****

**Music Education and Entertainment Simulation System:**

**A Digital Guitar Instrument Controller**

**with an Integrated Web Application**

**ICON College of Technology and Management**

**Department of Information Technology**

**Bachelor's Degree Dissertation Project | BSc Hons**

**Accredited by Falmouth University**

**Academic Supervisor: Dr. Mohammed Jaffer Hasan**

**Tivadar Debnar**

**17663** | **2022-2023**

**Tivadar Debnar**

**17663** | **2022-2023**

**Tivadar Debnar**

**17663** | **2022-2023**

**Tivadar Debnar**

**17663** | **2022-2023**

**Tivadar Debnar**

**17663** | **2022-2023**

**Tivadar Debnar**

**17663** | **2022-2023**

**Tivadar Debnar**

**17663** | **2022-2023**

**Tivadar Debnar**

**17663** | **2022-2023**

**Tivadar Debnar**

**17663** | **2022-2023**

**Tivadar Debnar**

**17663** | **2022-2023**

**Tivadar Debnar**

**17663** | **2022-2023**

**Tivadar Debnar**

**17663** | **2022-2023**

**Tivadar Debnar**

**17663** | **2022-2023**

**Tivadar Debnar**

**17663** | **2022-2023**

**Tivadar Debnar**

**17663** | **2022-2023**

**Tivadar Debnar**

**17663** | **2022-2023**

*A person playing a guitar

Description automatically generated with low confidence*

***Playing Alone on the Road***

*"*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*

*Cover Image by* Eclectic Music Atlanta

Table of Contents

[1. | Introduction 6](#_Toc127837607)

[1.1 | Inspiration 6](#_Toc127837608)

[1.2. | Project Aim and Objectives 7](#_Toc127837609)

[1.3. | Requirements and Specifications 8](#_Toc127837610)

[1.3.2 | Constraints 13](#_Toc127837611)

[1.3.2 | Cost, Marketability and Target Users 13](#_Toc127837612)

[1.3.1 | Feasibility 15](#_Toc127837613)

[1.3.3 | Risk Management and Analysis 16](#_Toc127837614)

[2. | Project Management 17](#_Toc127837615)

[2.1. | Relevant Methodologies 17](#_Toc127837616)

[2.1.1. | Linear Methodologies 17](#_Toc127837617)

[2.1.2. | Iterative Methodologies 18](#_Toc127837618)

[2.1.3. | Justification of Applied Methodologies 19](#_Toc127837619)

[2.3. | Work Breakdown Structure 20](#_Toc127837620)

[2.4. | Project Phases and Schedule 22](#_Toc127837621)

[2.5. | Tools 22](#_Toc127837622)

[3. | Literature Review 23](#_Toc127837623)

[3.1. | Terminology 23](#_Toc127837624)

[3.2. | Music Notations 24](#_Toc127837625)

[3.3. | Research for Supporting the Aim and Objectives 25](#_Toc127837626)

[3.3.1. | Existing Technologies 25](#_Toc127837627)

[3.3.2. | Comparison 27](#_Toc127837628)

[3.3.3. | Gap in Current Technologies 27](#_Toc127837629)

[3.4. | Console Design Research 28](#_Toc127837630)

[3.4.1. | Calculating Fret Distances 28](#_Toc127837631)

[3.4.2. | Key Switch Interface Design Concepts 29](#_Toc127837632)

[3.4.3. | Analog Voltage Dividers 31](#_Toc127837633)

[3.4.4. | Debounce Mechanism 32](#_Toc127837634)

[3.6 | Audio in the Browser 34](#_Toc127837635)

[4. | Design 36](#_Toc127837636)

[4.1 | Controller Design 36](#_Toc127837637)

[4.1.1. | Physical Design and Electrical Housing 36](#_Toc127837638)

[4.1.2. | Imitating Gibson SE Les Paul Style Guitars 37](#_Toc127837639)

[4.1.3. | Left Hand Control Design: Fret Buttons 37](#_Toc127837640)

[4.1.4. | Electrical Layout 41](#_Toc127837641)

[4.1.5. | Right Hand Control Design: Body and Strum 43](#_Toc127837642)

[4.2. | Frontend Design 46](#_Toc127837643)

[4.2.1. | User Activities - Activity Diagram 46](#_Toc127837644)

[4.2.2. | User Interface – Sequence Diagram 46](#_Toc127837645)

[4.2.3. | User Experience 46](#_Toc127837646)

[4.2.4. | App State Management without React 46](#_Toc127837647)

[4.2.5. | JQuery Style DOM Access and Utility Functions 46](#_Toc127837648)

[4.3. | Backend Design 46](#_Toc127837649)

[4.3.1. | User Interaction 46](#_Toc127837650)

[4.3.2. | ER Diagram (Ideal and Actual) 46](#_Toc127837651)

[5. Development 47](#_Toc127837652)

[5.1. | Hardware Development 47](#_Toc127837653)

[5.1.1. | Physical Materials (Buying, Cutting) 47](#_Toc127837654)

[5.1.2. | Development Images and Challenges 47](#_Toc127837655)

[5.1.3. | The Microcontroller 47](#_Toc127837656)

[5.1.3.1 | Microcontroller Selection (Leonardo versus Due) 47](#_Toc127837657)

[5.1.3.2 | Microcontroller Programming 47](#_Toc127837658)

[5.1.3.2 | Communication Protocol (and Options: Binary, Textual, Numerical, Adv Disadv) 47](#_Toc127837659)

[5.2. | Backend Development 48](#_Toc127837660)

[5.2.1. | Schemas and Database 48](#_Toc127837661)

[5.2.2. | API Calls 48](#_Toc127837662)

[5.3. | Frontend Development 48](#_Toc127837663)

[5.3.1. | Material Collection 48](#_Toc127837664)

[5.3.2. | Theme and Digital Design (Adobe Photoshop, CSS Variables instead of Bootstrap, Font) 48](#_Toc127837665)

[5.3.3. | Registration and Login 48](#_Toc127837666)

[5.3.4. | Game Play and Practice Options 48](#_Toc127837667)

[5.3.5. | Manipulating Audio: Jam Session 48](#_Toc127837668)

[5.3.6. | Chord Translation and Display 48](#_Toc127837669)

[5.3.7. | Interpreting Tablature 48](#_Toc127837670)

[5.3.8. | Game Play and Practice Options 48](#_Toc127837671)

[6. | Testing 48](#_Toc127837672)

[6.1. | Hardware Testing 48](#_Toc127837673)

[6.1.1. | Interaction Testing 49](#_Toc127837674)

[6.1.2. | Performance Testing 49](#_Toc127837675)

[6.2. | Software Testing 49](#_Toc127837676)

[6.2.1. | Automated Testing 49](#_Toc127837677)

[6.2.2. | API Testing 49](#_Toc127837678)

[6.2.3. | Function Testing 49](#_Toc127837679)

[6.2.4. | Unit Testing 49](#_Toc127837680)

[6.2.5. | Interaction Testing 49](#_Toc127837681)

[6.2.6. | Input Testing 49](#_Toc127837682)

[6.2.7. | Requirement Test Sheet 49](#_Toc127837683)

[6.3 | User Acceptance 49](#_Toc127837684)

[7. | Reflection 49](#_Toc127837685)

[7.1. | Manufacturing 49](#_Toc127837686)

[7.2. | Project Management 49](#_Toc127837687)

[7.3. | Challenges 49](#_Toc127837688)

[7.4. | Critiques: Overall User Experience 49](#_Toc127837689)

[7.5. | Recommendations: Prototype 49](#_Toc127837690)

[7.6. | Recommendations: Commercial Development 49](#_Toc127837691)

[7.7. | Personal Development 49](#_Toc127837692)

[References 51](#_Toc127837693)

[Bibliography 51](#_Toc127837694)

[Appendices 52](#_Toc127837695)

[Links 52](#_Toc127837696)

# 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 increases 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 the 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, and this time, determined to overdo his performance somehow. Little did I suspect that decades 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 the simple melody surprisingly well, though. So, I asked him.

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

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

Since then, I have been thinking about how many talented people waste their time playing on unauthentic consoles with merely five plastic buttons and a strum bar. If I could create a lightweight device that resembles an actual guitar, I could 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 sneer from Gabriel's face; he will be the first to play it.

## 1.2. | Project Aim and Objectives

**Aim**

RiffMaster aims to offer a comprehensive and unique simulated music experience, providing a hardware component with a naturalistic guitar layout and software to learn and play the instrument. The controller 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 application will be written for the browser. Users should be able to connect the device to any computer via a USB cable and use the application through 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**: Design and develop a digital guitar controller with a layout that accurately simulates the instrument's mechanism. The controller'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 controller'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 corresponding string. 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 like 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 signup 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 note accuracy and rhythm precision mistakes.
* **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 and deploy a backend application that reflects a simple startup's real-world business model for prototyping the web application. The business model must be limited to functionality constraints, as the project focuses on hardware and software prototyping rather than business implementation. The backend should communicate to a database and store user information, such as user name, songs, tablatures and scores.

## 1.3. | Requirements and Specifications

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 are user and system requirements, which detail how the user interacts with the system and can be used to test the project.

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 (Appendix/Concept.drawio)

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; 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 signup 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 signup 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 the user achieves 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. A scrollbar is shown if there are more songs than the available screen real estate.
   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 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 (Appendix/Pointing.drawio)

* 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 after the game.
     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**
   1. List an extended list of chords (1000+).
   2. Display an accurate finger positioning for the selected chord.
   3. Draw a guitar neck chord representation (SVG) from the lowest non-zero pressed fret.
   4. Barre chords (when one finger interacts with multiple strings) should be displayed standardly.
   5. Users may select root note, chord type and base note for filtering.
   6. Clicking on a chord should display its composition and play out individual notes.

### 1.3.2 | 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 Wi-Fi 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.

### 1.3.2 | Cost, Marketability and Target Users

**Cost**

The project's cost details 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 in large-scale production. Furthermore, our chosen Arduino board will be underutilised because we use only a fraction of the microcontroller's wide range of functionality and capability.

**Target Users**

If the project was commercialised, the following group of people might turn out to be the prosperous focus target for this product:

* Guitar Hero players who want to extend the difficulty and authenticity of the currently available game system,
* Guitar Hero players who would like to take on some real-world instrument skills while being entertained,
* Novice guitarist in need of guided instructions or up-to-date musical translations,
* Students and game-oriented hidden talents wanting to explore music while having fun,
* Guitarists that might need a silent instrument while practising at night,
* Beginner composers or upcoming bands with a lack of standard music notation and knowledge of chords,
* People who want to improve their fine motor skills, rhythm-sense, and overall musical 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 industry branch has been declining 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 margin 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.

### 1.3.1 | 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, the more intriguing and promising aspects of RiffMaster would be to build it around a community of users that appreciate the educational and social impact of games. As digital interactions and education become ever more convenient and natural for our modern society, it opens 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 increase the feasibility of the entire project and be a novelty. Players could organise online competitions, play simultaneously, publish compositions, like others' compositions, follow each other, group in bands, communicate, version and fork band music, 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. RiffMaster is not designed to succeed only to be a stand-alone digital instrument but also as an educational system and social platform; therefore, it has the potential to be impactful. Moreover, as a comprehensive social and educational gaming platform, it would stand out of the crowd, with a high promise to become successful.

Diagram

Description automatically generated

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

### 1.3.3 | 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, text, application, email

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, 2022)

This method assumes that the project requirements are accurate, permanent and fully complete before the development starts. Current software development companies tend to lean towards more agile solutions; however, this method is traditionally rooted in manufacturing so 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, 2022)

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 to mitigate 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.

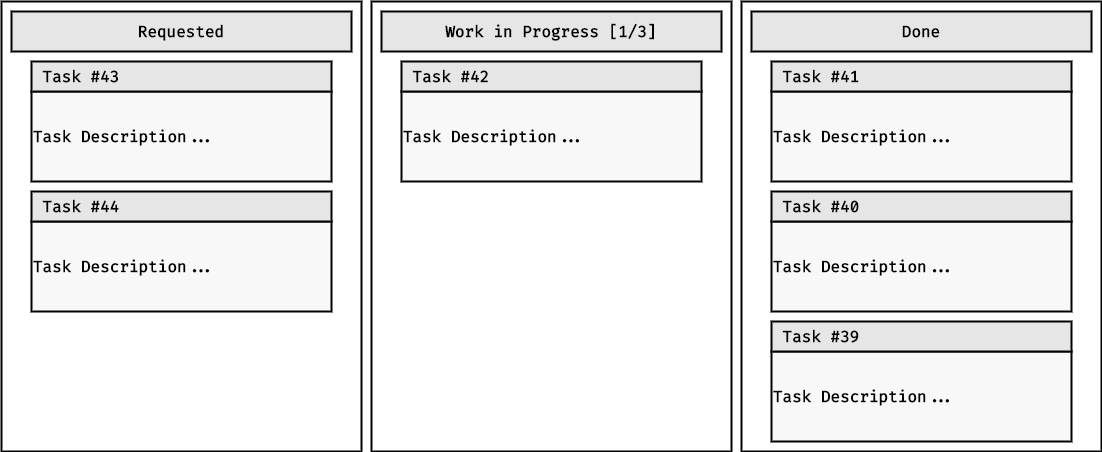


Figure | Kanban (https://kanbanize.com, 2022)

### 2.1.3. | Justification of Applied Methodologies

"*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

The Work Breakdown Structure is split into console and application development phases and does not include preliminary research and documentation. The console prototyping will follow a linear, manufacturer-like, step-by-step development style where each task is started when the last one ends. One of the most significant risk factors is the early finalisation of this part of the development because failing to engineer a working prototype would compromise the entire project.

Diagram

Description automatically generated

Figure | Console WBS

On the other hand, the application can be built more simultaneously, where many of the features may be developed in parallel, more or less independently. Therefore, the play and practice options or chord tutorial features may be started after creating our landing page. The order of the tasks, on the other hand not arbitrary. Some functionalities like tablature translation or the game environment may require extensive development time, and their program units are reusable. These software components precede miscellaneous, less vital functionality, like tutorials, social elements, and administrative sections.

Diagram, engineering drawing

Description automatically generated

Figure | Application WBS

## 2.4. | Project Phases and Schedule

The project console development phase follows a sequential order, and the software development is iterative. However, the subtasks are dependent on one another. Additionally, the schedule is based on a pessimistic scenario, leaving plenty of time to feature development, where the final user acceptance tests are scheduled at the end of March 2023.

Chart

Description automatically generated

Figure | Gantt Chart (Appendix/RiffMaster.pod)

## 2.5. | Tools

The following tools, programs and languages will be used throughout the development phases:

* **Hardware**:
  + Arduino Microcontroller: C++,
  + Physical Tools for wiring and housing,
  + TinkerCad: electrical simulations,
  + Additional Calculations: Python.
* **Software**:
  + VsCode Editor,
  + Frontend: JavaScript (TypeScript), HTML, CSS,
  + Backend: NodeJS (Express), MongoDB (Mongoose),
  + Testing: Jest,
  + Visual Design: Adobe Photoshop,
* **Documentation**:
  + GitHub, Excel, ProjectLibre, Microsoft Word,
  + Diagrams: DrawIO.

# 3. | Literature Review

## 3.1. | Terminology

**Parts of a Guitar:** Our controller has similar characteristics to an actual guitar. We will refer to the matching controller components, such as frets, bridge, and neck, as they were parts of a real instrument (Figure 23).



Figure | Parts of a Guitar (https://hellomusictheory.com, 2022)

**Audio and Sound: The key difference between sound and audio is their form of energy. Sound is mechanical wave energy (longitudinal sound waves) that propagates through a medium, causing variation in pressure within the medium. On the other hand, audio is made of electrical energy (analogue or digital signals) that represents sound electrically.**

**Chord:** "*A chord is three or more notes sounding simultaneously. It can be played on one instrument, like a guitar, or by many instruments at once, like a brass quartet or a choir*" (Eriksson, 2016)

**Riff:** The riff's etymology is unclear; however, it was first used in the 1930-s, probably to shorten the refrain. After that, the term stayed in pop music, and now, it refers to short, repeated melody patterns. Similarly to other literature, this project and the application will reference these patterns as riffs.

**Monophony and Polyphony:** We address monophony because some keyboards are wired so that only one keypress can be recognised at a given time. This limitation would result in a monophonic instrument. However, the guitar is a polyphonic instrument, and strings can be strummed simultaneously. Therefore, to create a complete experience, we must make the input device polyphonic.

**Hammer-On Pull-Off (HOPO):** This guitar technic allows the player to produce sound without strumming by hammering or pulling off fingers from a given string. A pseudo-HOPO is built in the major simulation games, and this function will also be introduced to RiffMaster in a simplified solution to mimic the original instrument better.

## 3.2. | Music Notations

Guitar literature uses two basic notations: standard music notation and tablature notation. Both have similar characteristics; however, standard notation is primarily used in classical music, while tablatures are more common for novice guitar players. Even though the standard notation is the lingua-franca of music, as it has no boundaries for the instrument it interprets, it might emerge as a hurdle for gameplayers because of its steep learning curve.

A close-up of a musical instrument

Description automatically generated with low confidence

Figure | Standard Music Notation (https://pegheadnation.com/string-school/music-notation-guide, 2022)

"*Because it is meant to be used by any instrument, common notation is not written to tell how to play notes on a specific instrument. Instead, common notation simply tells you what each note should sound like … Tablature focuses on telling how to play the notes on the instrument. A note is presented in terms of which string to play, and where to hold the string down on the fretboard*" (Schmidt-Jones, 2016). Because of its simplicity and descriptive feature, the application will use tabular (**tab**) notation in the gameplay.

The prototype will follow 4/4 common time, standard fret and note numbering, traditional chord name shortenings, and well-known rhythm signs as per convention. Potentially other tabular components may be added in later versions, such as optional lyrics, play instructions, side notes and musical notation walkthrough tutorials. Lastly, tablatures files are often saved in a TXT or TAB plain ASCII text format. We will opt for the TAB extension name throughout the application because of its explicit meaning.

![Diagram

Description automatically generated](data:image/jpeg;base64,/9j/4AAQSkZJRgABAQEAYABgAAD/4RDgRXhpZgAATU0AKgAAAAgABAE7AAIAAAAHAAAISodpAAQAAAABAAAIUpydAAEAAAAOAAAQyuocAAcAAAgMAAAAPgAAAAAc6gAAAAgAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAFN0ZXZlbgAAAAWQAwACAAAAFAAAEKCQBAACAAAAFAAAELSSkQACAAAAAzY2AACSkgACAAAAAzY2AADqHAAHAAAIDAAACJQAAAAAHOoAAAAIAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAyMDE5OjA2OjI4IDEwOjQ3OjIyADIwMTk6MDY6MjggMTA6NDc6MjIAAABTAHQAZQB2AGUAbgAAAP/hCxlodHRwOi8vbnMuYWRvYmUuY29tL3hhcC8xLjAvADw/eHBhY2tldCBiZWdpbj0n77u/JyBpZD0nVzVNME1wQ2VoaUh6cmVTek5UY3prYzlkJz8+DQo8eDp4bXBtZXRhIHhtbG5zOng9ImFkb2JlOm5zOm1ldGEvIj48cmRmOlJERiB4bWxuczpyZGY9Imh0dHA6Ly93d3cudzMub3JnLzE5OTkvMDIvMjItcmRmLXN5bnRheC1ucyMiPjxyZGY6RGVzY3JpcHRpb24gcmRmOmFib3V0PSJ1dWlkOmZhZjViZGQ1LWJhM2QtMTFkYS1hZDMxLWQzM2Q3NTE4MmYxYiIgeG1sbnM6ZGM9Imh0dHA6Ly9wdXJsLm9yZy9kYy9lbGVtZW50cy8xLjEvIi8+PHJkZjpEZXNjcmlwdGlvbiByZGY6YWJvdXQ9InV1aWQ6ZmFmNWJkZDUtYmEzZC0xMWRhLWFkMzEtZDMzZDc1MTgyZjFiIiB4bWxuczp4bXA9Imh0dHA6Ly9ucy5hZG9iZS5jb20veGFwLzEuMC8iPjx4bXA6Q3JlYXRlRGF0ZT4yMDE5LTA2LTI4VDEwOjQ3OjIyLjY1NjwveG1wOkNyZWF0ZURhdGU+PC9yZGY6RGVzY3JpcHRpb24+PHJkZjpEZXNjcmlwdGlvbiByZGY6YWJvdXQ9InV1aWQ6ZmFmNWJkZDUtYmEzZC0xMWRhLWFkMzEtZDMzZDc1MTgyZjFiIiB4bWxuczpkYz0iaHR0cDovL3B1cmwub3JnL2RjL2VsZW1lbnRzLzEuMS8iPjxkYzpjcmVhdG9yPjxyZGY6U2VxIHhtbG5zOnJkZj0iaHR0cDovL3d3dy53My5vcmcvMTk5OS8wMi8yMi1yZGYtc3ludGF4LW5zIyI+PHJkZjpsaT5TdGV2ZW48L3JkZjpsaT48L3JkZjpTZXE+DQoJCQk8L2RjOmNyZWF0b3I+PC9yZGY6RGVzY3JpcHRpb24+PC9yZGY6UkRGPjwveDp4bXBtZXRhPg0KICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgIAogICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgCiAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAKICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgIAogICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgCiAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAKICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgIAogICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgCiAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAKICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgIAogICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgCiAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAKICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgIAogICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgCiAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAKICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgIAogICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgCiAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAKICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgIAogICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgCiAgICAgICAgICAgICAgICAgICAgICAgICAgICA8P3hwYWNrZXQgZW5kPSd3Jz8+/9sAQwAHBQUGBQQHBgUGCAcHCAoRCwoJCQoVDxAMERgVGhkYFRgXGx4nIRsdJR0XGCIuIiUoKSssKxogLzMvKjInKisq/9sAQwEHCAgKCQoUCwsUKhwYHCoqKioqKioqKioqKioqKioqKioqKioqKioqKioqKioqKioqKioqKioqKioqKioqKioq/8AAEQgBHANQAwEiAAIRAQMRAf/EAB8AAAEFAQEBAQEBAAAAAAAAAAABAgMEBQYHCAkKC//EALUQAAIBAwMCBAMFBQQEAAABfQECAwAEEQUSITFBBhNRYQcicRQygZGhCCNCscEVUtHwJDNicoIJChYXGBkaJSYnKCkqNDU2Nzg5OkNERUZHSElKU1RVVldYWVpjZGVmZ2hpanN0dXZ3eHl6g4SFhoeIiYqSk5SVlpeYmZqio6Slpqeoqaqys7S1tre4ubrCw8TFxsfIycrS09TV1tfY2drh4uPk5ebn6Onq8fLz9PX29/j5+v/EAB8BAAMBAQEBAQEBAQEAAAAAAAABAgMEBQYHCAkKC//EALURAAIBAgQEAwQHBQQEAAECdwABAgMRBAUhMQYSQVEHYXETIjKBCBRCkaGxwQkjM1LwFWJy0QoWJDThJfEXGBkaJicoKSo1Njc4OTpDREVGR0hJSlNUVVZXWFlaY2RlZmdoaWpzdHV2d3h5eoKDhIWGh4iJipKTlJWWl5iZmqKjpKWmp6ipqrKztLW2t7i5usLDxMXGx8jJytLT1NXW19jZ2uLj5OXm5+jp6vLz9PX29/j5+v/aAAwDAQACEQMRAD8A+kaKKKACiiigAooooAKZFKkyFozkBmUn3UkH9QaSe4htbd57qWOGGMbnkkYKqj1JPArmdBu76TwRf3Wm7bq7+26i1rsVZA/+lzeXgGRA3GP41B9RQB1VFcl4V1Dxjd6lInifT/s1qIyUb7DDDlsjjKX059eNo+tc1rnhDQ/DvizVPFui6aX1yz024vprl5nkaSVwVT5CSo+6/RR0FAHqVFeffCxZ7dNVtL4W11dq0E8uqwxur3vmxBwZCzuSy5xw20DAAUcV6DQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRXMXUzwidNQ1CdLcX+2SYSmLavkghQVxtG7H9c5oA6eiqulvLJpNo9zuMrQqXLDBzjvVqgAooooA4n4nfD2T4kaLZ6Udam0u1juRNcrEm7z1AOFIyBwcHnI9q6fQ9Gs/Dug2Wj6YhS0soVhiDHJwBjJPcnqT61fooAKYIIllklWJBJIAHcKMuBnAJ74yfzp9FAFTTtK0/SLY2+k2FtYwM5cxW0KxqWPU4UAZPrVuiigAooooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKAGSyCGF5WyVRSxx7VH9ol/585vzT/4qi+/5B1z/ANcm/kanoAg+0S/8+c35p/8AFUfaJf8Anzm/NP8A4qp657UNYnXxdFYxXUNtYWdo15qU0mPlDNtiXceFB2yEn0Xt1oA2vtEv/PnN+af/ABVH2iX/AJ85vzT/AOKp9vPHdW0VxA26KVA6NgjKkZB5qSgCD7RL/wA+c35p/wDFUfaJf+fOb80/+KqeigCD7RL/AM+c35p/8VR9ol/585vzT/4qp6KAIPtEv/PnN+af/FUfaJf+fOb80/8AiqnooAg+0S/8+c35p/8AFUfaJf8Anzm/NP8A4qp6KAIPtEv/AD5zfmn/AMVR9ol/585vzT/4qp6KAIPtEv8Az5zfmn/xVH2iX/nzm/NP/iqnooAg+0S/8+c35p/8VR9ol/585vzT/wCKqeigCD7RL/z5zfmn/wAVR9ol/wCfOb80/wDiqnooAg+0S/8APnN+af8AxVH2iX/nzm/NP/iqnooAg+0S/wDPnN+af/FUfaJf+fOb80/+KpupXqadpV1eykBLaF5WLdMKCf6Vy3w98V3fiaykfVL60a9WGKWSxj0uezktg4yM+c5MinoHUBTg4z2AOr+0S/8APnN+af8AxVH2iX/nzm/NP/iqnooAg+0S/wDPnN+af/FUfaJf+fOb80/+KrD8aeJn8N6fafZ9oub65FtE7wPOsZ2sxbyo8PIcKQEUgkkAGrfhXVZta8N2t9cz2k8sgYPJZrIiEhiPuSDeh45RuVORk4oA0ftEv/PnN+af/FUfaJf+fOb80/8AiqnooAg+0S/8+c35p/8AFUfaJf8Anzm/NP8A4qp6KAIPtEv/AD5zfmn/AMVR9ol/585vzT/4qp6DwDgZPpQBB9ol/wCfOb80/wDiqPtEv/PnN+af/FVxPg/xrrGveKJLC9gs2g8iSWWO2Rlk0yRZNohnJchnZfmGFToeCMGu9oAg+0S/8+c35p/8VR9ol/585vzT/wCKqeigCD7RL/z5zfmn/wAVR9ol/wCfOb80/wDiqnrnPGniZ/Den2n2faLm+uRbRO8DzrGdrMW8qPDyHCkBFIJJABoA3PtEv/PnN+af/FUfaJf+fOb80/8AiqzvCuqza14btb65ntJ5ZAweSzWREJDEfckG9DxyjcqcjJxWxQBB9ol/585vzT/4qj7RL/z5zfmn/wAVU9FAEH2iX/nzm/NP/iqPtEv/AD5zfmn/AMVU9cJqXje4tviI+i/b7OwsIPs8by3GmTziWaUn9356uscJI2435JLcA0Adp9ol/wCfOb80/wDiqPtEv/PnN+af/FVPRQBB9ol/585vzT/4qj7RL/z5zfmn/wAVU9FAEH2iX/nzm/NP/iqPtEv/AD5zfmn/AMVU9FAEH2iX/nzm/NP/AIqmx3byxq6WkxVhkHKcj/vqrNQWP/IPg/65r/KgA+0S/wDPnN+af/FUfaJf+fOb80/+KqeigCD7RL/z5zfmn/xVH2iX/nzm/NP/AIqp6KAIPtEv/PnN+af/ABVH2iX/AJ85vzT/AOKqeigCD7RL/wA+c35p/wDFUfaJf+fOb80/+KqeigCD7RL/AM+c35p/8VR9ol/585vzT/4qp6KAIPtEv/PnN+af/FUfaJf+fOb80/8AiqnooAg+0S/8+c35p/8AFUfaJf8Anzm/NP8A4qp6KAIPtEv/AD5zfmn/AMVR9ol/585vzT/4qp6KAIPtEv8Az5zfmn/xVH2iX/nzm/NP/iqnooAg+0S/8+c35p/8VR9ol/585vzT/wCKqeigCD7RL/z5zfmn/wAVR9ol/wCfOb80/wDiqnooAg+0S/8APnN+af8AxVH2iX/nzm/NP/iqnooAg+0S/wDPnN+af/FUfaJf+fOb80/+KqeigCD7RL/z5zfmn/xVH2iX/nzm/NP/AIqp6KAIPtEv/PnN+af/ABVH2iX/AJ85vzT/AOKqeigCD7RL/wA+c35p/wDFUfaJf+fOb80/+KqeigCD7RL/AM+c35p/8VR9ol/585vzT/4qp6KAIPtEv/PnN+af/FUfaJf+fOb80/8AiqnooAg+0S/8+c35p/8AFUfaJf8Anzm/NP8A4qp6KAIPtEv/AD5zfmn/AMVR9ol/585vzT/4qpXcRxs7dFBJrh/BPji68U6qfNmsVtZ7b7RBai2ninjG7AxI/wC7nGPvFNuxuCDQB2f2iX/nzm/NP/iqPtEv/PnN+af/ABVT0UAQfaJf+fOb80/+Ko+0S/8APnN+af8AxVT0UAQfaJf+fOb80/8AiqPtEv8Az5zfmn/xVT0UAQx3BaYRPDJGxUsNxXnGM9CfUVNUD/8AIRh/65SfzSp6ACiiigCC+/5B1z/1yb+Ro+xxf3pv+/7/AONF9/yDrn/rk38jR9ol/wCfOb80/wDiqAILyEwWU0trBcXUyITHAt0ymRuwyWwPrXh3xJsNatPhTfya/pUkN7dXkVxdXbXiyh33YWMKCflVeBn+76mvePtEv/PnN+af/FVjeJ/D1p4u0ltM1m1vGs3Ks8Ucka7iDkHOc9fegCfw/Yzx6FaxajBLazRRrH5Yu2fhQBk4OBnHQZ+taX2OL+9N/wB/3/xqK2aaC3SJ4rqcqMeZK0e5vrggfpUv2iX/AJ85vzT/AOKoAPscX96b/v8Av/jR9ji/vTf9/wB/8aPtEv8Az5zfmn/xVH2iX/nzm/NP/iqAD7HF/em/7/v/AI0fY4v703/f9/8AGj7RL/z5zfmn/wAVR9ol/wCfOb80/wDiqAD7HF/em/7/AL/40fY4v703/f8Af/Gj7RL/AM+c35p/8VR9ol/585vzT/4qgA+xxf3pv+/7/wCNH2OL+9N/3/f/ABo+0S/8+c35p/8AFUfaJf8Anzm/NP8A4qgA+xxf3pv+/wC/+NH2OL+9N/3/AH/xo+0S/wDPnN+af/FUfaJf+fOb80/+KoAPscX96b/v+/8AjWRqviDwxoVytvrfiGy02dl3rFeaoImK+oDODj3rX+0S/wDPnN+af/FVy3jZf7W1jw94bk/497+6a4u1JwJIoAH2H1BfZkdxkUAdOltBJGrxySujAFWW4cgj1BzTvscX96b/AL/v/jSfa1854YIzM0WA4jdPkJGQCCQRwQfoaX7RL/z5zfmn/wAVQAfY4v703/f9/wDGj7HF/em/7/v/AI1FLqAgeJJoJEaZtkYZ4wXOM4Hzc8A1L9ol/wCfOb80/wDiqAD7HF/em/7/AL/40fY4v703/f8Af/Gj7RL/AM+c35p/8VR9ol/585vzT/4qgBJLQm1mjtriaCSRCqzbvMMZxwwD5HHuMVh6D4ZvrLVrvVtf1WLUtRubeO18y2tTaosaZI+Xe53FmJJyB0wBit37RL/z5zfmn/xVH2iX/nzm/NP/AIqgA+xxf3pv+/7/AONH2OL+9N/3/f8Axo+0S/8APnN+af8AxVH2iX/nzm/NP/iqAMzxB4eOsWdp9kuvsl7YXK3VpcSxmdVcAjDKWBZSGIIDA88EU3w14Zj8P6GLFrhp5ZJ5LmeaMGESSyMWchQflXJOBk49Sea1ftEv/PnN+af/ABVH2iX/AJ85vzT/AOKoAPscX96b/v8Av/jR9ji/vTf9/wB/8aPtEv8Az5zfmn/xVH2iX/nzm/NP/iqAD7HF/em/7/v/AI0fY4v703/f9/8AGj7RL/z5zfmn/wAVR9ol/wCfOb80/wDiqAD7HF/em/7/AL/40q2qLuw0vzAjmVj/ADNJ9ol/585vzT/4qj7RL/z5zfmn/wAVQByXhrwDNoOsWl1caut3b6day2tlDHa+U4SRw5Mz7z5jDAwQEHUkEmut+xxf3pv+/wC/+NH2iX/nzm/NP/iqPtEv/PnN+af/ABVAB9ji/vTf9/3/AMaPscX96b/v+/8AjR9ol/585vzT/wCKo+0S/wDPnN+af/FUAH2OL+9N/wB/3/xrM8QeHjrFnafZLr7Je2Fyt1aXEsZnVXAIwylgWUhiCAwPPBFaf2iX/nzm/NP/AIqj7RL/AM+c35p/8VQBleGvDMfh/QxYtcNPLJPJczzRgwiSWRizkKD8q5JwMnHqTzWr9ji/vTf9/wB/8aPtEv8Az5zfmn/xVH2iX/nzm/NP/iqAD7HF/em/7/v/AI0fY4v703/f9/8AGj7RL/z5zfmn/wAVR9ol/wCfOb80/wDiqAFFpGrAhpsg55mc/wBa53XvC2pa/qAjudbQaIZoZ3sTZAyho23AJMHGFLBSQyMeDgjPHQ/aJf8Anzm/NP8A4qj7RL/z5zfmn/xVADntkkcszSgn+7MwH5A037HF/em/7/v/AI0faJf+fOb80/8AiqPtEv8Az5zfmn/xVAB9ji/vTf8Af9/8azbPV/D+o6nNp2n63bXV9b5861g1HfLHg4O5A2Rg8citRJnckNBJHxnc20j9CTXmPgbVE07xEmmaV4oXXdLFvdXWpiSCOL+zpvNzzgB49xMmVlLH5cggA0AemfY4v703/f8Af/Gj7HF/em/7/v8A401LtpEDx2srKwyGVkII/wC+qd9ol/585vzT/wCKoAPscX96b/v+/wDjUFnaxtYwktNkoDxM47fWp/tEv/PnN+af/FVXtLl49PiJtpdqxjLbkAxjry1AFj7HF/em/wC/7/40fY4v703/AH/f/GorfUVu4FmtYJJYmJ2ujIQcHHB3eoqX7RL/AM+c35p/8VQAfY4v703/AH/f/Gj7HF/em/7/AL/40faJf+fOb80/+Ko+0S/8+c35p/8AFUAH2OL+9N/3/f8Axo+xxf3pv+/7/wCNH2iX/nzm/NP/AIqj7RL/AM+c35p/8VQAfY4v703/AH/f/Gj7HF/em/7/AL/40faJf+fOb80/+Ko+0S/8+c35p/8AFUAH2OL+9N/3/f8Axo+xxf3pv+/7/wCNH2iX/nzm/NP/AIqj7RL/AM+c35p/8VQAfY4v703/AH/f/GqOq32jaFarc63qsOnQM2xZbu/MSlvTLMBn2q99ol/585vzT/4quL+Ir2EU2i3uqeKW8MG1aeW3lMUZ3y+UQF3yBowQC3ykZbnaeKAOvt0tLy2juLS4aeCVQ8csV0zK6noQQ2CPepfscX96b/v+/wDjWX4Y1HU77wjpV5rFm66hcWqPcIiBNrkc/KxyM9cdq1PtEv8Az5zfmn/xVAB9ji/vTf8Af9/8aPscX96b/v8Av/jR9ol/585vzT/4qj7RL/z5zfmn/wAVQAfY4v703/f9/wDGj7HF/em/7/v/AI0faJf+fOb80/8AiqPtEv8Az5zfmn/xVAB9ji/vTf8Af9/8aPscX96b/v8Av/jR9ol/585vzT/4qj7RL/z5zfmn/wAVQBFcJaWdtJcXdw0EESl5JZbplVFHUklsAe9RadcaXrFil7pGoLf2rkhZ7W9MqMQcHDKxBweKZrd7pltoNzc+IoVTTogGn+0RCVQoI+ZgN3APOT0xk4AzXPfD65SaPX57EC+sX1Nnt9QjZT9uyibpAQdhAbK5UKDt6Z5IB132OL+9N/3/AH/xo+xxf3pv+/7/AONH2iX/AJ85vzT/AOKo+0S/8+c35p/8VQAfY4v703/f9/8AGj7HF/em/wC/7/40faJf+fOb80/+Ko+0S/8APnN+af8AxVAB9ji/vTf9/wB/8aPscX96b/v+/wDjR9ol/wCfOb80/wDiqckzuSGgkj4zubaR+hJoAy7PV/D+o6nNp2n63bXV9b5861g1HfLHg4O5A2Rg8citL7HF/em/7/v/AI15t4Bvza+J4tH0PxL/AMJDp7w3NxqCG3SM2FwZsgEAB4yxaQbJCzfLkYwa9J+0S/8APnN+af8AxVAB9ji/vTf9/wB/8aPscX96b/v+/wDjR9ol/wCfOb80/wDiqPtEv/PnN+af/FUAH2OL+9N/3/f/ABqjqt9o2hWq3Ot6rDp0DNsWW7vzEpb0yzAZ9qvfaJf+fOb80/8Aiq4v4ivYRTaLe6p4pbwwbVp5beUxRnfL5RAXfIGjBALfKRludp4oA6+3S0vLaO4tLhp4JVDxyxXTMrqehBDYI96l+xxf3pv+/wC/+NZfhjUdTvvCOlXmsWbrqFxao9wiIE2uRz8rHIz1x2rU+0S/8+c35p/8VQAfY4v703/f9/8AGj7HF/em/wC/7/40faJf+fOb80/+Ko+0S/8APnN+af8AxVACraxrnmRgwwQ8rMCPoTXK+H/AR0XWrS7n1Fbq30y2ltdMhW28t4Y5GDMJH3HzD8qgEBeByCea6n7RL/z5zfmn/wAVR9ol/wCfOb80/wDiqAD7HF/em/7/AL/40fY4v703/f8Af/Gj7RL/AM+c35p/8VR9ol/585vzT/4qgCjqt9o2hWq3Ot6rDp0DNsWW7vzEpb0yzAZ9qfptzpWs2S3mj6il/asSFntb0yoSOCAysRxVDxrrFxo/gfUb61UxXXl+VBux8sjsEUnGehYGruh6TD4b8O2Ol2ULPHbRLH8pGWOPmYljyScknqSaALv2OL+9N/3/AH/xo+xxf3pv+/7/AONQ22qR3qO9mhuFjkaJ2iljYK6naynDcEEEEdQRU32iX/nzm/NP/iqAI1hWLUYthc5if78jN3T1NW6zhqELa/BZSHyrlraWVYXddzIGjBYAE8AsBn3rRoAKKKKAIL7/AJB1z/1yb+RqeoL7/kHXP/XJv5Gj7HF/em/7/v8A40AT0VB9ji/vTf8Af9/8a5DW9Xs7HxUz6hrEmn6Po9r5l8zXLqJZpT+6jJzkkKjNtHJ3L2NAHbUVSs0t72xguojcBJ41kUNO+QGGRn5vepvscX96b/v+/wDjQBheM9e1vQLbTptB0GTWVmvUivBGTughP3pAoBLEewqbStSvPEulQ39nKdORb+RTG8O5pIopXjKMGxtLFc5HT361o3duYrKeS1SaedI2aOL7Sw3sBwuSwAyeOaraFoi6X4fsLGWSZpbe3RJHEzje4A3NwepOT+NAGtRUH2OL+9N/3/f/ABo+xxf3pv8Av+/+NAE9FQfY4v703/f9/wDGj7HF/em/7/v/AI0AT1k+Jbm9sdDnvbDULOwFqjTTTXlo1wgRQSflWSM598/hV/7HF/em/wC/7/41neJfD/8AwkHhq40dbprZLgqHkZTJlQ4ZlIJGQQCOvegBng+81nUfCdjfeJY7aK/uY/NaO2jaNUVuVBBdvmxjPzEZzW3Ubwo8ap8yqvQI5X+VM+xxf3pv+/7/AONAE9FQfY4v703/AH/f/Gj7HF/em/7/AL/40AQXmlR3M4uYJZLS7UbRPDjJH91geGHsfwxUPla7jZ9r08j/AJ6/Znz/AN87/wCtXfscX96b/v8Av/jR9ji/vTf9/wB/8aAK9npSW9wbq5le7vCCvny4yoP8KgcKPp17k1fqD7HF/em/7/v/AI0fY4v703/f9/8AGgCeioPscX96b/v+/wDjR9ji/vTf9/3/AMaAJ6Kg+xxf3pv+/wC/+NH2OL+9N/3/AH/xoAnoqD7HF/em/wC/7/40fY4v703/AH/f/GgCeioPscX96b/v+/8AjR9ji/vTf9/3/wAaAJ6Kg+xxf3pv+/7/AONH2OL+9N/3/f8AxoAnoqD7HF/em/7/AL/40fY4v703/f8Af/GgCeioPscX96b/AL/v/jR9ji/vTf8Af9/8aAJ6Kg+xxf3pv+/7/wCNH2OL+9N/3/f/ABoAnoqD7HF/em/7/v8A40fY4v703/f9/wDGgCeioPscX96b/v8Av/jR9ji/vTf9/wB/8aAJ6Kg+xxf3pv8Av+/+NH2OL+9N/wB/3/xoAnoqD7HF/em/7/v/AI0fY4v703/f9/8AGgCeioPscX96b/v+/wDjR9ji/vTf9/3/AMaAJ6Kg+xxf3pv+/wC/+NH2OL+9N/3/AH/xoAnoIyMGoPscX96b/v8Av/jR9ji/vTf9/wB/8aAKbaFbxu0mmyzac7HJ+zMAhPqUIK/jjPvSeRrkXCXtlOO3mWzK34kPj9BV37HF/em/7/v/AI0fY4v703/f9/8AGgCn5OuScNeWMA7lLZ3P4EuAPyNQWWhwT2kEmpTTX52KQk7Dyx/2zUBT9SCfetP7HF/em/7/AL/41BZ2sbWMJLTZKA8TOO31oAugBVAUAADAA7UtQfY4v703/f8Af/Gj7HF/em/7/v8A40AT0VB9ji/vTf8Af9/8aPscX96b/v8Av/jQBPRUH2OL+9N/3/f/ABo+xxf3pv8Av+/+NAE9VtSvU07Srq9lICW0LysW6YUE/wBKd9ji/vTf9/3/AMaSS0JtZo7a4mgkkQqs27zDGccMA+Rx7jFAHK/D3xXd+JrKR9UvrRr1YYpZLGPS57OS2DjIz5zkyKegdQFODjPbsq5vQfDN9Zatd6tr+qxalqNzbx2vmW1qbVFjTJHy73O4sxJOQOmAMVu/Y4v703/f9/8AGgCeioPscX96b/v+/wDjR9ji/vTf9/3/AMaAJ6Kg+xxf3pv+/wC/+NH2OL+9N/3/AH/xoAnoqD7HF/em/wC/7/40fY4v703/AH/f/GgCcnAyeBXCeB/G9x4m1aeO8v7OMP5z2tiNMngkeJJSgkWeR9kwxjOxcAsOfXtkt0jYlGkzjHzSs36Emud03wtqSeJIdX1/W49TezhlgsxHZC3ZVkYFjIQ5DthVGVVBwTjngA6eioPscX96b/v+/wDjR9ji/vTf9/3/AMaAJ6Kg+xxf3pv+/wC/+NH2OL+9N/3/AH/xoAnoqD7HF/em/wC/7/40fY4v703/AH/f/GgCeioPscX96b/v+/8AjR9ji/vTf9/3/wAaAJ6Kg+xxf3pv+/7/AONH2OL+9N/3/f8AxoAnoqD7HF/em/7/AL/40fY4v703/f8Af/GgCeioPscX96b/AL/v/jR9ji/vTf8Af9/8aAJ6Kg+xxf3pv+/7/wCNH2OL+9N/3/f/ABoAnoqD7HF/em/7/v8A40fY4v703/f9/wDGgCeioPscX96b/v8Av/jR9ji/vTf9/wB/8aAJ6Kg+xxf3pv8Av+/+NH2OL+9N/wB/3/xoAnoqD7HF/em/7/v/AI0fY4v703/f9/8AGgCeioPscX96b/v+/wDjR9ji/vTf9/3/AMaAMbUPC27UZNU0C+k0fUZSDO8cYkhusDA82I8MQP4gVbtuxxURtPG048qTVtFtV6NPBYSM5/3VaXap+u4exre+xxf3pv8Av+/+NH2OL+9N/wB/3/xoAydH8O2miaj5yPNd31zE32m/umDzT4K4yQAABk4VQFHYCt2qiwrFqMWwucxP9+Rm7p6mrdABRRRQBBff8g65/wCuTfyNH2iX/nzm/NP/AIqi+/5B1z/1yb+RqegCjeXN8LKb+z7EtdbD5QndQm7tuIJOPpXiXxL0rxFpHwhvLfW4rR1kvEuLi6ictJNOzjLNngDsABwAo7V73Wfq+habr9obXWbRby3JBMMhJQkcglc4JoAg0JdRstFt7bUrYNNCgjH2crtCqAByW5PHXA69K0PtEv8Az5zfmn/xVOtreO1t0gh37EGF3uzn8ySaloAg+0S/8+c35p/8VR9ol/585vzT/wCKqeigCD7RL/z5zfmn/wAVR9ol/wCfOb80/wDiqnooAg+0S/8APnN+af8AxVH2iX/nzm/NP/iqnooAg+0S/wDPnN+af/FUfaJf+fOb80/+KqeigCD7RL/z5zfmn/xVH2iX/nzm/NP/AIqp6KAIPtEv/PnN+af/ABVH2iX/AJ85vzT/AOKqeigCD7RL/wA+c35p/wDFUfaJf+fOb80/+KqeigCD7RL/AM+c35p/8VR9ol/585vzT/4qp6KAIPtEv/PnN+af/FUfaJf+fOb80/8AiqnooAg+0S/8+c35p/8AFUfaJf8Anzm/NP8A4qp6KAIPtEv/AD5zfmn/AMVR9ol/585vzT/4qp6KAIPtEv8Az5zfmn/xVH2iX/nzm/NP/iqnooAg+0S/8+c35p/8VR9ol/585vzT/wCKqeigCD7RL/z5zfmn/wAVR9ol/wCfOb80/wDiqnooAg+0S/8APnN+af8AxVH2iX/nzm/NP/iqnooAg+0S/wDPnN+af/FUfaJf+fOb80/+KqeigCD7RL/z5zfmn/xVH2iX/nzm/NP/AIqp6KAIPtEv/PnN+af/ABVH2iX/AJ85vzT/AOKqeigCD7RL/wA+c35p/wDFUfaJf+fOb80/+KqeigCD7RL/AM+c35p/8VR9ol/585vzT/4qp6KAIPtEv/PnN+af/FUfaJf+fOb80/8AiqnooAg+0S/8+c35p/8AFUfaJf8Anzm/NP8A4qp6KAIPtEv/AD5zfmn/AMVR9ol/585vzT/4qp6KAIPtEv8Az5zfmn/xVH2iX/nzm/NP/iqnooAg+0S/8+c35p/8VUFnPILGEC1mYBByCnPH+9V6oLH/AJB8H/XNf5UAH2iX/nzm/NP/AIqj7RL/AM+c35p/8VU9FAEH2iX/AJ85vzT/AOKo+0S/8+c35p/8VU9FAEH2iX/nzm/NP/iqPtEv/PnN+af/ABVT0UAQfaJf+fOb80/+Ko+0S/8APnN+af8AxVT0UAQfaJf+fOb80/8AiqPtEv8Az5zfmn/xVT0UAQfaJf8Anzm/NP8A4qj7RL/z5zfmn/xVT0UAQfaJf+fOb80/+Ko+0S/8+c35p/8AFVPRQBB9ol/585vzT/4qj7RL/wA+c35p/wDFVPRQBB9ol/585vzT/wCKo+0S/wDPnN+af/FVPRQBB9ol/wCfOb80/wDiqPtEv/PnN+af/FVPRQBB9ol/585vzT/4qj7RL/z5zfmn/wAVU9FAEH2iX/nzm/NP/iqPtEv/AD5zfmn/AMVU9FAEH2iX/nzm/NP/AIqj7RL/AM+c35p/8VU9FAEH2iX/AJ85vzT/AOKo+0S/8+c35p/8VU9FAEH2iX/nzm/NP/iqPtEv/PnN+af/ABVT0UAQfaJf+fOb80/+Ko+0S/8APnN+af8AxVT0UAQfaJf+fOb80/8AiqPtEv8Az5zfmn/xVT0UAQfaJf8Anzm/NP8A4qj7RL/z5zfmn/xVT0UAQfaJf+fOb80/+Ko+0S/8+c35p/8AFVPRQBB9ol/585vzT/4qj7RL/wA+c35p/wDFVPRQBB9ol/585vzT/wCKo+0S/wDPnN+af/FVPRQBB9ol/wCfOb80/wDiqPtEv/PnN+af/FVPRQBB9ol/585vzT/4qj7RL/z5zfmn/wAVU9FAFRZGk1GLdC8WIn++V55T0Jq3UD/8hGH/AK5SfzSp6ACiiigBHRZEZHGVYYI9RUP2OL+9N/3/AH/xqeigCD7HF/em/wC/7/40fY4v703/AH/f/Gp6KAIPscX96b/v+/8AjR9ji/vTf9/3/wAanooAg+xxf3pv+/7/AONH2OL+9N/3/f8AxqeigCD7HF/em/7/AL/40fY4v703/f8Af/Gp6KAIPscX96b/AL/v/jR9ji/vTf8Af9/8anooAg+xxf3pv+/7/wCNH2OL+9N/3/f/ABqeoby6jsbGe7nyIoI2lfAydqjJ/lQAn2OL+9N/3/f/ABo+xxf3pv8Av+/+NT0UAQfY4v703/f9/wDGj7HF/em/7/v/AI1PUM91HbzW8cmd1zIYo8D+IIz/AMkNACfY4v703/f9/wDGj7HF/em/7/v/AI1PRQBB9ji/vTf9/wB/8aPscX96b/v+/wDjU9QrdRvfS2gz5sUaStxxtYsB/wCgNQAn2OL+9N/3/f8Axo+xxf3pv+/7/wCNT0UAQfY4v703/f8Af/Gj7HF/em/7/v8A41PUNrdR3kLSQ5KrI8RyMfMjlG/VTQAn2OL+9N/3/f8Axo+xxf3pv+/7/wCNT0UAQfY4v703/f8Af/Gj7HF/em/7/v8A40t5dR2NjPdz5EUEbSvgZO1Rk/yqagCD7HF/em/7/v8A40fY4v703/f9/wDGp6KAIPscX96b/v8Av/jR9ji/vTf9/wB/8anooAg+xxf3pv8Av+/+NH2OL+9N/wB/3/xqeigCD7HF/em/7/v/AI0fY4v703/f9/8AGlguo7ia4jjzutpBFJkfxFFf+TipqAIPscX96b/v+/8AjR9ji/vTf9/3/wAanooAg+xxf3pv+/7/AONH2OL+9N/3/f8Axqeoby6jsbGe7nyIoI2lfAydqjJ/lQAn2OL+9N/3/f8Axo+xxf3pv+/7/wCNT0UAQfY4v703/f8Af/Gj7HF/em/7/v8A41PRQBB9ji/vTf8Af9/8aPscX96b/v8Av/jU9QtdRpfRWhz5ssbyrxxtUqD/AOhrQAn2OL+9N/3/AH/xo+xxf3pv+/7/AONT0UAQfY4v703/AH/f/Gj7HF/em/7/AL/41PUMF1HcTXEced1tIIpMj+Ior/ycUAJ9ji/vTf8Af9/8aPscX96b/v8Av/jU9FAEH2OL+9N/3/f/ABpFsYVUKpmAAwAJ34/WrFFAEH2OL+9N/wB/3/xo+xxf3pv+/wC/+NLeXUdjYz3c+RFBG0r4GTtUZP8AKpqAIPscX96b/v8Av/jR9ji/vTf9/wB/8anooAg+xxf3pv8Av+/+NH2OL+9N/wB/3/xqeigCD7HF/em/7/v/AI0fY4v703/f9/8AGp6KAIPscX96b/v+/wDjR9ji/vTf9/3/AMaW1uo7yFpIclVkeI5GPmRyjfqpqagCD7HF/em/7/v/AI0fY4v703/f9/8AGp6hurqOzhWSbIVpEiGBn5ncIv6sKAE+xxf3pv8Av+/+NH2OL+9N/wB/3/xqeigCD7HF/em/7/v/AI0fY4v703/f9/8AGp6KAIPscX96b/v+/wDjR9ji/vTf9/3/AManqGC6juJriOPO62kEUmR/EUV/5OKAE+xxf3pv+/7/AONH2OL+9N/3/f8AxqeigCD7HF/em/7/AL/40fY4v703/f8Af/Gp6hs7qO+sYLuDJinjWVMjB2sMj+dACfY4v703/f8Af/Gj7HF/em/7/v8A41PRQBB9ji/vTf8Af9/8aPscX96b/v8Av/jU9FAEH2OL+9N/3/f/ABo+xxf3pv8Av+/+NT0UAQfY4v703/f9/wDGj7HF/em/7/v/AI0sF1HcTXEced1tIIpMj+Ior/ycVNQBB9ji/vTf9/3/AMaPscX96b/v+/8AjU9FAEH2OL+9N/3/AH/xo+xxf3pv+/7/AONT1DdXUdnCsk2QrSJEMDPzO4Rf1YUAJ9ji/vTf9/3/AMaPscX96b/v+/8AjU9FAEH2OL+9N/3/AH/xo+xxf3pv+/7/AONT0UAQfY4v703/AH/f/Gj7HF/em/7/AL/41PUMF1HcTXEced1tIIpMj+Ior/ycUAJ9ji/vTf8Af9/8aPscX96b/v8Av/jU9FAEH2OL+9N/3/f/ABo+xxf3pv8Av+/+NLeXUdjYz3c+RFBG0r4GTtUZP8qmoAg+xxf3pv8Av+/+NH2OL+9N/wB/3/xqeigCKO2jik3rvLYIBeRmwD9T7CpaKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigArM8Tf8AIp6v/wBeM3/os1p1meJv+RT1f/rxm/8ARZoA06K5Pwhc6xPd3A1bxDpmrIEGxLKRGKHPU7VHFdZQAVmar/yEtE/6/m/9J5q065P4g3k9hpWn3Npex2MyXw23EqF1TMUgOQFY8g46Hr+NAHWUV49/wmGvf9Dpp3/gHJ/8j0f8Jhr3/Q6ad/4Byf8AyPQB7DWZb/8AI2X/AP1423/oyevMf+Ew17/odNO/8A5P/keoE8Va0uozTDxbYLI8UaNMbV8OAXIUDyOo3E9B94cnHAB7RRXj3/CYa9/0Omnf+Acn/wAj0f8ACYa9/wBDpp3/AIByf/I9AHsNZnh//kGy/wDX9d/+lEleY/8ACYa9/wBDpp3/AIByf/I9eheB5pLjwjbTTTLcSSSzu8yLgSEzOSwGBgHr0H0FAG/RRRQBmeJv+RT1f/rxm/8ARZrTrM8Tf8inq/8A14zf+izT7S/stRvzJp+rQ3ISPDQQSo4GT94459vSgDQooooAKKKKACiiigDM0r/kJa3/ANfy/wDpPDWnXJnxZouha9rNtqt75Ez3SSKvlO2VNvEM5UEdQal/4WN4V/6Cn/kvL/8AE0AdPRXMf8LG8K/9BT/yXl/+Jo/4WN4V/wCgp/5Ly/8AxNAHT1meJv8AkU9X/wCvGb/0Way/+FjeFf8AoKf+S8v/AMTVHXPHvhq88PajbW2pb5prWWONfIkG5ihAGSuOpoA7WiiigAooooAKzLj/AJGyw/68bn/0ZBWnXD/EDUrvS9R0qaw1OHTJGiuEM00RcMMxHbgI3PAPTt1oA7iivHv+Ew17/odNO/8AAOT/AOR6P+Ew17/odNO/8A5P/kegD2GszSv+Qlrf/X8v/pPDXmP/AAmGvf8AQ6ad/wCAcn/yPXa/D68nv9K1C5u72O+me+O64iQor4ijAwCqngDHQdPxoA6yiiigAooooAzPE3/Ip6v/ANeM3/os1eNygvBbYfeYzJnYduM4+90z7VR8Tf8AIp6v/wBeM3/os1l/8LG8K/8AQU/8l5f/AImgDp6K5j/hY3hX/oKf+S8v/wATR/wsbwr/ANBT/wAl5f8A4mgDp6K5j/hY3hX/AKCn/kvL/wDE0f8ACxvCv/QU/wDJeX/4mgDp6K5j/hY3hX/oKf8AkvL/APE1jeJPHlrc28I8NeIrezlVyZWntZCGXHAH7pqAOt8P/wDINl/6/rv/ANKJK06wPA80lx4RtpppluJJJZ3eZFwJCZnJYDAwD16D6Ct+gArM8Qf8g2L/AK/rT/0ojrTrH8U3MVnoP2m5bZDDdWskjYJ2qLiMk4HPQUAbFFcx/wALG8K/9BT/AMl5f/iaP+FjeFf+gp/5Ly//ABNAHT0VzH/CxvCv/QU/8l5f/iaP+FjeFf8AoKf+S8v/AMTQB09Zmlf8hLW/+v5f/SeGuZ17x/plxpZTw/r0VrebwRJNbSldvcf6tv5Vd+H15Pf6VqFzd3sd9M98d1xEhRXxFGBgFVPAGOg6fjQB1lFFFABWZ4Z/5FPSP+vGH/0WK065vwzf2V3o+iQW2rQmeCzjElrFKjM37sAhh1GD9KAOkooooAKKKKACiiigDM0r/kJa3/1/L/6Tw1p1yZ8WaLoWvazbare+RM90kir5TtlTbxDOVBHUGpf+FjeFf+gp/wCS8v8A8TQB09Fcx/wsbwr/ANBT/wAl5f8A4mj/AIWN4V/6Cn/kvL/8TQB09ZniD/kGxf8AX9af+lEdZf8Awsbwr/0FP/JeX/4mq1/400DVktbLT7/zriW+tdieTIucToTyVA6A0AdhRRRQAUUUUAFZmlf8hLW/+v5f/SeGtOvL/Emvalpfi7VYbDxBaaZG0sbmGa3Zyx8mMbsiJuOAOvbpQB6hRXj3/CYa9/0Omnf+Acn/AMj0f8Jhr3/Q6ad/4Byf/I9AHp3ib/kU9X/68Zv/AEWa068XvvFWtXGnXMM3i2wuI5ImR4UtXBkBBBUHyBgnp1H1Fe0UAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFMmhS4gkhmXdHIpR1PcEYIp9FAGTo/hjR9Bmkl0m0+zvKoVz5rtkf8AAia1qKKACoLuytb+Hyb62huYs7tk0Ydc+uDU9FAGZ/wjOg/9ATTv/ASP/Cj/AIRnQf8AoCad/wCAkf8AhWnRQBmf8IzoP/QE07/wEj/wo/4RnQf+gJp3/gJH/hWnRQBmf8IzoP8A0BNO/wDASP8Awo/4RnQf+gJp3/gJH/hWnRQBmf8ACM6D/wBATTv/AAEj/wAKv29vDaQLBawxwxJwscahVX6AVJRQAUUUUAMmhS4gkhmXdHIpR1PcEYIrM0fwxo+gzSS6TafZ3lUK5812yP8AgRNa1FABRRRQAUUUUAFFFFAFG50TSr2cz3mmWdxK2AZJbdGY49yKi/4RnQf+gJp3/gJH/hWnRQBmf8IzoP8A0BNO/wDASP8Awo/4RnQf+gJp3/gJH/hWnRQBmf8ACM6D/wBATTv/AAEj/wAKB4a0JWBXRdPBHIItU4/StOigAooooAKKKKACq15ptjqKoNQs7e6CZKieJX259MjirNFAGZ/wjOg/9ATTv/ASP/Cj/hGdB/6Amnf+Akf+FadFAGZ/wjOg/wDQE07/AMBI/wDCrlpZWthD5NjbQ20Wd2yGMIufXAqeigAooooAKKKKAGyRpLG0cqq6OCrKwyGB6gis7/hGdB/6Amnf+Akf+FadFAGZ/wAIzoP/AEBNO/8AASP/AAo/4RnQf+gJp3/gJH/hWnRQBmf8IzoP/QE07/wEj/wo/wCEZ0H/AKAmnf8AgJH/AIVp0UAZn/CM6D/0BNO/8BI/8KP+EZ0H/oCad/4CR/4Vp0UAR29vDaQLBawxwxJwscahVX6AVJRRQAVHcW8N3A0F1DHNE/DRyKGVvqDUlFAGZ/wjOg/9ATTv/ASP/Cj/AIRnQf8AoCad/wCAkf8AhWnRQBmf8IzoP/QE07/wEj/wo/4RnQf+gJp3/gJH/hWnRQBmf8IzoP8A0BNO/wDASP8Awq5aWVrYQ+TY20NtFndshjCLn1wKnooAKKKKACsbTvCWiaVqRv8AT7Hybk7gXErnr14Jx+lbNFABRRRQAUUUUAFFFFAFG50TSr2cz3mmWdxK2AZJbdGY49yKi/4RnQf+gJp3/gJH/hWnRQBmf8IzoP8A0BNO/wDASP8Awo/4RnQf+gJp3/gJH/hWnRQBmf8ACM6D/wBATTv/AAEj/wAKfD4f0a3mSaDSbGKVDuR0tkDKfUEDitCigAooooAKKKKACqNzomlXs5nvNMs7iVsAyS26Mxx7kVeooAzP+EZ0H/oCad/4CR/4Uf8ACM6D/wBATTv/AAEj/wAK06KAMweGtCVgV0XTwRyCLVOP0rToooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigArCe6vTBeLPfNFJHdbE+zQqWYFAQihs85PJI7HoOm3I6xRs7nCqCSfQCqNxFpt2pW6sROpbeRJZs2WxjPK9ccZoAt2vnfZIftW3z9g8zb03Y5/WparR3FvDEscMUkaIMKq27gKPQDFO+2Rf3Zv+/D/4UAT0VB9si/uzf9+H/wAKPtkX92b/AL8P/hQBPRUH2yL+7N/34f8Awo+2Rf3Zv+/D/wCFAE9FQfbIv7s3/fh/8KPtkX92b/vw/wDhQBPRUH2yL+7N/wB+H/wo+2Rf3Zv+/D/4UAT0VB9si/uzf9+H/wAKPtkX92b/AL8P/hQBPRUH2yL+7N/34f8Awo+2Rf3Zv+/D/wCFAE9FQfbIv7s3/fh/8KPtkX92b/vw/wDhQBPRUH2yL+7N/wB+H/wo+2Rf3Zv+/D/4UAT0VB9si/uzf9+H/wAKPtkX92b/AL8P/hQBPRUH2yL+7N/34f8Awo+2Rf3Zv+/D/wCFAE9FQfbIv7s3/fh/8KPtkX92b/vw/wDhQBPRUH2yL+7N/wB+H/wo+2Rf3Zv+/D/4UAT0VB9si/uzf9+H/wAKPtkX92b/AL8P/hQBPRUH2yL+7N/34f8Awo+2Rf3Zv+/D/wCFAE9FQfbIv7s3/fh/8KPtkX92b/vw/wDhQBPRUH2yL+7N/wB+H/wo+2Rf3Zv+/D/4UAT0VB9si/uzf9+H/wAKPtkX92b/AL8P/hQBPRUH2yL+7N/34f8Awo+2Rf3Zv+/D/wCFAE9FQfbIv7s3/fh/8KPtkX92b/vw/wDhQBPRUH2yL+7N/wB+H/wo+2Rf3Zv+/D/4UAT0VB9si/uzf9+H/wAKQ30KjLCYDIHMD/4UAWKKg+2Rf3Zv+/D/AOFH2yL+7N/34f8AwoAnoqD7ZF/dm/78P/hR9si/uzf9+H/woAnoqD7ZF/dm/wC/D/4UfbIv7s3/AH4f/CgCeioPtkX92b/vw/8AhR9si/uzf9+H/wAKAJ6Kg+2Rf3Zv+/D/AOFH2yL+7N/34f8AwoAnoqD7ZF/dm/78P/hSNfQqMsJgMgcwP/hQBYoqD7ZF/dm/78P/AIUfbIv7s3/fh/8ACgCeioPtkX92b/vw/wDhR9si/uzf9+H/AMKAJ6Kg+2Rf3Zv+/D/4UfbIv7s3/fh/8KAJ6Kg+2Rf3Zv8Avw/+FH2yL+7N/wB+H/woAnoqD7ZF/dm/78P/AIUfbIv7s3/fh/8ACgCeioPtkX92b/vw/wDhR9si/uzf9+H/AMKAJ6Kg+2Rf3Zv+/D/4UfbIv7s3/fh/8KAJ6Kg+2Rf3Zv8Avw/+FH2yL+7N/wB+H/woAnoqD7ZF/dm/78P/AIUfbIv7s3/fh/8ACgCeioPtkX92b/vw/wDhR9si/uzf9+H/AMKAJ6Kg+2Rf3Zv+/D/4UfbIv7s3/fh/8KAJ6Kg+2Rf3Zv8Avw/+FH2yL+7N/wB+H/woAnoqD7ZF/dm/78P/AIUfbIv7s3/fh/8ACgCeioPtkX92b/vw/wDhR9si/uzf9+H/AMKAJ6Kg+2Rf3Zv+/D/4UfbIv7s3/fh/8KAJ6Kg+2Rf3Zv8Avw/+FH2yL+7N/wB+H/woAnoqD7ZF/dm/78P/AIUfbIv7s3/fh/8ACgCeioPtkX92b/vw/wDhR9si/uzf9+H/AMKAJ6Kg+2Rf3Zv+/D/4UfbIv7s3/fh/8KAJ6Kg+2Rf3Zv8Avw/+FH2yL+7N/wB+H/woAnoqD7ZF/dm/78P/AIUfbIv7s3/fh/8ACgCeioPtkX92b/vw/wDhR9si/uzf9+H/AMKAJ6KijuY5ZNi7w2CQHjZcgfUe4qWgAooooAgvv+Qdc/8AXJv5Gp6gvv8AkHXP/XJv5Gj7PL/z+Tfkn/xNAE9FQfZ5f+fyb8k/+Jo+zy/8/k35J/8AE0AT0VB9nl/5/JvyT/4mj7PL/wA/k35J/wDE0AT0VB9nl/5/JvyT/wCJo+zy/wDP5N+Sf/E0AT0VB9nl/wCfyb8k/wDiaPs8v/P5N+Sf/E0AT0VB9nl/5/JvyT/4mj7PL/z+Tfkn/wATQBPRUH2eX/n8m/JP/iaPs8v/AD+Tfkn/AMTQBPRUH2eX/n8m/JP/AImj7PL/AM/k35J/8TQBPRUH2eX/AJ/JvyT/AOJo+zy/8/k35J/8TQBPXI+D/HB8XahfR26aStvaySRlYNU866XbIUBkhEY8sNtJHzHtXUpC6k7p5JARja4XH6AVyGi+F9WTxVZanq1to9jDpdpNa2yaUXxMsjg5ZGQCIAIDtDPyTzxyAdpRUH2eX/n8m/JP/iaUQSBgTdTEA9CE5/8AHaAOXvPHaxeNE0Cxt7O4dZlhmE2opBOWKhyYYmH70KpBb5lIzwDXXVwGt+A9Qv8AxBePZPZpp+pX1pfXNw8jrcQPBj5UUKQ4bYOSy7ctw3Su6eF2clbmVAf4VC4H5rQBLRUH2eX/AJ/JvyT/AOJo+zy/8/k35J/8TQBPRUH2eX/n8m/JP/iaPs8v/P5N+Sf/ABNADdRvk0zS7q+mSSSO2haZljXLMFBJAHrxXP8AhLxhP4ivruyvtMSxuLeCC6UwXJnjeKZSUyxRcPwcrg445Oa2tSsLq70a9tLbUZre4uIGjiuMLmFiCAwwB0zmuZ8DeE9V8P3N5Jerp+n280EUSWWmSvJG0i533DF0XDvkZG0/d5ZuwB2tFQfZ5f8An8m/JP8A4mj7PL/z+Tfkn/xNAE9VdU1GDSNJu9RvCRb2kLTSFRk7VGTj34p/2eX/AJ/JvyT/AOJqprGjDWNA1DS7i5kKXtu8Bchfk3KRkYA9aAM3wj4ql8TreO9rZRx27qqyWWpJdjJGTHJgKY5F43KQQM8Ma6SuO8JeF9V03WLvU9Yazt5ZLG3sI4dPld0KRbsSneq7WO7G3BwB95u3VfZ5f+fyb8k/+JoAnoqD7PL/AM/k35J/8TR9nl/5/JvyT/4mgB13cJZ2U9zL9yGNpG+gGT/Kuc8C+L38Z6SdRCaUkJClUsNU+2PGSM7JR5ahGAI4ya3Luzkn026t98dw00TIq3cQeIkgjDKMbl9RnpXPeGfDuq2+v32t63Fp1hc3FnBZJb6XK0kYWPcd+540wcuQF2kADqc0AdbRUH2eX/n8m/JP/iaPs8v/AD+Tfkn/AMTQBPRUH2eX/n8m/JP/AImj7PL/AM/k35J/8TQBPUF5/qF/66x/+hij7PL/AM/k35J/8TUF1BIIVzdTH97HwQn98f7NAF6ioPs8v/P5N+Sf/E0fZ5f+fyb8k/8AiaAJ6Kg+zy/8/k35J/8AE0fZ5f8An8m/JP8A4mgCeioPs8v/AD+Tfkn/AMTR9nl/5/JvyT/4mgCeioPs8v8Az+Tfkn/xNH2eX/n8m/JP/iaAJ6Kg+zy/8/k35J/8TR9nl/5/JvyT/wCJoAnqC8/1C/8AXWP/ANDFH2eX/n8m/JP/AImoLuCQQrm6mP7yPghP74/2aAL1FQfZ5f8An8m/JP8A4mj7PL/z+Tfkn/xNAE9FQfZ5f+fyb8k/+Jo+zy/8/k35J/8AE0AT0VB9nl/5/JvyT/4mj7PL/wA/k35J/wDE0AT0VB9nl/5/JvyT/wCJo+zy/wDP5N+Sf/E0AT0VB9nl/wCfyb8k/wDiaPs8v/P5N+Sf/E0AT0VB9nl/5/JvyT/4mj7PL/z+Tfkn/wATQBPRUH2eX/n8m/JP/iaPs8v/AD+Tfkn/AMTQBPRUH2eX/n8m/JP/AImj7PL/AM/k35J/8TQBPUV3cJZ2U9zL9yGNpG+gGT/Km/Z5f+fyb8k/+JqK7s5J9NurffHcNNEyKt3EHiJIIwyjG5fUZ6UAYfgXxe/jPSTqITSkhIUqlhqn2x4yRnZKPLUIwBHGTXUVyXhnw7qtvr99retxadYXNxZwWSW+lytJGFj3HfueNMHLkBdpAA6nNdN9nl/5/JvyT/4mgCeuR03x2ureMH0ext7OWCN5Y3f+0UF0pjO1pDbkZMW4FQ4YnP8ACBzXUpC6uC1zK4H8LBcH8lrhbXwHqFv4otXV7NNKs9Vn1aOeORxcu8qlTEybdu3Ln5txyABtHWgDv6KhMEhYkXUwBPQBOP8Ax2k+zy/8/k35J/8AE0AT0VB9nl/5/JvyT/4mj7PL/wA/k35J/wDE0AT1l+JNbXw74eutUeBrjyANsSnG5iwUZPYZIyewyau/Z5f+fyb8k/8AiayvFOjX2s+FbzTtPvjHczAbXlO1XAYExsVGQrAFSRk4J69KAG+FfEc3iCPUI72xWyvNNu2tLhIpjNEWADZSQqpYYYfwjByO1b1cv4M8O6hodjfx3ptLGO4ufNttP05y8FjHtA2IWRepBY4RQCxwO56H7PL/AM/k35J/8TQBPRUH2eX/AJ/JvyT/AOJo+zy/8/k35J/8TQBT8Q65B4d0ObUrlDIsZVEjBA3uzBVGT0ySOe1V/Cmvy+I9HN7LDZIPNZEl0/UEvLeYD+JJFAzzkEFVIIP1pnirw6/iLwvc6atwDMzpLE06gpvRw6hgoHykrg+xqv4V0DUNNfWL3VXht7zVrwXLwWUrSxQYRUAVnVckhck7RyfbJAOmoqD7PL/z+Tfkn/xNH2eX/n8m/JP/AImgCesrxPrkfhrwvqGszRiVbKBpfLaTYHI6Ddg4ye+KvfZ5f+fyb8k/+JqlrVhdXWhXEFolle3Jw0UeqR7oWYEEBto46cHBwecHGKAIvC2tyeIdGF/IdLZXchG0vUftsRA6/vPLTkHIIwfrWzXNeFtAv9Ol1m/1Q29teaveC5khsZDJHDhFQAOyKWJCgklR1x2zW99nl/5/JvyT/wCJoAnoqD7PL/z+Tfkn/wATR9nl/wCfyb8k/wDiaAJ6Kg+zy/8AP5N+Sf8AxNH2eX/n8m/JP/iaAB/+QjD/ANcpP5pU9VFjaPUYt0zy5if74XjlPQCrdABRRRQBBff8g65/65N/I0fbIv7s3/fh/wDCmanEk2k3kUo3I8DqwPcFTms37PrbWv8AZzvEI/uG/wDMPmNH/u44fHGc47+1AGr9si/uzf8Afh/8KPtkX92b/vw/+FSQQR21vHBAgSKNQiKOigDAFPoAg+2Rf3Zv+/D/AOFH2yL+7N/34f8AwqeigCD7ZF/dm/78P/hR9si/uzf9+H/wqeigCD7ZF/dm/wC/D/4UfbIv7s3/AH4f/CpjnadoBOOATiuW0fxNrd74zutC1DRbGCO0tlnmu7TUXnCFydkZDQJ8xAJ4JwMeooA6P7ZF/dm/78P/AIUfbIv7s3/fh/8ACp6KAPPviP4d0bxJq+haReaTY3F3qNziW6ktkaaO2iHmOquRlcnC8f3jiu68yG1VIVjdVVQFWOJiAOgHApZLS2luorqW3ie4gDCKVkBeMN94KeozgZx1qlc6dPHfPf6VKsc8gHnQy58ufAwCccqwAxuGeOCDgYALn2yL+7N/34f/AAo+2Rf3Zv8Avw/+FUWl1u5Hlpa21lnhpmmMpX/dXaMn6kfQ1dsrKKwt/Kh3HLFndzlpGPVmPcmgBftkX92b/vw/+FH2yL+7N/34f/Cp6KAIPtkX92b/AL8P/hR9si/uzf8Afh/8KnooAg+2Rf3Zv+/D/wCFH2yL+7N/34f/AAqeigCD7ZF/dm/78P8A4UfbIv7s3/fh/wDCp6KAIPtkX92b/vw/+FH2yL+7N/34f/Cp6KAIPtkX92b/AL8P/hR9si/uzf8Afh/8KnooAg+2Rf3Zv+/D/wCFH2yL+7N/34f/AAqeigCD7ZF/dm/78P8A4UfbIv7s3/fh/wDCp6KAIPtkX92b/vw/+FH2yL+7N/34f/Cp6KAIPtkX92b/AL8P/hR9si/uzf8Afh/8KnooAg+2Rf3Zv+/D/wCFH2yL+7N/34f/AAqeigCD7ZF/dm/78P8A4UfbIv7s3/fh/wDCp6KAIPtkX92b/vw/+FOS5SRwqrKCf70LAfmRUtFAHlltZnR/iXHq88Ph7VW1rWJbZJYIN95ZBYeB5+ewjw0YUY3fePf003casQVmyDjiFz/SqsOgaPb6zLq8Gk2MWpzLtlvUtkE0g44LgbiOB1PYVENGks+dFu2s07Wzr5kP4LkFfopA9qAL32yL+7N/34f/AAqC6u42hUBZv9bGeYXH8Y9qh+0a3FxJp9pP/tRXRXP/AAFk4/M1Ddz61LCoWytLYeZH88lwzkHeMfKFGf8AvoUAaf2yL+7N/wB+H/wo+2Rf3Zv+/D/4VSGi/aHD6xctfkHIiK7IVP8AuDr/AMCLVqUAQfbIv7s3/fh/8KPtkX92b/vw/wDhU9FAEH2yL+7N/wB+H/wo+2Rf3Zv+/D/4VPRQBB9si/uzf9+H/wAKPtkX92b/AL8P/hU9FAEH2yL+7N/34f8Awo+2Rf3Zv+/D/wCFT0UAQfbIv7s3/fh/8Kgu7uNoVAWb/WRnmFx/GPar1QXn+oX/AK6x/wDoYoAPtkX92b/vw/8AhR9si/uzf9+H/wAKnrB8aeKB4N8KXWttp11qS25UG3tFy5ywGfYDOSaANf7ZF/dm/wC/D/4UfbIv7s3/AH4f/Csax8SN4k02/k8LeQ8trIkKSXR/ds5RJGBCEnhXH4+3J6CgCD7ZF/dm/wC/D/4UfbIv7s3/AH4f/Cp6KAIPtkX92b/vw/8AhR9si/uzf9+H/wAKnqO4adbaRrSOOWcKTGkshRWbsCwBIHvg/SgBn2yL+7N/34f/AAo+2Rf3Zv8Avw/+Fc/4V8S6rruqavaalpFpZx6bKsH2m1vnuEllxuZRuijPygjJ5GTjsa6egCEXcbMAFmyTjmFx/SuC8R+GdD1/4qaPbvo9gZ7VDql7di2TzpNh2QoXxuI3ZbGf4AOnFeh1CLS2F6bwW8QumjERn2DeUBzt3dcZ5xQAr3KRuVZZSR/dhYj8wKb9si/uzf8Afh/8Kovp93YXUtxpDRskzF5bSZiqFjyWVgCVJ7jBBPPByaG/ti+HlNHFp0J4eRJfMlI9F4AX6nP0oAvfbIv7s3/fh/8ACj7ZF/dm/wC/D/4U61tobO1S3tkCRRjCqP8APJ96loAg+2Rf3Zv+/D/4UfbIv7s3/fh/8KnooAg+2Rf3Zv8Avw/+FH2yL+7N/wB+H/wqeigCD7ZF/dm/78P/AIUfbIv7s3/fh/8ACp6KAIPtkX92b/vw/wDhR9si/uzf9+H/AMKnooAg+2Rf3Zv+/D/4UfbIv7s3/fh/8KnooAg+2Rf3Zv8Avw/+FH2yL+7N/wB+H/wqeigCD7ZF/dm/78P/AIUfbIv7s3/fh/8ACp6KAIPtkX92b/vw/wDhR9si/uzf9+H/AMKnooAg+2Rf3Zv+/D/4UfbIv7s3/fh/8KnooAg+2Rf3Zv8Avw/+FH2yL+7N/wB+H/wqeigCD7ZF/dm/78P/AIUfbIv7s3/fh/8ACp6KAIPtkX92b/vw/wDhR9si/uzf9+H/AMKnooAiS5SQkKJAQM/PGyj8yK8x8C2Z0DxhbwTw+H9Rm1q3ur06pplv++XEwJV5snzUO8AHCYK4we3qdZ+n6Bo+k3VzdaVpNjZXF2264ltrZI3mOScuVALHJJ59TQBZ+2Rf3Zv+/D/4UfbIv7s3/fh/8Kzdf8PLrLWt1bXcmn6nYlmtL2Ibim7G5GU8OjbRuU9cAgggEUhf+NI18h9B0qeboLpNSeOE/wC0UMRZf90bvrQBtrMsuoxbA4xE/wB+Nl7p6irdc3o+gXFp4h/tbWr77fqs9s8e5F2Q20e5D5cSZOBkAliSzEDJwAB0lABRRRQBBff8g65/65N/I0fZ5f8An8m/JP8A4mi+/wCQdc/9cm/kaPtkX92b/vw/+FAB9nl/5/JvyT/4mj7PL/z+Tfkn/wATR9si/uzf9+H/AMKPtkX92b/vw/8AhQAfZ5f+fyb8k/8AiaPs8v8Az+Tfkn/xNH2yL+7N/wB+H/wo+2Rf3Zv+/D/4UAH2eX/n8m/JP/iaPs8v/P5N+Sf/ABNH2yL+7N/34f8Awo+2Rf3Zv+/D/wCFADo4nRstcSSDHRguP0ArG8NaHcaTNrNzfvFJc6lqElxviJP7vAWNTkDBCqBjpWv9si/uzf8Afh/8KPtkX92b/vw/+FAB9nl/5/JvyT/4mj7PL/z+Tfkn/wATR9si/uzf9+H/AMKPtkX92b/vw/8AhQAfZ5f+fyb8k/8AiaPs8v8Az+Tfkn/xNH2yL+7N/wB+H/wo+2Rf3Zv+/D/4UAH2eX/n8m/JP/iaPs8v/P5N+Sf/ABNH2yL+7N/34f8Awo+2Rf3Zv+/D/wCFAB9nl/5/JvyT/wCJo+zy/wDP5N+Sf/E0fbIv7s3/AH4f/Cj7ZF/dm/78P/hQAfZ5f+fyb8k/+Jo+zy/8/k35J/8AE0fbIv7s3/fh/wDCj7ZF/dm/78P/AIUAH2eX/n8m/JP/AImj7PL/AM/k35J/8TR9si/uzf8Afh/8KPtkX92b/vw/+FAB9nl/5/JvyT/4mj7PL/z+Tfkn/wATR9si/uzf9+H/AMKPtkX92b/vw/8AhQAfZ5f+fyb8k/8AiaPs8v8Az+Tfkn/xNH2yL+7N/wB+H/wo+2Rf3Zv+/D/4UAH2eX/n8m/JP/iaPs8v/P5N+Sf/ABNH2yL+7N/34f8Awo+2Rf3Zv+/D/wCFAB9nl/5/JvyT/wCJo+zy/wDP5N+Sf/E0fbIv7s3/AH4f/Cj7ZF/dm/78P/hQAfZ5f+fyb8k/+Jo+zy/8/k35J/8AE0fbIv7s3/fh/wDCj7ZF/dm/78P/AIUAH2eX/n8m/JP/AImj7PL/AM/k35J/8TR9si/uzf8Afh/8KPtkX92b/vw/+FAB9nl/5/JvyT/4mj7PL/z+Tfkn/wATR9si/uzf9+H/AMKPtkX92b/vw/8AhQAfZ5f+fyb8k/8AiaPs8v8Az+Tfkn/xNH2yL+7N/wB+H/wo+2Rf3Zv+/D/4UAH2eX/n8m/JP/iaPs8v/P5N+Sf/ABNH2yL+7N/34f8Awo+2Rf3Zv+/D/wCFAB9nl/5/JvyT/wCJo+zy/wDP5N+Sf/E0fbIv7s3/AH4f/Cj7ZF/dm/78P/hQAfZ5f+fyb8k/+Jo+zy/8/k35J/8AE0fbIv7s3/fh/wDCj7ZF/dm/78P/AIUAH2eX/n8m/JP/AImoLqCQQrm6mP72PghP74/2an+2Rf3Zv+/D/wCFQXV3G0KgLN/rYzzC4/jHtQBP9nl/5/JvyT/4mj7PL/z+Tfkn/wATR9si/uzf9+H/AMKPtkX92b/vw/8AhQAfZ5f+fyb8k/8AiaPs8v8Az+Tfkn/xNH2yL+7N/wB+H/wo+2Rf3Zv+/D/4UAH2eX/n8m/JP/iaPs8v/P5N+Sf/ABNH2yL+7N/34f8Awo+2Rf3Zv+/D/wCFAB9nl/5/JvyT/wCJo+zy/wDP5N+Sf/E0fbIv7s3/AH4f/Cj7ZF/dm/78P/hQAfZ5f+fyb8k/+Jo+zy/8/k35J/8AE0fbIv7s3/fh/wDCj7ZF/dm/78P/AIUAH2eX/n8m/JP/AImoLuCQQrm6mP7yPghP74/2an+2Rf3Zv+/D/wCFQXd3G0KgLN/rIzzC4/jHtQBP9nl/5/JvyT/4mj7PL/z+Tfkn/wATR9si/uzf9+H/AMKPtkX92b/vw/8AhQBjeF7G5XSpbq6luILi+uZbqRXjVWAZyIwwK5BEYjU/7tbP2eX/AJ/JvyT/AOJo+2Rf3Zv+/D/4UfbIv7s3/fh/8KAD7PL/AM/k35J/8TR9nl/5/JvyT/4mj7ZF/dm/78P/AIUfbIv7s3/fh/8ACgA+zy/8/k35J/8AE09I3RWzM8hI43hePyApn2yL+7N/34f/AAo+2Rf3Zv8Avw/+FAGR4Q0K40DwvFY3skbXrPLNczQkkPI7liwLAevcVr/Z5f8An8m/JP8A4mj7ZF/dm/78P/hR9si/uzf9+H/woAPs8v8Az+Tfkn/xNH2eX/n8m/JP/iaPtkX92b/vw/8AhR9si/uzf9+H/wAKAD7PL/z+Tfkn/wATR9nl/wCfyb8k/wDiaPtkX92b/vw/+FH2yL+7N/34f/CgA+zy/wDP5N+Sf/E0fZ5f+fyb8k/+Jo+2Rf3Zv+/D/wCFH2yL+7N/34f/AAoAPs8v/P5N+Sf/ABNH2eX/AJ/JvyT/AOJo+2Rf3Zv+/D/4UfbIv7s3/fh/8KAD7PL/AM/k35J/8TR9nl/5/JvyT/4mj7ZF/dm/78P/AIUfbIv7s3/fh/8ACgA+zy/8/k35J/8AE0fZ5f8An8m/JP8A4mj7ZF/dm/78P/hR9si/uzf9+H/woAPs8v8Az+Tfkn/xNH2eX/n8m/JP/iaPtkX92b/vw/8AhR9si/uzf9+H/wAKAD7PL/z+Tfkn/wATR9nl/wCfyb8k/wDiaPtkX92b/vw/+FH2yL+7N/34f/CgA+zy/wDP5N+Sf/E0fZ5f+fyb8k/+Jo+2Rf3Zv+/D/wCFH2yL+7N/34f/AAoAPs8v/P5N+Sf/ABNH2eX/AJ/JvyT/AOJo+2Rf3Zv+/D/4UfbIv7s3/fh/8KAD7PL/AM/k35J/8TR9nl/5/JvyT/4mj7ZF/dm/78P/AIUfbIv7s3/fh/8ACgA+zy/8/k35J/8AE0fZ5f8An8m/JP8A4mj7ZF/dm/78P/hR9si/uzf9+H/woAPs8v8Az+Tfkn/xNH2eX/n8m/JP/iaPtkX92b/vw/8AhR9si/uzf9+H/wAKAD7PL/z+Tfkn/wATR9nl/wCfyb8k/wDiaPtkX92b/vw/+FH2yL+7N/34f/CgA+zy/wDP5N+Sf/E0fZ5f+fyb8k/+Jo+2Rf3Zv+/D/wCFH2yL+7N/34f/AAoAPs8v/P5N+Sf/ABNH2eX/AJ/JvyT/AOJo+2Rf3Zv+/D/4UfbIv7s3/fh/8KAD7PL/AM/k35J/8TR9nl/5/JvyT/4mj7ZF/dm/78P/AIUfbIv7s3/fh/8ACgCNY2j1GLdM8uYn++F45T0Aq3VRZll1GLYHGIn+/Gy909RVugAooooAgvv+Qdc/9cm/kanpskayxPG/3XUqcehqL7PL/wA/k35J/wDE0AT0VB9nl/5/JvyT/wCJo+zy/wDP5N+Sf/E0AT0VB9nl/wCfyb8k/wDiaPs8v/P5N+Sf/E0AT0VB9nl/5/JvyT/4mj7PL/z+Tfkn/wATQBPRUH2eX/n8m/JP/iaPs8v/AD+Tfkn/AMTQBPRUH2eX/n8m/JP/AImj7PL/AM/k35J/8TQBPRUH2eX/AJ/JvyT/AOJo+zy/8/k35J/8TQBPRUH2eX/n8m/JP/iaPs8v/P5N+Sf/ABNAE9FQfZ5f+fyb8k/+Jo+zy/8AP5N+Sf8AxNAE9FQfZ5f+fyb8k/8AiaPs8v8Az+Tfkn/xNAE9FQfZ5f8An8m/JP8A4mj7PL/z+Tfkn/xNAE9FQfZ5f+fyb8k/+Jo+zy/8/k35J/8AE0AT0VB9nl/5/JvyT/4mj7PL/wA/k35J/wDE0AT0VB9nl/5/JvyT/wCJo+zy/wDP5N+Sf/E0AT0VB9nl/wCfyb8k/wDiaPs8v/P5N+Sf/E0AT0VB9nl/5/JvyT/4mj7PL/z+Tfkn/wATQBPRUH2eX/n8m/JP/iaPs8v/AD+Tfkn/AMTQBPRUH2eX/n8m/JP/AImj7PL/AM/k35J/8TQBPRUH2eX/AJ/JvyT/AOJo+zy/8/k35J/8TQBPRUH2eX/n8m/JP/iaPs8v/P5N+Sf/ABNAE9FQfZ5f+fyb8k/+Jo+zy/8AP5N+Sf8AxNAE9FQfZ5f+fyb8k/8AiaPs8v8Az+Tfkn/xNAE9QXn+oX/rrH/6GKPs8v8Az+Tfkn/xNNe0eRQHu5iAQeidQcj+H1FAFmioPs8v/P5N+Sf/ABNH2eX/AJ/JvyT/AOJoAnoqD7PL/wA/k35J/wDE0fZ5f+fyb8k/+JoAnoqD7PL/AM/k35J/8TR9nl/5/JvyT/4mgCeioPs8v/P5N+Sf/E0fZ5f+fyb8k/8AiaAJ6Kg+zy/8/k35J/8AE0fZ5f8An8m/JP8A4mgCeoLz/UL/ANdY/wD0MUfZ5f8An8m/JP8A4mmvaPIuHu5iMg9E6g5H8PqKALNFQfZ5f+fyb8k/+Jo+zy/8/k35J/8AE0AT0VB9nl/5/JvyT/4mj7PL/wA/k35J/wDE0AT0VB9nl/5/JvyT/wCJo+zy/wDP5N+Sf/E0AT0VB9nl/wCfyb8k/wDiaPs8v/P5N+Sf/E0AT0VB9nl/5/JvyT/4mj7PL/z+Tfkn/wATQBPRUH2eX/n8m/JP/iaPs8v/AD+Tfkn/AMTQBPRUH2eX/n8m/JP/AImj7PL/AM/k35J/8TQBPRUH2eX/AJ/JvyT/AOJo+zy/8/k35J/8TQBPRUH2eX/n8m/JP/iaPs8v/P5N+Sf/ABNAE9FQfZ5f+fyb8k/+Jo+zy/8AP5N+Sf8AxNAE9FQfZ5f+fyb8k/8AiaPs8v8Az+Tfkn/xNAE9FQfZ5f8An8m/JP8A4mj7PL/z+Tfkn/xNAE9FQfZ5f+fyb8k/+Jo+zy/8/k35J/8AE0AT0VB9nl/5/JvyT/4mj7PL/wA/k35J/wDE0AT0VB9nl/5/JvyT/wCJo+zy/wDP5N+Sf/E0AT0VB9nl/wCfyb8k/wDiaPs8v/P5N+Sf/E0AT0VB9nl/5/JvyT/4mj7PL/z+Tfkn/wATQBPRUH2eX/n8m/JP/iaPs8v/AD+Tfkn/AMTQBPRUH2eX/n8m/JP/AImj7PL/AM/k35J/8TQBPRUH2eX/AJ/JvyT/AOJo+zy/8/k35J/8TQBPRUH2eX/n8m/JP/iaPs8v/P5N+Sf/ABNAE9FQfZ5f+fyb8k/+Jo+zy/8AP5N+Sf8AxNAA/wDyEYf+uUn80qeoY7crMJXmkkYKVG4LxnGegHoKmoAKKKKACiiigCOaeO3jDzOEUuqAn+8zBVH4kgfjUlZniD/kGxf9f1p/6UR1ejuYZbiWCOVWlhx5iA8pkZGfrQBLRRRQAUUUUAFRxzxyySpG4ZoX2SAfwttDY/JgfxqSszSv+Qlrf/X8v/pPDQBp0UUUARyTxxSRJI4Vpn2Rg/xNtLY/JSfwqSszVf8AkJaJ/wBfzf8ApPNWnQAVHPPHbW8k87hIokLu56KoGSfyqSszxN/yKer/APXjN/6LNAGnRRRQAVHHPHLJKkbhmhfZIB/C20Nj8mB/GpKzNK/5CWt/9fy/+k8NAGnRRRQBGZ41uEgLgSujOqdyqkAn8Cy/nUlZlx/yNlh/143P/oyCtOgAooooAKKKKACo4Z47iMvC4dQ7ISP7ysVYfgQR+FSVmeH/APkGy/8AX9d/+lElAGnRRRQAUUUUARwzx3EZeFw6h2Qkf3lYqw/Agj8KkrM8P/8AINl/6/rv/wBKJK06ACiiigAqOGeO4jLwuHUOyEj+8rFWH4EEfhUlZnh//kGy/wDX9d/+lElAGnRRRQBGJ42uHgDgyoiuydwrEgH8SrflUlZlv/yNl/8A9eNt/wCjJ606ACiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooAjhnjuIy8Lh1DshI/vKxVh+BBH4VJWZ4f/wCQbL/1/Xf/AKUSVp0AFRmeNbhIC4ErozqncqpAJ/Asv51JWZcf8jZYf9eNz/6MgoA06KKKAI5p47eMPM4RS6oCf7zMFUfiSB+NSVmeIP8AkGxf9f1p/wClEdadABRRRQBHDPHcRl4XDqHZCR/eVirD8CCPwqSszw//AMg2X/r+u/8A0okrToAKjE8bXDwBwZURXZO4ViQD+JVvyqSsy3/5Gy//AOvG2/8ARk9AGnRRRQAUUUUAFFFFAEcE8dzbxzwOHilQOjjoykZB/KpKzPDP/Ip6R/14w/8AosVp0AFFFFABUcE8dzbxzwOHilQOjjoykZB/KpKzPDP/ACKekf8AXjD/AOixQBp0UUUAFFFFAEcE8dzbxzwOHilQOjjoykZB/KpKzPDP/Ip6R/14w/8AosVp0AFRxzxyySpG4ZoX2SAfwttDY/JgfxqSszSv+Qlrf/X8v/pPDQBp0UUUAFFFFABRRRQAUUUUAZPiWWODR0lmdY447y1Z3c4CgXEZJJ7CnDxJoAYkazpoJ6n7VHz+tW9Q0+11WwlstQi863lxvQkjOCCORz1ArB/4Vz4V/wCgX/5MS/8AxVAGp/wk2g/9BvTv/AuP/Gj/AISbQf8AoN6d/wCBcf8AjWX/AMK58K/9Av8A8mJf/iqP+Fc+Ff8AoF/+TEv/AMVQBqf8JNoP/Qb07/wLj/xo/wCEm0H/AKDenf8AgXH/AI1l/wDCufCv/QL/APJiX/4qj/hXPhX/AKBf/kxL/wDFUALr3iyC30sv4f1HSbq83gCOa8jC7e5++v8AOk8D391qdlqV3qAtxcSX3zi1kDx8QxAYIZgeAO55zR/wrnwr/wBAv/yYl/8Aiq2dJ0aw0Oza10q38iFnMhXezZYgDOSSewoAvUUUUAcx44v7rTLLTbvTxbm4jvvkF1IEj5hlBySygcE9xziuY/4TvxV/zy8O/wDgdF/8fr0DVtGsNcs1tdVt/PhVxIF3suGAIzkEHuaxv+Fc+Ff+gX/5MS//ABVAHMf8J34q/wCeXh3/AMDov/j9VtT8aeJbrSbu3uY9BEMsDpIYr2Nn2lSDtAmJJx0GD9K7D/hXPhX/AKBf/kxL/wDFUf8ACufCv/QL/wDJiX/4qgDmP+E78Vf88vDv/gdF/wDH6P8AhO/FX/PLw7/4HRf/AB+un/4Vz4V/6Bf/AJMS/wDxVH/CufCv/QL/APJiX/4qgDmP+E78Vf8APLw7/wCB0X/x+un8D391qdlqV3qAtxcSX3zi1kDx8QxAYIZgeAO55zR/wrnwr/0C/wDyYl/+KrZ0nRrDQ7NrXSrfyIWcyFd7NliAM5JJ7CgC9RRRQBmXH/I2WH/Xjc/+jIKvQLOqv9pkRyXJUou3C54HU8+9ZPiTwpYeKY7ddQeeP7OWKNCwB5xkcg+grQ0vTotI0u3sLZnaKBNimQgsR74AoAt0UUUAFFFFABXO6Rrmk2drcQXeqWUEyX11ujluEVlzcSEZBOehroq5668B+G728murnTd80zmSRhPINzE5JwGxQBe/4SbQf+g3p3/gXH/jR/wk2g/9BvTv/AuP/Gsv/hXPhX/oF/8AkxL/APFUf8K58K/9Av8A8mJf/iqANT/hJtB/6Denf+Bcf+NH/CTaD/0G9O/8C4/8ay/+Fc+Ff+gX/wCTEv8A8VR/wrnwr/0C/wDyYl/+KoA0fDUsc+jvLC6yRyXl0yOhyGBuJCCD3Fa1VtP0+10qwistPi8m3izsQEnGSSeTz1JqzQAUUUUAFeYv4s1/TL2+tNPj0U28d9c7DdXcaSczOTkGVSOSew4xXp1c9deA/Dd7eTXVzpu+aZzJIwnkG5ick4DYoA5L/hO/FX/PLw7/AOB0X/x+j/hO/FX/ADy8O/8AgdF/8frp/wDhXPhX/oF/+TEv/wAVR/wrnwr/ANAv/wAmJf8A4qgCt4K1bUNZ1bVLjVVslmWC3RRZSrIm3dKeSrMM5J4z6cV2FZmj+HdL0BZhpFr9n87HmfvGbdjOPvE+prToAKKKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigDi9FvPFC6zJbw6batov8AaFyDclh5m3zn3HG/+9kfdrtK5lPAelR+Jv7cSW6FyZjPsDr5e49eNue/rXTUAFcf411bUNG1bS7jSlsmmaC4RheyrGm3dEeCzKM5A4z68V2FZmseHdL19YRq9r9o8nPl/vGXbnGfukegoA4T/hO/FX/PLw7/AOB0X/x+j/hO/FX/ADy8O/8AgdF/8frp/wDhXPhX/oF/+TEv/wAVR/wrnwr/ANAv/wAmJf8A4qgDj9Q8aeJbm2RJ49BCieFx5d7GTuWRWXpMeMgZPYZOR1qz/wAJ34q/55eHf/A6L/4/XT/8K58K/wDQL/8AJiX/AOKo/wCFc+Ff+gX/AOTEv/xVAHMf8J34q/55eHf/AAOi/wDj9H/Cd+Kv+eXh3/wOi/8Aj9dP/wAK58K/9Av/AMmJf/iqP+Fc+Ff+gX/5MS//ABVAFrwZPLdeF4bi5EYmlnuHkETBk3GdydpBIIz0OT9a3arafp9rpVhFZafF5NvFnYgJOMkk8nnqTVmgArLg/wCRs1DH/Pjbf+hz1qVzviPwRpnie8iur+W5jlij8sGBlGRknnKn1NAG7bLOtugunSSYD52RdoJ9hk1LUcEK29vHChJWNAgJ64AxUlABRRRQAUUUUAc34e8Q6LB4Y0uKbV7COSOziV0e5QFSEAIIzwa0f+Em0H/oN6d/4Fx/41mH4deFixJ0rr6XEo/9mpP+Fc+Ff+gX/wCTEv8A8VQBqf8ACTaD/wBBvTv/AALj/wAaP+Em0H/oN6d/4Fx/41l/8K58K/8AQL/8mJf/AIqj/hXPhX/oF/8AkxL/APFUAan/AAk2g/8AQb07/wAC4/8AGjwz/wAinpH/AF4w/wDosVl/8K58K/8AQL/8mJf/AIqujt7eK0tYre3TZFCgjRR/CoGAPyoAkooooAKKKKAPKNM8aeJbXSbS3to9BMMUCJGZb2NX2hQBuBmBBx1GB9Ks/wDCd+Kv+eXh3/wOi/8Aj9dQfh14WLEnSuvpcSj/ANmpP+Fc+Ff+gX/5MS//ABVAHMf8J34q/wCeXh3/AMDov/j9dP4Hv7rU7LUrvUBbi4kvvnFrIHj4hiAwQzA8Adzzmj/hXPhX/oF/+TEv/wAVWzpOjWGh2bWulW/kQs5kK72bLEAZyST2FAF6iiigAooooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooA//2Q==)

Figure | Tab Notation (https://www.ultimate-guitar.com, 2022)

## 3.3. | Research for Supporting the Aim and Objectives

To create a bespoke and successful entertainment or educational product which is unique and not yet invented, 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 design, we must explore the current gaming and musical entertainment devices that might be relevant to our project. Brief precautionary research will prevent us from reinventing the wheel and will refine the project's outline more accurately.

### 3.3.1. | 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 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 is similar to RiffMaster; some parts may be the perfect blueprints for our hardware prototype, such as its strum bar OFF-ON-OFF momentary switch button component.

**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, 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 will cost as much as a decent acoustic or electronic instrument.

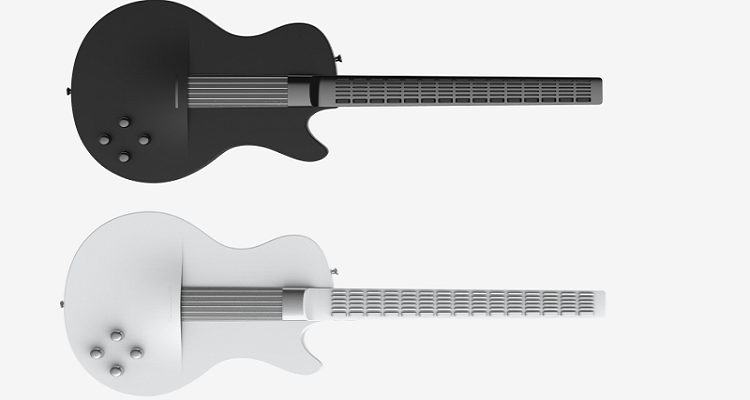


Figure | MI Digital Guitar Series (https://www.digitalmusicnews.com/2016/08/01/mi-guitar-easy-to-learn, 2022)

**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 drills, while others are unique features, like tuning and uploading the players' music.

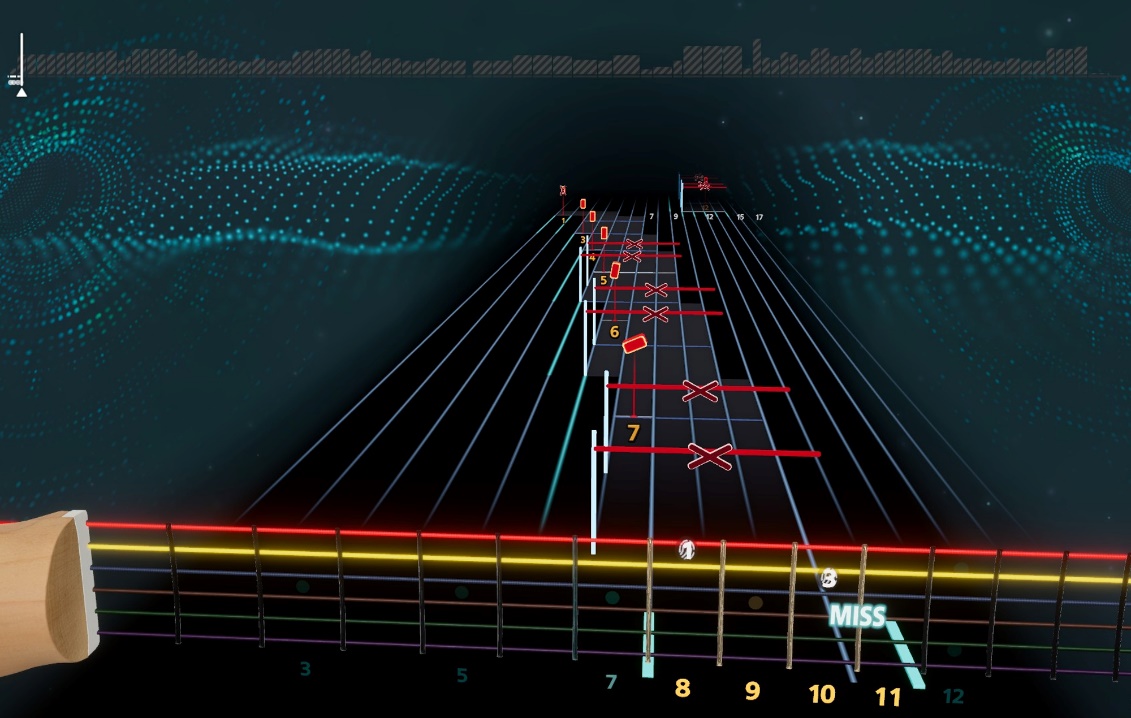


Figure | RockSmith Screenshot (https://www.pushsquare.com/games/ps4/rocksmith\_2014\_edition/screenshots, 2022)

### 3.3.2. | 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 the assessment of our findings and some further specifications.

Table

Description automatically generated

Figure | Technology Comparison (Appendix/Comparision.drawio)

### 3.3.3. | 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.

While Guitar Hero has outstanding gameplay, its unrealistic controller layout lacks a realistic user experience. On the other hand, MI's 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 must buy a decent-quality instrument 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.

## 3.4. | Console Design Research

### 3.4.1. | Calculating Fret Distances

As the specifications outline, the console layout must simulate the guitar experience as authentically as possible. Current devices like the MI Digital Guitar do not account for the distances between the frets and button cap sizes. Therefore, we must 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 and by basing our measurements on a Guitar Hero console for our experimental calculations.

**Text

Description automatically generated**

Table

Description automatically generated with low confidence

Figure | Fret Distances (Appendix/Fret-Distance-Table.drawio)

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 (Appendix/Fret-Distance-Neck-Diagram.drawio)

### 3.4.2. | **Key Switch Interface Design Concepts**

To build a successful guitar system based on time precision, we need to detect inputs from the user interface accurately and efficiently. From an electrical engineer's perspective, our console can be abstracted as an intricate keyboard input consisting 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.).

#### 3.4.2.1. | 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 Console 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.

#### 3.4.2.2. | 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.

Matrix keyboards use scanning algorithms to detect button presses, where rows and columns are individually read.

Diagram

Description automatically generated with medium confidence

Figure | Keyboard Scanning Flowchart (Appendix/Keyboard-Scanning.drawio)

Unfortunately, keyboard matrices introduce problems with simultaneous key presses, such as keyboard ghosting (unrecognised strokes) and masking (unpressed strokes mistakenly registered). These problems are well-known in the gaming community, and anti-ghosting keyboards are sold for professional gamers.

A picture containing text, crossword puzzle

Description automatically generated

Figure | Indistinguishable Keystrokes (https://www.microsoft.com/applied-sciences/projects/anti-ghosting, 2022)

Similarly, ghosting and masking may cause problems when the player simultaneously interacts with the controller's fret buttons, playing triads or chords. Although these problems may be corrected using diodes, our scenario may have better wiring alternatives.

### 3.4.3. | Analog Voltage Dividers

Traditionally buttons are registered through one of the digital pins because of the momentary press input's binary nature. However, with an ingenious trick, we may capture keypresses on analog pins, similar to how potentiometers work with microcontrollers or how values are read from a sensor. "*To get the value from the sensor, call analogRead() that takes one argument: what pin it should take a voltage reading on. The value, which is between 0 and 1023, is the representation of the voltage on the pin*" (Fitzgerald, 2012).

#### 3.4.3.1 | Logarithmic Resistor Ladders

This analog design could eliminate the ghosting and masking problems because only the topmost button press on each row (string) should be registered, similarly to the real-world instrument. An additional benefit of this design is the reduced number of pins used: six analog pins for strings and six digital pins for strums. One way of creating analog voltage dividers is to use sequential wiring of switches and identical resistors, often called resistor ladders.

Graphical user interface

Description automatically generated

Figure | Resistor Ladder Solution (Appendix/Resistance-Ladder.drawio)

Graphical user interface, application, table, Excel

Description automatically generated

Figure | ACD Logarithmic Value Readings

Unfortunately, even though this solution is straightforward and might work for projects with fewer switches, it is problematic. Even if we assume perfect resistors, we must interpret our voltage divider values on the ADC by hard-coding a logarithmic range. The reading would be inaccurate since each string consists of twenty buttons. Therefore, we need to extend our research for a more elegant solution that can precisely read button interactions without being compromised by the short distances between the upper values of the logarithmic curve.

**3.4.3.2. | Linear Resistor Ladders**

We can equalise the distance between the ADC readings by calculating the resistance for each button switch.

Finding twenty resistor values require more than a handful of calculations. Therefore, writing a small algorithm to give us the resistance values is more straightforward. Additionally, we can later adjust the values in the program if we need any changes in the electrical design.

Text, letter

Description automatically generated

Figure | Linear Resistance Ladder (Appendix/linear\_ladder.py)

### 3.4.4. | 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 (based on: https://docs.arduino.cc/built-in-examples, Appendix/debounce.cpp)

**3.5. Human Input Device Protocol**

We need a protocol that works through the USB connection to establish efficient communication between the console and the computer. The Human Interface Device Protocol "(*HID) is designed for common PC interface devices such as keyboard and mouse, but can be adapted for many custom applications*" (Murphy, 2017). Arduino uses HID communication through Serial class, a built-in method for simple I/O actions using ASCII characters; however, even though a USB cable powers most Arduinos, bidirectional USB communication is only possible on Arduino Leonardo.

Table

Description automatically generated

Figure | Protocol Example: Playing F#min7 Chord on Console (Appendix/Serial-Protocol.drawio)

ASCII keyboard communication seems the most straightforward solution because characters may be easily interpreted into corresponding user actions. However, a significant drawback is that using our console device as a keyboard might interfere with the user's PC keyboard, and the user will not be able to use it properly while playing the game. Fortunately, some Arduino libraries let us connect our device to the PC as a joystick, similar to Xbox controllers.

## 3.6 | Audio in the Browser

One of the fundamental elements of the application is to engage the player through music and seamless game interaction. "*Well-designed games create engagement by promoting a low state, a total absorption that makes the player gratifyingly oblivious to anything else. Good musical experiences also involve low states, and music classes are most effective when they foster flow*"**.** (Csíkszentmihályi, 2009)However, to create the right conditions for flow, the players need audio feedback of several kinds.

The most apparent audio experience is the chosen piece of music that the player will follow with the game controller. Nevertheless, there may be a need for additional audio feedback, such as a sound simulation the guitar controller generates. This generated sound may be a template guitar audio stock that can be distinguishable from the original music. Another benefit is that the player can hear the generated music, which is immediate feedback for errors. On the other hand, some inexperienced players may not hear out errors in their performance. Therefore, audio and visual feedback may be needed for mistakenly played or missed music notes.

Fortunately, web browsers support audio interactions, and there can be multiple audio files playing simultaneously, but the implementation is a costly business. Media files are typically larger than text-based files, and the application might need to preload some data to the cache. "*Web caching is the temporary storage of Web objects (such as HTML documents) for later retrieval. There are three significant advantages to Web caching: reduced bandwidth consumption (fewer requests and responses that need to go over the network), reduced server load (fewer requests for a server to handle), and reduced latency*" (Sulaiman, 2008). However, browsers do not handle caching uniformly, which might cause browser vendor compatibility problems, and users may also have different browser settings that prevent caching.

# 4. | Design

The entertainment system has multiple well-separable factors, such as the game controller's physical design, communication with the user interface, or the application design. While each of them will go through separate design phases, they must be consolidated into one compatible working interface.

## 4.1 | Controller Design

Our controller may be divided into three fundamental central units, the neck, where the user interacts with the fret inputs; an input solution for the strum; and a body. The main focus of our design is how to arrange our elements in a simple, feasible, and usable manner.

### 4.1.1. | Physical Design and Electrical Housing

When designing the game controller's hardware, we must know what physical elements our user interface and controller communication may involve. For instance, multiple types of inputs: 120 tactile momentary switches, a strum switch solution and an ON-OFF toggle switch. Additionally, the hardware must accommodate a microcontroller, wiring of hundreds of electrical components, a limited number of UI outputs (LEDs), and USB conversion plugs and cables. This extensive list of components suggests that using a toy-sized controller such as the Guitar Hero console may be too restrictive. Unlike industrial counterparts, the elements must be assembled into a hand-made device. Instead, reusing existing designs of real-sized instruments would be beneficial for a more reasonable physical real estate management. Many options exist, such as the classical acoustic or the modern electric guitar. However, the number of frets and physical constraints indicate that a semi-acoustic design would be a good candidate for our prototype.

A group of guitars

Description automatically generated with medium confidence

Figure | Guitar Types (https://www.pianodreamers.com/guitar-buying-guide/)

### 4.1.2. | Imitating Gibson SE Les Paul Style Guitars

Semi-acoustic guitars are often open-cavity instruments, which is ideal for our purposes. One of the most popular and well-recognisable designs is the Les Paul Gibson. Therefore, we may incorporate some of its fundamental features, such as the dimensions and body style. In addition, we may use open-source vector graphics to achieve better accuracy when creating the body's curves.

Diagram, engineering drawing

Description automatically generated

Figure | Editing Les Paul Gibson's Vector Graphic in Adobe Illustrator

### 4.1.3. | Left Hand Control Design: Fret Buttons

The user will interact with a range of buttons in a mesh arrangement on the left hand to communicate which string has active finger press positions associated. Hence, the neck width and depth must be small enough that all six buttons on a fret may be pressed simultaneously with an index finger for barre chords. However, it must be wide enough to leave sufficient space for the electronics. Les Paul uses a 42mm width closer to the headstock and 57mm at the junction of the neck and body.

The console neck will have a constant width to simplify the hardware development, and 50mm (48mm button space with 2mm margins) is a compromise that satisfies the usability requirement. At the same time, it can concisely allocate all the electrical components needed. The length of the neck is the standard 460mm, and the fret distances are calculated with the golden ratio discussed in previous chapters.

A screenshot of a computer

Description automatically generated with low confidence

Figure | Drilling Plan, Button Layout, Fret Distances (Appendix/Drilling Template.drawio)

The switch drill holes will be evenly allocated alongside the axis of the fret. The distribution of drill holes comfortably accommodates the 0.7mm wide (plus soldering) switch legs in a 6mm by 4mm rectangular arrangement.

Diagram

Description automatically generated with low confidence

Figure | Neck Button Arrangement (Appendix/Neck Design.drawio)

Every fret will receive string support with grooves. This additional element offers protection for tactile switches, provides a guide to the users' hand, and can be used to keep actual strings in place (for an optional string solution).

A picture containing diagram

Description automatically generated

Figure | Fret Support (Appendix/Fret Design.drawio)

The front of the neck and the buttons will face towards the user's fingers so that no other electronic components may fit the otherwise compressed interface. The switch legs must lead through the drill holes by wire extensions to reach the backside and compose a circuit.

The electronics must be protected from physical impacts; therefore, a back cover and side protection are required. Moreover, to securely hold the neck into position in the instrument body, the back cover and the side will be extended, and its height will be increased to fit the whole body's depth.

Diagram

Description automatically generated

Figure | Neck Cover (Appendix/Neck Cover.drawio)

### 4.1.4. | Electrical Layout

The electrical layout is one of the most challenging parts of the controllers' design. The below diagram includes only the components required for the left-hand side interactions; everything else is omitted.

A picture containing text, electronics

Description automatically generated

Figure | Neck Electric Circuits (Electric Design.draw.io)

The interface design research chapter discovered several possible electric solutions, such as the analogue voltage dividers like the linear and logarithmic resistor ladders. Nevertheless, the chosen solution is matrix wiring using diodes to eliminate ghosting and masking issues.

The following reasons directed me towards the selection:

* **Linear ladders**: building any of the solutions mentioned above will take weeks, and linear ladders may not read this many switch positions accurately, an unnecessary risk that would compromise the entire project.
* **Logarithmic ladders**: a working solution would be accurate; however, obtaining the necessary components with the exact resistance may take a long time.
* **Keyboard Matrices**: a button matrix may not seem the most elegant solution; however, it has certain aspects that are crucial to a successful, in-time completion of the project:
  + **Fewer types of components**: only switches, wires and diodes; any N1 range diode is suitable for our scenario (N14007 will be used).
  + **Accurate**: voltages are read accurately (tested on a 2 x 2 matrix).
  + **Simple, Clean and Proven**: the most critical factors, as 120 switches are involved.
  + **Development Time**: no significant difference in the assembly as the complexity and the number of components are the same for this many switches.
  + **Disadvantages:** a 6 x 20 solution in such a concise arrangement logically dictates using a stripped wire for the row bus (*Figure 44 Row Bus*). It may cause contact failures when too many components are placed and soldered together.

Graphical user interface, text, application

Description automatically generated with medium confidence

Figure | Electric Diagram of 2x6 Matrix (Appendix/Fret Electric.drawio)

### 4.1.5. | Right Hand Control Design: Body and Strum

The user will have multiple control options on the right-hand side that may serve as input or output. The most significant component is the strum unit, which triggers events that produce sound effects or gameplay actions. But beyond the strum, the instrument's body must allocate a standard-size ON / OFF switch, LEDs for major device events, USB connection, wiring, and support elements to fix the neck, cover, and overall body together. The available real estate suggests that some features may have limited possibilities for an elegant and executable solution, like bending perfect curves from plastic or creating supporting elements that fit surgically together. Hence, our design must consider an error rate of up to 0.5mm or even more in some cases.

Graphical user interface, diagram

Description automatically generatedFigure | Body Cutting and Drilling Instructions (Appendix/Body Design.drawio)

#### 4.1.5.1. | Body Design

The body is made of several plastic parts that have to be assembled in a way to allow disassembly for servicing. The body and the cover will be held together by several connector units that concise the support material and nuts for the screws. Additionally, these connectors are beneficial for providing extra support, as this handheld device may be put under more stress than other electrical devices. The body must withstand the weight of the user's right arm, and the pressure from the strumming must not cave in the hollow device.

Most importantly, the body needs to allocate the connection to the neck safely and functionally. One of the typical guitar design failures originates from weak neck-body links, exposing problems such as a bent or damaged guitar neck. And similarly to the actual guitar instrument, substantial pulling or pushing forces may easily damage the device or even break the controller apart, as the neck provides significant leverage. Therefore, the body must feature additional supports to reinforce the long neck and tolerate reasonable use.

Graphical user interface, application

Description automatically generated

Figure Body from Inside (Appendix/Body Support.drawio)

#### 4.1.5.2. | Strum Design Options

Several solutions may be applied for building a successful strum unit:

* **Vibration Sensor**: We can read which string has been strummed with designated vibration sensors, and even the amount of pressure and length of notes may be read. Even though it would be an ideal solution, the feasibility of building an accurate and concise strum unit this way is low.
* **Pressure Sensor**: Similarly, we can read the force applied on a string by a force meter. However, the cost and space requirement is significant.
* **Capacitive Sensors**: Proximity or capacity sensors may also read individual string strums. They are relatively cost-effective and would fit in the limited space. Still, they may add to the complexity of the solution and are sensitive to environmental factors, such as humidity and temperature.
* **Optical Sensors**: This solution may involve a light source above each string. When the user strums the string, the sensor can read the incoming light or changes in the light source. This intriguing solution, however, requires a lot of experimentation.
* **Magnetic Sensors**: We may also read the magnetic property changes on the strings as the user touches them. Unfortunately, this may result in false readings, as the user may touch the strings without the intention to apply any force on them.
* **Strum Bar**: We can cut corners using six Guitar Hero Strum Bars on the guitar. Unfortunately, these strum bars are difficult to find and require more space than available.
* **Momentary Tactile Switches**: The most straightforward solution is to use six buttons to trigger strum events. This method is the least user-friendly, as it does not simulate the guitar strumming experience but is an excellent backup if everything else fails.

#### 4.1.5.3. | Strum Solution: Mechanical String Switch Design with Levers

Our chosen method is a quirky compromise between a switch and a pressure sensor solution. Each string is between a fixed point and a momentary switch that can act on lateral forces to trigger strum events. The optimum tension may be set with tuning pegs or screws, and springs may be applied to mitigate or soften the pressures of solid strums. A medium-difficulty solution that resembles more of the actual instrument mechanism. It fits the available real estate, can be delivered quickly and has a decent enough string tension and accuracy to produce an authentic experience.

On the downside, the string switch is not an analogue input and is less versatile than sensor readings, and extensive fine-tuning is required to get the desired tension on the strings. Furthermore, the tension and strum vibration between the bridge and lever may produce a byproduct noise that must be muffled with a soft material. More importantly, levers are mechanical components and prone to physical damage. The user may be required to set the tension or adjust it periodically, as strings loosen with time, and users may damage the unit while tuning. Lastly, user opinion may be divided on such a solution; experienced users may find the option for adjusting string pulls authentic, while beginners may feel intimidated about tuning.

The mechanism works the following way:

Text

Description automatically generated

Figure | Lever Mechanism (Appendix/Lever\_Mechanism.drawio)

One of the most complex units in terms of physical design is our strum unit, and it comprises the following elements:

* Lever-Bridge (#1A/#3A),
* Unit Base (#2),
* Bridge (#1B/#3B),
* Six screws, nuts, springs for each side, and six levers,
* Hull-base (#4) and sides (#5A/B).

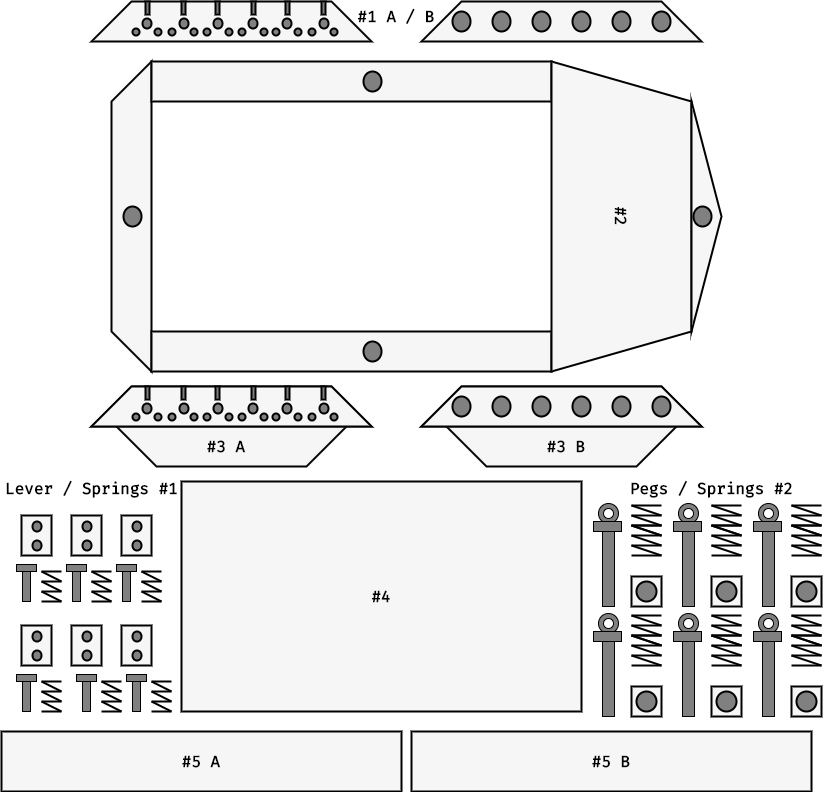


Figure | Strum Components (Appendix/Strum Components.drawio)

The strum unit is designed to be replaceable and completely separate from the main console body. The unit's hull sits in the body cavity in a way that requires a minimum elevation from the surface (10mm – 15mm) but provides space for comfortable strumming (approx. 30mm). The finalised strum unit follows the assembly diagram below:

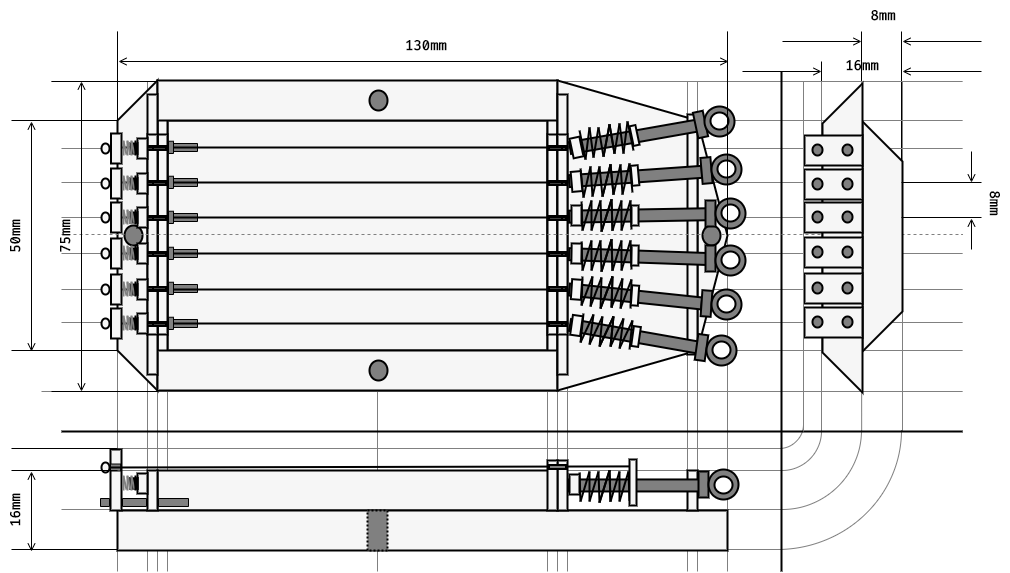


Figure | Strum Unit Assembly Projection Diagram (Appendix/Strum\_Unit.drawio)

#### 4.1.5.4. | Toggle and LEDs

One of the specification requirements is to allow the user to interact with the device without disrupting traditional inputs. However, a USB keyboard communication may be utilised depending on the final communication protocol. In this case, an involuntary strum may send keyboard messages through the USB port. Hence, the device must have an ON/OFF switch that disables any serial communication towards the computer. Additionally, three LEDs will provide further information about the device's state:

* RED: Device connected to USB,
* GREEN: USB communication is ON,
* BLUE: The user interacted with the device by pressing any button or strum (for testing and troubleshooting).

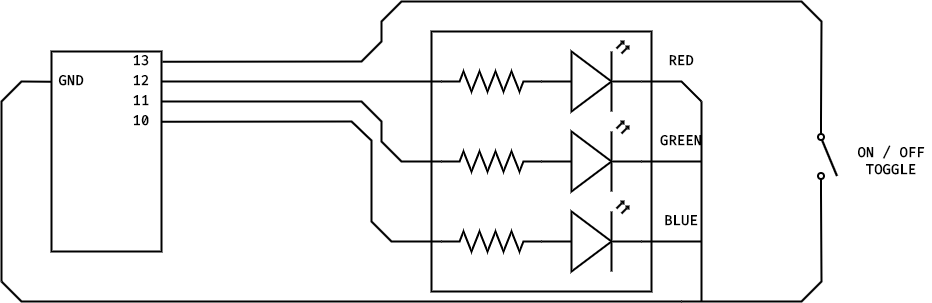


Figure | Toggle and LEDs Electrical Diagram (Appendix/Toggle\_and\_LEDs.drawio)

### 4.1.6 | Head Stock Design

Even though the user can interact with the fret by buttons and use any finger position or chord pattern on the left-hand side of the console, controls may not feel authentic enough for some users. Therefore, users may string up the fretboard using the headstock, which provides fixed positions to attach the strings, improving the playing experience by providing a guideline to the left hand. The headstock unit is an optional accessory and must be attachable and detachable to the guitar neck.

Diagram, engineering drawing

Description automatically generated

Figure |Head Stock Projection Diagram (Appendix/Head\_Stock.drawio)

## 

## 4.2. | Frontend Design

The frontend application for the prototype will be built using traditional web technologies, such as HTML5, CSS and JavaScript, and will be based on the wireframes provided in the requirement specifications. A feature-rich commercial implementation would probably benefit from specific frameworks' advantages, such as app state management (React) or CSS libraries (SaSS). However, these may add extra complexity for prototyping and will be omitted.

### 4.2.1. | User Activities

The following user activities are the most important ones for presenting a working prototype's basic functionalities:

A screenshot of a computer

Description automatically generated with low confidence

Figure | Activity Diagram

The signup and login procedures are not essential for the controller prototype; however, a well-rounded application should include those functionalities, as they clarify the complete process and align with our specifications.

### 4.2.2. | User Experience

The prototype will highlight the fundamental design principles a commercial app implementation would be based on. And because a diverse audience of varying demographics will use the application, the navigation and menu options must be straightforward and clear of unnecessary functionalities.

A high contrast must be used for our visual design, and it needs to follow a certain logic. For instance, the guitar strings will be colour coded with bright, high-visible, well-distinguishable colours to help the user play the instrument. The following palette includes some web-compatible colours that may be used for our interface:

Background pattern

Description automatically generated with medium confidence

Figure | String Colour Code Palette

We may apply this colour palette as a standard guideline for the rest of the app to keep the UI consistent. However, using bright colours means the background and the theme should be dark to create a high-contrast environment. As a desktop game, a standard 16:9 or 16:10 screen-size ratio may be used for the base, unlike the popular mobile-first applications. A responsive CSS for tablet and mobile sizes will constitute additional functionalities and is out of our current scope.

### 4.2.3. | App State Management without React

The prototype will not use any framework, so handling the app state must be considered before development. React with Redux would offer ways to pass data between components and maintain the component state efficiently, but it would be overkill for a prototype. On the other hand, using vanilla JS does not mean we cannot manage the state without frameworks. For instance, the state may be stored in a global object. While this method is relatively simple for a single-page app, our multi-page structure requires holding data in cookies or local storage. On the bright side, multiple pages may share the storage, and it may help speed up the development; it is readable and straightforward. It is vital to note that a commercial implementation should be written using a framework because an extensive feature-rich app would be cumbersome to maintain and impossible to scale up using only a simplified global state.

### 4.2.4. | JQuery Style DOM Access and Utility Functions

The application will rely heavily on DOM addressing and manipulation; some developers may prefer libraries for these functionalities, like JQuery. Unfortunately, our scenario would not justify JQuery's size (78KB – 85KB), as only DOM addressing (select single / select all), append, and attribute manipulation would be used often. For this reason, we may easily create our own utility functions with these functionalities in a non-verbose fashion. We may keep the syntax similar to JQuery, so other developers may easily read our code. The drawback of this method is that it is non-standard, and working with a team requires that every team member is familiar with these functions.

Graphical user interface, text, application, email

Description automatically generated

Figure | Utility JQuery-Like Functions (Appendix/Code/util/misc.js)

## 4.3. | Backend Design

We may choose from several options regarding application architecture, such as WebSocket or GraphQL; however, this application will use RESTful architecture (Representational State Transfer). One of the main advantages of using RESTful is that it is scalable, stateless, easy to use, and built on HTML standards, using traditional (GET, PUT, POST) request verbs.

### 4.3.1. | User Interaction

When the user interacts with the server, for instance, to access and display basic user information, request a guitar tablature or store the scores, the frontend will communicate these intentions with the server side. In exchange, the server will send back the requested information in a JSON-formatted response. We can use JSON Web Tokens (JWT) for authentication, a modern, secure standard where each request header receives a JWT token.

Table

Description automatically generated

Figure | RESTful Login Request with JWT Tokens

### 4.3.2. | ER Diagram

Our RESTful application needs to communicate with a server, and various options are available, such as Flask, Django or NodeJS. NodeJS is one of our best candidates, as it is built on the Chrome V8 JavaScript engine (ES6 support) and is a web-native, easy-to-adapt server environment. A commercial implementation would most likely be made around a MERN (Mongo-Express-React-NodeJS) stack; therefore, using MongoDB for the prototype is a logical and appropriate design decision. The following entity relationship may be applied to the prototype's database:

A picture containing engineering drawing

Description automatically generated

Figure | Generic Entity Relationship Diagram (Appendix/ER.drawio)

Because of the separation of concerns, the User table is used for authentication, and the Profile table stores every other information about the client. As deleting a user may cause referencing problems (for instance, empty user reference in a score chart or creator of tablature), an active status property must be stored on the User table. A separate profile makes it possible to delete personal information, as only the User table references it. The user can choose an avatar for their profile, stored as a string. The contact information is included in the profile details for simplicity, which would otherwise receive a separate Address or Contact table and appropriate conjunction tables.

The tablatures will be stored in the Tab table with basic tab information and the tab content and will be referenced by several other tables, such as Like, Comment or Play. When a user plays a tab to the end, the score will be stored in the Play table with the points achieved and other statistics, such as timing and accuracy. Lastly, users may collect achievements. The condition or criteria of the achievement is stored along with the value and basic description. Because of the many-to-many relation, a UserAchievement conjunction table may be created to normalise this relation with reference to UserID and AchievemntID as foreign keys. Alternatively, because we use a no-SQL database, there are options for non-normalised solutions, such as using a list of object IDs or hybrid snapshot storage. A complete commercial solution would need many additional tables, such as Message, Follow, Instrument, Contact, Skill, and Comment.

# 5. Development

## 5.1. | Hardware Development

The hardware design in previous chapters fits hand-crafted manufacturing processes with limited tools and materials. Despite the abundance of third-party services, DIY seems the most feasible option, considering the time frame, cost and complexity. For instance, 3D printing requires detailed modelling and in-time delivery of complex multi-part components is not guaranteed. Similar to PCB printing, which also involves circuit design that may be faulty and needs to be reordered.

However, even hand manufacturing requires project management tools beyond Post-It notes, and many software options are available to work with KanBan. Some of the most recommended ones are Monday, a paid application, Trello, with some limited features or even Google Draft. In this scenario, Draft and Excel can perfectly cover the functionalities this particular solo project requires; they are free and straightforward tools.

Graphical user interface, application

Description automatically generated

Figure | Draft and Logbook (Appendix/Logbook.drawio)

### 5.1.1. | Physical Materials

Several materials, such as polystyrene or acrylic sheets, carbon fibre, wood or even metals like aluminium, are suitable for building our prototype:

A picture containing text

Description automatically generated

Figure | Materials and Properties Rating (Appendix/Material\_Properties.drawio)

We will use polystyrene sheets for the prototype as we will need two different thickness sizes for the sheets (2mm and 3mm). They are available in most hardware shops and are the cheapest options (£30 - £50 depending on size). In addition, our material must be bendable, and polystyrene bends on heat well; it is relatively light, can be cut without specialised tools, and is an excellent electrical insulator.

However, a commercial product should be built with carbon fibre, which is more durable for heavy use. It doesn't snap and break easily when cutting and can be used for manufacturing. Most importantly, it is more eco-friendly and would be a more environmentally conscious choice for mass production.

### 5.1.2. | Console Prototype Development – Images, Steps and Challenges

The prototype development started with the fretboard because it is the heart of the device and the most plausible point for failure. As expected, the 240 components (switches and diodes), the extensive wiring and the more than 1000 soldering points were complicated to cram in a 50mm by 500mm space and required more than 60 hours.

|  |  |  |
| --- | --- | --- |
| A picture containing guitar  Description automatically generated | A picture containing text, indoor, electronics, display  Description automatically generated | A picture containing text, indoor, black, music  Description automatically generated |

Figure | Developing the Guitar Neck

Next, the console body covers were cut, and a template body was created to bend the body side walls to the appropriate curvature after several unsuccessful attempts of free-handed bending resulting in broken pieces of already cut and sanded components.

|  |  |  |
| --- | --- | --- |
|  | A picture containing outdoor, ground, bench, wooden  Description automatically generated | A picture containing indoor, bag, clothes  Description automatically generated |

Figure | Guitar Side and Covers

The side walls material (2mm polystyrene) required several rounds of heating on every inch before being clamped to the template to force it into shape.

|  |  |  |
| --- | --- | --- |
| A picture containing text  Description automatically generated | A picture containing indoor  Description automatically generated | A picture containing wall, indoor  Description automatically generated |

Figure | Guitar Body

Body supporting components and the side and back cover of the neck were attached to the device, and the nuts were pressured and pasted to the drilling holes.

|  |  |  |
| --- | --- | --- |
| A picture containing indoor  Description automatically generated | Graphical user interface, website  Description automatically generated | A picture containing metalware, catch  Description automatically generated |

Figure | Supporting Components

The neck attachment was the most dangerous part of the assembly. If it had failed to hold together in a precise manner, the whole project might have been jeopardised, as Gorilla glue and CT1 cannot be broken apart without severe damage. After a coat of matt black paint, the microcontroller, wiring, USB socket, LEDs and toggle also found their final place.

|  |  |  |
| --- | --- | --- |
|  | A picture containing guitar  Description automatically generated | A picture containing engine  Description automatically generated |

Figure | Electric Housing

Although the strum unit was reasonably straightforward to assemble, the screws, levers, and string tension adjustment took considerable time to work together correctly.

|  |  |  |
| --- | --- | --- |
| A picture containing outdoor  Description automatically generated | A picture containing indoor  Description automatically generated | A close-up of a machine  Description automatically generated with low confidence |

Figure | Strum Unit

Finally, we could attach the headstock and string up the device. As the strings were slipping aside on the plain switch buttons; therefore, each received a custom cap, individually hand grooved, resized, and milled.

|  |  |  |
| --- | --- | --- |
|  | A picture containing wall, indoor, metal, appliance  Description automatically generated | A picture containing indoor, guitar, black  Description automatically generated |

Figure | Head Stock and Strings

The final controller device development time exceeded even the worst pessimistic estimations (over 200 hours); hence, the early start of the assembly paid off its dividends.



Figure | Final Product Prototype

The complete photo gallery with more than 180 development images is available on the link in the Appendices Section.

### 5.1.3. | The Microcontroller

### 5.1.3.1 | Microcontroller Selection

As already discussed in the Literature Review, there are limited options of microcontrollers to choose from because we must have a minimum of 26 pins for the fretboard, six for the strum unit, and four for the toggle and LEDs, leaving us with Arduino Leonardo, Due and Mega.

Table

Description automatically generated

Figure | Arduino Comparision (Appendix/Arduino\_Selection.drawio)

Even though Due is not the most economical option, it has more than the required amount of pins (we may expect pin failures), a powerful processor, and, most importantly, it can operate as a USB host. That means that one of its USB ports can be used as a traditional keyboard device.

### 5.1.3.2 | Microcontroller Programming

Our ultimate goal is to read the state of the device safely, and when state changes happen, it needs to be communicated through USB as a sequence of keyboard presses. Hence, we must differentiate between previous and current states for the frets and strums. For example, we may initialise two arrays for the strums, each with a length of six, and two 2D matrices for the frets with a size of 6/20. As we used pull-up resistors, the initial values will be set to one: one means inactive, and zero means active.

Calendar

Description automatically generated

Figure | Controller State Initialisation (Appendix/Code/Hardware/controller.ino)

We activate all the column pins with each microcontroller loop and read the individual row values, updating previous and current readings for each array element.

Text

Description automatically generated

Figure | State Update (Appendix/Code/Hardware/controller.ino)

Suppose state changes are detected (new and old values in a position are different). In that case, a message is sent by emulating a predefined set of keypresses signifying the event and the location of the state change.

Text, letter

Description automatically generated

Figure | Messaging (Appendix/Code/Hardware/controller.ino)

#### 5.1.3.3. | Debounce Handling

However, as discussed in previous sections, momentary tactile switches are not guaranteed to be immune to physical bouncing and may trigger unsolicited actions from the controller. However, our device has two types of inputs, behaving symmetrically opposing.

**The Fret Button**

The more traditional perspective on inputs is that they trigger an action when pressed, which is valid for our fret switches. When a fret press happens, we may assume that the state change was intentional after a particular time, typically around 50 to 100ms.

**The Strum Switch**

On the other hand, strum switches should be considered active when released, as they activate a sound after the strings have been put under pressure and let go. We may expect more noise from our solution, as our switches use physical strings attached to levers with springs that may trigger involuntary on-off changes. Therefore we cannot apply the same method to strums and frets. To solve this problem, we may use a state machine to determine the current state of the activated string. After the first release, we can measure the time of string state changes and only act if the last change happens after the debounce time.

Diagram

Description automatically generated with low confidence

Figure | Debouncing

The improved version of our debounce handling looks the following:

Text, letter

Description automatically generated

Figure | Fret Debounce Handling

### 5.1.3.4. | Communication Protocol

As the message between the console device and host computer are sent as keypresses, we must agree on a suitable and efficient protocol for our scenario.

* MIDI/OSI: the commercial version of the guitar console would most likely use MIDI, as it is an industry standard. However, we agreed on a simplified communication for the prototype, and midi would be difficult and time-consuming to implement.
* Binary: Binary communication is one of the most efficient ways of communicating with the computer, but translating state changes to binary may end up with a longer message than other solutions.
* Textual: Text-based communication provides an easy-to-implement solution for the problem, as we can describe a specific state with relatively few characters. As each individual keypress may take some time to travel through the DOMs event propagation hierarchy, the shorter the message, the more efficient the protocol becomes.
* Numerical: As the numerical solution is essentially a specified textual solution, it has the same strengths as sending text keypresses, only with a restricted set of characters for clarity.

The numerical solution will be used because development time, simplicity and efficiency are critical factors. Each message will consist of four digits: digit #1 is the event (pressed or released), digits #2 and #3 signify the fret position from 0 to 20, and digit #4 will be the string number. The following example shows a simple button press and strumming.

A picture containing qr code

Description automatically generated

Figure | USB Serial Communication Protocol (Appendix/Protocol.drawio)

These messages may be translated on the frontend by adding keypress event listeners and triggering functions, such as display notes or play audio. As the USB connection is set to be visible to the computer as a traditional keyboard, we can create a console event listener that monitors the state of our application and calls handling functions on each state change.



Figure | Deciphering Messages (Appendix/Frontend/console.js)

## 5.2. | Backend Development

NodeJS Runtime Environment is an appropriate choice for our application because it is asynchronous (non-blocking). Therefore, NodeJS applications are ideal for I/O intensive programs, such as requesting user or tablature information from a database or network-intensive processes. The application requests will flow through the Express pipeline with the following sequence: admin, error and logging, database connectivity, schema modelling, and server response.

Graphical user interface, diagram

Description automatically generated

Figure | Node Express Request Pipeline (Appendix/Pipeline.drawio)

### 5.2.1. | Database Models and Schemas

Although we can use NodeJs in SQL-style referencing, one of the benefits of NodeJs is the ability to break away from strict referential table structures; instead, we use embedding and object references. Embedding is when the parent table stores information about a referenced table, providing a snapshot of related data, a high-performance method that may sometimes be memory intensive. Object references require less memory but may necessitate multiple requests to the database. For instance, we do not need to create a separate conjunction table for user achievements, but we can store them in a list of object IDs and query them individually on a need basis.

Graphical user interface, application, Teams

Description automatically generated

Figure | Profile Model - Achievements (Appendix/Code/Backend/Models/profile.js)

### 5.2.2. | API Endpoints

When the client application calls a server request, the result will be returned in a JSON format with the properties: success (boolean), message (on an error), and resources. The following API routes are not the complete solution, but they are the most necessary to create a working prototype application:

Timeline

Description automatically generated

Figure | API Route Endpoints (Appendix/API\_Endpoints.drawio)

Some API endpoints must redirect our users, such as when registration is complete or when the email link is activated to confirm a valid user. This type of two-phase registration is useful against DOS and resource consumption attacks because an attacker cannot create a flood of registered users, as the email address must be unique and valid to be able to be delivered to confirm registration.

Text, letter

Description automatically generated

Figure | Send Email with Node Mailer (Appendix/Code/Backend/Routes/Subscribe.js)

Our simple confirmation email:

Graphical user interface, text, application, email

Description automatically generated

Figure | Email Confirmation

#### 5.2.3. | Authentication, Authorisation, and Security

JSON Web Tokens (JWT) are appended in every API call's header to allow secure communication between a client end server. JWT stores information about the data payload, such as the user ID or admin privileges and expiry, the encryption algorithm used and provides a unique signature as well. As the JWT private key is stored in the server's environmental variable, it cannot be decrypted by a third party (unless accidentally pushed in a Git repository with an uncareful commit). Therefore, our JWT Token will be passed along with every request to authenticate and authorise every API endpoint.

When users register on RiffMaster, they must send their authentication information to our server, including the password. There are two directly opposing opinions on approaching this problem: hashing on the front or backend side of the application. The first approach asserts that no password information should be sent to a server in plain-text format, and passwords should be hashed before submitting a request.

Graphical user interface

Description automatically generated

Figure | Password Hashing and Encryption

This approach dramatically reduces the chances of a Man-in-the-Middle attack intercepting our password on a transfer. However, it is vulnerable to a Pass-the-Hash attack, as the literal hashed password stored on the database has already been hashed for any malevolent activities. They can also be used for Replay attacks, where the complete HTTP request is recorded and replayed later to gain access to unauthorised resources. Regarding JWT, the accepted method is to sign and verify signatures on the backend of our application, as the private key is secure, and tampered tokens are automatically refused.

Chart

Description automatically generated with low confidence

Figure | Bycrypt Salting and Encryption (Appendix/Code/Backend/Routes/user.js)

## 5.3. | Frontend Development

### 5.3.1. | Material Collection

#### 5.3.1.1. | Audio Cutting and Songs (Audacity)

#### 5.3.1.2. | Chord List

### 5.3.2. | Theme and Digital Design (Adobe Photoshop, CSS Variables instead of Bootstrap, Font)

### 5.3.3. | Registration and Login

### 5.3.4. | Landing Page

### 5.3.5. | Manipulating Audio: Jam Session

### 5.3.6. | Chord Translation and Display

### 5.3.7. | Tablature Parsing

Reusable functionalities for parsing, displaying and manipulating multidimensional tablatures. Three Different representations for the same structures, guitar neck, tablature editing studio, and play.

Event delegation for editing tablatures.

Text

Description automatically generated

Figure | Different Translation After String Parsing

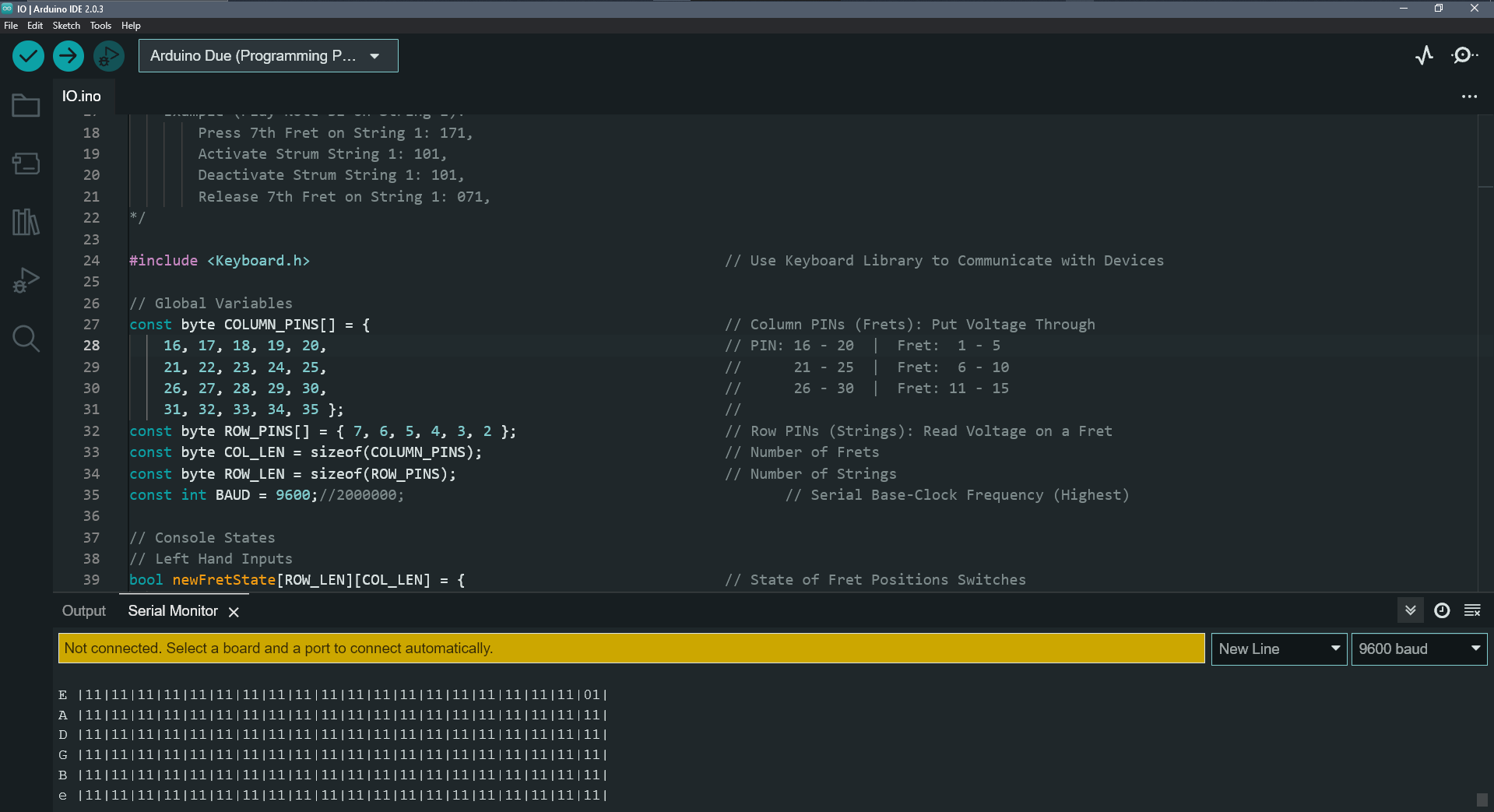
## 

## 5.3.8. | Game Play and Practice Options

Games that emulate guitars must have a graphical representation of the music being played, with colour-coded notes representing a fret or string action. This user interface component is often called note highway or track.

# 6. | Testing

## 6.1. | Hardware Testing



### 6.1.1. | Interaction Testing

### 6.1.2. | Performance Testing

## 6.2. | Software Testing

### 6.2.1. | Automated Testing

Justify automated testing:

Test code more frequently in less time,

Catch bugs before deploying a release,

More confidence in source code,

Secure refactoring

Focus more on quality.

### 6.2.2. | API End-to-End Testing

A screenshot of a computer

Description automatically generated with medium confidence

Figure | Postman

### 6.2.3. | Function Testing

### 6.2.4. | Unit Testing

Test functions without external dependencies, such as database or web services. Cheap to write, executes fast, but don't give a lot of confidence in the reliability of the function of the seperation of dependencies.

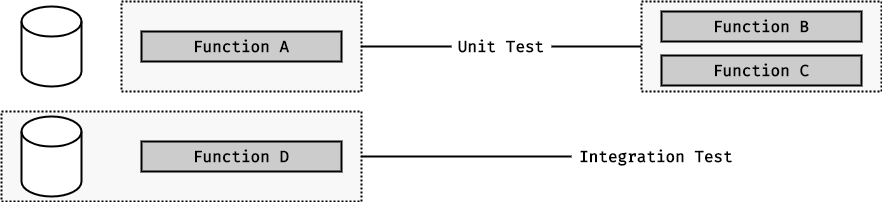


Figure | Unit versus Integration Testing (Appendix/Unit.drawio)

### 6.2.5. | Integration Testing

Misconseption: some developers think that testing two or more funcitons together describes unit testing. In reality integration testing tests the class or component with external dependencies. Take longer to execute because of the I/O on files or databases, but gives confidence on the general health of the application.

Text

Description automatically generated

### 6.2.6. | Input Testing

### 6.2.7. | Requirement Test Sheet

## 6.3 | User Acceptance

# 7. | Reflection

Match user and system requirements table with reality (suitable for an Excel Table), draw conclusions, missing features, challenges, and time to develop them.

## 7.1. | Manufacturing

## 7.2. | Project Management (Work Monitor Analysis) and Version Controlling

Chart, bar chart

Description automatically generated

Figure | Labour Statistics (Appendix/Labour Statistics.ipynb)

**Refresh Labour Statistics**

## 7.3. | Challenges

## 7.4. | Critiques: Overall User Experience

## 7.5. | Recommendations for an Improved Console Prototype

This project will be continued after the submission as a pet project. Explain why and what development will be done. Explain patenting issues.

## 7.6. | Recommendations for an Improved Music App

Use of Artificial Intelligence, Frameworks, Graph DB for Users, Hardware Specs, App Features: Social, Tutorials, Responsive Design.

## 7.7. | Recommendations for Commercialisation

## 7.7.1. | Commercial Development

7.7.2. | Patenting

7.7.3. | Monetizing Strategies

## 7.7. | Personal Development

Justify the documentation length as it embraces a broad spectrum of computing disciplines, such as hardware development, manufacturing labour, frontend, backend and microcontroller programming.

# References

Arduino.cc, 2..A. (2022) *Arduino*. Available from: <https://docs.arduino.cc/built-in-examples/digital/Debounce,> [Accessed 12 October 2022].

Chrzanowski. (2015) *Music Video Game with User Directed Sound Generation*. Patent No. 9061205B2. US:

Csíkszentmihályi, M. (2009) *Flow: The psychology of optimal experience. New York: Harper Perennial Modern*.

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

Dribin, D.. (2000) *Keyboard matrix help*.

Edgard, V. (n.d.) *https://erickuehnl.com/1994/12/30/a-brief-history-of-computer-music/* [Accessed 02 October 2022].

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

Esparrago Jr, R.A. (1988) Kanban. *Production and Inventory Management Journal*. p.6.

Fitzgerald, S..S.M. (2012) *The Arduino Project Book*. Torino: Arduino LLC.

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

Ibrahim Rihan, M. (2022) *The Feasibility Study* Available from: <https://www.academia.edu/29462582/The_Feasibility_Study> [Accessed 14 October 2022].

Janzen, D.a.S.H. (2005) Test-driven development concepts, taxonomy, and future direction. 38. Computer. pp.43-50.

Kader, W.M..R.H..M.M.a.B.M.A.S.. (2012) *Advancement of CMOS Schmitt trigger circuits.*. 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.

Murphy, R. (2017) *USB HID Basics with PSoC 3 and PSoC 5LP*. Cypress.

Schmidt-Jones, C. (2016) *Common Notation for Guitar Tabluature Readers*.

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

Sulaiman, S..S.M.. (2008) Web caching and prefetching: What, why, and how? *Proceedings - International Symposium on Information Technology*

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

Warren, G. (2015) *Exploring the Raspberry Pi 2 with C++.* Springer.

# Bibliography

**Edstrom**, B., 2016., Arduino for Musicians, Oxford University Press.

# Appendices

**Logbook, Gantt, Ethical Form and Risk Assessment**

****

**Diagrams**



**Code**



# 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> |
|  | Gallery | https://drive.google.com/drive/folders/17AnjHB91FEG5995bXSGXAZzvXx8nxK1b?usp=sharing |