GUITAR TUITION

6COSC006W – Final Year Project Report

This report is submitter in partial fulfilment of the requirements for the BSC (HONS) COMPUTER SCIENCE degree at the University of Westminster.

STUDENT: BEATRICE ANTONIU
SUPERVISOR: JEFFREY FERGUSON
DEGREE: BSC (HONS) IN COMPUTER SCIENCE
School of Computer Science & Engineering
College of Design, Creative and Digital Industries
University of Westminster
DATE: 27/04/2021

Declaration

This report has been prepared based on my own work. Where other published and unpublished source materials have been used, these have been acknowledged in references.

Word Count: 4854

Student Name: Beatrice Antoniu

Date of Submission: 27.04.2021

List of figures

Table of Contents

1. Introduction

Problem Statement

The goal of the project is to teach people of all ages how to start their guitar playing journey. As a self-taught guitar player, I had a hard time trying to get started and motivated as there were not many software, applications or even videos of how to play this instrument and other instruments in general. Although now there are quite a few options to learn how to play guitar, I personally do not find them extremely efficient: they either require a subscription or they do not have the right amount of information. What I intend to do with this project is trying to create something that is easy to understand by anybody and gives you at least a good base that you will be able to use to evolve further. The software/application will offer the user a tuition-like experience through 3D objects, animations, and other elements, educating them at the same time.

Aim and Objectives

The aim is to create a thought through project that is well-organised, designed and that fits well the proposed idea. Not only creating a project that will benefit others but that will help expanding my current knowledge and learning new skills, such as new programming languages or new software, during the developing time. The final result should be a user-friendly and easy to use application for every age group, making use of the visual world of 3D and animations.

Objectives:

- Planning, deciding, and designing the project withing the timeline. Dividing the plan into different categories for easier time management and identifying problems.
- Researching features that guitar players found useful when learning the instrument.
- Learning C# and motion capture. I have some basic knowledge of C# that needs to be sharpened and afterwards motion capture in Motion Builder for animations and other effects.

- To test different software for the final product as the 3D Objects will be tested in Unity, 3Ds Max and Blender.
- Stick to the deadlines and the Project Planner.
- Obtain supervisor feedback on current work and use it to make changes that will benefit the final product.
- Testing different prototypes through questionnaires, asking people with or without guitar experience for their feedback.

2. Background

Review of projects/applications

I have been testing a range of software/applications or other methods of learning guitar throughout the years and as I have mentioned, they seem to lack a lot of features that to me are vital. I have chosen a few apps that I had a proper look at and that are more similar to my project.

There are two apps that I have come across that are similar to my project. Both using 3D elements to teach guitar. **Guitar 3D** and **Basic Guitar Chords 3D** created by **Polygonium**, are two apps for both Android and iOS mobiles and macOS. **Guitar 3D** is a bit more evolved than the later as it teaches rhythm and notes while **Basic Guitar Chords 3D** only teaches guitar chords shapes in different scales and keys. The interfaces of the two are almost identical, the second one focuses more on the fingers and hand positions so you can play with the camera position around that area. **Guitar 3D** has more camera positions, you can see the character playing the guitar from multiple views and choose which hand you want to focus on, you can also choose your chords for the character to play at a chosen tempo. A flaw with **Guitar 3D** is that certain features require a subscription to unlock while **Basic Guitar Chords 3D** does not.

Figure 1 Guitar 3D

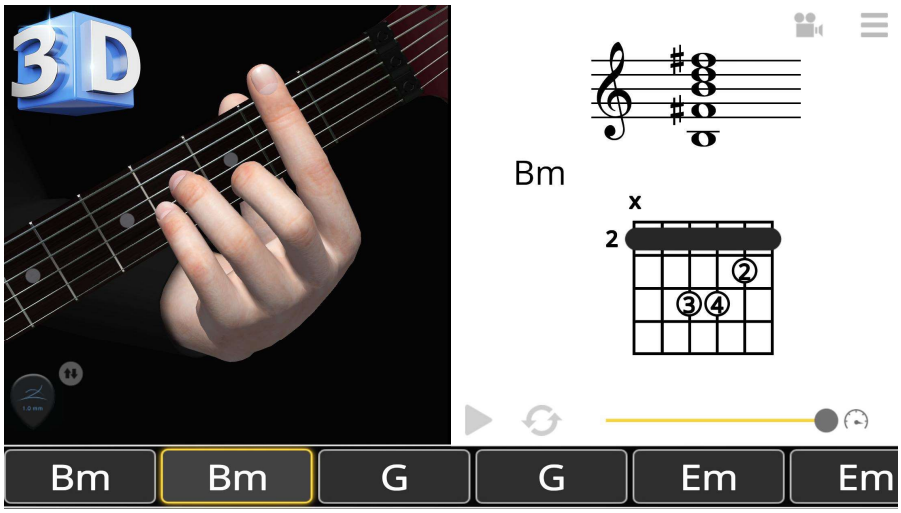


Figure 2 Basic Guitar Chords 3D

Yousician is a Finnish interactive music service to learn and play a musical instrument. The application is available on iOS, Android, Windows and macOS platforms, but I have only tested it on iOS so far. It has a nice design, simple with suggestive tabs and categories. It has 3 ways of learning which are **learning through songs** (for people that already know how to play), **learning through theory and workouts** (for people that are just starting or want to learn more) and **learning through challenges** (again, for people that know how to play and want to test their skills). The app itself has

reminders for lessons, levels of difficulty for songs and lessons and a variety of songs that you can play or learn along with explanatory videos for certain techniques. I would say it has everything you need to get started or continue your training with the exception that you need to pay a subscription in order to use all of these features.

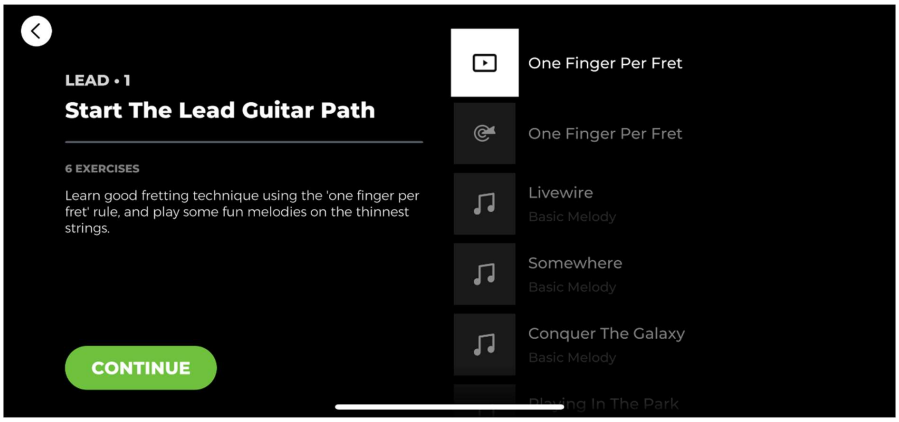


Figure 3 Yousician

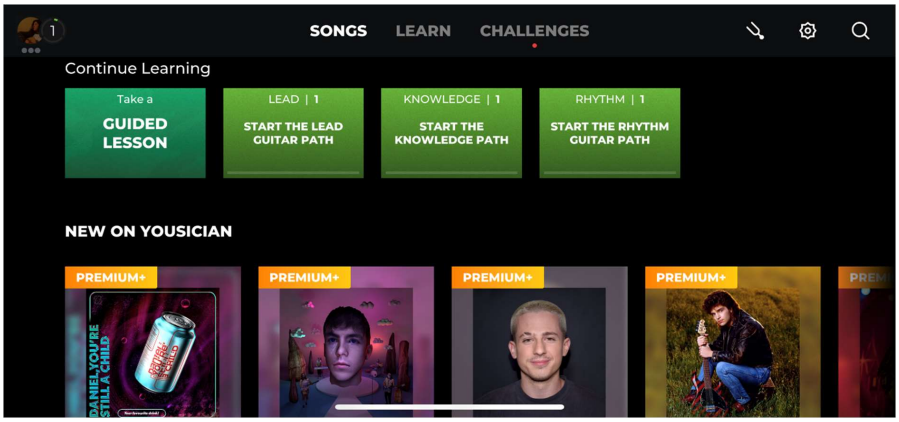


Figure 4 Yousician

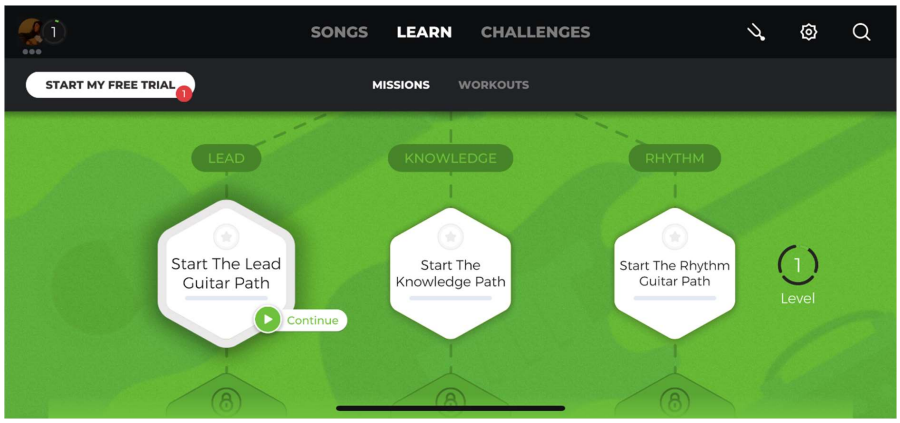


Figure 5 Yousician

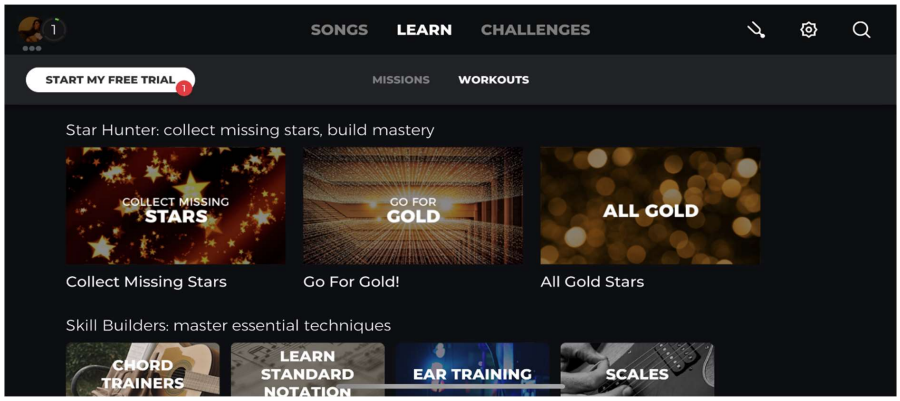


Figure 6 Yousician

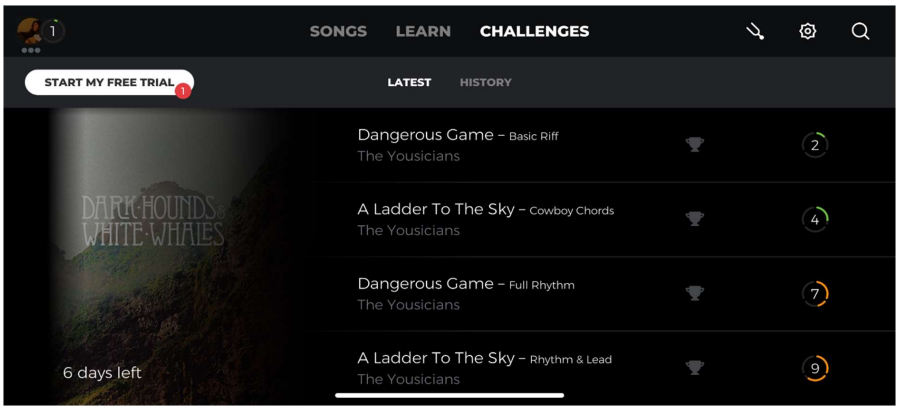


Figure 7 Yousician

SimplyGuitar is a mobile app that teaches you guitar in a similar way to **Yousician**. Also, with a simple and straightforward design, **SimplyGuitar** offers you fewer lessons than the other apps and less features. Main problem with it is that if **Yousician** lets you play even if for a few minutes a day without a subscription, this one will not. You play the first introductory lessons and then you are required to pay.

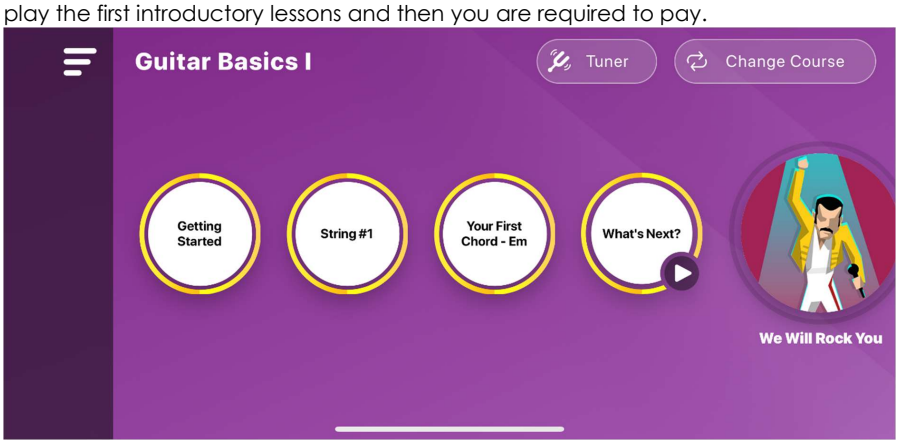


Figure 8 SimplyGuitar

Ultimate Guitar is the largest guitarist community website including guitar and bass guitar tablature, chord sheets, reviews of music and equipment, interviews with notable musicians, online written and video lessons, and forums. (Wikipedia, 2021)

I have been using it ever since I started playing guitar and it has evolved since 2014 a lot. It used to be based around guitar only: chords, tablatures, and forums. It has a Phone version as well, but you need to buy the app, otherwise the web version is mostly free. It is quite different than the other similar applications I have mentioned as its purpose was just to offer music sheets and simply teach people guitar chords. It

is not the best way of learning, but it is perfect for people that already have some knowledge and that want to contribute to help others.

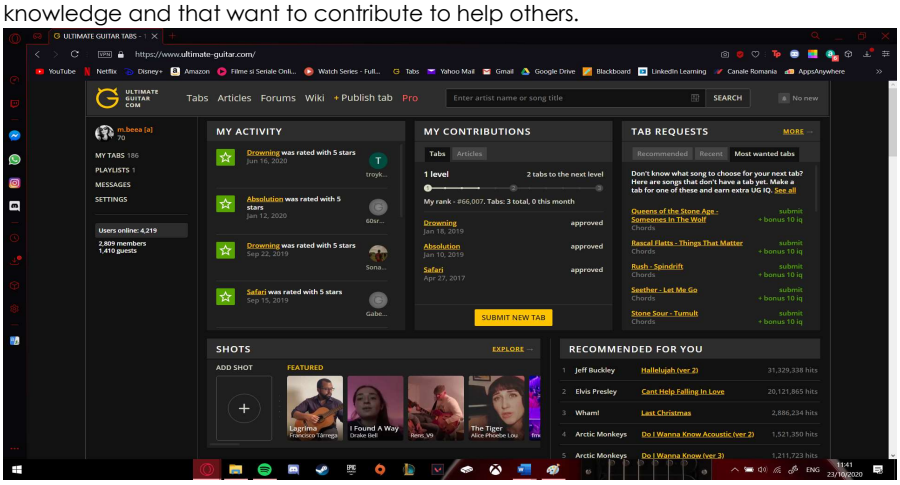


Figure 9 Ultimate Guitar

These are just a few examples. There are other apps, blogs, YouTube channels that teach guitar, but my idea is to have a little bit of everything in order to create a complex yet simple application. I have created a table that makes a simple comparison between these examples and my project idea to summarise what I have been explaining above.

	Their Main Features	My Main Features
Guitar 3D	Guitar Simulator Training Mode Strumming and Rhythm Learning	Strumming Patterns Learning Chord Shapes Learning Songs Learning Basic Knowledge, such as parts of the guitar, tuning the guitar, fingers positioning. Reading tabulatures Review Mode Progress Tracking Different Levels of Learning
Basic Guitar Chords 3D	Chord Shapes Learning Review Mode	
Yousician	Play Along Progress Tracking Practice Mode	
SimplyGuitar	Play Along Progress Tracking Practice Mode	
Ultimate Guitar	Strumming Patterns Learning Chord Shapes Learning Video Explanations Community Feedback	

For me, the obvious choice was **Unity** since it is a cross-platform game developing engine that I have used before, but there are other tools that would have helped developing this project in the same way. According to quite a few blog posts, there are plenty free tools similar to Unity.

	Installation & Ownership	2D/3D	Ease of Use	Integration & Compatibility	VR Support	Customer Support
Unreal Engine	***	Both	***	******	Yes	***
Amazon Lumberyard	***	3D Only	*****	***	Yes	***
CryENGINE	***	Both	***	******	Yes	***
Unity	***	Both	***	*****	Yes	***
GameMaker: Studio	******	2D Only	*****	***	No	***
Godot	*****	BothBoth	*****	***	No	*****
Cocos2d	*****	2D Only	*****	***	No	***

Figure 10 Top Game Engines 2021 Comparison Table

Commented [BA1]: https://www.incredibuild.com/blog/top-7-gaming-engines-you-should-consider-for-2020

My project requires both 2D and 3D elements so not all of them fit the criteria. **Unreal Engine**, **CryEngine** and **Godot** would be great replacements for my scenario.

Godot is free to use and open source so you can build 2D and 3D games and sell them however you want. It has a unique approach compared to other engines and it is much easier to use. It does not have as many resources available as Unity or Unreal and it has its own programming language, **GDScript**, which resembles **Python**, which can be a red flag for most users that do not want to take on yet another language.

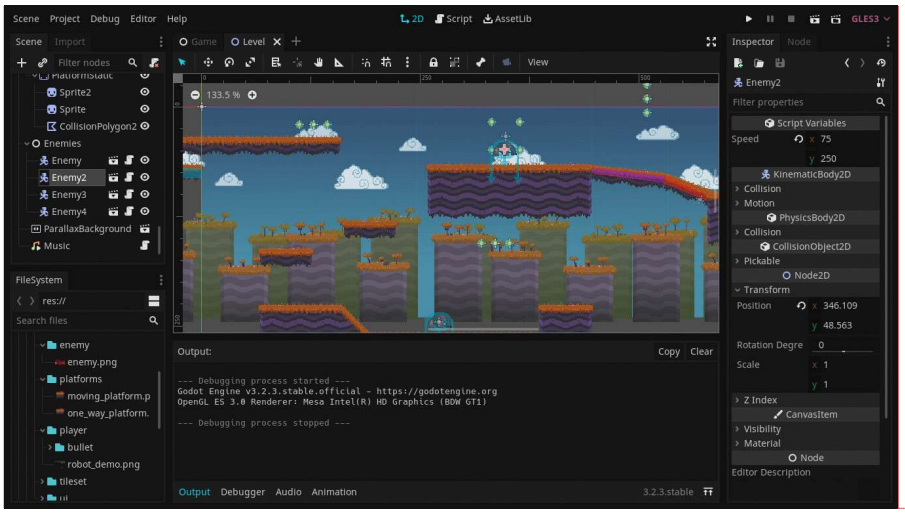


Figure 11 Godot 2D Workspace UI

Unreal Engine is one of the most popular and widely used game engines. It is multi-platform engine designed to handle more complicated tasks more efficiently. Like **Godot**, it is also open source, and it has a marketplace as Unity does where you can take free assets from. It is suited for larger projects and so requires more powerful computers in order to work because of the high-end graphics.

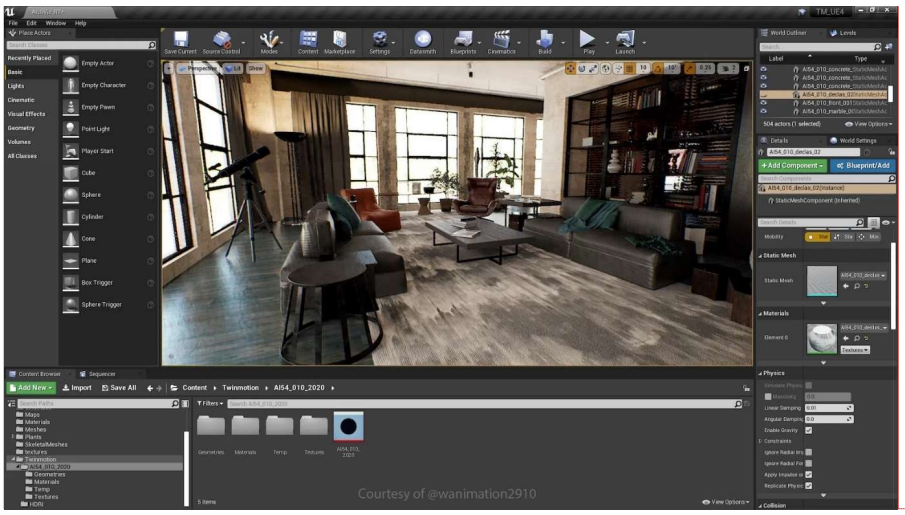


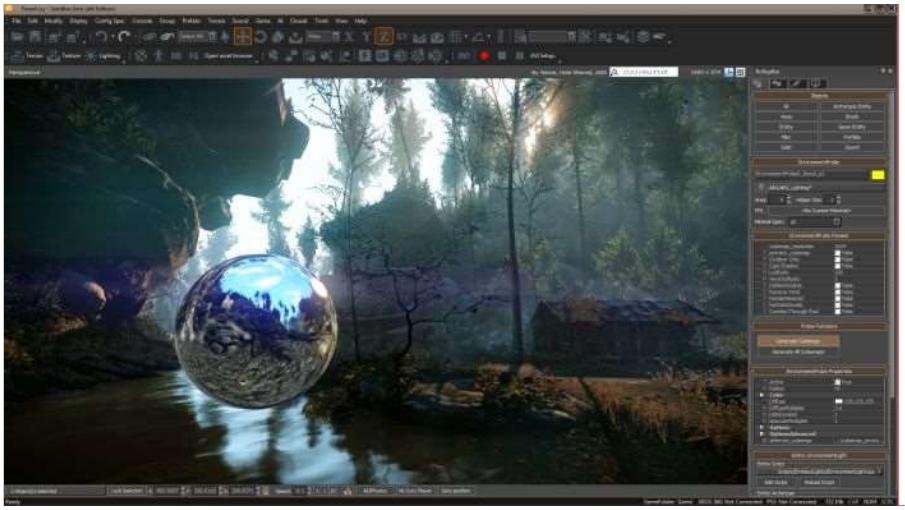
Figure 12 Twinmotion Importer for Unreal Engine

CryENGINE is a free to use platform recognised for its photorealistic games. It has great visual capabilities such as open world environments, it is easy to iterate and

Commented [BA2]: https://en.wikipedia.org/wiki/Godot_(game_engine)#/media/File:UI_of_Godot_Game_Engine.jpg

Commented [BA3]: https://www.unrealengine.com/marketplace/en-US/product/21cab2d84a794b7bab359eb1ba5d3e74

learn and it does not require a lot of programming. You can access both the engine and the editor's source code. It is much smaller than **Unity** and **Unreal**, so it does not have that much support. It is also not that easy to use straight away and it is not recommended for fast-paced games or complex RPGs.



Commented [BA4]: https://www.geeks3d.com/20110818/cryengine-3-free-sdk/

Figure 13 CryENGINE 3 Sandbox

3. Requirements

Stakeholders

The administration of the project will be entirely done by me as it has only one purpose and it does not have a login system or anything that needs to be monitored by several people at once. My project is meant for people that want to learn guitar or continue learning it, that have access and can use a computer to a minimal level. It is designed for both experienced and non-experienced users.

Gathering requirements

My process of gathering the requirements for this project consisted in researching the initial list of requirements I have set and going through what I could do and could not. First things first, I have started by creating a prototype in Adobe XD with all the features that I wanted so I can decide on which fit my idea and I can implement as well. After implementing the interface according to the prototype, I have looked up different UI/UX examples and started playing around with the

design. Once the design was done, I have started developing the back end of the project. I have watched different videos and read the documentations for the tools I was using, while creating the scripts. I did not set a schedule of gathering requirements before starting the implementation. I would focus on one feature at a time and try to develop it fully and then go to the next one and come back if anything needed to be readjusted. Blogs and forums have been also useful to some of the issues that I had, comparing my errors with other users that had similar problems and trying to adjust the solution to my code.

Modelling requirements and relevant diagrams

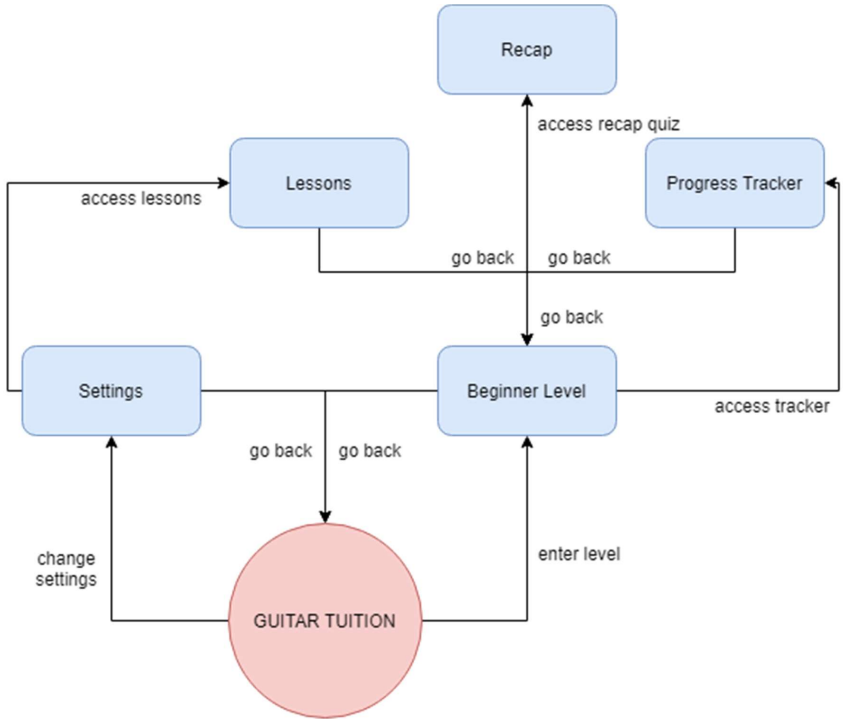


Figure 14 Context Diagram

List of project requirements

Functional:

- *The user will have access to the project through Unity.*
- *The user will be interacting with the project using the mouse.*
- *The user will be able to easily interact with the UI.*

Non-functional:

- *Have arranged meetings with the supervisor once every two weeks. In order to get the necessary feedback and ask the essential questions, this requirement is a must for the project to be on the right track.*
- *Create a schedule to achieve the final goal. It is a good way of keeping everything clean and organised and for myself to manage the amount of time I need to spend working on every detail.*
- *To research the new skills the project requires and develop them further. C# is one of the most valuable skills that the project needs in order to function, although I have previous coding experience with C#, the bar needs to be raised as my knowledge is at a basic level and mostly borrowed from other programming languages.*
- *To research and practice motion capture and modelling techniques in Unity and Motion Builder.*
- *Create different prototypes of the current stage of the project and apply the feedback received from the supervisor.*
- *Keeping up with deadlines and the goals I have set are vital.*

Essential:

- *The project will be based on Windows 10.*
- *The project will support Unity 2020.*
- *The project will include responsive audio.*

Desirable:

- *The user will be able to mute the audio.*
- *The user will be able to change resolution and quality.*
- *The user will be able to change window mode.*
- *The project will have a Test Knowledge feature.*

Luxury:

- *The project will be cross-platform.*
- *The project will work outside of Unity as a standalone application.*
- *The project will have a Progress Tracking feature.*
- *The project will support user interaction during animations play time.*
- *The Main Menu will have a Notifications feature.*

Legal, social, and ethical issues

This project does not gather any personal information or location, so the user's privacy is completely safe. It is completely free of use, does not ask any personal

questions or bank details in order to be used. Any external sounds, models, icons, or other resources that were used in this application, were taken under a free license and if not were referenced or credited for. Unfortunately, it is not an application dedicate for people with visual or hearing impairments as it has not been developed to that level. Does not contain any graphic, physical, or sexual content, nor it discriminates based on age, gender, race, or disability.

4. Methodology

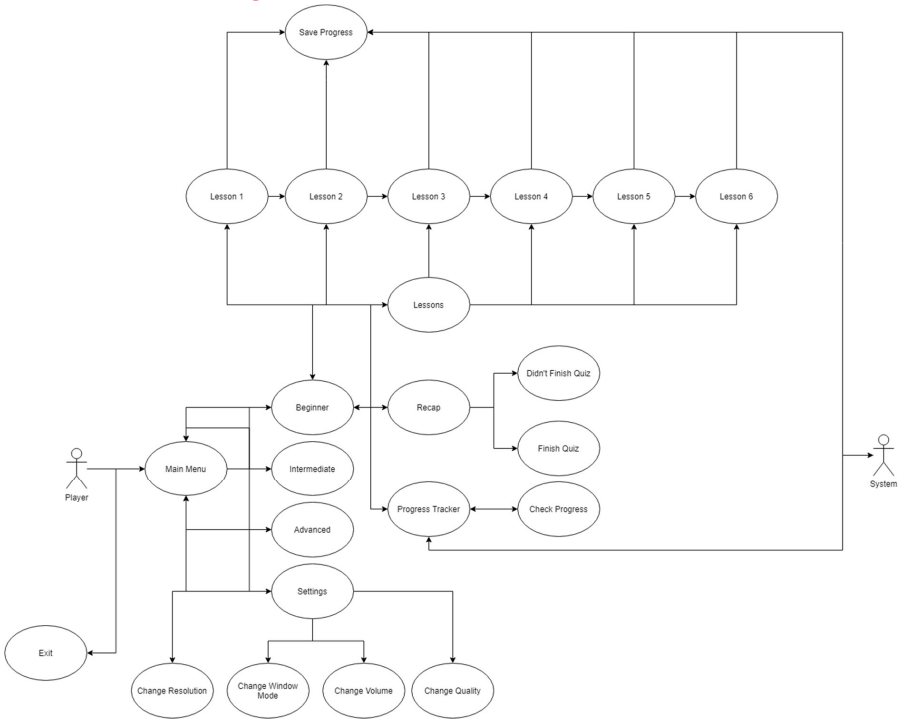


Figure 15 Use Case

5. Design

Initially I have realised a prototype interface with Adobe XD which I have followed mostly in order to start implementing the user interface and start developing the application. I have kept different versions of the project, including the initial prototype version from Adobe XD, in order to decide on a final design that fits the best.

The User Interface was created in **Unity** using the **Canvas** and **UI** elements. Sprites and Icons were made and modified in **Photoshop** and **Illustrator**. At this stage, the interface does not present any issues and you can always change the resolution and quality from the **Settings** menu on the **Main Menu** to fit any screen and frame

rate up to **1920x1080 75Hz**. Changing the User Pre-sets (resolution, quality, audio, and window mode), caused a few problems as they were either not saving properly or changing the resolution in a way the UI could no longer be accessed. I have managed to store the settings through a script and solved all the issues.

The design is fully functional and is not too loaded so it could at the same time remain simple and intuitive. All UI buttons have a click sound effect when you interact with them and a suggestive name or icon to what they are supposed to do. There were a few issues when adding the 3D objects to the scene as they require a camera and I needed to re-adjust the design to fit all the elements. I created a custom cursor from an icon and made a few changes in **Photoshop** to match the design of the project.

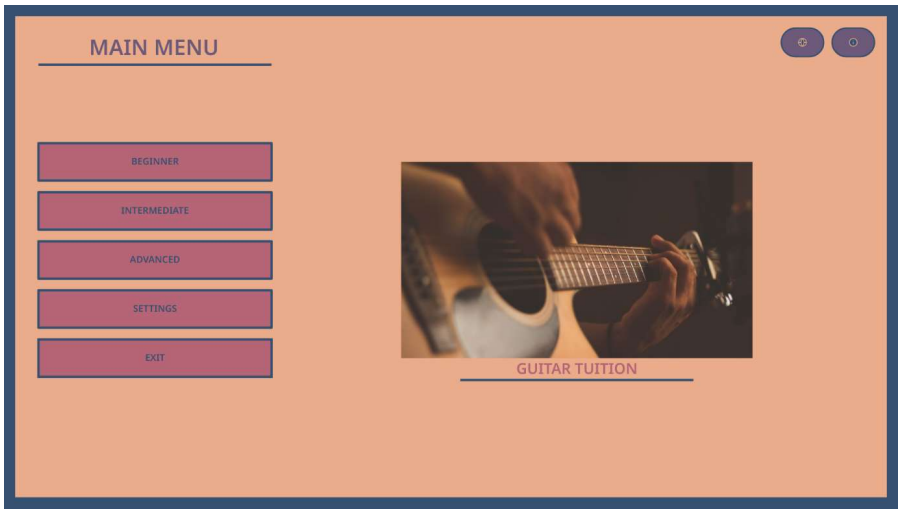


Figure 16 Adobe XD Prototype

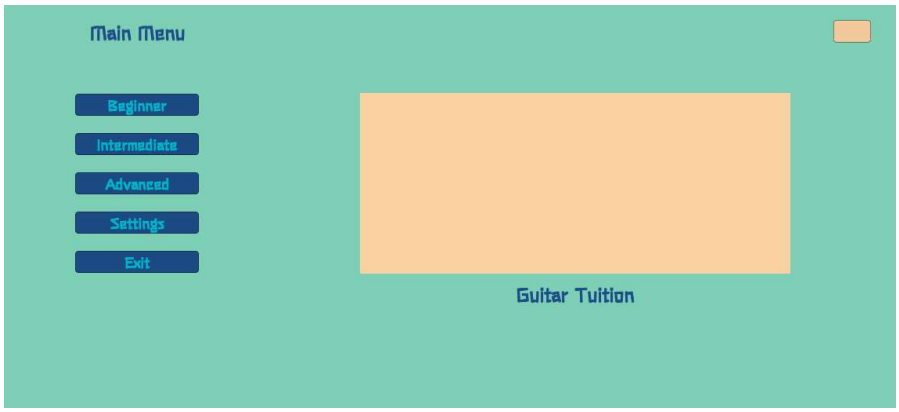


Figure 17 Unity First Implementation

Figure 18 Unity First Build

Figure 19 Unity Final Build

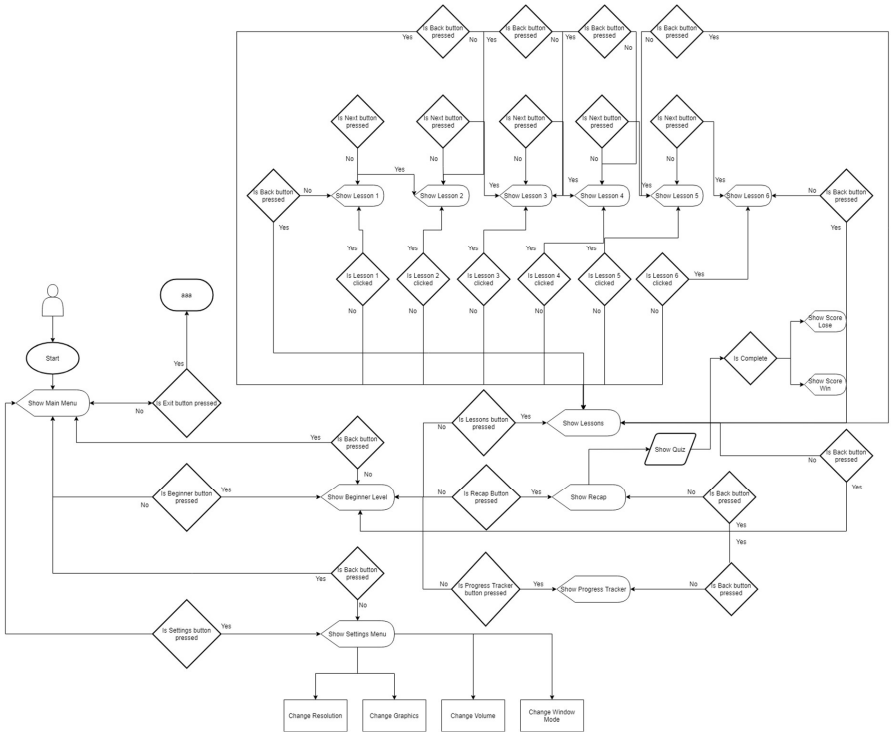


Figure 20 Flowchart

6. Tools and implementation

Tools

Tools

My project was developed in **Unity** using **C#** as the main programming language. Since it is a 3D application/game, I have made use of other tools for 3D modelling and animation. **Autodesk Motion Builder** played the most important role out of all as it was used to create all the animations for the final product. I have used the **animator** and **animator controllers** provided by **Unity** to link the animations created with **Motion Builder** and to also create some other small animations and transitions. With the help of **Blender** and **Autodesk 3Ds Max** I have reshaped, re-applied textures and created rigs for the models that I have used in order to use them for my purpose. To record the guitar sounds for each chord shape and strumming pattern I have used **Garage Band** to record myself playing. I have adjusted the animation according to the sound through storyboards created in **Motion Builder**. I have created sprites for the chord shapes and strumming patterns with the help of **Photoshop** and **Illustrator.**

Skills

I have used **Unity, Blender** and **3Ds Max** before the start of this project but I certainly improved my skills a lot after hours of researching and applying what I have learned to the application. **Motion Builder** I managed to pick up faster than I expected in the beginning as I have never used this tool before and creating animations and storyboards took a whole less time. My **C#** skills have improved as well since I was just starting to learn the language in the beginning of the project. I was not extremely familiar with any of the **Adobe** tools either so working in **Photoshop** and **Illustrator** are new skills I have acquired along the way.

Implementation

As the user starts the project, they will be provided with a **Main Menu** consisting of three levels of difficulties, a settings menu, and the exit button. Moving around the application is realised by switching between the scenes on button click. Once the button is clicked the scene's name is passed into the method and loads the scene with that name. This script is used to advance to the next lesson as well, in the **Lessons** menu in the **Beginner Level**.

```
public void changeScene(string sceneName)
{
    SceneManager.LoadScene(sceneName);
}
```

```
    public void exitScene()
    {
        Application.Quit();
    }
```

Lesson 1 and **Lesson 2** of the Beginner Level share a similar script. On **Lesson 1**, the user will be presented with two buttons: Acoustic Guitar and Electric Guitar, which they can click and will reveal the respective guitar. They will be able to go back and see the other option without returning to the Beginner Level's menu. The user will be able to hover on different parts of each guitar that will reveal the name of the highlighted area.

```
    void OnMouseEnter()
    {
        partSelected.SetActive(true);
        startcolour = GetComponent<Renderer>().material.color;
        GetComponent<Renderer>().material.color = new Color (255/255f,
148/255f, 0/255f);
    }

    void OnMouseExit()
    {
        partSelected.SetActive(false);
        GetComponent<Renderer>().material.color = startcolour;
    }
```

Lesson 2 works in a similar way. The user can interact with the Hand object to get a better understanding of the finger positioning on the fretboard. Hovering over any of the four fingers they need to use to create chord shapes would reveal the number of the finger that is displayed on the tabulature and the position on the fretboard.

This scene also contains a **Text Writer** script that has been adopted from CodeMonkey. (CodeMonkey, 2019)Basically this script displays text character by character inside a box which the user can click to display the text faster and go to the next message. I made a little modification to this script so the text would display in a certain order and once it reaches the end of the array it just repeats the last message, instead of randomizing all the messages in the array as the original script does.

```
    private void Awake() {
        messageText =
transform.Find("message").Find("messageText").GetComponent<Text>();
        clickAudioSource =
transform.Find("click").GetComponent<AudioSource>();

        transform.Find("message").GetComponent<Button_UI>().ClickFunc =
() => {
            if (textWriterSingle != null &&
textWriterSingle.IsActive())
            {
                // Currently active TextWriter
                textWriterSingle.WriteAllAndDestroy();
            }
            else
            {
                string[] messageArray = new string[] {
```

```
                    "Let's start with the name and position of the
fingers on our left hand, shall we?",
                    "We're going to use our fingers to play notes,
scales and chords.",
                    "We'll take the G chord as an example because it
uses all fingers.",
                    "If you hover over each finger you will see their
positions on the fretboard.",
                    "We use our thumb only to hold the neck of the
guitar.",
                    "Let's see what the right hand does.",
                    "With this hand we're going to strum the strings.",
                    "Each finger corresponds to a string or a set of
strings but only when plucking the strings.",
                    "Thumb is plucking the E, A and D strings.",
                    "The index finger is plucking the 3rd string (G).",
                    "The middle finger is plucking the 2nd string
(B).",
                    "The ring finger is plucking the 1st string (e).",
                    "We don't use our pinkie to strum nor pluck the
strings.",
                    "Now that we got this covered, how do we read
guitar tabulatures?",
                    "Tabulatures can become complicated when doing more
complex songs.",
                    "But we're not there yet...",
                    "Let's have a look at the example on the right.",
                    "Hovering over the numbers will give you a proper
explanation of what to pluck or strum.",
                };

                if (index == messageArray.Length)
                {
                    string message = messageArray[index - 1];
                    StartSound();
                    textWriterSingle =
TextWriter.AddWriter_Static(messageText, message, .05f, true, true,
StopSound);
                }
                else
                {
                    string message = messageArray[index];
                    StartSound();
                    textWriterSingle =
TextWriter.AddWriter_Static(messageText, message, .05f, true, true,
StopSound);
                    index++;
                }
            }
        };
    }
```

Lesson 3 is quite similar to the previous lesson. In this scene the user is learning about tuning the guitar using a certain method. While they can also click the box to reveal the walkthrough of the lesson, they can hover over the guitar strings and several elements will appear and some will be highlighted. Hovering over the first string would highlight the corresponding key, the fret that you need to use and the string

itself. It will also display the name of the string they are highlighting and the number of the fret.
Lesson 4 and **Lesson 5** also share a similar script the difference being the animations they hold. **Lesson 4** is meant for the user to learn chord shapes; they can go back and forth through the animations using the buttons provided and they can also replay the animation. Once the animation is triggered a chord chart will also be displayed that shows the position of the fingers along with the chord sound. **Lesson's 5** purpose is learning strumming patterns and behaves in the same manner as the previous lesson.

```
    public void PreviousChord(){

    if (count > 0 && count <= 8)
    {
        count -= 1;
        anim.ResetTrigger(showChords[count + 1]);
        showCharts[count + 1].SetActive(false);
        anim.SetTrigger(showChords[count]);
        showCharts[count].SetActive(true);
        chordsAudio[count].Play();
    }
    else if (count == 0)
    {
        anim.SetTrigger(showChords[count]);
        showCharts[count].SetActive(true);
        chordsAudio[count].Play();
    }
}
```

Lesson 6 teaches the user their first song based on what they have learned from the previous lessons. It also has control buttons that play, pause, and replay the animation and sound. Once the Play button was clicked it will become inactive until one of the Pause or Replay buttons are clicked. On the right of the model will be the chord charts that were used to play the song, being animated at the same time with the main animation. It also has the strumming pattern used in the animation.

```
public void StartSong()
{
    if(count == 0)
    {
        anim.SetTrigger(startSong);
        anim2.SetTrigger(startSong);
        songsAudio.Play();
        count = 1;
    }
    else if(count == 1)
    {
        anim.enabled = true;
        songsAudio.Play();
        count = 0;
    }
    play.enabled = false;
}

public void RepeatSong()
```

```
    {
        anim.Play("Idle", 0, 0f);
        anim.SetTrigger(startSong);
        anim2.SetTrigger(startSong);
        songsAudio.Stop();
        songsAudio.Play();
        play.enabled = false;
    }

    public void PauseSong()
    {
        anim.enabled = false;
        anim2.enabled = false;
        songsAudio.Pause();
        play.enabled = true;

    }
```

The **Recap** scene from the Beginner Level menu allows the user to go through fifteen questions related to what they have learned in this level. The questions are stored inside a Scriptable Object. The quiz is simple, questions (audio, image, text) would be displayed one by one until the user gets them all right or gets three wrong. After either finishing all the questions or losing all the lives, the user will get a screen with their final score. I have used a piece of code to shuffle the list of questions from (swapnilrane24, 2019).

```
    public void SetQuestion(Question question)
    {
        this.question = question;

        switch(question.questionType)
        {
            case QuestionType.TEXT:
                questionImage.transform.gameObject.SetActive(false);
                break;
            case QuestionType.IMAGE:
                ImageHolder();
                questionImage.transform.gameObject.SetActive(true);
                questionImage.sprite = question.questionImage;
                break;
            case QuestionType.AUDIO:
                ImageHolder();
                questionAudio.transform.gameObject.SetActive(true);
                audioLength = question.questionClip.length;
                StartCoroutine(PlayAudio());
                break;
        }

        questionText.text = question.questionInfo;

        List<string> answerList =
ShuffleList.ShuffleListItems<string>(question.options);

        for(int i = 0; i < options.Count; i++)
        {
            options[i].GetComponentInChildren<Text>().text =
answerList[i];
```

```
            options[i].name = answerList[i];
            options[i].image.color = normalColour;
        }

        answered = false;
    }
```

Settings menu has different options for the user to change the resolution, quality, window mode and audio. Changing any of these options requires saving the Player Preferences for the whole project even after terminating the application. Both **Resolution** and **Graphics** are represented by dropdown lists, first one is populated in the script by calling the Resolution array and it will fill the dropdown list up to the resolution 1920x1080 with the frame rate of 75hz.

```
Resolution[] resolutions;
```

For the **Graphics** list I have added the options in the dropdown element with the name for the supported quality settings that Unity provides.

The **Window** is triggered by a toggle that switches between Window Mode and Fullscreen Mode. I look for the last state of this toggle in the Awake() method so I can keep the window mode after quiting the application.

```
screenInt = PlayerPrefs.GetInt("togglestate");
```

The **Volume** is changed using a slider and will change the volume of the entire project. If the user wants to mute or unmute the sound of the application, they need to return to the Main Menu and change it from the settings and will apply instantly through the PlayerPrefs.

```
public void SaveVolume()
{
    PlayerPrefs.SetFloat(audioName, slider.value);
}
```

7. Testing

Functional Testing

Test Case ID	Test Scenario	Test Steps	Test Data	Expected Results	Actual Results	Pass/ Fail
B1	Go to Beginner Scene	Click Beginner Button	NA	Changes scene to Beginner Level	Changes scene to Beginner Level	Pass
B2	Click sound	Click Beginner Button	NA	Audio Source plays sound	Audio Source plays sound	Pass
S1	Go to Settings Scene	Click Settings Button	NA	Changes to Settings Scene	Changes to Settings Scene	Pass

S2	Click sound	Click Settings Button	NA	Audio Source plays sound	Audio Source plays sound	Pass
S3	Return to Main Menu	Click the return button	NA	Returns to Main Menu	Returns to Main Menu	Pass
S4	Select Resolution	Click Resolution dropdown list	Resolution's array	Selected Resolution	Selected Resolution	Pass
S4	Select Quality	Click Graphics dropdown list	Graphics Values	Selected Quality	Selected Quality	Pass
S5	Change Volume	Move slider	Slider value	Save volume	Save volume	Pass
S6	Toggle Fullscreen	Click toggle	Toggle value	Fullscreen on/off	Fullscreen on/off	Pass
E1	Quit application	Click Exit Button	NA	Closes application	Closes application	Pass
E2	Click sound	Click Exit Button	NA	Audio Source plays sound	Audio Source plays sound	Pass
BL1	Return to Main Menu	Click the return button	NA	Returns to Main Menu	Returns to Main Menu	Pass
BL2	Click sound	Click the return button	NA	Audio Source plays sound	Audio Source plays sound	Pass
LS1	Go to Lessons Scene	Click the Lessons Button	NA	Changes to Lessons Scene	Changes to Lessons Scene	Pass
LS2	Click sound	Click the return button	NA	Audio Source plays sound	Audio Source plays sound	Pass
R1	Go to Recap Scene	Click the Recap Button	NA	Changes to Recap Scene	Changes to Recap Scene	Pass
R2	Click sound	Click the return button	NA	Audio Source plays sound	Audio Source plays sound	Pass
PT1	Go to Progress Tracker Scene	Click the Progress Tracker Button	NA	Changes to Progress Tracker Scene	Changes to Progress Tracker Scene	Pass
PT2	Click sound	Click the return button	NA	Audio Source plays sound	Audio Source plays sound	Pass

8. Conclusions and reflections

This project still needs some work and can be developed further as I intended from the beginning. I have only created one level of learning for the prototype stage of the application. The only uncomplete stage of this level is the progress tracking which I did not get around implementing fully so I left it out of the final build. This project is still in an early developing stage and it is likely to lack in features and learning resources. A very important strength of this application is that there are barely any similar projects to this and if developed further it could become something unique. One of the further steps would be unlocking the next two levels to provide new lessons for Intermediate and Advanced guitar players as well. The user would be able to choose their level instead of having to go through all the lessons to unlock their desired course. For further developing a login system for the users to keep their progress and history of what they have learned would be something to consider as well as the notifications panel proposed in the initial list of requirements, that will update the user with new lessons or reminders to continue a certain lesson. The animations, models and design could always be improved to have a more professional look and more lessons could be added to have more learning materials and tips for each level. A weak spot of the application is the lack of user interactivity during the lessons that could come as not appealing to many users, but the purpose of the project was to provide an educational application and not a game; a zoom in feature or even rotation of the camera to see the model better would not change the feel of the project. Another further step in fully developing this application would be evolving the Recap menu into a Training Mode feature where the user can test their knowledge by playing their instrument and getting feedback on what they are doing right or wrong. In a similar way, adding more songs with a certain difficulty level and a Play Along feature as a way of applying what they have learned further.

9. References

AshMesh, 2020. *TurboSquid.* [Online]
Available at: https://www.turbosquid.com/3d-models/3d-low-poly-acoustic-guitar-model-1635007
[Accessed 2020].

Blender, 2021. *Blender.* [Online]
Available at: https://docs.blender.org/manual/en/latest/
[Accessed 2020].

Builder, A. M., 2021. *Autodesk Motion Builder.* [Online]
Available at: https://knowledge.autodesk.com/support/motionbuilder?sort=score
[Accessed 2020].

CodeMonkey, 2019. *CodeMonkey.* [Online]
Available at: https://unitycodemonkey.com/video.php?v=ZVh4nH8Mayg
[Accessed 2021].

Max, A. 3., 2021. *Autodesk 3Ds Max.* [Online]
Available at: http://docs.autodesk.com/3DSMAX/15/ENU/3ds-Max-

Help/index.html?url=files/GUID-8677D7F6-B959-43E2-9E5D-78C3EA4F56FB.htm,topicNumber=d30e5351
[Accessed 2020].

mod3ler, n.d. *TurboSquid.* [Online]
Available at: https://www.turbosquid.com/3d-models/white-guitar-3d-model-1199062

muratakdan, n.d. *TurboSquid.* [Online]
Available at: https://www.turbosquid.com/3d-models/3d-arm-hand-finger-model-1346955

swapnilrane24, 2019. *swapnilrane24.* [Online]
Available at: https://github.com/swapnilrane24/Quiz/blob/master/Assets/Quiz/Scripts/ShuffleList.cs
[Accessed 2021].

Technologies, U., 2021. *Unity Scripting API.* [Online]
Available at: https://docs.unity3d.com/ScriptReference/index.html
[Accessed 2020].

Wikipedia, 2021. *Wikipedia.* [Online]
Available at: https://en.wikipedia.org/wiki/Ultimate_Guitar
[Accessed 2021].

10. Bibliography

AshMesh, 2020. *TurboSquid.* [Online]
Available at: https://www.turbosquid.com/3d-models/3d-low-poly-acoustic-guitar-model-1635007
[Accessed 2020].

Blender, 2021. *Blender.* [Online]
Available at: https://docs.blender.org/manual/en/latest/
[Accessed 2020].

Builder, A. M., 2021. *Autodesk Motion Builder.* [Online]
Available at: https://knowledge.autodesk.com/support/motionbuilder?sort=score
[Accessed 2020].

CodeMonkey, 2019. *CodeMonkey.* [Online]
Available at: https://unitycodemonkey.com/video.php?v=ZVh4nH8Mayg
[Accessed 2021].

Max, A. 3., 2021. *Autodesk 3Ds Max.* [Online]
Available at: http://docs.autodesk.com/3DSMAX/15/ENU/3ds-Max-Help/index.html?url=files/GUID-8677D7F6-B959-43E2-9E5D-

78C3EA4F56FB.htm,topicNumber=d30e5351
[Accessed 2020].

mod3ler, n.d. *TurboSquid.* [Online]
Available at: https://www.turbosquid.com/3d-models/white-guitar-3d-model-1199062

muratakdan, n.d. *TurboSquid.* [Online]
Available at: https://www.turbosquid.com/3d-models/3d-arm-hand-finger-model-1346955

swapnilrane24, 2019. *swapnilrane24.* [Online]
Available at:
https://github.com/swapnilrane24/Quiz/blob/master/Assets/Quiz/Scripts/ShuffleList.cs
[Accessed 2021].

Technologies, U., 2021. *Unity Scripting API.* [Online]
Available at: https://docs.unity3d.com/ScriptReference/index.html
[Accessed 2020].

Wikipedia, 2021. *Wikipedia.* [Online]
Available at: https://en.wikipedia.org/wiki/Ultimate_Guitar
[Accessed 2021].

Appendix I

Final Year Project Demo

https://youtu.be/8nXN5j3n0ys