, as opposed to incrementing the ones that have been. For example, if macOS and Android had been selected, the selected set becomes  $L \cup M$ . If we were then to call *incMac()* followed by *incLinux()*, then the emulators within the set  $L \cap M$  would be incremented twice. In this situation, the only emulator (that runs on the five main platforms) we need to decrement the variable for is Cemmu.

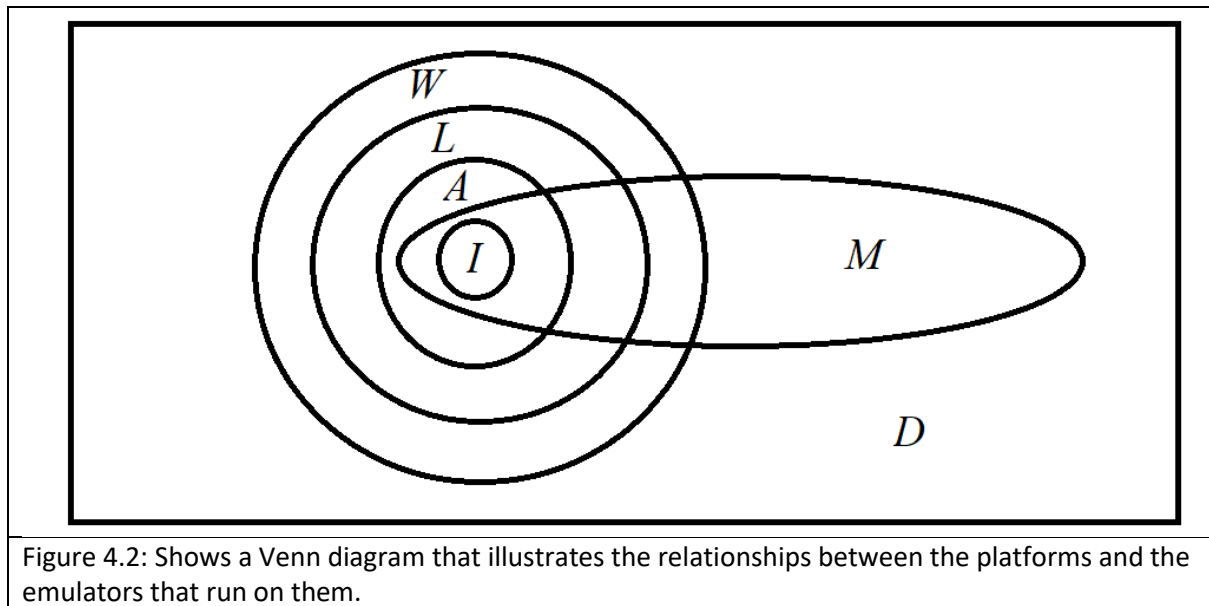


Figure 4.2: Shows a Venn diagram that illustrates the relationships between the platforms and the emulators that run on them.

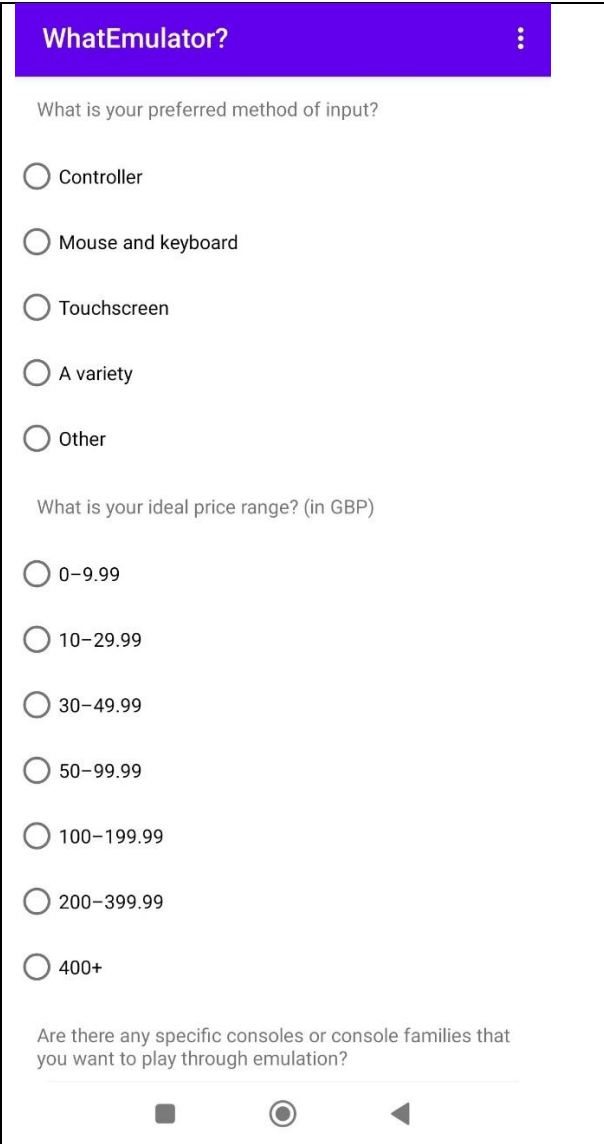
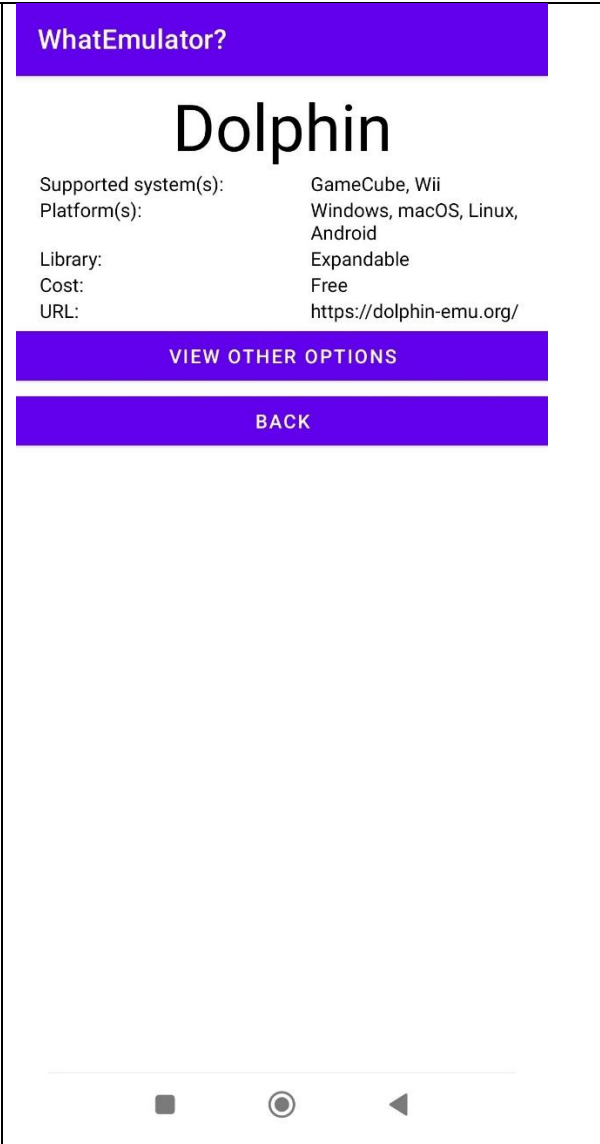
#### 4.5 Stage Five

For the final stage of development, I created new activities for each possible emulator (output). These activities contained a small factsheet about their respective emulator. These activities meant that the user could actually receive useful information about their chosen emulator, which is a key purpose of the app. These activities all also had two buttons below the factsheet. One for going back to the main activity, and one for going to another new activity created in this stage of development. This activity is essentially a list of buttons with each one taking you to an activity describing an emulator. I did this so that users can see the other options that are available to them, and not just the one chosen for them by the application. I believe this is an important feature to have, as the app is not perfect and will not necessarily choose the perfect option for them. In addition to this, I believe that many people may want more than one emulator, and so easily being able to see other options could be beneficial to them.

## 5 Results

Once development was finished, I was left with a fully working mobile application that fulfilled all of the project aims outlined in section 1.2. Figures 5.1, 5.2, and 5.3 show parts of each activity in the application in their final forms. Figure 5.1 shows that I have given users a form that allows them to

answer questions. Figure 5.2 displays a potential output from the application. Figure 5.3 shows that users are able to go to a screen to view other emulator options. Therefore, these screenshots prove that at least three of the aims from section 1.2 have been fulfilled.

 <p>WhatEmulator?</p> <p>What is your preferred method of input?</p> <p><input type="radio"/> Controller</p> <p><input type="radio"/> Mouse and keyboard</p> <p><input type="radio"/> Touchscreen</p> <p><input type="radio"/> A variety</p> <p><input type="radio"/> Other</p> <p>What is your ideal price range? (in GBP)</p> <p><input type="radio"/> 0-9.99</p> <p><input type="radio"/> 10-29.99</p> <p><input type="radio"/> 30-49.99</p> <p><input type="radio"/> 50-99.99</p> <p><input type="radio"/> 100-199.99</p> <p><input type="radio"/> 200-399.99</p> <p><input type="radio"/> 400+</p> <p>Are there any specific consoles or console families that you want to play through emulation?</p>	 <p>WhatEmulator?</p> <h1>Dolphin</h1> <p>Supported system(s): GameCube, Wii</p> <p>Platform(s): Windows, macOS, Linux, Android</p> <p>Library: Expandable</p> <p>Cost: Free</p> <p>URL: <a href="https://dolphin-emu.org/">https://dolphin-emu.org/</a></p> <p>VIEW OTHER OPTIONS</p> <p>BACK</p>
Figure 5.1: A screenshot of the start of the main activity.	Figure 5.2: A screenshot of the Dolphin emulator page.

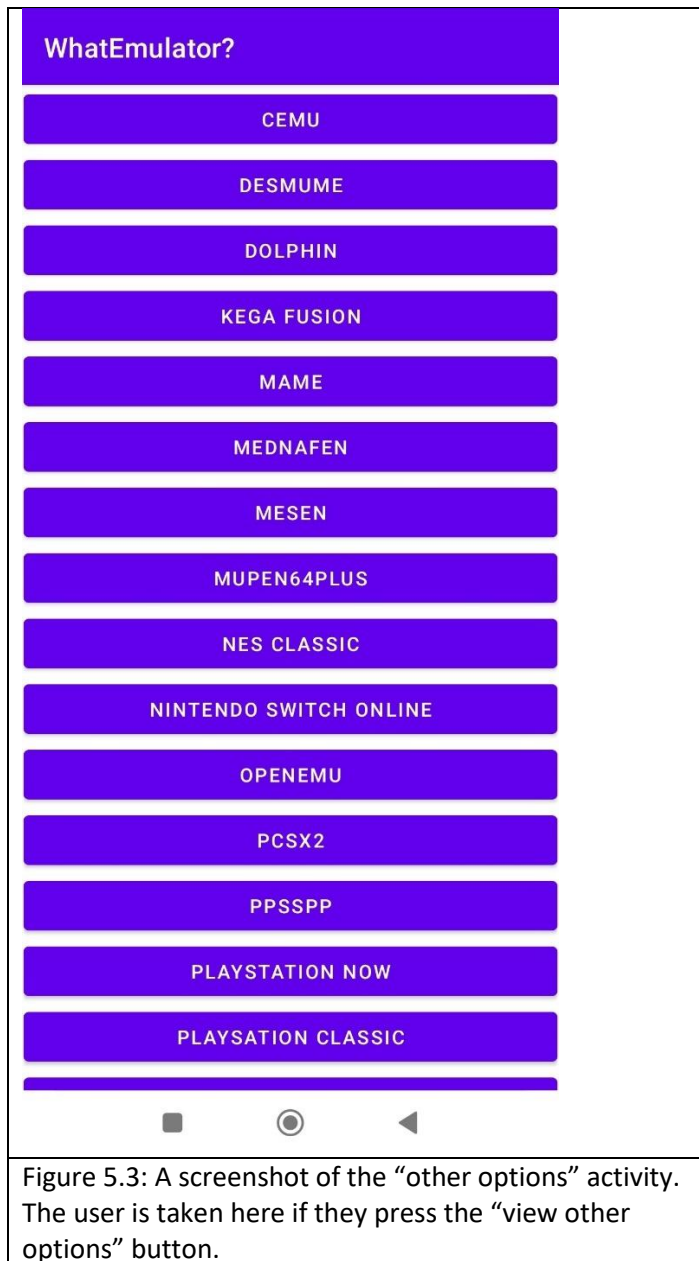


Figure 5.3: A screenshot of the “other options” activity. The user is taken here if they press the “view other options” button.

## 6 Conclusion

Overall, I am very happy with how my project turned out. However, looking at it critically, there is much that could be improved. In addition to this, it doesn't really add anything new to the field of computer science. If you were to compare it to popular mobile applications, such as YouTube for Android, the design quality looks very poor. Aesthetically, it is not pleasing to look at and the most popular mobile applications often are. On the other hand, my goal was not to make a “good-looking” app. It was to make one that works, and for this, I have succeeded.

My main criticism would be that my own work was barely influenced by others. This meant that my approach was not as scientific as I would have liked. This issue arose because I was not able to find published works involving the type of selection algorithm that I produced.

On a structural level, I could have used a database to store information about all of the different emulators. This would probably make the project scale better if more emulators were to be added. It could also reduce the amount of code as there would be less “hard-coded” lines.

With regards to the selection algorithm, I have set it up in a way that allows for easy modification, and so if I was to do a large amount of testing with users then I would be able to fine-tune the output very easily.

In conclusion, I got the results I wanted but I am very aware of this project’s shortcomings.

## Bibliography

- [1]B. Jack, "What Are Video Game Generations and Why Do We Use Them?", *MUO*, 2021. [Online]. Available: <https://www.makeuseof.com/what-are-video-game-generations/>. [Accessed: 30- Oct- 2021].
- [2]"The 8 Generations of Video Game Consoles", *Bbc.co.uk*, 2021. [Online]. Available: <https://www.bbc.co.uk/archive/the-8-generations-of-video-game-consoles/zvcjky>. [Accessed: 30- Oct- 2021].
- [3]"Home video game console generations - Wikipedia", *En.wikipedia.org*, 2021. [Online]. Available: [https://en.wikipedia.org/wiki/Home\\_video\\_game\\_console\\_generations#Console\\_generation\\_overview](https://en.wikipedia.org/wiki/Home_video_game_console_generations#Console_generation_overview). [Accessed: 31- Oct- 2021].
- [4]D. Kindy, "Thirty Years Ago, Game Boy Changed the Way America Played Video Games", *Smithsonian Magazine*, 2019. [Online]. Available: <https://www.smithsonianmag.com/innovation/thirty-years-ago-game-boy-changed-way-america-played-video-games-180972743/>. [Accessed: 31- Oct- 2021].
- [5]G. Gurwin, "The History of the Xbox | Digital Trends", *Digital Trends*, 2021. [Online]. Available: <https://www.digitaltrends.com/gaming/the-history-of-the-xbox/>. [Accessed: 31- Oct- 2021].
- [6]D. Ewalt, *Forbes.com*, 2006. [Online]. Available: [https://www.forbes.com/2006/11/13/wii-review-ps3-tech-media-cx\\_de\\_1113wii.html?sh=6012ad6c75bb](https://www.forbes.com/2006/11/13/wii-review-ps3-tech-media-cx_de_1113wii.html?sh=6012ad6c75bb). [Accessed: 31- Oct- 2021].
- [7]K. Dube, "What Are the Classic Editions of Retro Game Consoles?", *Online Tech Tips*, 2021. [Online]. Available: <https://www.online-tech-tips.com/gaming/what-are-the-classic-editions-of-retro-game-consoles/>. [Accessed: 31- Oct- 2021].
- [8]"Atari PONG - Game Console - Computing History", *Computinghistory.org.uk*. [Online]. Available: <http://www.computinghistory.org.uk/det/4007/Atari-PONG/>. [Accessed: 31- Oct- 2021].
- [9]"A Brief History of Game Console Warfare | BusinessWeek", *Web.archive.org*, 2007. [Online]. Available: [https://web.archive.org/web/20070509092239/http://images.businessweek.com/ss/06/10/game\\_consoles/source/3.htm](https://web.archive.org/web/20070509092239/http://images.businessweek.com/ss/06/10/game_consoles/source/3.htm). [Accessed: 31- Oct- 2021].
- [10]R. Rigues, "The history of backward compatibility on consoles", *Olhar Digital*, 2020. [Online]. Available: <https://olhardigital.com.br/en/2020/02/14/games-e-consoles/a-historia-da-retrocompatibilidade-nos-consoles/>. [Accessed: 31- Oct- 2021].
- [11]J. Parish, "35 Years Ago, Nintendo's First Brush With Video Disaster", *USgamer.net*, 2014. [Online]. Available: <https://web.archive.org/web/20190502051532/https://www.usgamer.net/articles/35-years-ago-nintendo-had-its-first-brush-with-video-game-disaster>. [Accessed: 31- Oct- 2021].
- [12]D. McFerran, "How Nintendo's Game & Watch Took "Withered Technology" And Turned It Into A Million-Seller", *Nintendo Life*, 2021. [Online]. Available: [https://www.nintendolife.com/news/2021/01/feature\\_how\\_nintendos\\_game\\_and\\_watch\\_took\\_withered\\_technology\\_and\\_turned\\_it\\_into\\_a\\_million-seller](https://www.nintendolife.com/news/2021/01/feature_how_nintendos_game_and_watch_took_withered_technology_and_turned_it_into_a_million-seller). [Accessed: 01- Nov- 2021].

- [13]D. Frear, "Remembering the Super Game Boy", *Nintendo Life*, 2011. [Online]. Available: [https://www.nintendolife.com/news/2011/10/feature\\_remembering\\_the\\_super\\_game\\_boy](https://www.nintendolife.com/news/2011/10/feature_remembering_the_super_game_boy). [Accessed: 01- Nov- 2021].
- [14]"Nintendo GameCube Game Boy Player - IGN", *IGN*, 2002. [Online]. Available: <https://www.ign.com/articles/2002/11/18/nintendo-gamecube-game-boy-player>. [Accessed: 01- Nov- 2021].
- [15]B. Kuchera, "Nintendo Wii: the Ars Technica review", *Ars Technica*, 2006. [Online]. Available: <https://arstechnica.com/gadgets/2006/11/wii/>. [Accessed: 01- Nov- 2021].
- [16]"The Wiire - Fils-Aime Talks American Launch and More", *Web.archive.org*, 2006. [Online]. Available: [https://web.archive.org/web/20061121000932/http://www.thewiire.com/news/473/1/FilsAime\\_Talks\\_American\\_Launch\\_and\\_More](https://web.archive.org/web/20061121000932/http://www.thewiire.com/news/473/1/FilsAime_Talks_American_Launch_and_More). [Accessed: 01- Nov- 2021].
- [17]"Nintendo Switch - Wikipedia", *En.wikipedia.org*, 2021. [Online]. Available: [https://en.wikipedia.org/wiki/Nintendo\\_Switch](https://en.wikipedia.org/wiki/Nintendo_Switch). [Accessed: 01- Nov- 2021].
- [18]K. Knezevic, "Nintendo Switch Online Service: More Details Revealed In Nintendo Direct", *GameSpot*, 2017. [Online]. Available: <https://www.gamespot.com/articles/nintendo-switch-online-service-more-details-reveal/1100-6461758/>. [Accessed: 01- Nov- 2021].
- [19]L. Doolan, "Reminder: The Rewind Feature Is Now Live For Nintendo Switch Online NES Games", *Nintendo Life*, 2019. [Online]. Available: [https://www.nintendolife.com/news/2019/07/reminder\\_the\\_rewind\\_feature\\_is\\_now\\_live\\_for\\_nintendo\\_switch\\_online\\_nes\\_games#:~:text=On%20July%2017%2C%20a%20Rewind,a%20section%20of%20the%20game!](https://www.nintendolife.com/news/2019/07/reminder_the_rewind_feature_is_now_live_for_nintendo_switch_online_nes_games#:~:text=On%20July%2017%2C%20a%20Rewind,a%20section%20of%20the%20game!). [Accessed: 01- Nov- 2021].
- [20]D. Coldewey, "Review: The NES Classic Edition and all 30 games on it", *Techcrunch.com*, 2016. [Online]. Available: [https://techcrunch.com/2016/11/07/review-the-nes-classic-edition-and-all-30-games-on-it/?guccounter=1&guce\\_referrer=aHR0cHM6Ly93d3cuZ29vZ2x1LmNvbS8&guce\\_referrer\\_sig=AQAAG\\_t02QVjUQ7yMLz7ZL7oflN2vOIEXnbXmuNW2VqYsXT-quADLYzcvA8ol2FxaustMsBvrY5KE2AcCEjuhEG151ne-kYndO6pZcpQXQWyRd27T7Cy\\_U4WEyd2ulAYAozg\\_UpfPZePkxy8ihjwfyXPEadzH9pMFN1IF2aQFwSwq](https://techcrunch.com/2016/11/07/review-the-nes-classic-edition-and-all-30-games-on-it/?guccounter=1&guce_referrer=aHR0cHM6Ly93d3cuZ29vZ2x1LmNvbS8&guce_referrer_sig=AQAAG_t02QVjUQ7yMLz7ZL7oflN2vOIEXnbXmuNW2VqYsXT-quADLYzcvA8ol2FxaustMsBvrY5KE2AcCEjuhEG151ne-kYndO6pZcpQXQWyRd27T7Cy_U4WEyd2ulAYAozg_UpfPZePkxy8ihjwfyXPEadzH9pMFN1IF2aQFwSwq). [Accessed: 01- Nov- 2021].
- [21]M. Kamen, "Nintendo SNES Classic Mini review: amazing games marred by hardware oversights", *WIRED UK*, 2017. [Online]. Available: <https://www.wired.co.uk/article/snes-mini-classic-review>. [Accessed: 01- Nov- 2021].
- [22]"PlayStation® Cumulative Production Shipments of Hardware | CORPORATE INFORMATION | Sony Computer Entertainment Inc.", *Web.archive.org*, 2011. [Online]. Available: [https://web.archive.org/web/20110722094946/http://scei.co.jp/corporate/data/bizdataps\\_e.html](https://web.archive.org/web/20110722094946/http://scei.co.jp/corporate/data/bizdataps_e.html). [Accessed: 01- Nov- 2021].
- [23]P. Cohen, "Study: PSP UMD outlook 'bleak'", *Macworld*, 2006. [Online]. Available: <https://www.macworld.com/article/179784/umd.html>. [Accessed: 01- Nov- 2021].

[24]D. Reisinger, "Sony: PS3 is hard to develop for--on purpose", *CNET*, 2009. [Online]. Available: <https://www.cnet.com/home/smart-home/sony-ps3-is-hard-to-develop-for-on-purpose/>. [Accessed: 01- Nov- 2021].

[25]R. Whitwam, "Sony's PlayStation Now uses custom-designed hardware with eight PS3s on a single motherboard - ExtremeTech", *ExtremeTech*, 2014. [Online]. Available: <https://www.extremetech.com/gaming/175005-sonys-playstation-now-uses-custom-designed-hardware-with-eight-ps3s-on-a-single-motherboard>. [Accessed: 01- Nov- 2021].

[26]A. Vjestica, "How to transfer PS4 save data to PS5", *TechRadar*, 2020. [Online]. Available: <https://www.techradar.com/uk/how-to/how-to-transfer-ps4-save-data-to-ps5>. [Accessed: 01- Nov- 2021].

[27]A. Webster, "PlayStation Classic review: a journey through the exciting, awkward days of early 3D games", *The Verge*, 2018. [Online]. Available: <https://www.theverge.com/2018/11/27/18112685/playstation-classic-review-3d-games>. [Accessed: 01- Nov- 2018].

[28]S. Hollister, "Xbox One will not be backwards compatible with Xbox 360 games", *The Verge*, 2013. [Online]. Available: <https://www.theverge.com/2013/5/21/4350662/new-xbox-has-no-backwards-compatibilty>. [Accessed: 01- Nov- 2021].

[29]R. McCaffrey, "The Untold Story of Xbox One Backwards Compatibility - IGN", *IGN*, 2017. [Online]. Available: <https://www.ign.com/articles/2017/10/23/the-untold-story-of-xbox-one-backwards-compatibility>. [Accessed: 01- Nov- 2021].

[30]R. Leadbetter, "Inside Xbox Series X: the full specs", *Eurogamer.net*, 2020. [Online]. Available: <https://www.eurogamer.net/articles/digitalfoundry-2020-inside-xbox-series-x-full-specs>. [Accessed: 01- Nov- 2021].

[31]J. Conley, E. Andros, P. Chinai, E. Lipkowitz and D. Perez, "Use of a Game Over: Emulation and the Video Game Industry, A White Paper", *Northwestern Pritzker School of Law Scholarly Commons*, 2004. [Online]. Available: <https://scholarlycommons.law.northwestern.edu/njtip/vol2/iss2/3>. [Accessed: 01- Nov- 2021].

[32]N. Carpenter, "Nintendo awarded \$2.1M in pirated games lawsuit", *Polygon*, 2021. [Online]. Available: <https://www.polygon.com/22462914/nintendo-lawsuit-2-million-damages-rom-universe-pirated-games>. [Accessed: 01- Nov- 2021].