

**ICON College of Technology and Management**

**Department of Information Technology, BCS Hons Computing**

**Accredited by Falmouth University**

**Bachelor Dissertation and Final Project**

**Music Education and Entertainment System: A Digital Guitar Instrument Console with an Integrated Web Application**

**Tivadar Debnar**

**ID No: 17663**

**Session: September 2022**

**Academic Supervisor: Dr Mohammed Jaffer Hasan**

*A person playing a guitar

Description automatically generated with low confidence*

***Cover | Playing Alone on the Road***(Anon, n.d.)

*"*I long for instruments ob*e*dient to my thought and which, with their contribution of a whole new world of unsuspected sounds, will lend themselves to the exigencies of my inner rhythm.*"*

*Edgard Varese, 1917*

Table of Contents

[1. | Introduction 4](#_Toc116894318)

[1.1 | Inspiration 4](#_Toc116894319)

[1.2. | Outlining the Aim and Objectives 5](#_Toc116894320)

[1.2.1. | Existing Technologies 5](#_Toc116894321)

[1.2.2. | Comparison 7](#_Toc116894322)

[1.2.3. | Gap in Current Technologies 7](#_Toc116894323)

[1.3. | Project Aim and Objectives 8](#_Toc116894324)

[1.4. | Requirements 9](#_Toc116894325)

[1.4.2 | Constraints 14](#_Toc116894326)

[1.5.2 | Cost, Marketability and Target Users 14](#_Toc116894327)

[1.5.1 | Feasibility and Opportunities for Business Success 16](#_Toc116894328)

[1.5.3 | Risk Management and Analysis 17](#_Toc116894329)

[2. | Project Management 17](#_Toc116894330)

[2.1. | Relevant Methodologies 17](#_Toc116894331)

[2.1.1. | Linear Methodologies 17](#_Toc116894332)

[2.1.2. | Iterative Methodologies 17](#_Toc116894333)

[2.1.3. | Critical Comparision of Methodologies 17](#_Toc116894334)

[2.2. | Justification of Applied Methodology 17](#_Toc116894335)

[2.3. | Work Breakdown Structure 17](#_Toc116894336)

[2.4. | Project Phases and Schedule 17](#_Toc116894337)

[2.5. | Tools 17](#_Toc116894338)

[2.5.1. | Hardware Technologies 17](#_Toc116894339)

[2.5.2. | Software Technologies 17](#_Toc116894340)

[3. | Literature Review 17](#_Toc116894341)

[3.1. | Terminology 17](#_Toc116894342)

[3.2. | Digital Music 17](#_Toc116894343)

[3.3. | Console Design Research 18](#_Toc116894344)

[3.3.1. | Calculating Fret Distances 18](#_Toc116894345)

[References 22](#_Toc116894346)

[Bibliography 22](#_Toc116894347)

[Appendices 22](#_Toc116894348)

# 1. | Introduction

We, Earthly creatures, are very fortunate. Extremely few places in the vast vacuum-filled Universe have a suitable medium that supports audio signals to travel. However, here on Earth, sound vibrations can move through the atmosphere, providing information about our environment. As a result, mammalian evolution adapted to transform soundwaves into electrical signals, genetically engineering us to detect sounds. Hearing sounds increased our survival chances by identifying danger outside our visual zone and extending our communication channels.

Even though humans are not the only species communicating by creating sounds, we discovered a way of self-expression that conveyed a broader spectrum of emotional range beyond mere spoken words; music. And from as early as 40000 years ago, music has played an essential part in our everyday life. Our innate musicality drove us to experiment with new ranges of sounds, inventing the primary types of instruments. Ideophones (clapping and bells), membranophones (drums), aerophones (flute), and most importantly, chordophones (harp and guitar).

Although the exact origin of the modern guitar is debated, the instrument is already mentioned in the Bible, and it can be traced back to the Greek kithara κιθάρα and Arabic qitharah قيثارة words. By the 17th century, it became popular among amateurs. With the advent of the jazz age, the electric guitar's success elevated its status to become the instrument of virtuosos and rock stars. However, this is not the final step on the guitar's evolution ladder. The modern digital era opened opportunities to combine the latest technology with musical skills. This project's goal is to bring digital technology, musical entertainment, and education under the same roof.

## 1.1 | Inspiration

I hold in my hand my old buddy, Gabriel's Guitar Hero. Again, I am ready for the next round; this time determined to overdo his performance. Little did I suspect that years of sketchy guitar practices on my side would not score against a seasoned hero like him. After several failed attempts to show off my talent, he concluded that even though I had guitar experience, rhythm sense, and some music theory in my pocket, my chances of winning against him were astronomical as a first-timer.

How about him, I asked myself, what type of guitarist would he make, with all those hours of playing the virtual guitar console? The answer came weeks later when he visited me, and I handed him my electric guitar and taught him the intro of a song I knew he liked. Soon enough, he could play a simple piece surprisingly well, though. So, I asked him.

- Why do you waste your time practising an imaginary instrument? You'd become a great guitarist by this time.

- You'll see me playing when they invent guitars for the console. – He answered with a smirk on his face.

Since then, I have been thinking about the talented people wasting their time playing on five plastic buttons and a strum bar. If I could create a lightweight device that resembles an actual guitar, I would be able to develop an online interface that is free, available for everyone, vendor-independent, and educational. I am confident it would be at least as attractive an entertainment option as playing Guitar Hero. Well, the time has come to wipe off the smirk from Gabriel's face; he will be the first to play.

## 1.2. | Outlining the Aim and Objectives

To create a bespoke and successful entertainment or educational product, one must set themselves apart from the crowd and clarify the aim and objectives of the project in reflection of the existing market. Hence, before proceeding to any project specification, we must explore the current gaming and musical entertainment devices that might be relevant to our project. A brief precautionary research will prevent us from reinventing the wheel and will refine the project's outline more accurately.

### 1.2.1. | Existing Technologies

**Guitar Hero**

As this pop culture phenomenon was the initial inspiration behind the project's idea, it might as well serve as a perfect starting point. Harmonix Music System, the former owner of Guitar Hero, defined the project in its patent as "*a simulated musical instrument that may be used to alter the audio of a video game*" (Chrzanowski, 2015). It was first released in 2005 and has seen several iterations since then. The product features five fret buttons, a strum bar, a whammy bar, and some additional control buttons relevant to Xbox's console interface. One of the main limitations of this console is the restricted number of fret buttons and the single strum bar, which prevents it from being used as an authentic educational device. However, the product shows several similarities to our project idea; therefore, some parts of it might be a perfect blueprint for our hardware prototype, such as its hollow, emptied plastic body.

**Diagram

Description automatically generated**

Figure | Guitar Hero Controller Layout (https://fccid.io/VFIBW95123805/User-Manual/Users-Manual-814804, 2022)

**MI Digital Guitar**

Numerous products have attempted to bring this acoustic instrument closer to the digital world; thus, our project is not a unique invention in this aspect. One of the most prominent and promising technologically-enhanced instruments is the MI Digital Guitars from Magic Instruments, which is currently in the prototyping phase. MI will have an excellently smooth, modern design and will be a stand-alone instrument rather than an entertainment console because it can be run on an amplifier. Among the digital devices on the market, the MI's layout resembles the most to an actual guitar because of its fret design and built-in digital strings. Unfortunately, the device is only meant to teach rhythm and fundamental chord progressions and lacks features of finger-style playing or riffs. Although the instrument is not in the premium price category, it can cost as much as a decent acoustic or electronic instrument.



Figure | MI Digital Guitar Series (https://www.digitalmusicnews.com/2016/08/01/mi-guitar-easy-to-learn, 2022)

**RockSmith**

A video game developed by Ubisoft brought music education to the next level. The game teaches acoustic, electric or bass guitar, adjusting its difficulty to the player. It utilises accurate, real-time play feedback and is currently one of the leading software technologies focusing on musical autodidactic training. However, RockSmith is exclusively a software solution, and the player must own a guitar that can be connected to the game through a real tone cable. Therefore, RockSmith can be considered a specialised training software rather than just a video game. Some of its characteristics resemble the iconic Guitar Hero, such as the practice play or riffs, while others, like tuning and uploading the players' music, are unique features.

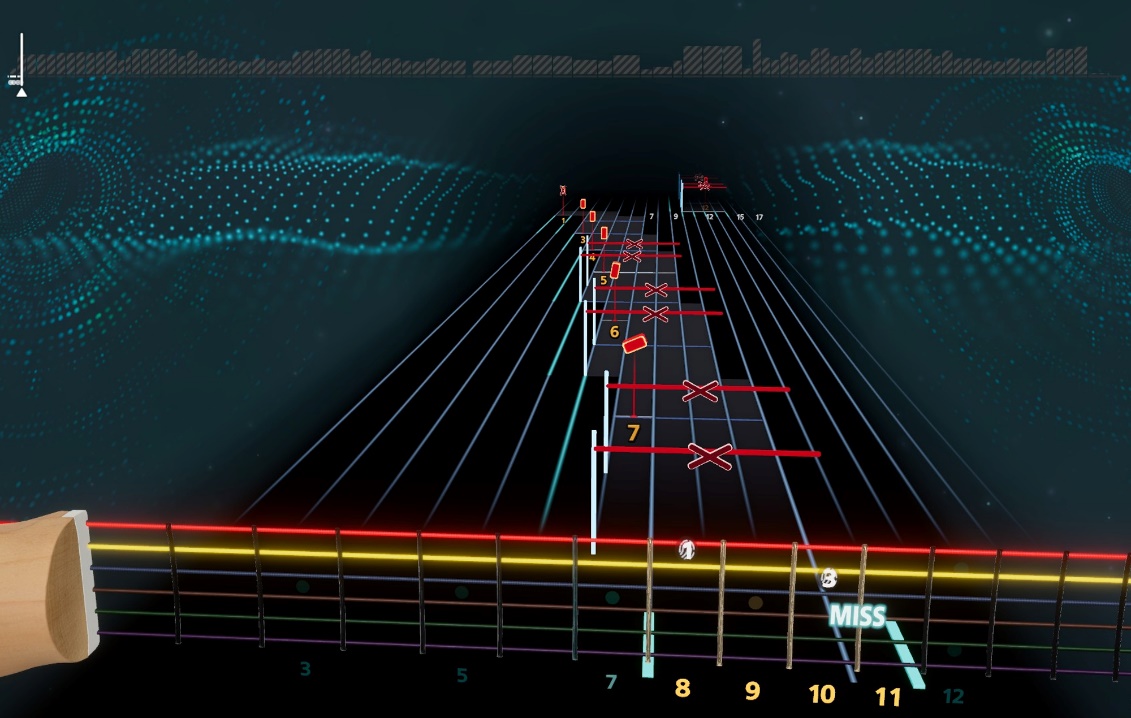


Figure | RockSmith Screenshot

### 1.2.2. | Comparison

Even though the examples mentioned above are only a fraction of the myriad of applications and devices currently available on the market, we can see that they all serve different purposes and have a specific user target. And because our project will mainly focus on playfully learning guitar riffs, chords and songs, it will be referred to RiffMaster from this point forward. The following table concludes our findings and some further specifications.

Table

Description automatically generated

Figure | Technology Comparision

## 1.2.3. | Gap in Current Technologies

As we can see, the presented options have decent coverage of usability and functionality that a novice guitarist would require. But unfortunately, they are sparsely isolated throughout several projects and products with distinct merits and limitations. Ideally, these features should be consolidated into one comprehensive product.

Firstly, while Guitar Hero has outstanding gameplay, it lacks a realistic user experience because of its console layout. Similarly, MI Digital Guitar has an excellent practical and artistic design and a more natural interface but inaccurate fret distances. Unfortunately, it cannot play notes, only limited chords through approximate button presses. Therefore it is an artificially invented music system, not unlike Guitar Hero. Thirdly, while RockSmith offers an exceptionally realistic application that teaches fundamental music skills, the user has to buy a decent-quality instrument to be able to start to play.

Finally, all these systems are vendor-specific, proprietary or licenced. As a software developer, I want to be able to write applications around a digital guitar instrument free from licences or the concern of litigations. Additionally, the device should have specifications, protocols and documentation available for every software developer in the community to produce a range of competition and further innovation. Most open-source projects enjoy eager crowds of professional and amateur developers' contributions; therefore, after the finalisation of the project, it will be shared for non-commercial use.

## 1.3. | Project Aim and Objectives

**Aim**

RiffMaster aims to offer a comprehensive simulated music experience, providing a hardware component with a naturalistic guitar layout and software to learn and play the instrument. The console device should have a minimalistic design to be affordable to a broader range of players. Apart from the actual device, the users are not required to own or buy any software licence; therefore, the software application will be written for the browser. Users should be able to connect the device to any computer via a USB cable and enjoy the application using the internet. The software will support the requirements of learning the instrument from the basics, and users should be able to track their progress after creating an account.

**Objectives**

**Objective 1**: Create a digital guitar device with a layout that accurately simulates the instrument's mechanism. The instrument's look, dimensions, and operability must be of a guitar, while the materials used may differ and be similar to a mock guitar. However, the guitar console's neck and the frets' distances must translate to a real guitar's exact proportions to enhance the players' precision in muscle memory and help them gain an easily transferable skill.

**Objective 2**: Ideally, six strings, alternatively six strum bars, may be used to activate a note, and follow-up research and experimentations will be included in the document regarding the design decision. Guitars are polyphonic instruments; consequently, one or more strings may be played simultaneously. If string activity is detected, the device should communicate the uppermost active frets position on the respective strings. The activated note must be transmitted through a USB port using a well-defined, simple protocol. The hardware must be safe to use and in accordance with safety regulations.

**Objective 3**: Design and develop a software application that accepts, detects and listens to user inputs from the device through a USB port without interrupting keyboard events. Integrate these inputs into the web app similarly to DOM key event states, such as note started, playing, and stopped. Optimise the application to accept and process simultaneous input information asynchronously.

**Objective 4**: Build a user-friendly frontend web application with a home page with sign-up and login options. Validate user login, and after successful sign-in or login, the following options should be available:

* **Jam Option**: the user can freely play the RiffMaster device and listen to the generated music,
* **Compose option**: the user can record the device, and the produced piece is translated into tablature notation, which can be manually edited, saved, played, or deleted.
* **Practice Option**: the user can load a tablature or follow a tutorial, play along with a song, and practice at different speeds. Different sections of the music may be selected for repeated practice.
* **Play Option**: the user can play a piece of selected music. The application will score the performance according to accurate real-time feedback, considering the player's number of mistakes in note accuracy or rhythm precision.
* **Chords Explorer**: chords will be clickable throughout the application. When pausing a running session, the player may check chords.

Finally, create a restricted number of demo songs and tablatures to test the prototype and the application.

**Objective 5**: Build a backend application that reflects a simple business model that will serve the web app. The business model must be limited to functionality constraints, as the project focuses on technical solutions rather than business implementations. The backend should communicate to a database and store user information, such as user name, songs, tablatures and scores.

## 1.4. | Requirements

Requirements are as crucial for large-scale projects as for smaller or individual ones because they concisely and unambiguously capture the project's parameters. "*Understanding user requirements is an integral part of information systems design and is critical to the success of interactive systems"* (Maguire, 2002)*.* In our scenario, the most important ones are user and system requirements, classified as critical, moderate and optional categories.

1. **Prerequisites**
   1. The user must have access to a RiffMaster console.
   2. The user must have a desktop or laptop with an available USB-A socket.
   3. A USB-A/B cable connects the host to the console.
   4. The user must have internet access.
   5. The application will be publicly accessible at http:\\www.tschiboka.co.uk\projects\riffmaster\index.html web domain address.

Diagram

Description automatically generated

Figure | RiffMaster's Fundamental Concept Diagram

1. **Hardware**
   1. There are 20 fret buttons in six rows, six strum bar switches and a power switch allocated on the guitar console.
   2. The user interacts with the website with traditional mouse and keyboard inputs.
   3. The user interacts with game functionalities with the guitar console.
   4. A 5V USB connection powers the console, and no batteries are required to operate the hardware.
   5. Interacting with the strum bar does not affect the application unless the user is in the practice, compose, jam or play options.
2. **Software**
   1. The application is a web-based Graphic User Interface.
   2. The application is a supporting software for the RiffMaster guitar console.
   3. The software runs on all major vendor browsers (Edge, Firefox, Chrome, Safari, Explorer).
   4. The application uses a dark theme and bright buttons for better accessibility.
   5. Upon reaching the landing page, there is a sign-up and a login option.

Line chart

Description automatically generated

Figure | Landing Page (Appendix/Wire Frames Landing Page.drawio)

1. **Authentication**
   1. The user must sign in to access the application functionality.
   2. If the user has no account, they need to create one:
      1. Pressing the sign-up button redirects the user to a registration form.
      2. The registration form has a first name, last name, avatar, email, password, confirm fields, and a submit button.
      3. The registration form's submit button is disabled until all fields are filled in.
      4. The registration page does not submit invalid forms.
      5. Invalid form fields are highlighted, and error messages specify the reason for the invalidity.
      6. A first and last name field is invalid if empty or non-alphabetic values are provided.
      7. An email address is invalid if it does not match the standard email format requirements.
      8. A password is invalid if less than eight characters or more than 32 characters.
      9. A password requires lowercase, capital letters, and numbers to pass validation.
      10. A confirm password field must completely match the password input to be valid.
      11. The terms and conditions check box must be checked to submit the form.
      12. The check box label has a link to the terms and conditions page.

Diagram

Description automatically generated

Figure | Wireframes Registration (Appendix/Wireframes – Registration.drawio)

* 1. If the user submits the registration form, the submit button will immediately turn disabled again, preventing submitting multiple forms.
  2. The user will see a notification informing them that an activation email will be sent to the email address.
  3. When the user clicks on the email link, it will activate their account and redirects them to the login.
  4. If the user has an account, they need to sign in.
     1. The login form has email and password fields, and a submit button.
     2. The submit button is disabled until both the email and password input fields contain content.
     3. Upon submission, the server will match the email with the password and sends a response to the website.
     4. If the email has no matching password, a message will be provided.
     5. After the fifth unsuccessful login attempt, the form will lock down for five minutes, and the user must wait.
     6. If the password matches the one provided for the email address, the user will be signed in and redirected to the home page.

A picture containing graphical user interface

Description automatically generated

Figure | Login Wireframe (Appendix/Wireframe – Login.drawio)

1. **Home Page**
   1. The home page displays Profile Information, a Playlist, and a menu with six items: Practice, Jam Session, Play, Compose, Chords Explorer and Sign out.
   2. **Profile Information**
      1. The Profile includes the user's chosen avatar and name.
      2. The Profile displays the total scores achieved by the user and the user's ranking.
      3. The Profile displays a list of the last played songs in chronological order.
      4. The last played songs' information includes the author, title and the scores achieved on a particular piece.
      5. If there are more songs than the available screen real estate, a scrollbar is shown.
   3. **Playlist**
      1. The Playlist consists of Album cards.
      2. An Album card consists of the album cover, author and title.
      3. By clicking an Album cover, the user is directed to the Play page of the chosen song.
      4. If the albums run out of available space, a scrollbar appears.
   4. **Main Menu**
      1. The main menu's colour sequence will follow the same sequence as the string colour codes: yellow, orange, pink, purple, azure, and aquamarine.
   5. The footer displays copyright information with the current year.

Diagram

Description automatically generated

Figure | Home Wireframe (Appendix/Wireframes – Home.drawio)

1. **Play Menu Option**
   1. The Play menu option lets the players compete with others.
   2. The player hits start, and the application counts back down from three to one.
   3. The music starts after the countdown.
   4. The player can pause, resume and stop the game by pressing the corresponding buttons.
   5. In the header, the author and title are displayed.
   6. **Tablature**
   7. One or two tablature lines will represent the music that is playing, and it has the following elements:
      1. Base Chord: the base chord is clickable when the game is paused and shows the fret and finger positions.
      2. Finger Positions: Each line represents a string, and the numbers represent fret numbers.
      3. Rhythm Notation: standard notation assuming 4/4 at this development phase.
   8. The Guitar Animation is the screen representation of the console:
      1. It shows frets, finger positions and strum bars.
      2. The active finger positions and strums are displayed alongside the music.
      3. If the player fails to hit the appropriate fret buttons and the strum bars at the precise timing, a "missed" message is displayed.

A picture containing table

Description automatically generated

Figure | Play Wireframe (Appendix/Wireframes – Play.drawio)

1. **Scoring and Saving**
   1. The user can see the current score while playing.
   2. If ten, twenty or thirty consecutive notes are correctly strummed in a streak, a multiplier is given to the points.
   3. Time precision will add another multiplier depending on the player's timing.

Graphical user interface

Description automatically generated

Figure | Pointing System

* 1. The user can see the correct notes consecutively played as the streak in the footer.
  2. Every missed note will reset the streak to zero, and the streak multipliers to one.
  3. Every strum stroke resets the precision according to the accuracy of the last played note.
  4. The user can see their highest score on a particular song. The top score by the user community is shown in the bottom right corner.
  5. **Score meters**
     1. The top score meter represents the best general accuracy ever achieved on the song.
     2. The middle score meter gives a general value to the average score on the current performance.
     3. The bottom score meter has the accuracy of the last ten strums on the background and the last strum on the front.
     4. The general score is the product of note accuracy (binary) and the time precision percentage.
  6. **Saving Performance**
     1. Performance and score information is saved on the database when the game is over.
     2. The general date, score and performance are saved to the list of scores for the album.
     3. The score is rewritten if the score is higher than the personal best.
     4. The all-time best score is rewritten if the user breaks the record.

1. **Practice, Jam, Compose Menu Options (similar features)**
   1. **The practice** option is identical to the Play option but does not save the performance.
   2. A warning is placed to notify the user with an optional "Don't show again" checkbox.
   3. **Jam Session** has only the guitar animation that responds to strums and keynote presses.
   4. Strums produce the corresponding guitar sounds.
   5. **Compose Menu** option records the console input and produces sound.
   6. The notes are translated on the tablature when the player presses the record button.
   7. The notes are cleared when pressing the delete button.
2. **The chord Explorer** menu option listschord patterns in chronological order.

### 1.4.2 | Constraints

**Scope**

"*The scope of the study refers to the parameters under which the study will be operating*" (Simon, 2013). This project's scope is to conceptualise and prototype hardware and software components; therefore, the research study extensively focuses on technical exploration, design and delivery of requirements. Consequently, some exciting topics, such as refining the target audience, possible business models, brand design, marketability, mass manufacturing, industrial design, patent collision and litigation, will be investigated briefly but will lack in-depth research.

**Limitations**

"*Limitations are matters and occurrences that arise in a study which are out of the researcher's control*" (Simon, 2013). This project is a simplified version of a complete video game system with a strict timeframe; therefore, several features will be limited or omitted.

* The console will have a coarse functional design lacking the polished sophistication of an industrial end product.
* Similarly, the console is restricted to wired communication, as developing WiFi solutions would introduce a strain on the timeframe.
* The game application will be designed for desktop screen sizes, and mobile development is entirely out of the prototype's scope.
* Some nuances of user stories are intentionally omitted, such as:
  + Retrieval of forgotten passwords,
  + Deleting a user account (deactivation option may be provided),
  + Social features: multiplayer mode, messaging, following players, seeing other players' performance,
  + Statistics and data visualisation of users' performance,
  + Sophisticated high-resolution gameplay animations.
* A limited number of sample songs (minimum five),
* The tabular notation format may lack standards or details to conform to the gaming environment.

Finally, the hardware and software prototypes are unsuitable or intended for industrial or commercial use, and their sole purpose is to present a proof of concept.

### 1.5.2 | Cost, Marketability and Target Users

**Cost**

The project's cost details will refer to the prototype model rather than a merchantable end product. Unlike the hardware components, the software solution requires no additional costs (such as acquiring new software licences). However, building the game console requires several items to be purchased.

Table

Description automatically generated

Figure | Cost Estimation(Appendix/Cost-Table.xlsx)

The total estimated cost of the prototype will be around £240; however, this price includes buying tools, for instance. This price would be significantly lower in mass production. Depending on the materials' quality, it would cost between £40 to £100, mainly because cheaper microchip solutions may replace the Arduino microcontroller. We use only a fraction of Arduino's functionality.

**Target Users**

If the project is commercialised, the following groups might be an ideal focus target for this product:

* Guitar Hero players who want to extend the difficulty of the game,
* Guitar Hero players who would like to take on some instrument skills,
* Novice guitarist in need of guided instructions,
* Students wanting to explore music entertainingly,
* Guitarists that might need a silent instrument (playing at night),
* Beginner composers with a lack of standard music notation (rock bands),
* People who want to improve their fine motor skills.

**Marketability**

Consequently, the target audience may be a good indicator of the device's success in the market. For instance, MI Digital Guitar raised over £360.000 in funds for product development from their future target community (14.10.2022). This success is promising because our target audience intersects with MI's community. Another relevant market intersection is music gaming. Even though comparing the project to giants like Guitar Hero or Rock Band would seem overly ambitious, it is worth understanding that this branch of the industry has been on the decline for a decade.

Chart, line chart, scatter chart

Description automatically generated

Figure | Guitar Hero Trendline (www.engadget.com, 2009)

The oversaturation of the simplified simulation game instruments on the market caused a severe decline in the industry. Unfortunately, this means that relying solely on gaming features would leave a narrow window of opportunity for success. Therefore, RiffMaster needs to emphasise authentic instrument digitalisation and build a social community rather than jump on the train of the existing gaming schemas.

### 1.5.1 | Feasibility

"*A feasibility study evaluates the project's potential for success; therefore, perceived objectivity is an important factor in the credibility of the study for potential investors and lending institutions*" (Ibrahim Rihan, 2022). And sticking to the gaming industry would navigate the project towards dangerously thin ice. Therefore, one of the most intriguing and promising aspects of RiffMaster would be its educational and social impact. As digital interactions and education become ever more convenient and natural for our modern society, it opens up unchartered territories.

Such territory is the online fandom of music enthusiasts, composers and amateur players. Therefore, a feature-rich social web platform that integrates the product would not only increase the feasibility of the entire project but would be a novelty. Players could organise competitions, publish compositions, like others' compositions, follow each other, group in bands, communicate, teach, or post music instructions.

Naturally, this completeness is way out of the scope of a one-person project. Hence, we can state that RiffMaster as a video game would most likely fade into the giants' shadows. It is not designed to be a stand-alone digital instrument, but as an educational system, it has the potential to be impactful. However, as a comprehensive social and educational gaming platform, it would stand out of the crowd, with a high potential to become successful.

Diagram

Description automatically generated

Figure | Feasibility Diagram (Appendix/Feasibility-Diagram.drawio)

### 1.5.3 | Risk Management and Analysis

Every hardware or software project is inherently risky, and those risks must be identified and addressed. "*To identify risks, the following techniques can be used: brainstorming, work breakdown analysis, risk breakdown structure, checklists, among others*" (Menezes Jr, 2013.). Hence, we will collect, identify and propose actions for the project in the following table:

Graphical user interface, table

Description automatically generated

Figure | Risk Assessment (Appendix/Risk-Assessment.xlsx)

For this project, scheduling is the risk with the highest probability and severity; therefore, the design, experimentation and development phase will start as early as possible to avoid delays in delivering the hardware or any software features.

# 2. | Project Management

We can approach the workflow of our project tasks in several ways; hence, if we examine the relevant software development methodologies (SDM), we can select the best solution that suits our needs. After identifying the chosen method, we can create a detailed timeline of the project development steps.

## 2.1. | Relevant Methodologies

Software or other technical developments, such as hardware development, can be divided into linear and iterative categories. However, both linear and iterative approaches have distinct benefits and limitations, which can determine the project's outcome. Additionally, when selecting our development method, we must remember that this project consists of hardware and software components and their integration, and there might be a need to use a mixed methodology.

### 2.1.1. | Linear Methodologies

When engineers discuss sequential or linear software development mode, they essentially refer to the Waterfall model. The Waterfall is one of the oldest engineering models dating back to the 1960s. "*Waterfall Model predominately emphasises on the freezing of requirement specifications or the high-level design very early in the development life-cycle, prior to engaging in more thorough design and implementation work*" (Van Casteren, 2017). However, this inflexibility is often problematic because Waterfall may proceed towards only one direction, and the project cannot be further modified after the specification and design.

Graphical user interface, application

Description automatically generated

Figure | Waterfall (https://www.softwaretestinghelp.com/what-is-sdlc-waterfall-model)

This method assumes that the project requirements are accurate, permanent and fully complete before the actual development starts. Current software development companies tend to lean towards more agile solutions; however, as this method is traditionally rooted in manufacturing, this approach might be needed for the hardware components.

### 2.1.2. | Iterative Methodologies

#### 2.1.2.1. | Agile

Agile is not a methodology, as it does not stipulate the exact steps of the development phases. It, instead, can be considered a philosophical background for writing successful software. The agile manifesto is mere 68 words, yet its values and principles completely revolutionised traditional software development. For example, one principle states that "*our highest priority is to satisfy the customer through early and continuous delivery of valuable software*" (Fowler, 2001). This continuous delivery, however, cannot be achieved through sequential approaches. Additionally, Agile lacks the solid documentation of traditional development.

Chart

Description automatically generated

Figure | Agile Manifesto (https://agilemanifesto.org)

As soon as Agile solidified in the developer community's collective awareness, several methodologies came to light based on its principles, such as Scrum, Extreme Programming (XP), Lean, Feature-Driven Development (FDD), Spiral, or DevOps. Even though many of these methods may be tailored or trivialised to suit a one-person project, they essentially revolve around managing processes for a team of developers and departments. For instance, Scrum requires Scrum Master, Product Owner and Developers, or XP increases quality by pair programming. Therefore, we must discover methodologies more aligned with a single-developer project scenario.

#### 2.1.2.2. Test-Driven Development

Waterfall methodology tests code and business functionality retroactively, which carries a potential risk of failing mitigating problems early. TDD, an agile practice, "*requires writing automated tests prior to developing functional code in small, rapid iterations*" (Janzen, 2005). The functions must be granulated into individual testable units to achieve such automation.

A picture containing shape

Description automatically generated

Figure | TDD Algorithm (https://developer.ibm.com/articles/5-steps-of-test-driven-development, 2022)

As we can see, TDD is a test-first approach. Units and functions are tested deterministically, avoiding any side effects.

#### 2.1.2.3. KanBan

The Japanese company Toyota introduced the KanBan system in the 1950s to help visualise development work. "*KanBan means card, literally, a visible record used as a means of communication, of conveying ideas and information*" (Esparrago Jr, 1988). And soon, the software development industry introduced this method to improve efficiency, where the workflow is divided into requested, in-progress and done categories in a board system. One fundamental characteristic of the methodology is that it limits the number of in-progress cards to ensure manageability by reducing multitasking.

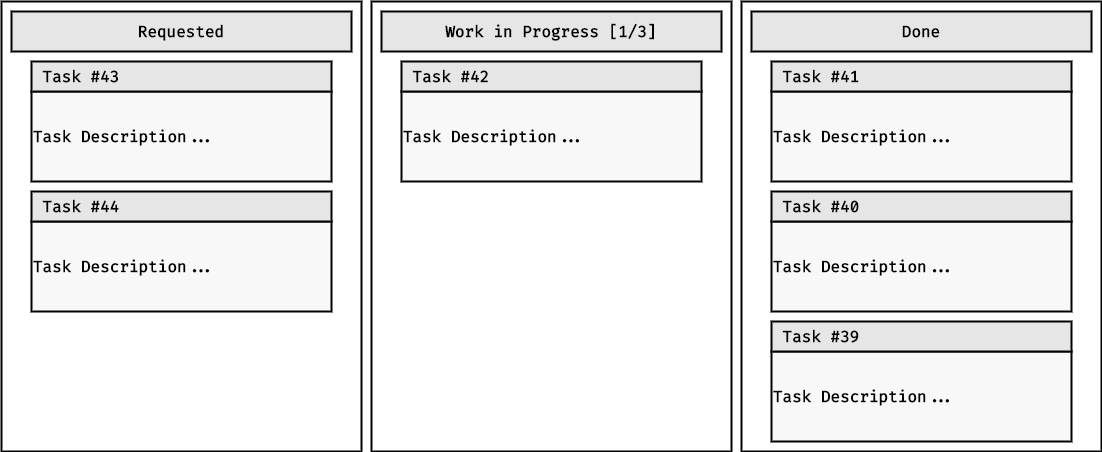


Figure | Kanban (https://kanbanize.com, 2022)

## 2.2. | Justification of Applied Methodology

"*When an organisation states that it uses a particular methodology, it often applies a combination of smaller, finer-grained methodologies on a project scale instead*" (Janzen, 2005). Similarly, this project will not be an exception. The project's design and development will be distinguished into hardware and software development and handled separately. The hardware development will follow a linear progression, similar to the Waterfall method, because our requirements are fixed, no changes will be expected, and the manufacturing has well-defined stages. However, most software development will be halted until the fully-functional console is ready and tested. The software engineering will follow TDD principles to ensure quality, and tests will be automated with Jest. The software features will be divided and developed iteratively. Both software and hardware will apply KanBan because its simple and practical approach is well-suited for a single-developer project and can support the level of organisation required in a small-scale environment. The maximum number of in-progress cards will be set to two to mitigate the accumulation of unfinished tasks.

## 2.3. | Work Breakdown Structure

Diagram

Description automatically generated

Figure | Console WBS

Diagram, engineering drawing

Description automatically generated

Figure | Application WBS

## 2.4. | Project Phases and Schedule

Chart

Description automatically generated

Figure | Gantt Chart

## 2.5. | Tools

# 3. | Literature Review

## 3.1. | Terminology

## 3.2. | Digital Music

## 3.3. | Console Design Research

### 3.3.1. | Calculating Fret Distances

As outlined in the specifications, the console layout must simulate the guitar experience as authentically as possible. Current devices, such as the MI Digital Guitar, do not account for the distances between the frets and button cap sizes. Therefore, we need to calibrate the console neck to a natural proportion where the 12th fret is halfway to the scale length between the nut and the saddle. The fret positions may be calculated by dividing the remainder of a fret-saddle distance by 17.817 recursively. For our initial calculations, we can base our measurements on the length of a Guitar Hero console.

**Text

Description automatically generated**

Figure | Calculate Fret Distances (appendix/fret\_distances.py)

The resulting array has the following values:

Table

Description automatically generated with low confidence

Figure | Fret Distances

These distance values can be used to create the first approximate diagram of the interface's physical layout. However, this layout does not yet consider a wiggle room between the buttons, which should ideally be at least one millimetre.

A picture containing shoji

Description automatically generated

Figure | Fret Distances on The Console Guitar's Neck

**3.3.2. |** Key Switch Interface Design Concepts

To build a successful guitar system based on time precision, we need to be able to detect inputs from the user interface accurately and efficiently. From an electrical engineer's perspective, our console can be abstracted down as an intricate keyboard input that consists of an array of key switches, similar to a conventional computer keyboard. "*Depending on how individual switches are connected, mechanical keypads are commonly available in two forms – matrix and common bus*" (Dave, n.d.).

#### 3.3.2.1. | Keyboard Buses

In our scenario, we need six times 20 fret buttons and six strum switches, totalling 126 switches. Unfortunately, connecting switches directly to pins would be inelegant even if we could find a microcontroller with 126 digital pins. The fundamental way of sequential wiring design is to use a common bus.

Calendar

Description automatically generated

Figure | Common Bus Consol Interface Wiring Schema (appendix/CommonBus.draw.io)

This solution in the above form is not ideal for our console; however, other linear solutions will be discussed in the following chapters.

#### 3.3.2.2. | Keyboard Matrix

User interfaces, such as keyboards and keypads, often use a keyboard matrix to consolidate a greater number of input switches to a limited number of microcontroller pins. "*When a key is pressed, a column wire makes contact with a row wire and completes a circuit. The keyboard controller detects this closed circuit and registers it as a key press*" (Dribin, 2000). For example, PC keyboards usually range from 63 to 105 keys, depending on the layout and the existence of a numerical pad. In the same way as conventional keyboards, the guitar interface can be arranged in a matrix. Meshing the switch wires would result in a drastically reduced digital pin requirement.

Diagram

Description automatically generated with medium confidence

Figure | Keyboard Scanning

#### 3.3.2.3. Debounce Mechanism

"*The physical action of pushing a button might require a half-second or so, so we tend to think in those terms. On the other hand, a digital circuit can react to a million of events in the same time frame*" (Warren, 2015). Because of the switch mechanisms, when a push button is pressed, it may register multiple interactions with the input device in a relatively short interval. In the case of a game console interface, it would result in a disastrous user experience. Various electrical solutions, such as flip-flops and Schmitt triggers, have been used to solve this problem. A "S*chmitt trigger circuit relies on changing the voltage or current threshold levels by means of positive feedback in the analogue loop*" (Kader, 2012), improving the immunity to analogue disturbances.

Diagram, engineering drawing

Description automatically generated

Figure | Schmitt Trigger (https://www.watelectronics.com, 2022)

However, we can solve debouncing using software engineering by measuring oscillation time. For example, Arduino microcontrollers have a millis function that "*returns the number of milliseconds passed since the Arduino board began running the current program. This number will overflow (go back to zero) after approximately 50 days*" (Arduino.cc, 2022). Therefore, we can prevent debouncing by measuring switch state changes. The time of the state changes should be recorded, and an intentional debounce delay should be applied to compensate for the noise. The change should be ignored when low-state changes happen in unreasonably short intervals.

Graphical user interface, text

Description automatically generated

Figure | Compensating Debouncing on Arduino (https://docs.arduino.cc/built-in-examples/digital/Debounce, 2022)

# References

Anon, n.d. *https://www.eclecticmusicatlanta.com/making-the-most-out-of-music-lessons.* [Online]   
[Accessed 02 10 2022].

Arduino.cc, 2. A., 2022. *Arduino.* [Online]   
Available at: https://docs.arduino.cc/built-in-examples/digital/Debounce,  
[Accessed 12 10 2022].

Chrzanowski, J. H. j. L. G. L. R., 2015. *Music Video Game with User Directed Sound Generation.* US, Patent No. 9061205B2.

Dave, V. E. R. B., n.d. *Reading Matrix and Common Bus Keypads.* s.l.:(PSoC 1)., Cypress AN2034..

Dribin, D., 2000. *Keyboard matrix help..* s.l.:s.n.

Eriksson, J., 2016. *Chord and Modality Analysis..* s.l.:KHT, School of Computer Science and Communication (CDC), Speech, Music and Hearing, TMH, Speech Communication and Technology.

Fowler, M. H. J., 2001. The agile manifesto.. *Software development,* 8(9), pp. 28-35.

Ibrahim Rihan, M., 2022. *The Feasibility Study.* [Online]   
Available at: https://www.academia.edu/29462582/The\_Feasibility\_Study  
[Accessed 14 10 2022].

Janzen, D. a. S. H., 2005. Test-driven development concepts, taxonomy, and future direction. In: s.l.:Computer, pp. 43-50.

Kader, W. R. H. M. M. a. B. M., 2012. *Advancement of CMOS Schmitt trigger circuits..* s.l.:Modern Applied Science, 6(12), p.51..

Maguire, M. B. N., 2002. *User requirements analysis. In IFIP World Computer Congress.* Boston, MA.: Springer.

Menezes Jr, J. G. C. a. M. H., 2013.. Defining indicators for risk assessment in software development projects. *Clei electronic journal,* 16(1), p. 11.

Simon, M. a. G. J., 2013. *Assumption, Limitations, Delimitations, and Scope of the Study (Doctoral Dissertation, Dissertation and Scholarly Research: Recipes for Success).,* s.l.: s.n.

Van Casteren, W., 2017. The Waterfall Model and the Agile Methodologies: A comparison by project characteristics. *Research Gate,* Volume 2, pp. 1-6.

Warren, G., 2015. *Exploring the Raspberry Pi 2 with C++..* s.l.:Springer.

# Bibliography

# Appendices

# Links

|  |  |  |
| --- | --- | --- |
| A close-up of a fetus  Description automatically generated with low confidence | GitHub | <https://github.com/tschiboka/RiffMaster> |
| Shape  Description automatically generated with low confidence | Project | <https://tschiboka.co.uk/projects/riffmaster/index.html> |