Gamified Drum Learning System Using Instructional Design Principles

Student Name: Kaho Yeung

Student Number: N00220055

Supervisor: Stefan Paz Berrios

Second Reader: Timm Jeschawitz

Declaration of Authorship

I am aware of the Institute’s policy on plagiarism and certify that this thesis is substantially my own work. Any external assistance, including the use of AI-assisted development tools and generative AI systems, has been appropriately documented and acknowledged within this report and appendices.

Signed: Kaho Yeung

Date: 1st May 2026

Acknowledgements

I would like to thank my supervisor, Stefan Paz Berrios, for their guidance, support, and feedback throughout the development of this project and Fintan Gallagher for introducing me to drumming. I would also like to thank friends and family that have supported me and lecturers in the course for providing the knowledge and foundation that supported this project.

Abstract

This project investigates the design and development of a gamified drum-learning system for beginner drummers using instructional design principles, motor learning theory, and real-time interaction. Traditional drum instruction often relies on notation-based teaching and passive video content, which can limit engagement, delay skill acquisition, and provide limited immediate feedback. The aim of this project was to develop an interactive system that supports coordination, rhythmic accuracy, and learner motivation through a more engaging and practice-oriented approach.

The system was developed using Unity and integrates multiple technologies, including MIDI input through an electronic drum kit, beatmap generation using Music21, and audio processing using third-party transcription (Drum2Notes) and drum software (BDF Player). The application allows users to import songs, synchronize audio and beatmaps, adjust tempo, trim practice sections, and receive immediate visual and auditory feedback during gameplay. Gamification elements such as scoring, combo tracking, and accuracy feedback were implemented to encourage sustained engagement and deliberate practice.

Research into motor learning, gamification, instructional design, and music pedagogy informed the structure and interface design of the system. Principles such as scaffolding, mastery learning, multimodal learning, and cognitive load reduction guided the development of the learning content and gameplay systems. User testing was conducted to evaluate usability and functionality, leading to improvements in synchronization, interface clarity, and visual feedback systems. The findings suggest that combining gamification with instructional design and real-time feedback can provide a more engaging and interactive alternative to traditional beginner drum instruction. The project demonstrates the potential of integrating physical musical hardware with digital learning systems to support skill acquisition, motivation, and music education through interactive technology.

Table of Contents

1 Introduction

Learning to play the drums requires the development of timing, coordination, and motor skills. Traditional beginner drum instructions rely on notation-based learning and passive video lessons, which can limit engagement and delay skill development. Research in music education and motor learning suggests that beginners often struggle to translate symbolic notation into coordinated physical movement, which can negatively affect motivation and early skill acquisition (Repp, 2005; Schmidt & Lee, 2011)

Recently, gamification and interactive learning systems such as Duolingo (Daily streak tracker) and Kahoot! (Competitive quizzes) have shown to improve motivation and learning outcomes by using elements such as progress tracking, immediate feedback, and goal-oriented challenges to sustain user motivation (Muttaqin, M., Zuhdi, H., & Ridwan, 2025). In music education, similar approaches have been shown to support rhythmic accuracy and sustained practice by providing real-time feedback and adaptive difficulty (Hamari et al., 2014; Keeler, 2020).

By implementing gamification combined with musicality, this project aims to make music learning more fun and interactive. Giving the user the ability to import their own song of choice and allowing them to break it down into chunks and learn, makes the process of learning a song more enjoyable.

This report investigates how a gamified learning system, using a physical electronic drum kit, can be designed to support effective drum learning, informed by principles from musicology and instructional design. It draws on research in music pedagogy, motor learning, gamification, and instructional design to guide its approach. The report aims to identify design strategies that enhance motivation, coordination, and skill development. By providing learners with access to new grooves, enabling them to import their own songs, adjust tempo, and segment tracks into smaller sections, the system supports a more engaging and enjoyable learning experience.

1.1 Project Overview

A Digital Learning System (DLS) refers to learning facilitated by technology that gives students flexible access to content such as videos, courses, and quizzes. This DLS uses an electronic drum kit as the primary input. The system will detect drum hits using MIDI (Musical Instrument Digital Interface) data and identify notes based on the corresponding MIDI number.

Gamification techniques will be used (combo, scoring, accuracy) to improve learner motivation and engagement using gamified User Interface (UI) elements.

The learning experience will be provided by integrating immediate feedback, motor learning, visual, and auditory learning underpinned using Instructional design principles.

The system is aimed at beginner drummers who get overwhelmed by notation-based music theory and prioritize deliberate practice and progressive learning, making drum learning more interactive, engaging and effective. Notation learning can cause barriers in improvisation skill development because it may lead students to rely heavily on visual cues rather than developing aural skills (Winters 2013). Multisensory learning (aural and visual) has been more effective than exclusive notation-based learning, especially for students with dyslexia (Winters, 2013. Brady, Personal Interview 2025).

Traditionally to do this, multiple tools and programs need to be combined. I am proposing consolidating all these together into a system in Unity.

1.2 Problem

The problem to be solved is traditional teaching methods, which are notation-based and rely on passive video instruction, make it difficult for beginners and self-taught musicians to learn effectively due to low motivation and stimulation.

Traditional teaching methods lack immediate feedback, visual stimulation, and progressive learning structures, which are vital for effective motor skill acquisition. O'Connor (2012) noted that visual stimulation and gradually building skills prevent students from feeling overwhelmed and reduces cognitive load.

Brady (Personal Interview 2025, Appendix A) observed that students find rudimental studies repetitive and unstimulating, but also emphasized rudiments create foundational motor patterns, which are necessary to create complex rhythm patterns.

1.3 Computing Areas

- **Game Design / Gamification**
  The project uses game design principles such as interactivity, immediate feedback and progress tracking
- **Digital Learning / Education Technology**
  Instructional design principles are used to inform lesson sequence, support skill progression and guide deliberate practice
- **Music Technology / Digital Audio**
  The system uses MIDI input from an electronic drum kit to detect drum hits and timing

- **Systems Integration**
  The project involves integrating multiple technologies (API, Python and Unity) into a unified application
- **Interaction Design (UI/UX)**
  This is a key part of the project that utilizes iterative design, testing, and user interface principles to facilitate an easy and enjoyable learning experience for fellow musicians.

2 Aim and Objectives

2.1 Overall Aim

The aim of this project is to design and develop a gamified digital learning system for beginner drummers, supporting the development of coordination and foundational skills informed by best-practice musical pedagogy.

2.2 Data Flow

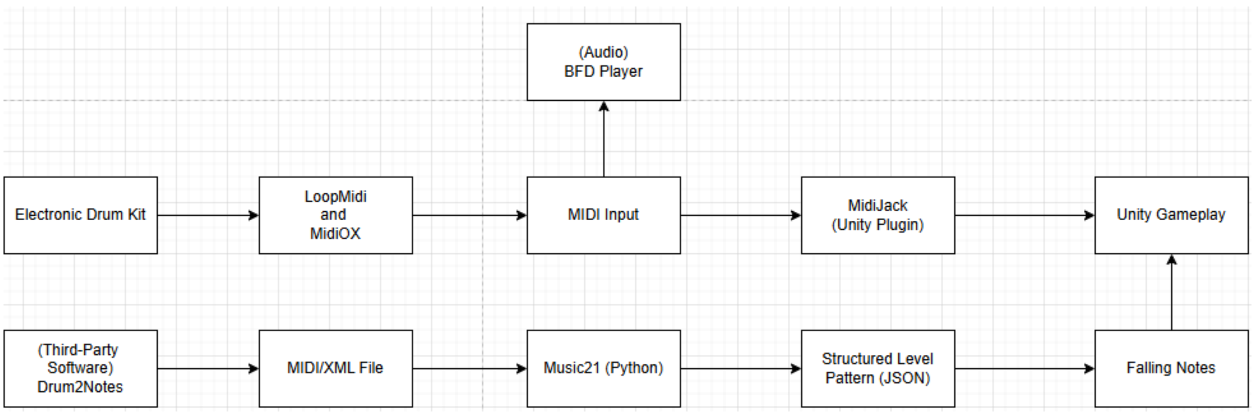

Figure 1 Data Flow

2.3 Objectives

- Connect electronic drum kit to computer and split the midi port so it can be used by two different software BFD Player and Unity, this can be done using LoopMidi (creates a port for BFD and Unity) and MidiOX (routes the new ports into each software)
- Detect Midi input in Unity using MidiJack, a Unity Plugin that allows Midi detection in Unity. This allows inputs from the physical kit to light up the kit in Unity, giving visual feedback.
- Use third-party software (Drum2Notes) to extract and export drum patterns from mp3 files as midi/xml.
- The MIDI and MP3 files will be imported and parsed using Music21 python toolkit to generate level patterns. These patterns are visualised using falling notes which will light up the pad on contact.
- Sync the song with a generated level pattern, combining these creates a Beatmap. This supports deliberate practice by allowing the user to play-along.
- Provide the user with the opportunity to adjust the song tempo. This allows the user to study the song and play along slowly, altering the tempo incrementally to their desired speed.

- Implement immediate feedback using audio from the kit and gamified UI elements (combo, score, accuracy, and hit feedback). The user can audibly notice when they are playing wrong if their own playing doesn't match the song. Visual feedback is shown to inform the user of late and early hits.
- Create a scaffolding course and assessment strategy. The course will contain drum learning content such as demonstration videos, grooves, and rudiments. The user will be assessed at the end of playing a song, where their score, accuracy, and number of late/early notes will be displayed.

3 Requirements and Feasibility Study

This section aims to assess this project's practicality by identifying strengths and weaknesses and confirming that the project is achievable and deliverable within this project's timeframe. The functional and non-functional requirements will be defined to guide the design and development.

3.1 Requirements

3.1.1 Functional Requirements

1. **Detect MIDI input from an electronic drum kit**
   The system will detect drum hits using MIDI input and identify notes based on corresponding MIDI number.
2. **Support file import**
   User can import a MIDI/XML file containing a drum-isolated song.
3. **Convert MIDI/XML data into a level pattern**
   The imported file will be converted into a JSON file containing a level pattern. JSON is used as it is easily readable, compact, and can communicate with virtually any language (Baranwal, 2019).
4. **Provide structured learning content**
   The system will have instructional learning content covering rhythmic and rudimental studies.
5. **Enable interactive practice and gameplay**
   Users will be able to apply learned concepts into gameplay.
6. **Display gamified UI elements**
   Gamified UI elements will be displayed during gameplay providing immediate feedback on performance accuracy.
7. **Support audio playback**
   Users will be able to hear the drum kit and play along to synchronized audio playback.
8. **Allow metronome and tempo control**
   Users can toggle metronome and tempo control to support deliberate practice.
9. **Review performance**
   Users can listen to recordings of previous practice sessions.

3.1.2 Non-Functional Requirements

1. **Usability**
   Intuitive and accessible for beginner drummers with no knowledge of music theory.
2. **Scalability**
   Expansion of instructional content, and game modes.
3. **Cross-platform**
   Deployable on desktop platforms capable of running Unity applications.

3.1.3 Feature Backlog

Priority	Feature
HIGH	MIDI Detection
HIGH	MIDI File Import
HIGH	Level Pattern Generator
HIGH	Learning Content
HIGH	Audio Playback
HIGH	Gameplay
MEDIUM	Gamified UI Elements
MEDIUM	Metronome and Tempo control
LOW	Review Performance
LOW	Storing Replays

High priority features represent the minimum viable product (MVP) required to function as a learning tool.

Medium priority features enhance motivation and usability but are not essential for the system to function.

Low priority features are additional addons that may be simplified if time constraints arise.

3.1.4 Constraints and Assumptions

The system assumes access to a MIDI compatible electronic drum kit.

The system assumes access to a computer capable of running Unity applications.

Current use of third-party software (Drum2Notes) to export MIDI/XML files is a paid subscription. Finding MIDI files for a specific song is difficult to find for free. Other methods like finding a transcription (StephenMackDrumming) and converting it into a MIDI (MuseScore) is possible but may require editing the MIDI file to match the drumkit MIDI numbers.

3.2 Feasibility Study

3.2.1 Technical

The project will be developed using well-documented and industry-standard technologies. The system consists of three main components:

1. MIDI is the standard communication method for electronic instruments. MIDI data from the electronic drum kit is transferred to the computer via USB cable, the MIDI inputs will be used to interact with Unity. Unity provides existing plugins such as MidiJack allowing real-time MIDI detection.

2. Music21 is a python toolkit used for music analysis and level pattern generation. Music21 will parse and convert imported MIDI/XML files into a JSON file containing the level pattern which Unity can visualize for gameplay.

3. Instructional content will be integrated into Unity to create a seamless transition from learning to gameplay to freeplay, resulting in a unified application that reduces cognitive load and combines all the moving parts into one system.

3.2.2 Economic

The project is economically feasible as it primarily relies on free or student-licensed software and tools. Unity and MidiJack Plugin (Student-Licensed), Python, and Music21.

Optional third-party tools such as paid transcription tools (Drum2Notes) will not be required for the core functionality, as free alternatives are available. Although this tool was initially intended to be integrated into this system, the cost of API access necessitates that its use remain optional.

No financial investment is required beyond existing hardware (drum kit and computer).

3.2.3 Operational

This system is designed to be usable for beginner musicians with minimal technical setup. Users will require an electronic drum kit with MIDI output and access to a computer capable of running the application. The unified application reduces the need to switch between multiple tools, improving usability, and learning flow.

The instructional design supports progressive learning through structured lessons, immediate feedback, and gamified progression. These features follow common learning platforms such as Duolingo, Linkedin Learning, and Drumeo, allowing users to learn content and apply them in assessment and interactive gameplay (freeplay).

Testing will be conducted throughout development, focusing on compatibility, integrations, usability, timing, accuracy and learner feedback.

3.2.4 Feasibility Conclusion

Based on technical, economic, and operational factors, this project is feasible for the Major Project. By using well-documented and industry-standard technologies, the project avoids unnecessary technical risks.

This feasibility study confirms that the project is realistic and can be successfully developed within the project timeframe.

4 Technologies

This section discusses the justification for the technology used to develop this project, and outlining the specific research that will be conducted relating to instructional design and gamification.

4.1 Core Technologies

4.1.1 Unity (C#)

Unity is a game engine which will be used as the primary environment for developing this application. Unity is very popular for game development as it is suited for interactive, real-time systems and provides MIDI support, and audio playback. Unity supports UI systems that allow integration of learning content and gameplay within a single application, aligning with the project goal of a unified application. I wanted to challenge myself with this project by expanding the way Unity is traditionally used by implementing physical computing, gamification, and music education together.

4.1.2 MIDI

The system will use MIDI data from an electronic drum kit to detect drum hits in real time. MIDI provides precise timing and information on each note.

4.1.3 Music Analysis (Python)

Music21 is a python-based toolkit used for analyzing musical data. MIDI or XML files will be parsed and converted into a structured JSON level pattern that can be imported into Unity.

4.1.4 JSON

A JSON format will be used to represent rhythmic patterns, note timing and positions. JSON is human-readable, compact and easily parsed in Unity, making it the best choice for transferring musical data (Baranwal, 2019).

4.1.5 User Interface and Learning Content

Unity's UI system will be used to design the learning interface, including lesson navigation and gameplay feedback. Tabs will separate instructional content (learning mode) from interactive practice (Freeplay mode) and assessment (Gameplay mode) supporting instructional design principles such as scaffolding and mastery learning.

4.2 Justification of Technology

Unity was chosen over other game engines, such as Unreal Engine, as it was covered within the course and provides a built-in UI system. Although the UI system in Unity is weak, it was capable of displaying basic scaffolding techniques and allowing course progression such as progress bars and ensuring the previous lesson is completed before proceeding. Future plans may involve an embedded React WebView, which will allow better animations, state objects, and an overall better interface. React allows flexible layouts suitable for all device sizes and can be linked with other tools and frameworks (Mark, 2025).

Music21 analyses symbolic notation data instead of complex sound waves. This toolkit is referenced in academic research (Cuthbert, 2010 https://dspace.mit.edu/handle/1721.1/84963) related to music analysis, supporting its suitability for this project.

MIDI is the standard communication method for electronic instruments by storing the instruments notes, timing and velocity. MIDI data will be used to communicate inputs to the interactive game mode, enabling real-time feedback on timing and accuracy.

JSON is used as the data interchange between Music21 and Unity due to the ability to communicate between languages, lightweight, and human-readable format (Baranwal ,2019). This allows musical data to be parsed efficiently.

5 Research

This section discusses the research that was conducted to justify the use of gamification, instructional design, electronic drum kit, and User Interface used.

5.1 Game Interface

The design of the user interface (UI) plays a critical role in supporting skill acquisition and motivation for everyone. Research in music education and motor learning highlights that immediate, interactive feedback enhances coordination, timing and overall learning outcomes (Repp, 2006; Schmidt & Lee, 2011) Several features commonly used in drumming instruction can be translated into a digital interface to support these principles.

A metronome function provides consistent timing cues which is essential for drummers to develop rhythmic accuracy and sensorimotor coordination. Brady (Personal Interview, 2025), noted that his students rely heavily on metronome during lessons to internalize tempo. Studies in music pedagogy confirm that consistent feedback accelerates skill development. (Repp, 2005).

The ability to slow down songs supports deliberate practice, allowing learners to focus on coordination and muscle memory before performing at full tempo. Brady (Personal Interview, 2025) observed that students often attempt to play at full speed, when they should slow down the tempo first to induce muscle memory. Slower practice aligns with motor learning theory, which mentions the importance of controlled repetition for forming muscle memory. Slow practice provides "thinking space" for musicians to develop accurate motor patterns and muscle memory through controlled repetition before increasing tempo (Maxfield, 2018). It is especially effective when segmenting tasks, reducing cognitive load and assisting coordination (Maxfield, 2018).

Playing along to music tracks is another key component. Based on the principle of specificity, practicing to the song you want to learn will enhance rhythmic perception and musicality by developing the aural, timing and sense of groove (Keeler, 2020; Pesek et al., 2024). According to Brady (Personal Interview, 2025) playing along to songs is “100% important” for developing their ear, timing and sense of groove.

Lastly, self-assessment through playing back their own recordings reinforces learning through reflective practice. Learners who evaluate their own performance are better able to identify errors and track progress, based on principles of mastery learning and formative assessment (Evans, 2025). Brady (Personal Interview, 2025) emphasized his students to record and listen to their own playing as “they are their own best critic”.

All these interface features are supported by research as effective mechanisms for promoting motor skill acquisition, coordination and sustained learner engagement in beginner drummers.

5.2 Why Visualize the drum kit?

Brady (Personal Interview, 2025):

- Tutors use bird’s-eye-view kit demonstrations
- They teach beginners where drums are located physically
- Visual demonstration is more beneficial than notation for early learners (especially dyslexic learners)
- Students rely heavily on visual layout to understand coordination

Gibson’s (1979) Affordance theory proposes that learners can directly see what actions are possible when the environment makes those possibilities obvious. Displaying the drum kit visually allows learners to see which drum pad to strike. Aligning the in-game visuals with the physical drum kit supports intuitive perception.

Norman (1988) argues that users learn systems more effectively when interfaces reflect familiar real-world structures through clear conceptual models. The visual display of the drum kit leverages the learners understanding of the drum kit layout, reducing cognitive load and minimizing the need for notation-based translation. This is beneficial for beginners and dyslexic learners who often rely more on audio and visual information when developing coordination and motor skills

These theories together support the use of visualised drum kit as an interface that promotes intuitive perception, coordination and skill transfer from in-game to real-world drumming skill.

5.3 Why gamification?

Gamification increases learner motivation by providing clear goals and a visible sense of progress, which are key factors in early skill development. Gamified learning systems like Duolingo have been proven to be effective for learning by implementing a daily streak tracker, points, leaderboard and social interactions (Muttaqin, M., Zuhdi, H., & Ridwan, 2025). These implementations increase engagement and motivation, by having a daily streak it motivates users to be consistent and learn for even five minutes a day just to keep the streak ongoing. For beginners who often find repetitive practice unengaging, a gamified structure makes drills more engaging and encourages longer, more focused practice sessions (Deci & Ryan, 2000; Hamari et al., 2014).

Real-time visual and auditory feedback allows learners to identify and correct timing errors immediately, which is essential for developing sensorimotor coordination. Immediate feedback supports faster skill acquisition by reinforcing correct movements. (Repp, 2005; Schmidt & Lee, 2011).

5.4 Immersion

Keeler, Kevin R. Jr. (2020) "Video Games in Music Education"

Talks about the similarities in the 'flow' state in video games and playing an instrument and whether it made a difference in their rhythmic skills. The 'flow' state is a psychological state of total absorption, focus, and deep enjoyment that can occur during composition, listening, or performance. In this state the person is fully immersed in what they are doing.

The flow state is achieved when a person is fully immersed in the activity they are doing. If the activity is too easy, boredom can set in, causing them to be less focused, but if the activity is too difficult, anxiety can arise causing them to rush and overthink. Both boredom and anxiety are barriers to flow. Any activity that requires a lot of motivation and concentration, and is enjoyable, can help create a sense of flow. (Bonaiuto et al., 2016).

By having full control over the song choices and tempo, the user can tune it to their desired skill level which induces the 'flow' state. During this state the user has reduced negative emotions, which encourages them and results in sustained practice. This makes the user internalize the content more effectively, leading to improved long-term mastery (Newsome, 2016).

5.5 Rhythmic Foundations

A well-developed sense of rhythm is fundamental for drummers, as rhythmic perception and motor coordination develop simultaneously. Research in motor learning suggests that learners must establish a stable internal sense of timing before more complex coordination can emerge (Repp, 2005). Core rhythmic concepts such as beats, bars, time signatures, and subdivisions provide a structural framework that enables learners to organise movement in time.

While music theory and notation can support musical understanding, they are not essential for the development of core drumming skills. Some prominent drummers have developed their skills with limited reliance on notation. For example, Buddy Rich was famous for his inability to read sheet music, instead he focused on learning and performing material by ear and memory (CMG Worldwide, 2023).

5.5.1 Rhythm Subdivision

Brady (Personal Interview, November 27, 2025) emphasized that drummers should first learn the basics of rhythmic subdivisions before progressing to complex patterns. He noted that effective instruction begins with quarter note grooves, then progresses to eighth, sixteenth and triplet grooves.

This progression reflects a progressive/hierarchical approach to rhythmic learning starting from simple to complex. From a pedagogical perspective this aligns with scaffolding and mastery learning principles as learners build on top of previous skills before advancing (Schmidt & Lee, 2011).

5.5.2 Groove timing

Groove timing involves the experience of timing, feel and musical context. Studies in music education have shown that practicing along to recordings enhances rhythmic perceptive and timing by reinforcing auditory motor skills (Keeler, 2020; Pesek et al., 2024) This highlights the principle of specificity: to improve a skill, you need to practice that skill.

5.5.3 Rudiments

Rudiments are like the alphabet of drumming. They are the foundational motor patterns that support complex rhythmic structures. From a motor learning perspective, rudiments enable the formation of consistent movement patterns through repetitions, strengthening muscle memory and coordination (Schmidt & Lee, 2011). Instructional systems that introduce rudiments allow learners to build technical skills while minimizing cognitive overload.

5.5.4 Stick Grip

The first skill beginners should master is how to hold the drumsticks (Brady, K. (2025)

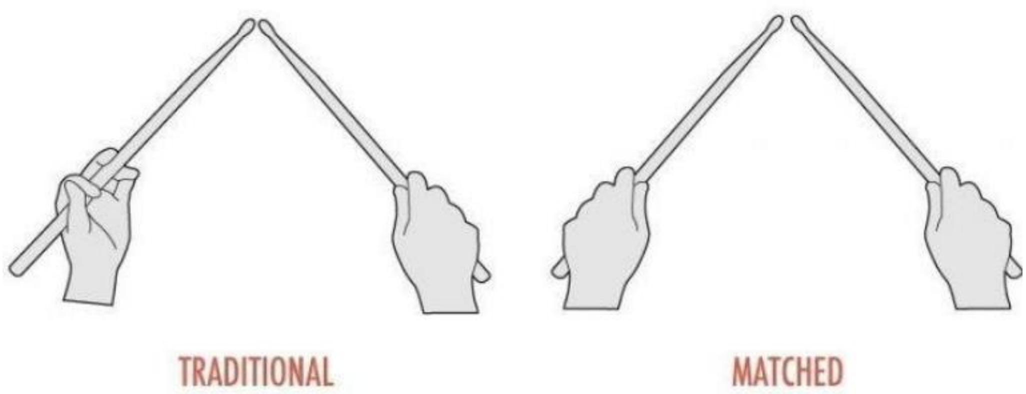

Figure 2 Drumstick Grips

Drumstick grip represents the primary interface of drums influencing control, comfort and endurance. Matched grip is both hands with matching grip hence the name. Both hands are holding the drumsticks overhand. Matched grip has become more dominant due to its symmetrical approach, and it allows more natural wrist movement for better power transfer.

Traditional grip is more specific and is used in marching and military bands, it requires more specialized coordination and typically introduced at later stages. Traditional grip is favoured for its finesse and control for more dynamic sounds. (Strain, J. A. (2002).

Together these theoretical foundations inform the structure of instructional content within a gamified drum learning system by aligning theory, motor learning principles, and pedagogical scaffolding. Beginner learners can develop rhythmic skills, coordination and musicality more effectively.

5.5.5 Pedagogy

Pedagogy refers to the method and practice of teaching and learning, encompassing teaching approaches, teaching theories, and feedback and assessment strategies (Editorial, T., 2025).

5.5.6 How Drumeo and LinkedIn Learning use pedagogy

Drumeo and LinkedIn Learning use similar teaching methods, particularly scaffolding, where learners build skills gradually from simple to more complex tasks.

5.5.6.1 Drumeo

5.5.6.1.1 Teaching approaches

- **Video lessons with step-by-step demonstrations**
  Drumeo uses structured video lessons (around 10–15 minutes) where instructors break down techniques into clear, manageable steps. Lessons often begin at a slow tempo so learners can understand the movement, then gradually increase in speed to match real performance conditions.
- **Tiered lesson structure (beginner → intermediate → advanced)**
  Content is organised by skill level, allowing learners to start at an appropriate point and progress at their own pace. This supports personalised learning and ensures that users are not overwhelmed or under-challenged.
- **Guided practice through play-along tracks**
  Learners can practice alongside audio tracks, helping them apply skills in a musical context rather than just isolated drills.

5.5.6.1.2 Teaching theory

- **Repetitive, focused skill-building**
  Drumeo emphasizes repetition of specific techniques (e.g., rudiments), reinforcing accuracy and consistency over time.
- **Scaffolding**
  Skills are introduced in a logical sequence, with each new concept building on previously learned material. This gradual increase in complexity supports long-term skill development.

- **Motor learning**
  The platform focuses on coordination, timing, and muscle memory. Learners are encouraged to master foundational movements before progressing, which aligns with how physical skills are developed in motor learning theory.

5.5.6.1.3 Feedback and assessment

- **Mastery before progression**
  Learners are encouraged to fully understand and perform a skill correctly before moving on, even though progression is largely self-paced.
- **Self-assessment through play-along exercises**
  Users evaluate their own performance by playing along with tracks, listening for timing and accuracy, which develops independent learning and critical listening skills.

5.5.6.2 Linked Learning

5.5.6.2.1 Teaching approaches

- **Short, focused video lessons**
  Lessons are broken into small segments, making them easy to follow and helping learners stay engaged without feeling overwhelmed.
- **Interactive exercises embedded in lessons**
  Many courses include hands-on activities that allow learners to immediately apply new knowledge, reinforcing understanding.
- **Courses organised by skill level and topic**
  Content is structured so learners can follow a clear pathway, from introductory concepts to more advanced material.

5.5.6.2.2 Teaching theory

- **Cognitive Load Theory**
  Content is delivered in short, manageable chunks to reduce mental overload, making it easier for learners to process and retain information.
- **Constructivism**
  Learners actively build knowledge by applying what they've learned through exercises and real-world examples, rather than passively watching content.
- **Mastery Learning**
  Progression often depends on completing tasks or understanding key concepts, encouraging learners to fully grasp material before moving forward.

5.5.6.2.3 Feedback and assessment

- **Built-in quizzes and knowledge checks**
  Learners receive immediate feedback on their understanding, helping them identify gaps and reinforce learning.
- **Certificates of completion**
  These provide motivation and a sense of achievement, while also offering a way to demonstrate skills to employers.

5.6 Comparison of Drumeo and LinkedIn Learning

Both Drumeo and LinkedIn Learning use structured, scaffolded learning approaches, but they differ in emphasis due to subject context. Drumeo focuses on motor skill acquisition, using slow-to-fast demonstrations, repetitive practice, and play-along exercises to develop timing, coordination, and muscle memory. Progression is strongly dependent on physical skill mastery, reflecting principles of motor learning and procedural repetition.

LinkedIn Learning, focuses more on conceptual and knowledge-based learning. It uses short, segmented videos to reduce cognitive load, supported by quizzes, progressive learning and interactive tasks to reinforce understanding.

Despite these differences, both platforms share key pedagogical principles: structured progression by skill level, incremental difficulty, and mastery-based advancement. Drumeo applies these principles to physical performance skills, while LinkedIn Learning applies them to cognitive skill development.

5.7 Instructional design

Instructional design is how the process of learning products and experiences are designed, developed, and delivered to support effective learning outcomes. (Evans, L. 2025)

Online learning platforms such as Linkedin Learning and Drumeo demonstrate instructional design in practice by organizing content progressively, incorporating interactive elements, and aligning learning activities with clear objectives.

Instructional design principles inform the structure of the gamified drum-learning system, ensuring the lessons progress from simple to complex, provide timely feedback and support sustained learner engagement.

5.7.1 Immediate feedback and motor learning

Building on the pedagogical and theory foundations discussed earlier the instructional design of this system integrates principles of motor learning and multimodal instruction to support beginner drum skills. Immediate visual and auditory feedback is embedded throughout gameplay, allowing learners to identify and correct timing errors as they occur. This approach aligns with established motor learning research, which shows that immediate feedback accelerates sensorimotor learning, supports coordination, and strengthens muscle memory (Repp, 2005; Schmidt & Lee, 2011; Pesek et al., 2024).

5.7.2 Visual and Auditory Learning

Visual and auditory instructional elements are combined through a visualized drum kit and synchronized sound feedback. The visual layout reflects common teaching practice such as bird's-eye view demonstrations used by drum tutors in professional practice (Brady, 2026), supporting spatial understanding and coordination (Norman, 1988). By pairing these visuals with physical interaction on a real drum kit, the system supports learning with perception principles. (Gibson, 1979).

This multimodal instructional design is beneficial to beginner learners and dyslexic users who struggle with notation-based teaching as it prioritizes perceptual and physical learning over notation-based learning (Brady, 2025)

6 Design

This section describes the process of designing each part of the interface and justifying the use of the designs based on best practice.

6.1 Paper Prototypes / Prototypes

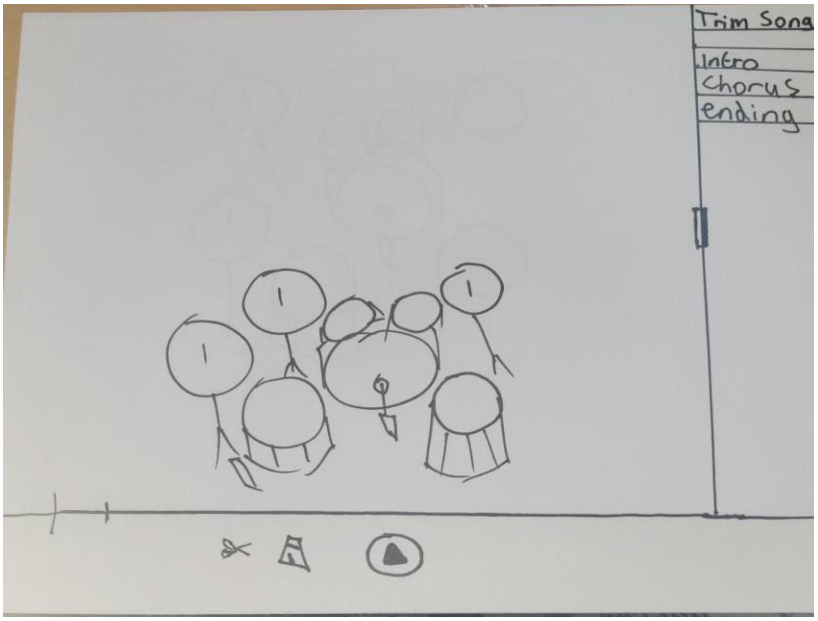

Figure 3 First iteration

In the first iteration of the gameplay paper prototypes, the design was very minimal, and some features like the trimmer were unclear.

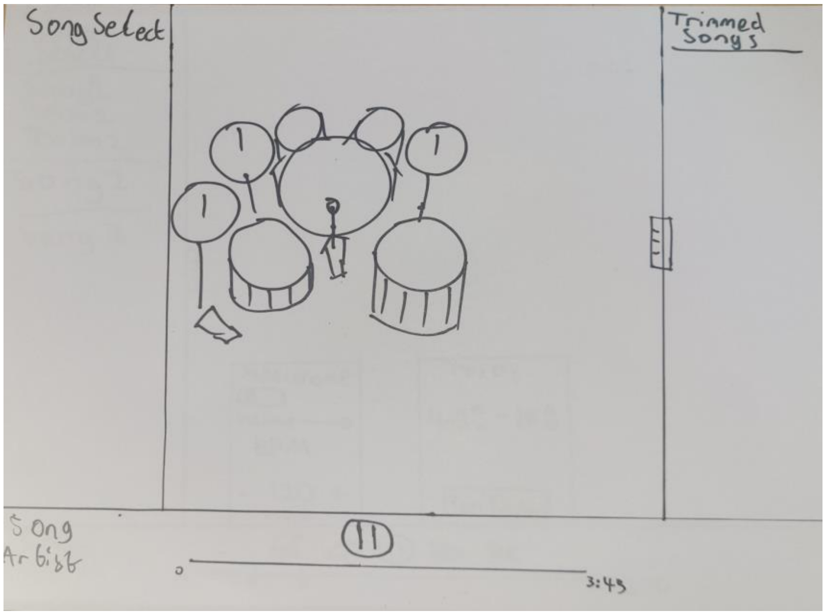

Figure 4 Second Iteration

The second iteration showed similarities to the music player Spotify. The layout showed clearly the length of the song and song title. Too many sidebars made this design clutter the screen with unnecessary information causing cognitive overload.

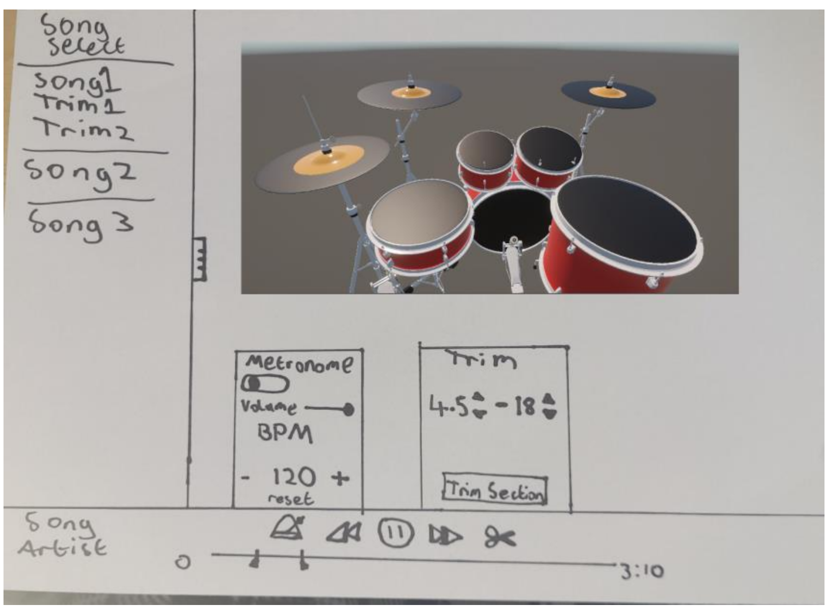

Figure 5 Third Iteration

In the third iteration more features were implemented such as metronome and song trimming, the sidebars were combined into one, reducing cognitive load.

6.2 App User Interface

This section justifies the design that was implemented during development.

6.3 Importing a song

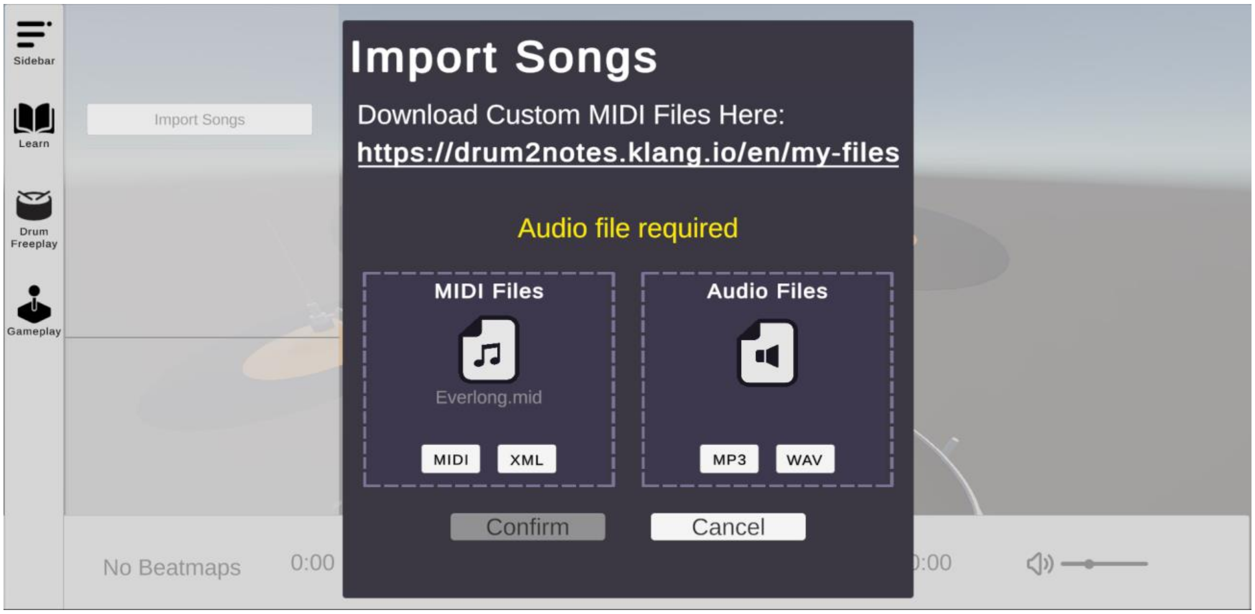

Figure 6 Import Song

This component uses a modal window which disables the main content. A modal window is effective for this component because it directs the user towards the information, requiring the user to act immediately before proceeding. This reduces cognitive load as it keeps the user focused on the specific task (Fessenden, 2017).

The upload components are boxes with dotted outlines, visually indicating that they are interactive elements. File format labels clearly communicate the types of files accepted. When a box is selected, the file explorer is opened with filters applied to restrict selectable file types, helping to prevent invalid file uploads. This design aligns with usability principles, specifically error prevention. (Nielsen, 1994)

6.3.1 Early design

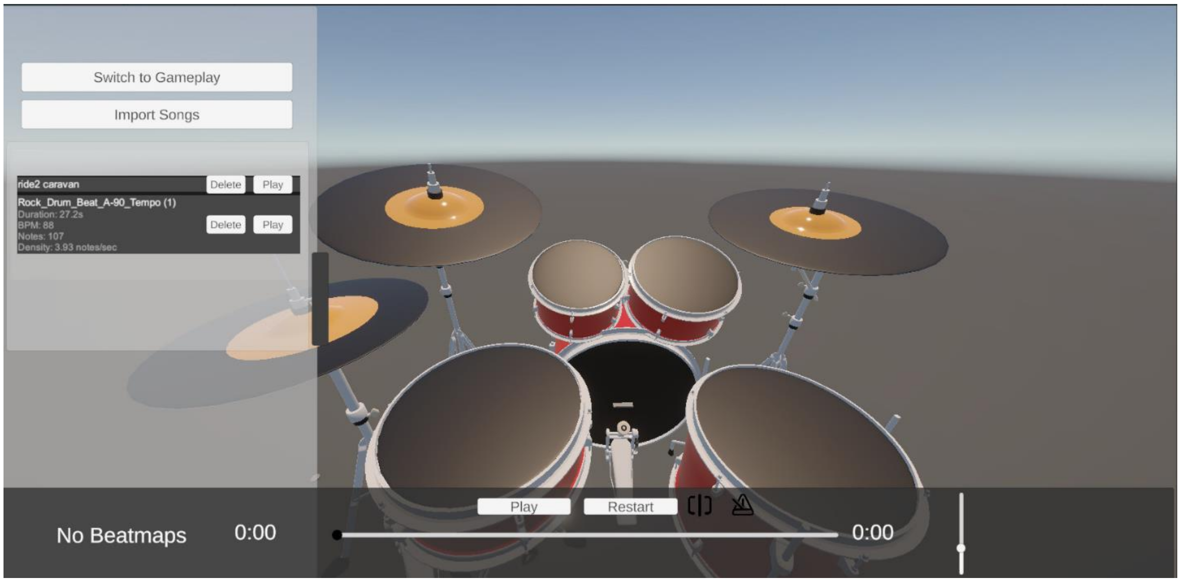

Figure 7 Import Song First Iteration

In the first iteration, a button opens the file explorer.

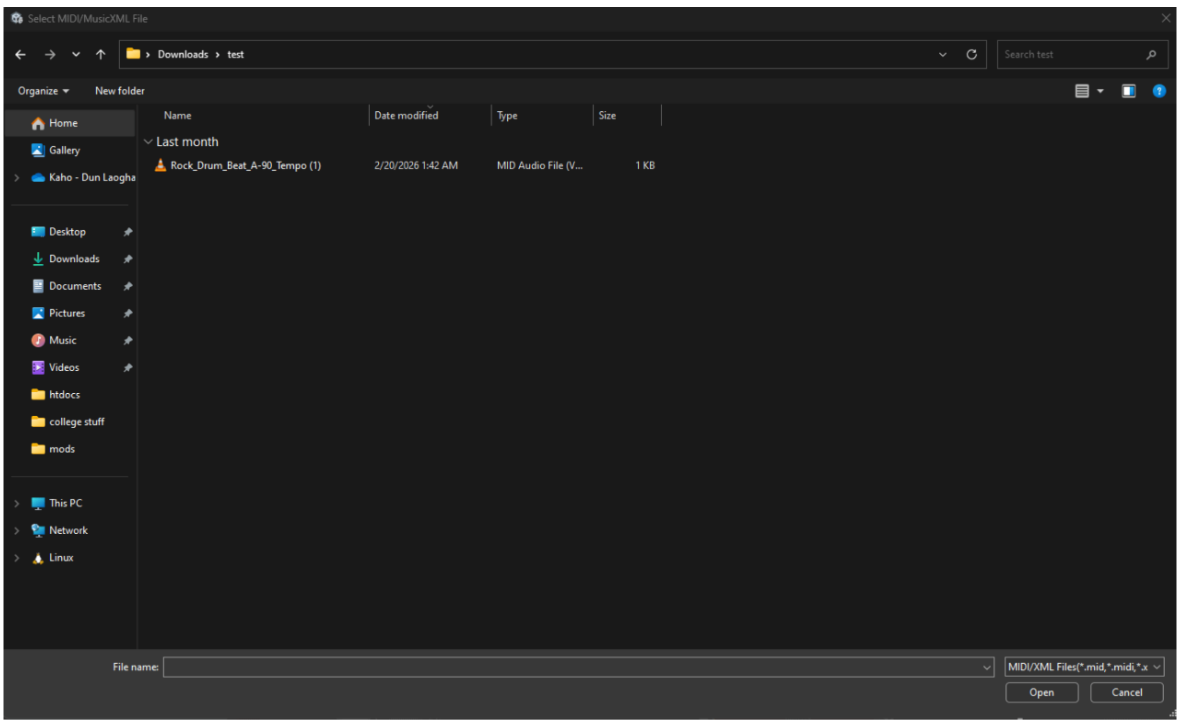

Figure 8 Import File Explorer

The file explorer filters the midi file first, then the mp3 audio file. This design lacks Visibility of System Status, Help and Documentation, User Control and Freedom.

This design was unclear about what files were being accepted and where to access midi files.

6.4 Syncing audio with notes

Figure 9 Audio Sync

This component appears right after importing a song, A modal window is also used in this component to maintain user focus and reduce cognitive load by limiting interaction to the current task (Fessenden, 2017). A waveform is displayed based on the imported mp3 file. This design uses recognition over recall and flexibility of use, both are usability principles (Nielsen 1994). The waveform supports recognition over recall by visually displaying the audio, reducing the need to remember exact timestamps. The user can choose to click on the waveform for approximate timing and use the input field for precise timing, both of which give the user flexibility of use. A marker gives the user visual feedback of where the drums are marked.

6.5 Early Designs

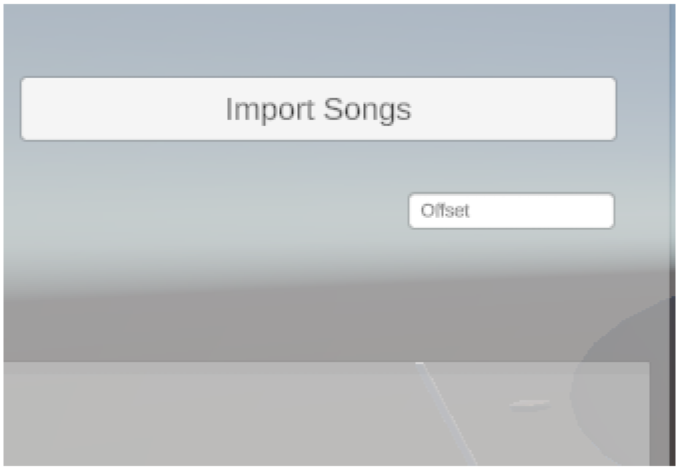

Figure 10 Audio Sync First Iteration

In the first iteration, users were required to manually input the drum offset before importing a song. This approach assumed the user knew the exact timing of the drums, which is unrealistic for beginners. The design lacked flexibility and efficiency of use, limited user control and freedom, and failed to support error prevention.

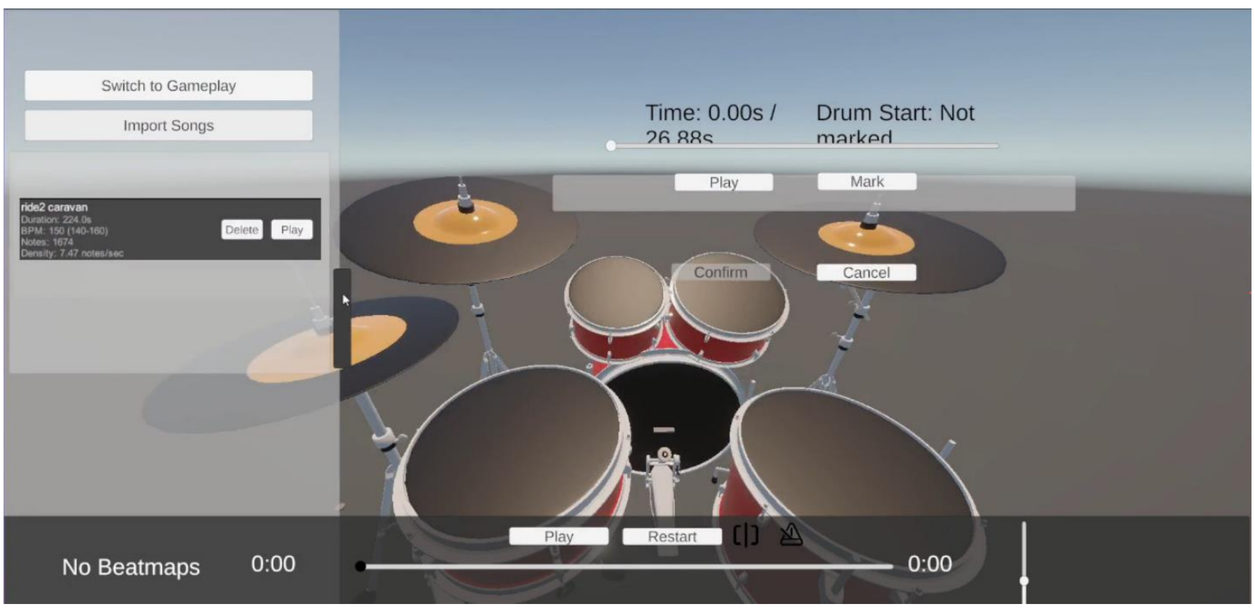

Figure 11 Audio Sync Second Iteration

In the second iteration, the usability principles improved, but it still lacks visibility of system status. This iteration was mainly used to test functionality. Compared to the final iteration, this struggled with precise timing and visual feedback of where the drums are marked. This panel made it unclear on what to focus on as there are many things going on like the sidebar, media player and the drums in the background. This was why a modal view was the preferred choice for this component.

6.6 Audio Trimming

Figure 12 Audio Trimming

This component also utilises a modal view to maintain user focus and consistency across the interface. It is visually similar to the audio sync component, featuring an audio waveform based on the imported track. Two adjustable handles allow for approximate trimming, with the selected area highlighted to distinguish it from the rest of the waveform. A thin play head indicator displays the current playback position.

For precise control, input fields with increment controls are provided for precise start and end times. An additional input field allows the user to assign a name to the trimmed audio. These features align with established usability principles, including visibility of system status, user control and freedom, and flexibility of use (Nielsen, 1994).

6.6.1 Audio Trimming First Iteration

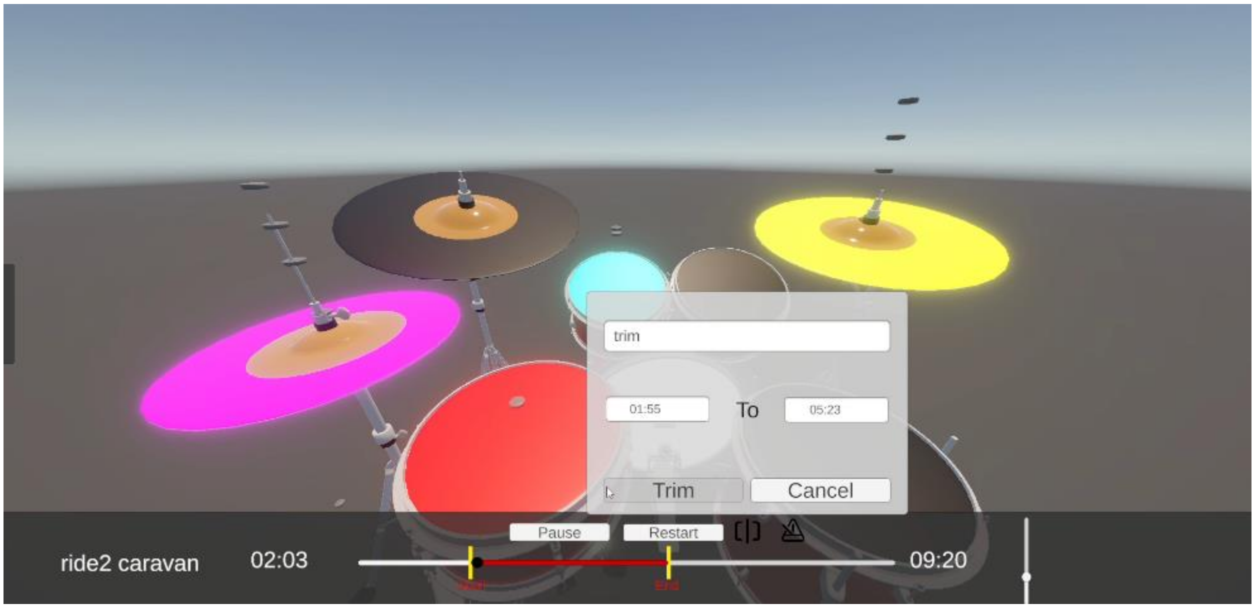

Figure 13 Audio Trimming First Iteration

This iteration had a problem with visibility of system status and cognitive load. The trimmer was using the main timeline slider instead of the waveform and the handlebars had a problem with the hitbox detection.

6.7 Beatmap Selection

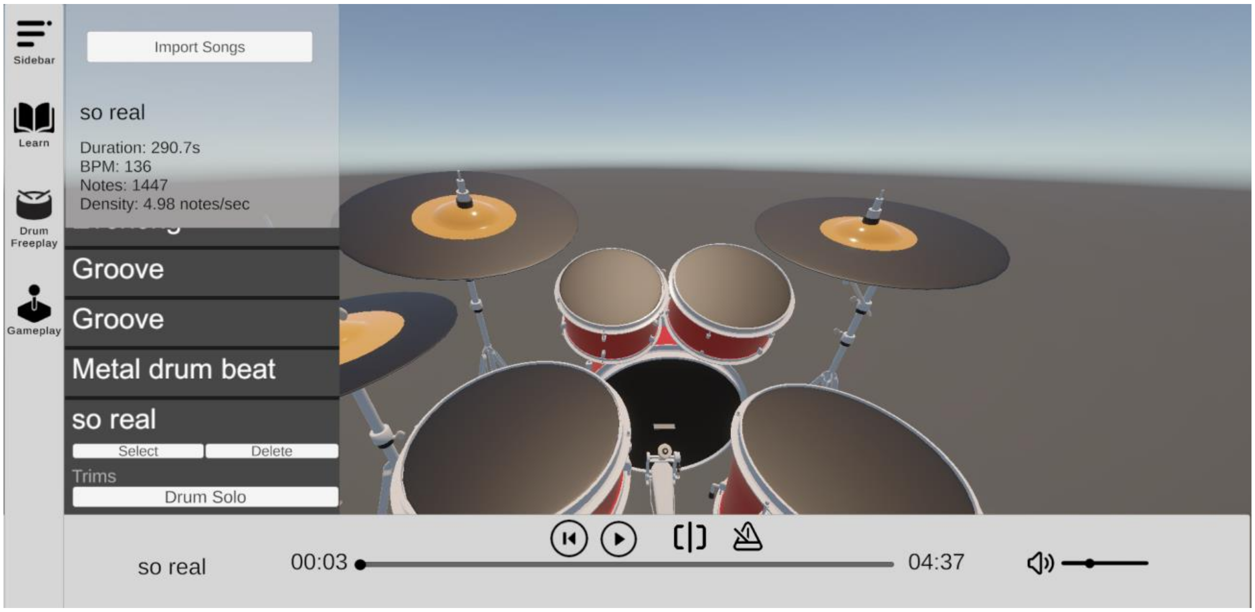

Figure 14 Beatmap Selection

A side panel is used for the beatmap selection to quickly choose between songs. This component uses a scrollable accordion user interface allowing only one song to be chosen. Accordions reduce clutter by showing only relevant information until it is interacted with, lowering cognitive load (*Wang, H.-H. 2023)*. Similar layouts are used in other rhythm games to easily scroll and scan through a large selection of songs.

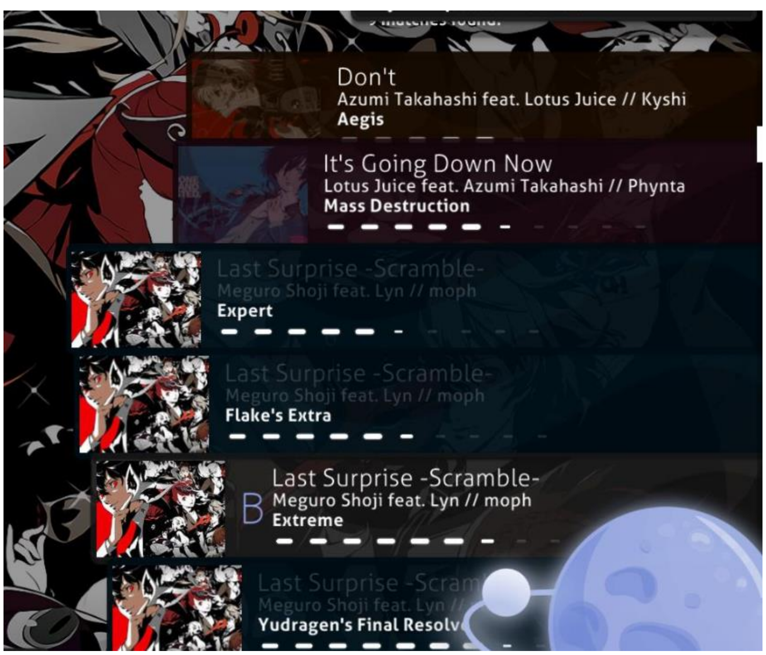

Figure 15 Osu! Rhythm Game using accordion layout

When a song is selected, more information on the song is displayed. Trimmed songs are grouped under the same song.

6.7.1 Early Designs

Figure 16 Beatmap Selection Second Iteration

The play button on the accordions is misleading as this only selects the song, not play it. This confused some users that were testing the app.

6.8 Metronome

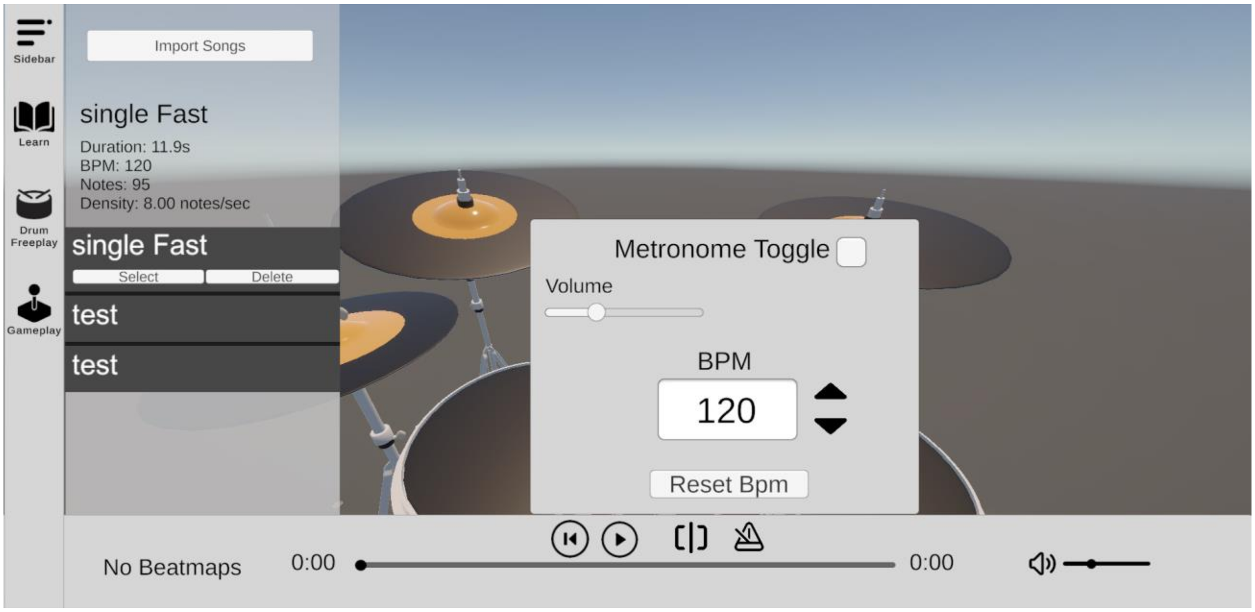

Figure 17 Metronome

The metronome control is represented by a recognizable icon, supporting the usability heuristic of recognition over recall by allowing users to identify its function (Nielsen, 1994). Activating the control opens a small panel that gives the option to toggle the metronome, adjust volume, and modify tempo (BPM).

This design supports user control and freedom by allowing users to enable or disable the metronome and adjust settings in real time. The use of clearly labelled controls and immediate feedback when changes are made also aligns with the visibility of system status. Grouping all metronome related settings within a single panel promotes consistency and reduces cognitive load (Nielsen, 1994).

6.9 Gameplay

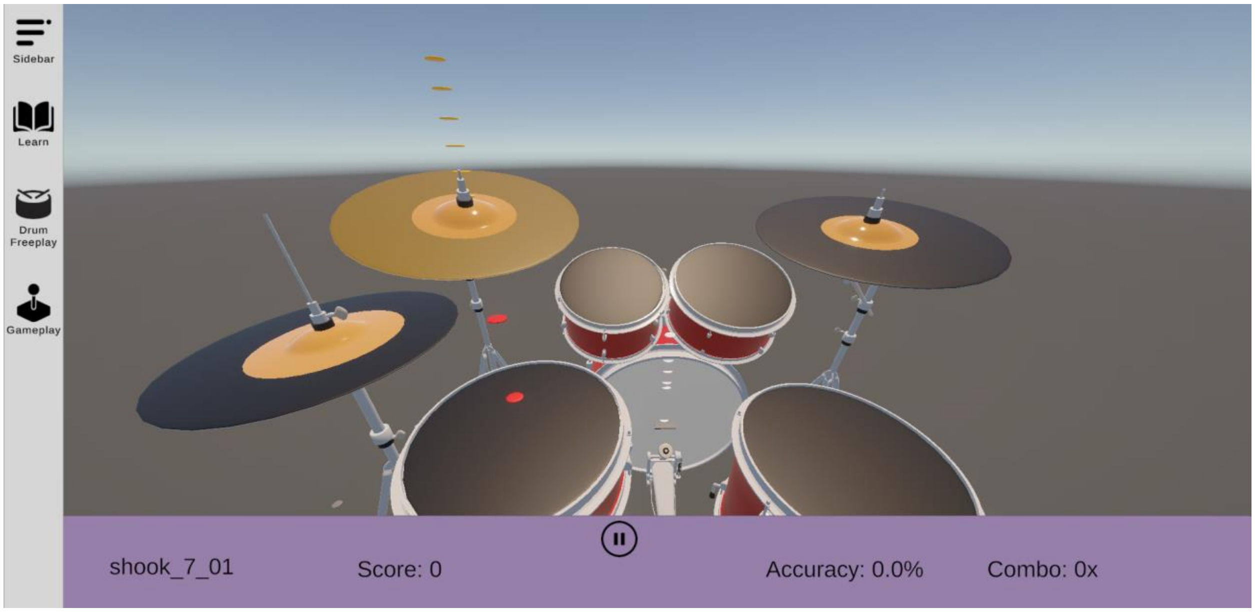

Figure 18 Gameplay

Gameplay mode is an assessment environment where user performance is evaluated. Unlike Free Play mode, users do not have access to controls such as play time adjustment, trimming, or tempo control, and must perform the full song. As the user interacts with incoming notes, feedback is provided based on timing accuracy (e.g., early or late hits), which contributes to an overall score and accuracy.

The falling notes are based off other rhythm games such as Guitar Hero as it directs the note to the specific drum pad, gives visual cue of when to hit, visualizes drum patterns and rhythm timing based on the spacing between notes. This allows the user to study the songs drums visually, instead of listening to the full song where other instruments could mask the audio of the drum.

To visually distinguish gameplay mode from other modes, the bottom panel changes colour. The use of colour as a visual indicator aligns with usability principles of visibility of system status, providing immediate feedback to the user about the current mode (Nielsen, 1994).

Figure 19 Stat Screen

This component appears after assessment, the panel displays the user's performance, showcasing all their hits, early and late, their highest combo (number of hits without missing), accuracy and score.

By providing clear feedback, users are able to monitor their performance. This encourages repeated engagement, as users aim to improve previous attempts. Performance metrics help motivate users by showing their progress and encouraging improvement (Deci & Ryan, 2000).

Presenting all relevant information in a single panel aligns with usability principles such as visibility of system status (Nielsen, 1994).

6.10 Course Content

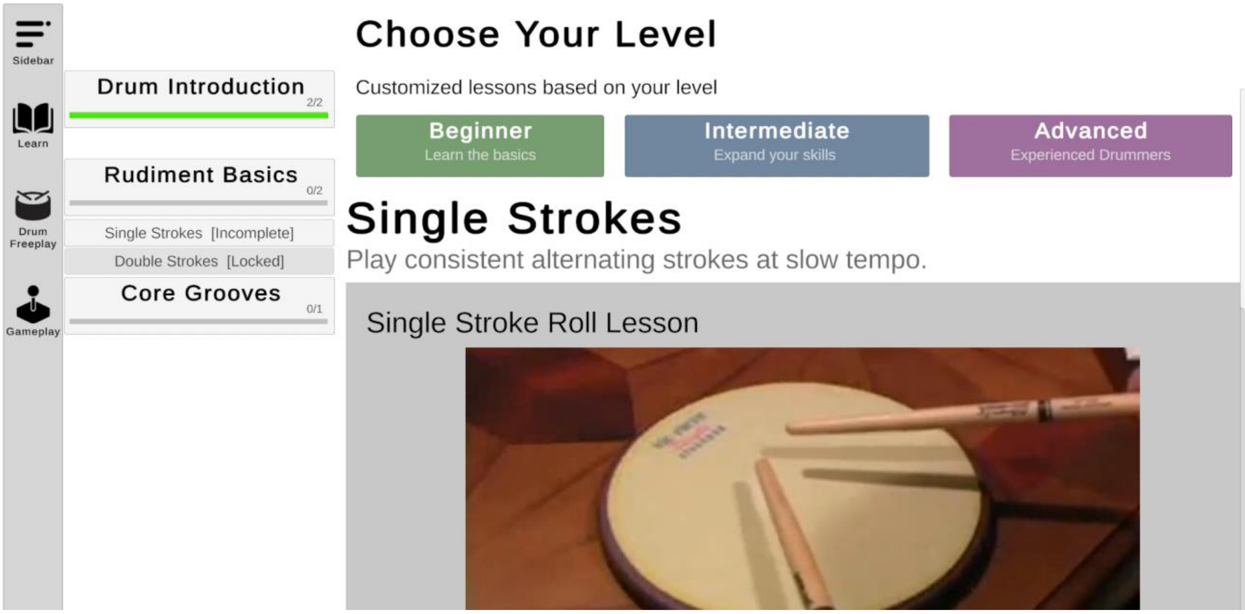

Figure 20 Course Page

The UI for the instructional content follows the design of Drumeo and LinkedInLearning. The content is organised according to skill level, including Beginner, Intermediate, and Advanced. A sidebar is used to quickly navigate between modules within the selected course. Each module includes a progress indicator and requires the completion of a lesson before progressing to the next, supporting principles of mastery learning (Schmidt & Lee, 2011).

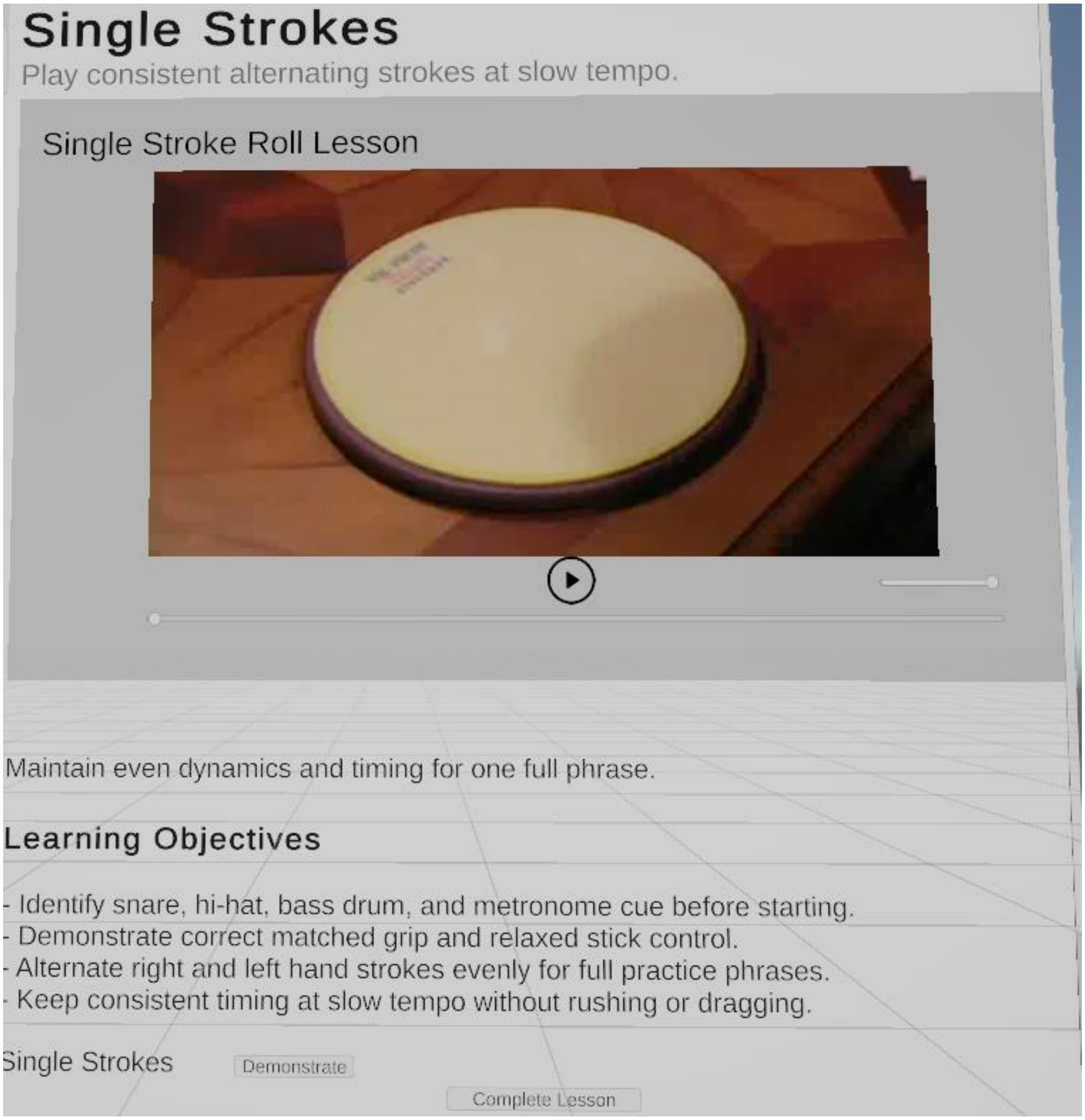

Figure 21 Full Lesson View

The lesson interface contains instructional videos and learning objectives. The video is positioned at the top of the page in line with usability guidelines (Nielsen 1994), which recommend prioritizing key content so that it is immediately visible without scrolling.

Both text and video are included, as users often prefer to scan written content before engaging with video material (Harley, 2020).

6.11 Navigation

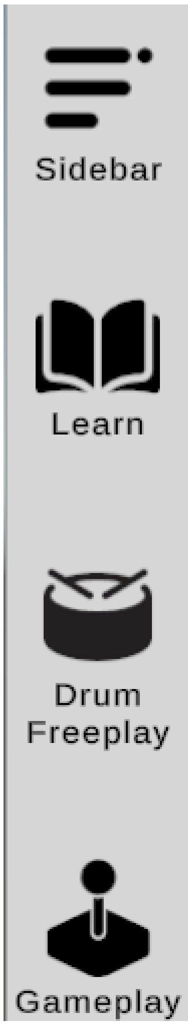

Figure 22 Sidebar Navigation

The sidebar navigation uses a combination of icons and text labels, including Learn, Free Play, and Gameplay. This design supports the usability heuristic of recognition over recall by allowing users to quickly identify functions through familiar icons, reinforced by text labels (Nielsen, 1994).

The use of both icons and text improves clarity and reduces uncertainty for new users who may not immediately recognize icon meanings. This aligns with accessibility and usability best practices.

A sidebar layout is used to provide consistent and persistent navigation across the application, supporting user control and freedom by allowing users to switch between sections at any time.

The fixed positioning of the sidebar enables users to develop familiarity with the interface layout.

A toggle button is included to collapse or expand the sidebar, allowing users to adjust the interface based on their preference. This supports flexibility of use and helps optimise screen space, particularly during gameplay where visual focus is important.

7 Implementation

This section documents the process of developing the application, using multiple plugins and libraries into a unified app and connecting the electronic drum kit for MIDI and sounds.

Much of the code in this project was produced using AI-assisted development tools such as GitHub Copilot which was built-in to Visual Studio Code. Copilot was used as it can use files within the project for context, and it was available with the GitHub student pack. I defined the project concept, system architecture, and functional requirements, and was responsible for refining, validating, and integrating the generated code into a system (Appendix B).

7.1 Initial Idea

The initial idea was to use a Raspberry Pi with external piezo sensors attached to the drum kit to detect hits and trigger LED responses. However, the drum kit already contains built-in piezo sensors used for MIDI detection, which simplifies the system and reduces latency. This removes the need for additional hardware, resulting in a direct signal chain from the electronic drum kit (MIDI output) to the computer.

An Optical Music Recognition (OMR) software Audiveris, was used to convert scanned sheet music into an XML file.

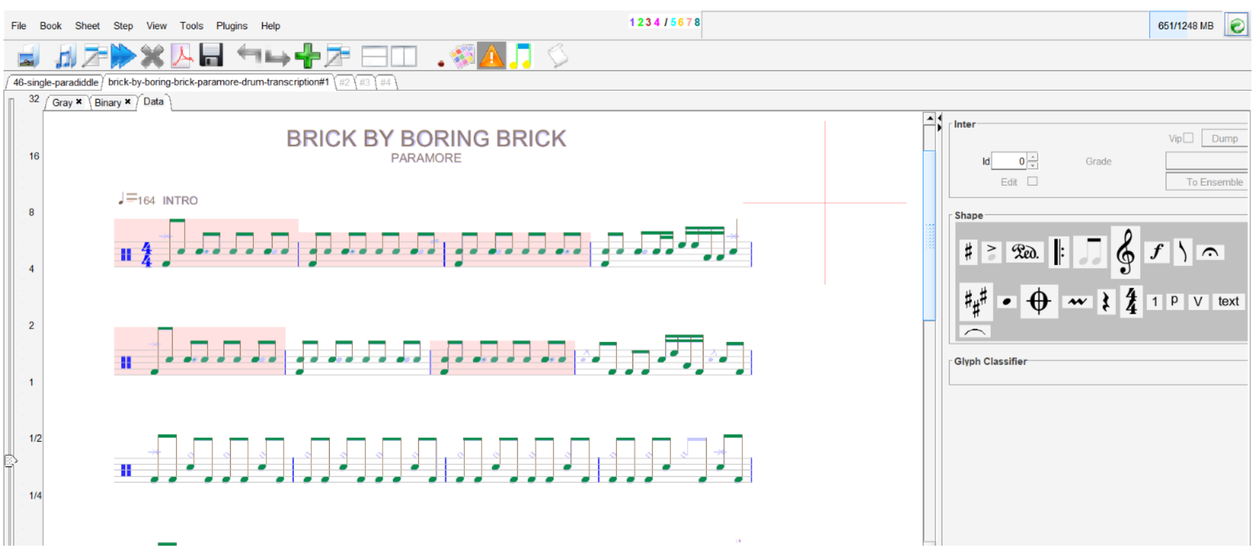

Figure 23 Audiveris

The exported XML file resulted in poor note detection as Audiveris lacked the ability to read drum notation and specialises in piano/orchestral notation. Drum notation is unpitched (percussion clef) meaning it depends on position/instrument mapping, not pitch.

Music21 was used instead as it is used in other academic papers (Cuthbert, 2010) and perfectly fits this project. Audiveris deals with symbols, while music21 deals with semantics (what the music means). Another benefit of using music21 is the various files it accepts. Audiveris only works with clean score music, while music21 accepts Midi, XML, and datasets, making it more flexible.

Since Audiveris was no longer usable, an alternative option was an auto-transcriber [Drum2Notes] that exports as MIDI or XML. This is a paid auto transcriber that uses AI to analyse imported audio files and transcribes the selected instrument. This app was on sale, making it affordable.

Although Drum2Notes was able to export as MIDI, the notes in the MIDI file were often not on time and the bpm detection is often off by a few beats. This causes the notes to be on time at first, as the song continues playing the notes start to fall behind, causing the notes to not sync with the song.

7.2 Connecting electronic drum kit

The next step was connecting the electronic drum kit to detect MIDI inputs. Most electronic instruments can be connected to a computer using a MIDI to USB cable. Next was testing if the computer could detect MIDI inputs.

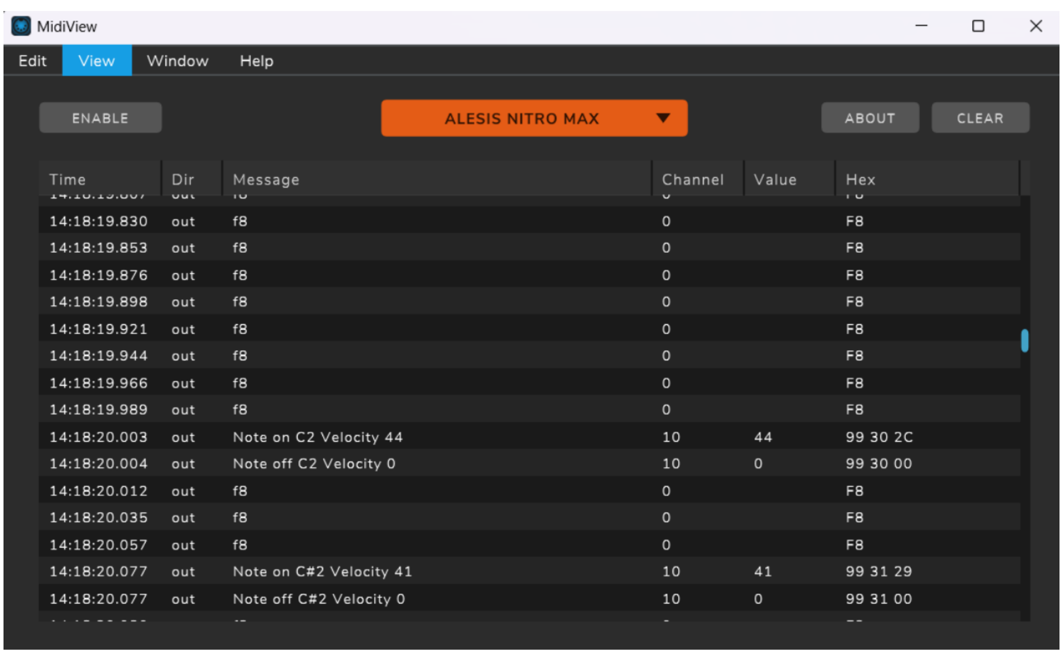

Figure 24 MidiView

Software to monitor MIDI inputs was used to detect inputs. This displayed the specific note, time, and velocity of each hit. These will be used when developing to calculate the hit timings and specific notes.

7.3 Connecting electronic drum kit to Unity

MidiJack is a MIDI input plugin for Unity. Having a MIDI controller built into Unity reduces latency between the electronic drum kit and the app, reducing latency is vital for instant feedback and accurate timings.

To test MidiJack, a small script was created to create a circle when a note was hit.

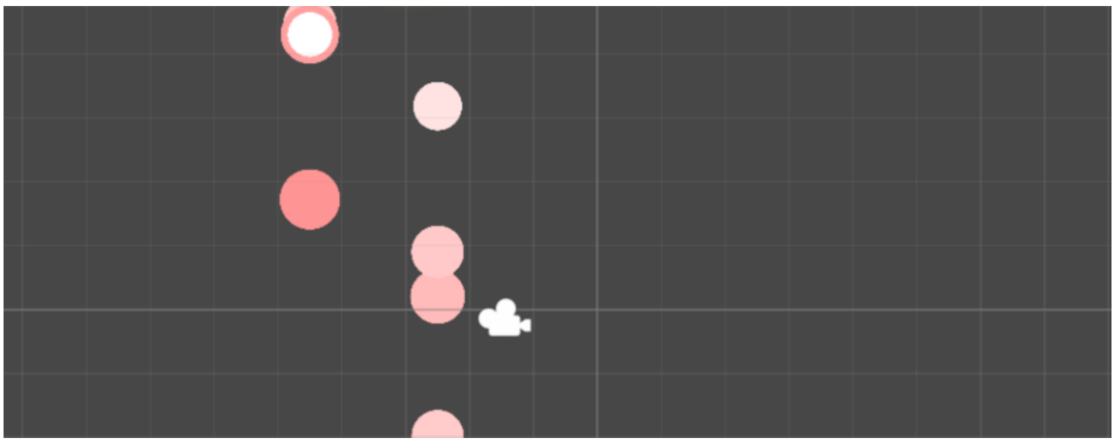

Figure 25 MIDI Input Test

When a MIDI input is detected, a circle prefab is instantiated which creates a new circle object. This was first tested on a 2d project for quick testing.

7.4 Audio

A huge part of playing the drums is being able to hear your own playing and the song track. The drum module from the electronic kit, which includes all the drum sounds, was unable to be heard from the computer. An initial approach involved using downloaded audio samples of drum hits with varying dynamics to simulate performance intensity. However, this was later replaced following feedback from the supervisor, who recommended using dedicated Virtual Studio Technology (VST) software for improved realism and integration.

A search for suitable software led to the official product page for the drum kit, where it was discovered that it includes Virtual Sound Technology (VST) software, specifically BFD Player. BFD Player provides realistic, high-quality drum sounds and can be connected to the drum kit via a MIDI-to-USB. Although it does not use the same sounds as the drum module, the audio quality and dynamic response were found to be significantly improved.

The main challenge was connecting the drum kit to both Unity and BFD Player simultaneously. It was identified that MIDI devices can be restricted to a single application at a time. To address this, the MIDI signal was split into multiple virtual ports, allowing it to be routed to both applications concurrently.

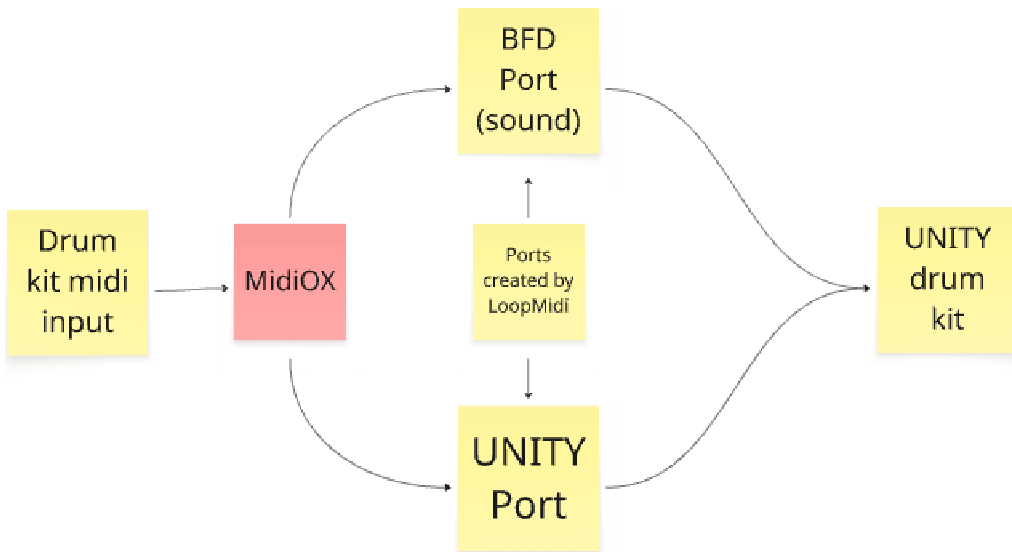

Figure 26 MIDI Data Flow

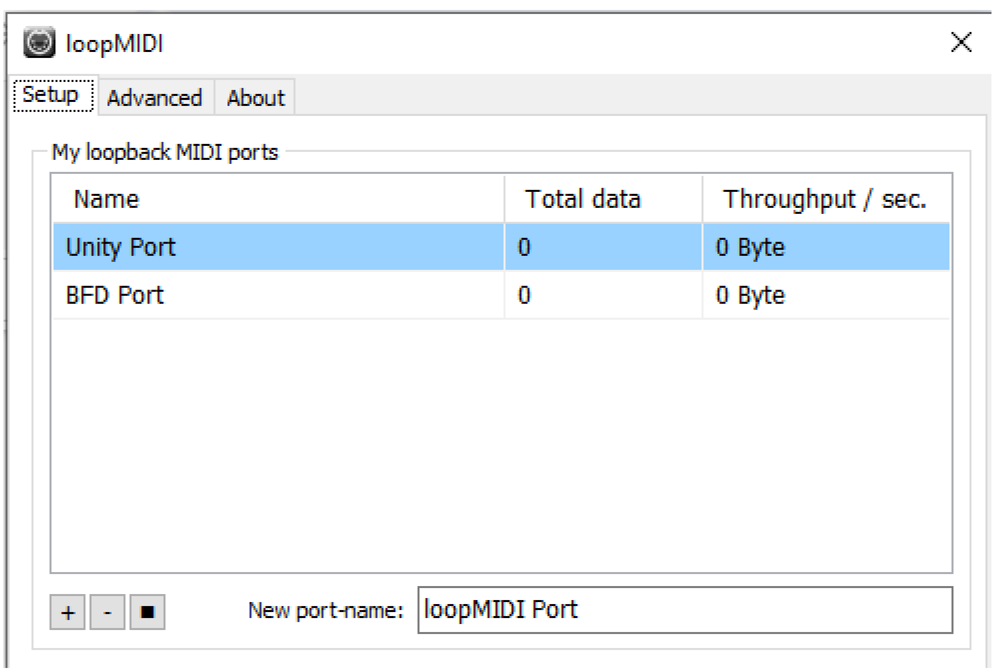

Figure 27 LoopMIDI

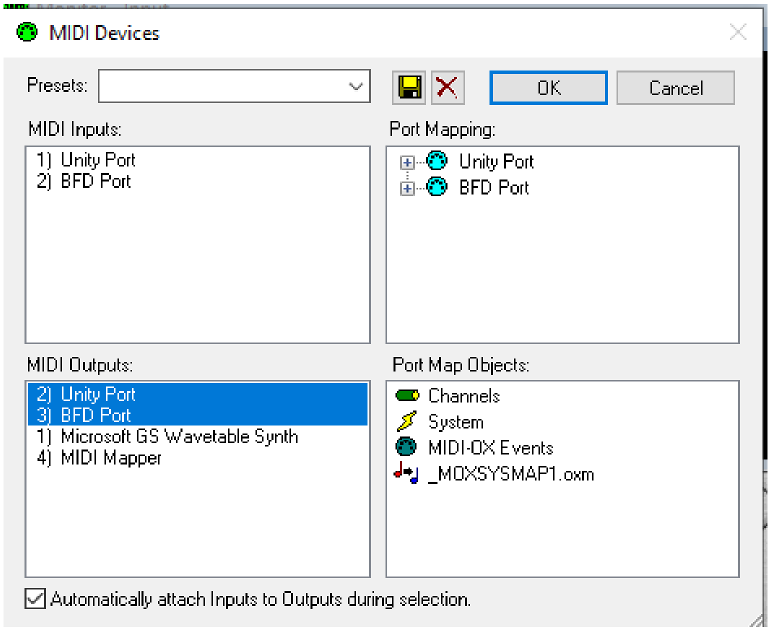

Figure 28 MIDI-OX

MIDI-OX is used to route the drums to each of those ports.

LoopMIDI is used to create more ports for BFD Player and Unity.

BFD Player can choose which port to use.

Unity then automatically uses the port that is available, the Unity Port.

With all components combined, the drum kit in Unity reacts to the physical drum kit which simultaneously flashes the kit on hit and produces the audio.

7.5 Visualizing the drum kit in Unity

Figure 29 Drumkit Asset

A 3D drum asset sourced from Sketchfab, created by AssetFactory, was used to represent the drum kit. The model is a low-poly representation of a generic drum set, which was selected to reduce performance load.

7.6 Importing Beat maps

To generate the notes that fall onto the drum kit, beatmaps must first be parsed and imported. This process begins by acquiring both a MIDI file and a corresponding MP3 audio file. The MIDI file can be generated using Drum2Notes, which requires the MP3 file as input for transcription.

Both MIDI and MP3 files are then imported into Unity. The MIDI file is processed using Music21, which parses the data and converts it into a structured JSON format. This JSON file contains key information such as note spawn time, target hit time and assigned lane (drum note).

The JSON file is then used within Unity to instantiate the note objects at the correct time and drum note, synchronized with the audio track.

7.7 Hit Detection

(5.2) Pad MIDI Note Numbers

Trigger	MIDI Note Number
Kick	36
Snare	38
Snare Rim	40
Tom 1	48
Tom 1 Rim	50
Tom 2	45
Tom 2 Rim	47
Tom 3	43
Tom 3 Rim	58
Tom 4	41
Tom 4 Rim	39

Trigger	MIDI Note Number
Ride	51
Crash 1	49
Crash 2	57
Hi-Hat Open	46
Hi-Hat Half-Open	23
Hi-Hat Closed	42
Hi-Hat Pedal	44
HH Splash	21

Figure 30 Alesis Nitro Max Pad MIDI Numbers

Each note on the drums is linked to the MIDI note on the physical drum kit. Each drum pad has a black cover that has their corresponding colour to light up when the MIDI note is detected.

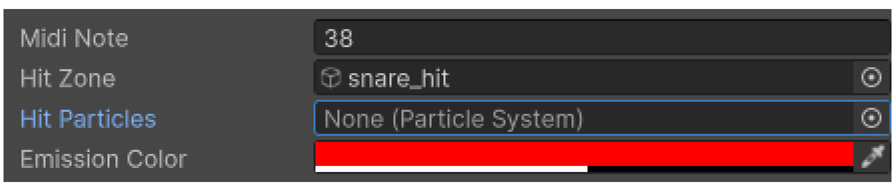

Figure 31 Unity Input Field

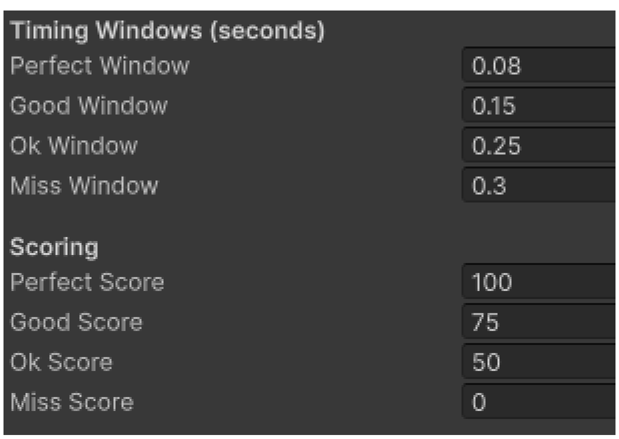

Figure 32 Unity Threshold Input Field

The timing windows are the threshold timing of each note. The MIDI data outputs the timestamp of each hit, which is then used for the gameplay mode. By getting the difference between the note timestamp and hit timestamp, it determines the grade of each hit.

7.8 Trimming

Trimming the beatmap requires the user to input a start and end time using an input field or the audio waveform. The audio and notes outside of the selected field are deleted and creates a new beatmap. The trimmed beatmaps are grouped with the original beatmap.

7.9 Metronome and Tempo Control

The metronome and tempo control are synced together. The initial bpm is taken from the selected beatmaps JSON file. The tempo ranges from double or quarter of the original bpm. Changing the bpm alters the speed of the notes, audio track, and metronome.

7.10 Course Content


```json
"courses": [
  {
    "id": "Beginner",
    "title": "Drum Foundations",
    "description": "Build timing, coordination, and groove fundamentals.",
    "difficulty": "Beginner",
    "modules": [
      {
        "id": "module-drum-introduction",
        "title": "Drum Introduction",
        "description": "Get comfortable with the drum kit layout and foundationa
        "lessons": [
          {
            "id": "lesson-holding-drumsticks",
            "title": "How to Hold Drumsticks",
            "description": "Learn matched grip, fulcrum control, and relaxed mot
            "objective": "Establish correct hand position and controlled rebounc
            "learningObjectives": [
              "Identify proper thumb and index fulcrum placement.",
              "Demonstrate relaxed matched grip without excess tension.",
              "Control rebound with even stick height in both hands.",
              "Maintain balanced posture and wrist motion while striking."
            ],
            "learningVideos": [
              {
                "title": "Learn Matched and Traditional Grip",
                "videoFilePath": "Videos/Rudiments/drumstick.mp4",
                "youtubeUrl": "https://www.youtube.com/watch?v=KdlKH0BhRsU"
              }
            ]
```


Figure 33 Course JSON

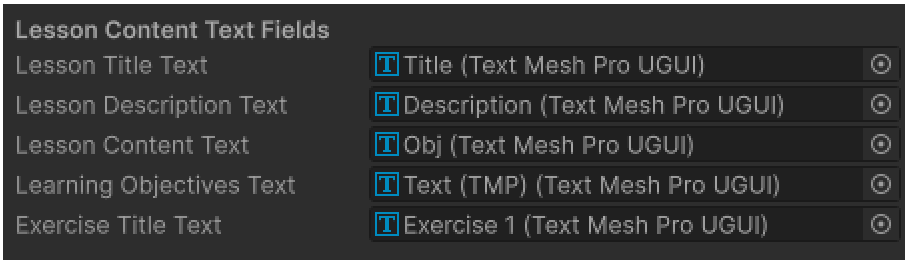

Figure 34 Unity Text Input Field

The contents of the course are stored in a JSON file. The titles of each course, module, and lesson can be dragged into Unity's text fields, displaying them. Basic styling and layouts are used to fit the contents into a single page. Buttons are used to interact between different modules and courses.

7.11 Difficulties faced

7.11.1 Windows Midi Service Blocking Unity

During development, an issue was found where Unity would freeze when receiving MIDI input from the drum kit. Troubleshooting involved disabling game objects and reviewing scripts. However, the issue persisted across multiple projects, this rules out Unity as the issue.

Checking the data flow confirmed that MIDI routing via loopMIDI and MIDI-OX was correctly configured and unchanged from previous working states. Windows task manager revealed that a Windows MIDI Service process was active.

Research indicated that a recent Windows 11 update (February/March 2026), added a Windows MIDI Service. This caused conflicts with virtual MIDI ports, resulting in devices becoming unresponsive or undetectable in Unity.

The issue was resolved by rolling back the Windows update, which disabled the new MIDI service. MIDI input functionality was restored, confirming that the operating system update was the source of the problem.

7.11.2 Displaying Course Content

While implementing the text within Unity, text and assets appeared blurry. The supervisor questioned Unity's ability to display large text content as it was difficult to read.

Figure 35 Unity Pixelated Text

Searching through Unity's settings revealed that the display scaling was default to 1.3x. This caused assets within Unity to be pixelated, changing the scaling to 1x fixed the pixelation issue.

7.11.3 Review Performance

In the requirements chapter, reviewing previous performance was a feature. This was not implemented because audio was routed through BFD Player rather than Unity, preventing Unity from capturing the drum audio.

7.11.4 Syncing Audio with the Drums

When syncing audio with the drums, some MIDI files have different start times. Currently the component assumes the MIDI notes start at 0:00. For example, the drums in Everlong by Foo Fighters start at 0:13, If the MIDI note starts at 0:00 then the offset needs to be set 0:13 into the song. If the MIDI note start at 0:05 then the offset needs to be at 0:08. This could be difficult to sync if the user has no knowledge of the MIDI files first note time.

8 Testing

Testing was conducted to evaluate the functionality, usability, and performance of the system following the implementation of core features, including MIDI hit detection, beatmap importing, and audio trimming. The primary aim of testing was to assess whether the system met its functional requirements. This aligns with design principles, where continuous evaluation is used to refine usability and interaction design (Nielsen, 1994).

8.1 User Testing

User testing sessions were carried out with participants interacting with key features of the system. The focus was on identifying usability issues, interaction difficulties, and accuracy of system feedback. The table summarizes the key findings.

	Feature	Feedback	Expected Result
T1	Importing and syncing Beatmap	Ui was unclear. Timing was off on some songs. Marked location unclear	Audio Track and notes on-time
T2	Trimming	The handlebars were difficult to click and drag.	Trimmed audio correctly trimmed with the updated notes
T3	Midi hit detection	Hit pads were too bright, notes were difficult to read when there was high density of notes. Thresholds were overlapping with other notes	Hit detection on time, display different thresholds correctly

8.2 Improvements

Based on the feedback from both users and the project supervisor, several design improvements were implemented in the latest iterations.

To improve clarity and usability:

- Modal panels were introduced for audio syncing and trimming to focus user attention and reduce cognitive load

- An audio waveform was added to support visual recognition of timing, improving accuracy in synchronization tasks
- Input fields were implemented to allow precise control alongside drag-based interaction

To address visual feedback issues:

- Hit pad brightness was reduced to improve readability and minimise visual strain
- Timing thresholds were refined to prevent overlap and improve accuracy of feedback
- UI elements were reorganised, with clear titles added to each component to improve visibility of system status (Nielsen, 1994)

Additional usability enhancements included:

- Loading indicators to communicate system processing
- Error messages to prevent invalid actions
- Improved layout consistency across components

9 Project Management

9.1 Methodology

This project followed the Agile Development Methodology. Agile was selected due to the iterative nature of the project, where features such as MIDI integration and audio handling required continuous testing and refinement. This methodology offers flexibility by breaking the project into small sprints. Sprints are a short period usually 1-2 weeks, to complete a specific task. Sprints are very similar to progressive learning, where a larger subject is broken down into many smaller subjects. This improves productivity and reduces cognitive load.

Regular meetings with the assigned supervisor will act as Sprint Reviews, ensuring consistent progress is demonstrated.

A Kanban Board will be used to track progress on Miro. This separates small tasks into different stages (Backlog, In Progress, Done).

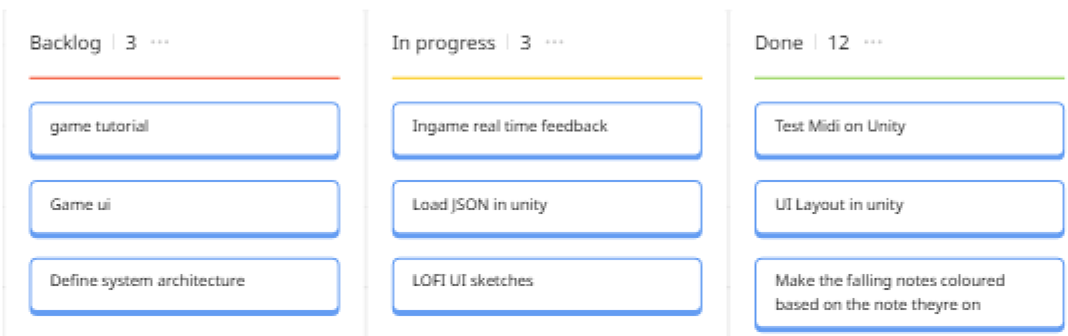

Figure 36 Kanban Board

A timeline in Miro was used to display the sprints broken down into weekly periods

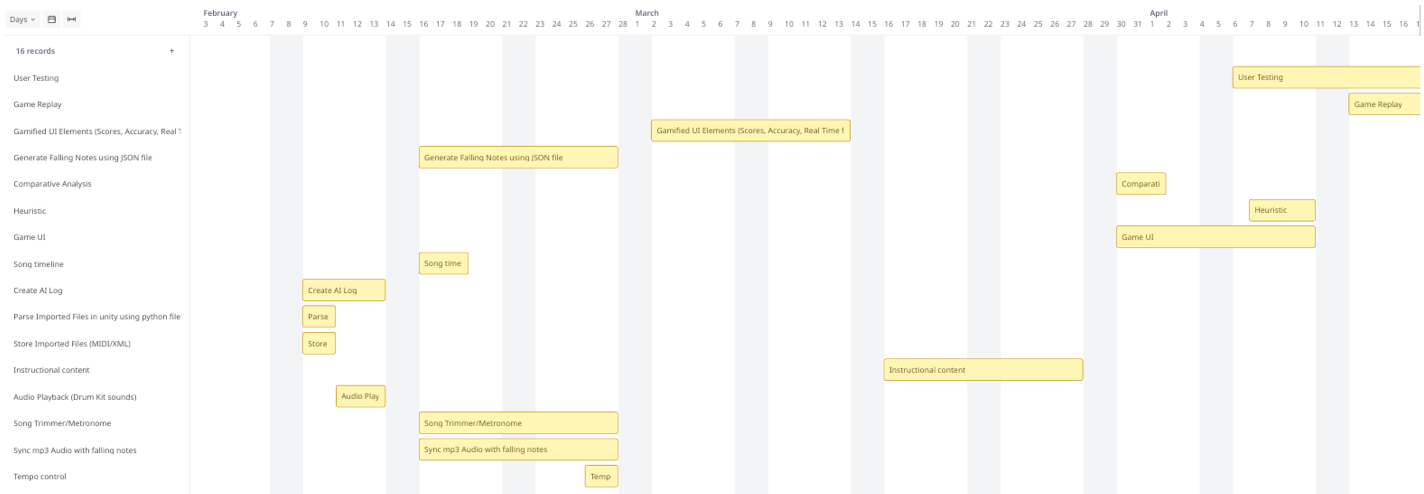

Figure 37 Project Timeline

Version control will be handled using GitHub, with regular commits to document development progress.

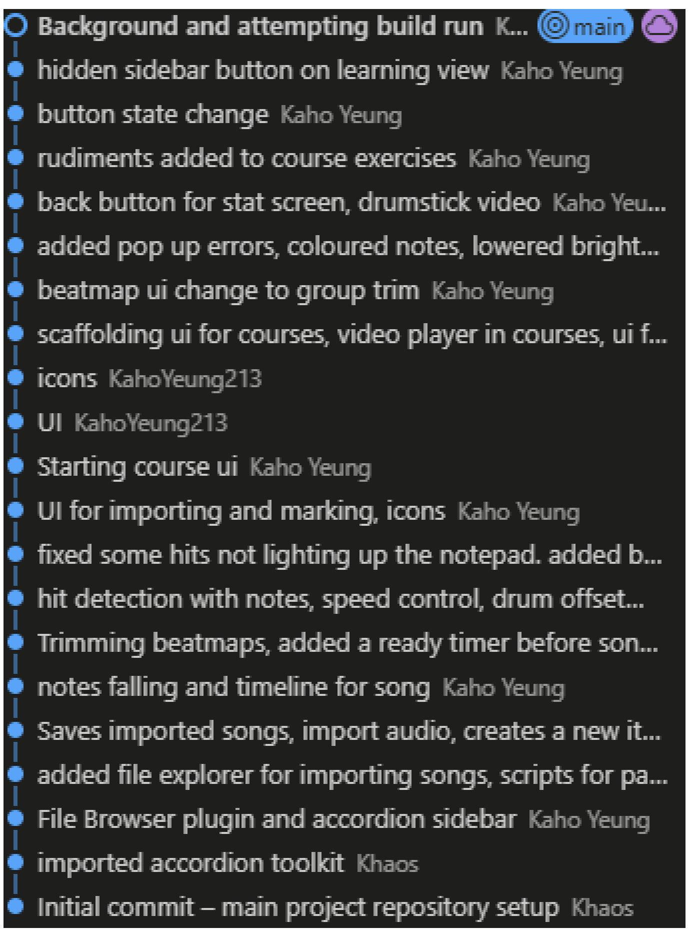

Figure 38 Git Commits

10 Conclusion

This project set out to design and develop a gamified drum-learning system that supports beginner drummers in developing rhythmic accuracy and coordination. By combining principles from motor learning, pedagogy, and usability design, the application provides a structured environment that encourages skill development through practice and feedback. The findings from both the research and implementation suggests this approach addresses limitations in traditional drum instruction.

Traditional learning methods, which rely heavily on notation and passive video content, often fail to provide the immediate feedback, engagement, and structured progression required for effective motor skill development. As highlighted in motor learning research, the development of coordination and timing depends on continuous sensorimotor feedback and repetition (Repp, 2005; Schmidt & Lee, 2011). The system developed in this project directly addresses these limitations by integrating real-time MIDI input, visual feedback, and gamified performance metrics such as accuracy, score, and timing evaluation. This aligns with research indicating that immediate feedback accelerates skill acquisition and supports motor patterns (Repp, 2005).

The use of gamification further strengthens learner engagement and motivation. By incorporating elements such as progress tracking and scoring systems, the system reflects principles of self-determination theory, which emphasizes the importance of competence, autonomy, and feedback in sustaining motivation (Deci & Ryan, 2000; Hamari et al., 2014). These features are particularly important for beginner drummers, who may otherwise find repetitive practice disengaging. The ability to import songs, adjust tempo, and segment songs into manageable sections also supports deliberate practice, which is essential for developing musical skill (Maxfield, 2018).

Instructional design principles shaped the learning experience. The system incorporates scaffolding through structured course content, gradual progression from simple to complex rhythms, and mastery-based advancement. This mirrors existing educational platforms such as Drumeo and LinkedIn Learning, which use chunking and progressive difficulty to reduce cognitive load and improve retention. The use of visual and auditory learning, through a visual drum kit and synchronized audio feedback, supports multimodal learning and aligns with Gibson’s (1979) affordance theory and conceptual models (Norman, 1988). This is particularly beneficial for beginners and learners who struggle with traditional notation-based approaches.

The project demonstrates successful integration of multiple technologies, including MIDI input handling using MidiJack within Unity, beatmap parsing using Music21, audio processing through

Drum2Notes and audio output through BFD Player. These components work together to deliver synchronised gameplay and performance feedback, meeting the core functional requirements of the project.

An iterative development approach was used throughout the project, supported by AI-assisted tools such as GitHub Copilot. This approach enabled rapid prototyping and problem-solving. Through this process, a deeper understanding of system integration was developed.

Despite completing the primary objectives, several limitations remain. Certain planned features were not fully implemented due to time and technical constraints, and issues such as MIDI timing inconsistencies required workarounds rather than complete solutions. The reliance on external tools for audio and MIDI processing introduces dependencies that may affect portability and scaling.

Future work may include training my own AI model specifically for this project as Drum2Notes also transcribes other instruments and doesn't focus on the drums, this causes some transcriptions to be slower than the original, inaccuracies and missing notes such as open high-hat. The course content can be expanded to include more fundamental skills and more advanced modules, perhaps migrating it to a React WebView which can then be embedded into Unity would be beneficial as React has been shown to be suitable for instructional content (Mark, 2025). Recording the audio of your own playing would require Unity to produce the drum audio instead of using BDF Player. Other ways of recording drum audio is using a software that records desktop audio.

In conclusion, the project demonstrates the potential of combining gamification and physical technology to support drum learning. It highlights both the technical challenges and opportunities involved in developing real-time music applications, while also contributing to an understanding of how digital tools can enhance skill-based learning experiences.

11 References

Baranwal, S. (2019, November 4).
What is JSON? Why to use it?
https://dev.to/delta456/what-is-json-why-to-use-it-1el8

Bouse, R. (2016).
Drum stick grips. Renjamin Bouse's Blog.
Available at: https://renjaminbouse.wordpress.com/2016/07/12/traditional-grip-is-a-sensitive-subject/
(Accessed: 9 December 2025).

CMG Worldwide (2023).
Buddy Rich: A World Class Drummer.
Available at: https://www.cmgworldwide.com/2023/03/23/buddy-rich-a-world-class-drummer/

Cuthbert, M. S., & Ariza, C. (2010).
Music21: A toolkit for computer-aided musicology.
International Symposium on Music Information Retrieval (ISMIR 2010).
https://dspace.mit.edu/handle/1721.1/84963

Deci, E. L., & Ryan, R. M. (2000).
The "what" and "why" of goal pursuits: Human needs and the self-determination of behavior.
Psychological Inquiry, 11(4), 227–268.

Editorial, T. (2025, August 18).
What is pedagogy?
Tes Magazine.
https://www.tes.com/magazine/teaching-learning/general/what-is-pedagogy

Evans, L. (2025, April 16).
What is instructional design? [5 examples + overview].
University of San Diego Online Degrees.
https://onlinedegrees.sandiego.edu/what-is-instructional-design-examples

Fessenden, T. (2017, April 23).
Modal & nonmodal dialogs: When (& when not) to use them.
Nielsen Norman Group.
https://www.nngroup.com/articles/modal-nonmodal-dialog/#toc-disadvantages-of-modal-dialogs-1

Gibson, J. J. (1979).
The ecological approach to visual perception.
Houghton Mifflin.

Hamari, J., Koivisto, J., & Sarsa, H. (2014).
Does gamification work? A literature review of empirical studies on gamification.
Proceedings of the 47th Hawaii International Conference on System Sciences, 3025–3034.

Harley, A. (2020, January 17).
Videos as instructional content: User behaviors and UX guidelines.
Nielsen Norman Group.
https://www.nngroup.com/articles/instructional-video-guidelines/

Keeler, K. R. Jr. (2020).
Video games in music education: The impact of video games on rhythmic performance.
Visions of Research in Music Education, 37, Article 5.
Available at: https://opencommons.uconn.edu/vrme/**vol37/iss1/5**

Mark. (2025, April 8).
Advantages of using react.js for developing engaging educational applications.
ArticleCube.
https://www.articlecube.com/advantages-using-reactjs-developing-engaging-educational-applications

Muttaqin, M., Zuhdi, H., & Ridwan. (2025).
The use of gamification-based Duolingo application in increasing student motivation is reviewed from the theory of self-determination.
AL-WIJDAN: Journal of Islamic Education Studies, 10(1), 81–95.
http://ejournal.uniramalang.ac.id/index.php/alwijdan

Newsome, E. (2016).
The Flow Music Method: Optimal experience tips and strategies for musicians and music teachers.
Queensland Conservatorium Griffith University, Australia.

Nielsen, J. (1994, April 24).
10 usability heuristics for user interface design.
Nielsen Norman Group.
https://www.nngroup.com/articles/ten-usability-heuristics/

Norman, D. A. (1988).
The psychology of everyday things.
Basic Books.

O’Connor, M. (2012).
20 Points of Creativity for The O’Connor Method Book One.

Pesek, M., Hirci, N. K., & Marolt, M. (2024, May 25).
Enhancing music rhythmic perception and performance with a VR game - virtual reality.
SpringerLink.
https://link.springer.com/article/10.1007/s10055-024-01014-y

Repp, B. H. (2005).
Sensorimotor synchronization: A review of the tapping literature.
Psychonomic Bulletin & Review, 12(6), 969–992.

Schmidt, R. A., & Lee, T. D. (2011).
Motor control and learning: A **behavioral emphasis (5th ed**.).
Human Kinetics.

Strain, J. A. (2002).
The evolution of snare drum grips.
Percussive Notes, June 2002, Vol. 66.

Wang, H.-H. (2023, July 30).
Accordions on desktop: When and how to use.
Nielsen Norman Group.
https://www.nngroup.com/articles/accordions-on-desktop/

Winters, C. (2013, January 13).
Models and Patterns in Jazz Improvisation.
Improv Pathways.
https://www.improvpathways.com/uploads/1/6/2/6/16262424/research_on_modeling_and_patterns.pdf

11.1 Figures

12 Appendices

12.1 Appendix A - Summary of interview with Kevin Brady

- First skills beginners should master
    - **Proper grip** (match or traditional)
    - **basics of rhythm** - how it is subdivided
      quarter notes - 1 2 3 4
      eighth notes - 1 & 2 & 3 & 4
      sixteenth notes - 1 e & a 2 e & a 3 e & a 4
    - **Basic rudiments**:
      single stroke
      double stroke
      paradiddle
    - Coordination using simple grooves
      quarter note groove
      then work up to eighth note grooves then to sixteenths
    -
- Mistakes beginners commonly make
    - Holding sticks wrong
    - habit of not listening to what they are actually playing, advises them to **listen back to their own playing** by recording
- What helps students practice at home
    - **ability to slow down the song**
    - memorizing rhythm
- What do students find boring/hard
    - **rudiments**, show how they can **use rudiments in fills, patterns and styles**
    - sight reading
- What skills need the most repetitions
    - **Rudiments** - always have to be on point, the hands can't let you down
    - more advanced - polyrhythms, odd timings e.g 7/8, 5/8
- What forms of feedback do you give students
    - Its a slow burn learning something new so **give them confidence**, if they do something correctly give them **immediate feedback**
    - **simplify and slow it down** if they're struggling
- How important is playing along to music
    - **100% hugely important**
      develops their ear, sense of timing and groove
- What overloads beginners the most
    - getting frustrated when they cant play on/off beat movements

    - when playing at speeds they try too hard to play at speed when they should **slow it down** to see the coordination and the muscle memory kicks in
- uses **sheet music, ear, and demonstrating** to teach
    - sometimes uses **visual aid**, if they have learning difficulties like his student with dyslexia he struggles with reading notation so he focuses more on his aural skills more
      mentions although notation is really important you can get away with not learning it by developing the ear and memory to a high level and understanding the fundamentals
- **starts through 4 basic grooves**
  quarter note groove
  eighth note groove
  sixteenth note groove
  triplet groove

  e.g basic quarter note groove
  double kick to a snare you should be able to switch to 1/8 notes on the hh with 1/16 and triplets too builds sense of beat sub dividing
- audio visual part helps, used to teach online during covid using a **birds eye view** to get a better perspective

- **Limb Separation**
  start with quarter note patterns then elevate to eighth note patterns
- **Techniques**
    - **Syncopation** is everything for a drummer, usually starts eighth note patterns first, can be applied to many styles (maybe do this after skills beginners should master first)

12.2 Appendix B – Summary of AI Prompts

AI Model	Prompt	Output
Figma make ai	design a layout for a drum learning system using instructional design principles and music pedagogy, make different tabs for 1. lessons and	Lofi version

	learning, 2. gameplay/practice using the MUSIC model of motivation	
Claude 4.5	Remove audio mappings	Removed audio mappings. no longer using imported audio drum sounds. using VST drum sounds instead
Claude 4.5	i have a button that says import songs, i want to make the button open a panel ive made. in this panel there are 2 boxes where you can upload files. one for midi, one for mp3. in this panel there is a confirm and cancel button. i have text in the panel that is a link, i want this to be clickable	Import Songs button now opens your upload panel instead of immediately starting import. Added panel-level confirm and cancel behavior: Confirm starts import only when both MIDI and MP3/audio are selected. Cancel closes the panel and clears selected files. Moved readiness logic to the panel confirm button: Main Import Songs button stays usable. Confirm button is enabled only when both files are selected. Added clickable link support for TMP text in the panel: If your text uses TMP link tags, it opens the clicked link. If no link tag is clicked, it opens a fallback URL.
Claude 4.5	i want to store the imported files so they can be loaded later on. these files will be later used to load a beatmap	Persistent Storage: Files are stored using PlayerPrefs and game restarts Automatic Validation: Missing files are automatically removed on load Easy Integration: Simple API for adding and retrieving beatmaps Scene Independent: Singleton pattern with DontDestroyOnLoad Flexible: Store audio-only or complete audio+beatmap entries
Claude 4.5	i want it so you can import first the midi file and secondly the mp3 audio file. i also want to be able to display the	How It Works Import Flow: Click "Import MIDI" → Select .mid file → Stored with metadata from filename Click "Import Audio" → Select .mp3 file → Linked + reads ID3 tags for title/artist Generate Beatmap → MidiXmlImporter creates JSON View in Accordion → Shows "Artist - Title" with status icon Select & Play → Loads beatmap and audio

	beatmaps in an accordion which i already have. how do i make it so it reads the title/artist of the song and then it loads them into an accordion for the song/beatmap selection	
agent	summary of chat for implementing trimmer ready timer and syncing	Initial scope started on BeatmapSnipper UI timing/marker behavior, then rapidly pivoted to core rhythm timing architecture. User introduced a hard global rule: always include a 3-second ready timer for every beatmap, regardless of song offset. Major requirement conflict cycle focused on offset semantics: Whether offset affects audio only, notes only, or both Whether countdown is silent vs audio-playing Whether audio should start at offset, offset-3, or from song start Whether beatmap time should reset vs continue from offset point User repeatedly redefined the intended model using concrete scenario (offset=13): Audio should be forwarded to pre-drum section Ready timer must still exist Notes should land in sync with that delayed drum entry Another major pivot: preserve full note content vs skipping intro notes. One phase requested filtering early notes to preserve countdown behavior Later phase explicitly requested full JSON note stream from 0:00 with no skipping Simplification request reset the strategy: “Add 3 seconds of no notes when generating beatmap JSON” as baseline behavior. Final timing model iteration requested: User-entered offset becomes additional note-time basis Audio starts at offset-3 for readiness Then requested universal grace period behavior for no-offset songs too. Feature scope expanded beyond sync: Added requirement for real-time timeline seeking behavior

		so correct notes appear when scrubbing/jumping. Feature scope expanded again into gameplay systems: Implement hit-pad timing capture Compare hit timestamp vs note timestamp Compute timing delta for scoring/accuracy pipeline.
agent	summary for manual syncing of drums, gameplay note hit thresholds, metronome, tempo control	Offset automation requested User asked to remove manual drum offset entry and improve import flow. Strategy change after MIDI limitation User clarified MIDI starts at 0:00 and does not indicate real drum entry. Direction shifted from MIDI-based offset to interaction-based sync. Tap-to-sync UX implemented User chose tap-to-sync. Workflow moved to: preview audio, mark first drum onset, use marked timestamp as offset for import. Dense-hit detection issue raised User reported rapid-note sequences causing wrong note matching (early hit affecting later notes). Requirement became chronological hit handling for close note clusters. Real-time BPM/speed control requested User asked for runtime tempo control via slider/input that affects both audio playback and falling-note speed. BPM-centric control model requested User asked to align speed control with metronome using BPM input. Requirement became: change BPM value, have both metronome timing and gameplay speed follow it. Metronome-linked panel + metronome volume control requested User requested SpeedPanel visibility tied to metronome icon toggle. User also requested direct metronome volume adjustment control.
agent	summary for importing beatmaps/audio, selecting songs from an accordion UI, loading playback, and adding an interactive timeline	Added full beatmap playback layer with a new player controller in BeatmapPlayer.cs. Added an interactive song timeline system in TimelineUI.cs, including seek/scrub behavior and playback UI integration. Connected beatmap selection UI to playback so selecting an entry can load beatmap data and audio via BeatmapAccordionUI.cs. Updated note spawn initialization compatibility in Notes.cs to match current falling-note initialization flow. Iterated UI interaction model for accordion selection, then

		settled on explicit per-item Play-button wiring (instead of runtime auto-button creation) in BeatmapAccordionUI.cs. Final accordion behavior: each beatmap item is expected to expose a dedicated Play button path, and clicking it loads beatmap + audio into player flow via BeatmapAccordionUI.cs and
agent	lane/hitzone based note spawning was built, MIDI mappings were unified and expanded, beatmap deletion management was added, and beatmap/audio timing was redesigned	Core note generation system added Implemented a full NoteSpawner flow that spawns falling notes per lane/hitzone from beatmap timing data. Integrated lane-based spawning logic and hitzone mapping for drum gameplay. MIDI-to-lane mapping architecture standardized Unified mapping across Python beatmap conversion and Unity runtime systems. Added support for multiple MIDI notes mapping to single lanes (especially hi-hat variants). Extended tom mappings with extra MIDI alternatives (including 47 and 58) to reduce missing/wrong lane assignments. Beatmap management upgraded Added beatmap deletion support at both single-item and clear-all levels. Updated library/UI behavior to reflect deletions reliably through events and data refresh. Beatmap conversion timing model redesigned Python converter updated to support spawn lead plus global timing offset behavior. Introduced configurable pre-start timing offset concept (Osu-like lead-in approach) in generated beatmaps. Audio/time sync model revised for gameplay timing Note spawn timing shifted to track playback timeline behavior instead of raw scene time. Added metadata-driven timing fields (including audio offset in beatmap metadata) and connected them to runtime playback time interpretation. Data model expanded Beatmap metadata schema extended to carry timing offset information needed for synchronization behavior across import and playback layers.
agent	importing, trimming and syncing changes	1. Added and refined beatmap snipping so trimmed audio includes a built-in countdown while note timing stays aligned.

		2. Updated snip export logic to correctly convert imported beatmap time into audio file trim positions, preserving sync between audio and notes. 3. Introduced a new drag-and-drop import UI with separate drop zones for MIDI/MusicXML and audio files. 4. Extended import support so the MIDI side accepts .mid, .midi, .xml, and .musicxml, while the audio side accepts .mp3, .wav, and .ogg. 5. Wired the importer so clicking a drop box opens the appropriate file picker for that file type. 6. Kept the existing tap-to-sync preview import flow, with preview playback used to mark drum start before generating the beatmap.
agent	changes to suit new ui mainly for importing/syncing songs	Reworked the song import flow so the main import button opens a custom import panel instead of immediately launching file selection. Switched the upload boxes to click-only file pickers and removed drag-and-drop behavior entirely after repeated nested dialog issues. Restricted file picker filters so MIDI/XML and audio types are separated correctly, with only supported audio formats shown. Added a clickable link text handler in the import panel so TMP rich-text links can open URLs. For drum marking / preview: Added a waveform-based audio preview UI for the drum-mark step. The waveform is generated from the loaded audio clip and can be clicked or dragged to seek. Added a playhead overlay that moves with playback time. Reduced preview audio loudness so marking doesn't blast the user. Improved waveform rendering so loud/compressed songs don't produce a fully maxed waveform. Precise marking controls added later: Added a precise time input field for entering exact mark time. Added up/down nudge

		buttons for fine time adjustment. Added a marker image that sits on the marked waveform position.
agent	course system	Introduced a full course system architecture on top of the drum app: course data model, course loading, progression tracking, and course-specific UI flow. Added JSON-driven course content support with hierarchical structure for course, module, lesson, and exercise, plus a starter course dataset. Added persistent course progress and linear unlock logic so lessons/exercises can be gated by completion. Expanded app mode handling from a simple learning/gameplay toggle into a multi-mode model that includes dedicated course mode. Built a new scaffolding-style course interface controller that drives course selection, module navigation, lesson selection, exercise selection, and detail panel updates. Migrated module navigation toward accordion behavior instead of separate panel-only lesson switching, including module expand/collapse state handling. Added dedicated prefab routing for different list item types so course/module/lesson/exercise rows can use different UI prefabs. Added dynamic module progress display logic so module rows can reflect completion status based on finished lessons. Added runtime UI normalization for instantiated list items to reduce prefab-state inconsistency during clone creation. Added resilient path-based text binding controls for multiple item types so runtime titles can be mapped to custom prefab hierarchies.