SENG302 Project Concept

Ali's Scale Jar

12th August 2016

Our Mission

Once upon a time there was a little girl called Ali. She had a scale jar. The rest is history… We're going to build an tool for learning theory basics, learning harmony basics, ear training, reference, and much more. Features will include;

- Enter and maintain knowledge
- Hear notes, scales, chords, …
- Train yourself to recognise musical elements
- Review and manage your individual performance data
- Assessing and indicating expertise
- …

Roles/Personas:

We'll use various terms, some general and some more specific, in our stories. Examples include musician, beginner, improver, teacher, researcher, user (generic), … You may find it helpful to develop detailed example personas to help you identify relevant queries about stories. For example, "Mary is an 8 year old girl who has been learning piano for 3 months and ..." suggests clear bright GUIs with little text might be appropriate. OTOH, Dave is an 85 year old man and …" suggests large fonts are in order.

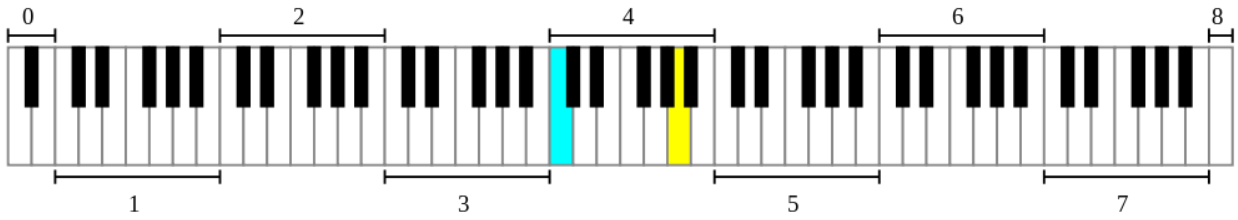
Domain background primer

Musical instruments come in many forms: sound may be produced by pressing a key (piano, organ, …), plucking or bowing a string (guitar, violin, …), blowing in a tube (recorder, saxophone, …), or just being there (theremin). Notes correspond to sound waves of a particular frequency (e.g. an orchestra typically tunes instruments so that the note A is at 440Hz). Actually, this A is sometimes referred to as A4 to distinguish it from other As with lower or higher pitches. Keyboard instruments, such as piano, and fretted string instruments, such as guitar (ignoring string bending) produce discrete notes. In the western context, successive notes are a

semitone apart. The terms “sharp” and “flat” refer to raising and lowering the frequency of a note.

Some instruments, such as violin, trombone or dobro, can produce sounds over continuous ranges — including “in between” notes. We will assume that we are dealing with notes which are discrete and a semitone apart: sharpening/flattening a note corresponds to moving one semitone to the next higher/lower note.

In this system, note names consist of a letter (A, B, C, D, E, F or G) and an optional “modifier.” When we get to G we just carry on with another ‘A’ and so on. The range of 8 note names from one A to the next is known as an octave. For many purposes, we don’t need to concern ourselves with the specifics of exactly which ‘A’s we’re considering. However, for other purposes (such as actually playing notes) the actual pitch does matter. For example, frequencies 440Hz, 880Hz and 220Hz each correspond to an ‘A’.



From the musician’s point of view, a ‘note’ is something like ‘middle C.’ Based on the ‘standard’ 88-note piano keyboard, middle C is in octave 4 and is denoted C4 accordingly. More information, including the image above, is available at https://en.wikipedia.org/wiki/Piano_key_frequencies. Middle C is the blue key and A4 the yellow key in the image above. Almost every modern piano has 52 white keys and 36 black keys for a total of 88 keys (seven octaves plus a bit, labelled from A0 to C8[1]). The three ‘A’s of the previous paragraph would be referred to as A4, A5, and A3 respectively if distinguishing them was necessary. Where an octave specifier is omitted, we’ll assume that it is 4, unless directed otherwise.

Starting from middle C (C4) on a piano keyboard the (conventionally) white notes are C4, D4 E4, F4, G4, A4, B4, C5, D5, and so on. To reach a (conventionally) black note, we can start on the nearest white note below it and go up a semitone: alternatively, we can start on the nearest white note above and go down a semitone. For example, starting on G and going up a semitone gives G sharp (G♯) while starting on A and going down a semitone gives A flat (♭). Thus the same note on a piano can have two equivalent (enharmonic) names.[2] We’ll meet other modifiers — such as double sharp (𝄪), double flat (𝄫) and natural (♮) — in due course.

[1] This is the convention in Scientific Pitch Notation. In some applications, including MIDI, middle C is sometimes called C5 (see story MIDI note names) but we will call it C4.
[2] This is sufficiently accurate for our purposes but be warned that a piano tuner will debate this at length!

For convenience, we'll use the common convention of using # and b to denote sharps and flats in plain text.

Similarly, on a fretted stringed instrument notes corresponding to each fret are a semitone apart. For example, the lowest string on a bass is (usually) tuned to E1: fret 3 on that string corresponds to G1, fret 4 to G# (and Ab) and fret 5 to A.

For completeness, we should note that the "white" notes can also come in sharpened and flattened forms. Sometimes we'll refer to F as E# because it is a semitone above E. This is called *enharmonic equivalence* and we'll encounter it in other contexts too. There is a good explanation of this, and some handy illustrations, at http://musicnotation.org/tutorials/enharmonic-equivalents/.

The 'distance' between two notes is called an *interval*. The interval between neighbouring keys on a piano, or neighbouring frets on a guitar, is a semitone and this is the smallest interval we'll consider. An interval of 2 semitones is (*quelle surprise!*) a *tone*. An interval of 12 semitones takes us to the next occurrence of the note we started on and is called an *octave*. This is a sufficiently special interval that the 12th fret of a guitar is likely to be decorated with inlaid materials. We'll encounter more intervals soon. You will probably find you actually know quite a lot about them already since many popular songs are characterised by particularly recognisable intervals[3]. See http://www.people.vcu.edu/~bhammel/theory/new_menu/resources/interval_songs.htm or http://www.easyeartraining.com/learn/interval-reference-songs-that-youve-actually-heard-of/ for examples.

The term "scale" is used to denote a sequence of notes, ordered by pitch, with a fundamental relationship. Each scale has a name which specifies the starting note, together with the relationships between the notes. These relationships are the intervals between the notes. For example, the intervals between the notes of a major scale are tone, tone, semitone, tone, tone, tone, semitone. The notes of the scale of C major are: C, D, E, F, G, A, B, C. The final note (another C) is an octave higher than the starting note: although only a single octave is shown here, the pattern is regarded as repeating. (Students typically start with C major scale because it only uses the white notes on a piano) Note that each of the 8 letter names appears: this is called "spelling." The scale of D major consists of the notes D, E, F#, G, A, B, C#, D — the spelling rules tell us that the 3rd note is F# not Gb. For some scales it will be necessary to use double flats and double sharps in order to spell them correctly. We'll mostly be dealing with *ascending* scales where the notes occur in order of increasing pitch — as for the C major and D

[3] For example the theme tune for *The Simpsons* begins with a 6-semitone interval: this famous interval has several names including 'tritone,' 'flat 5' and 'blue note'. The tritone is so grating that is it used for things like train horns where it is important to grab the listener's attention and Its association with evil and the devil has led to problems in the past.

major scales shown above. The descending major scale for C major would be C B, A, G, F, E, D, C.

Natural minor scales (we'll also meet other kind of minor) have intervals tone, semitone, tone, tone, semitone, tone, tone. Thus C natural minor ("C minor" for short) is C, D, Eb, F, G, Ab, Bb, C. If we take a particular major scale (say C major) and step down 3 semitones we get to A. Interestingly, the notes of the natural minor scale starting on A are the same as the major scale starting on C. Check it out! This relationship involves the *relative* minor and has a number of handy applications. For convenience, we'll often just say 'minor' when we mean 'natural minor' if the meaning is clear from the context.

Some instruments (e.g. trumpet) can only play one note at a time, but others (piano, guitar, …) don't have this limitation. The term *chord* is used for three or more notes played simultaneously. There are some fundamental relationships between chords and the scales from which their notes are taken. For example, the first, third and fifth notes of a scale (C, E, G for C major; C, Eb, G for C minor) constitute a characteristic chord. In general, if we take a scale and take every second note we can build up bigger and bigger chords: they can be described as "stacked" (major, minor) thirds.

After 2 octaves we get back to the starting note (e.g. C, E, G, B, D, F, A, C) and have collected the notes for a 'big' chord. The higher notes (in the second octave) correspond to intervals such as the 9th (D) rather than 2nd.

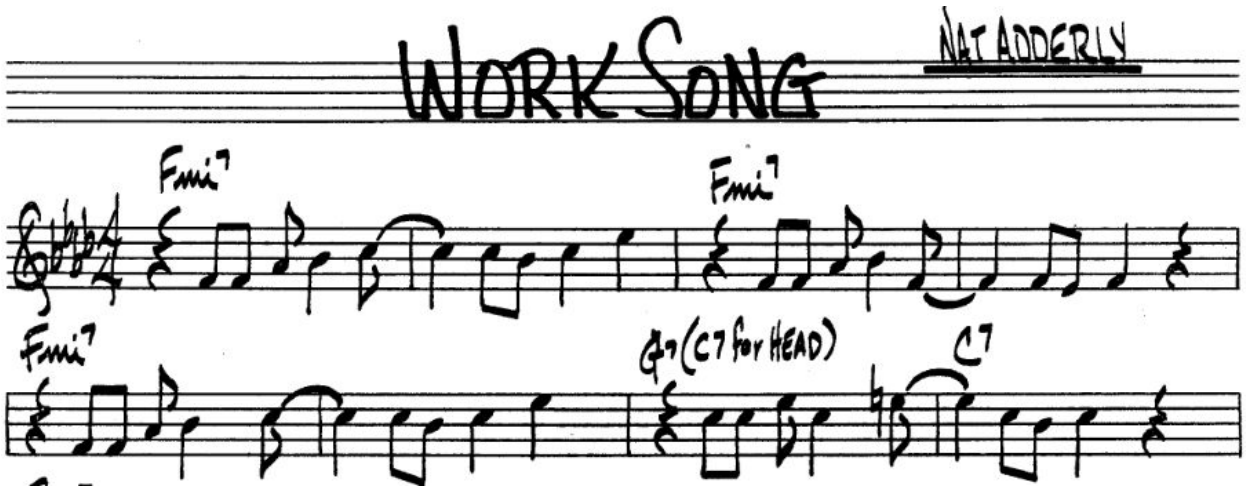
Grouping notes into "music" involves some additional concepts. There is a basic tempo (think clock speed) for counting time (tapping feet, headbanging, dance steps, …). Beats are grouped into "bars" with an underlying periodicity: when we waltz we think "one two three, one two three, …" but when we march we think "left right, left right, …". A *time signature* indicates both the number of beats in a bar and the duration of each beat. Common time signatures are 4/4 ("common time") and ¾ (waltz). See https://en.wikipedia.org/wiki/Time_signature for further detail.

Some musical forms involve grouping bars into common patterns. Examples include the "12 bar blues" and the 32 bar AB (2 x 16 bar sections) and AABA (4 x 4 bar sections) forms of many jazz standards.

Adding accents gives even more potential variation. Often the first beat of each bar will be emphasised (e.g. A waltz is really more like "ONE two three, ONE two three, …"). Emphasising beats 2 and 4 in common time leads to the familiar "backbeat" feel of rock & roll.

Written music includes a number of elements. We'll mainly be concerned with simplified documents called lead sheets. These are widely used in genres such as jazz, where musicians improvise extensively over the basic framework provided, and popular music, where a basic structure of units such as verse, chorus and bridge is set out.

Here's part of a "real" lead sheet:



Note that this one has more than just chord symbols: there's a treble clef, time signature, key signature (4 flats so it's F minor) and a stave with melody notes as well. Initially, we could just show the chords and bars like this:
Fm7 | Fm7 |Fm7 | Fm7 |
Fm7 | Fm7 | G7 | C77|

The 5-line stave (aka staff) is used to indicate notes and rests (for more detail on the notation see e.g. https://en.wikipedia.org/wiki/Note_value). Pitch is indicated by the position of the note head, which is either on a line or between lines (additional ledger lines are added if necessary for higher or lower notes). Duration is indicated by the detail of the note head and the "flag" attached. The first bar of *Work Song* starts with a crotchet rest, then has 2 F4 quavers followed by an Ab4 quaver, a Bb4 quaver and a C5 quaver "tied" to a C5 crotchet in the next bar. The additional notation details aren't important right now, but are included to give an indication of the sort of thing we would encounter if we were to develop a more sophisticated system --- in which case we'd probably make use of something like musicXML under the hood.

There are a number of very common chord progressions and you'll recognise some of them. The "amen" sung at the end of hymns is a IV - I; II - V - I progressions abound in jazz, and if you're resilient enough to sit through Heart and Soul then you'll have had enough I - VI - II -V to last you a lifetime...

We (by this stage of the course) will have seen the dominant (7) chord as the V chord in a major key. The (groan) dominant role of a V chord is to resolve to the corresponding I chord (this underlies the circle of fifths encountered in story *Key signatures (major)*). Thus we often see progressions like G7 CMaj7, Dm7 G7 CMaj7 and Em7 Am7 Dm7 G7 Cmaj7 in C major

pieces. In terms of their diatonic function, these are V I, II V I, and III VI II V I respectively. *Functionalising* in this way allows easy application to other keys. A *secondary dominant* is a ***dominant*** chord that is a 5th above another ***diatonic*** note. In other words, it resolves not to the I but to something else. For example, in C major the VI chord is Am7: changing it to A7 makes it a dominant a 5th above D (the V of the II). Changing the chord quality in this way helps instil a sense of "forward motion" in music. In a major key, any chord can be made a secondary dominant with the exception of the IV. It is not the 5th of a diatonic note (e.g. in C major the VII is B and the fifth above B is F#, not F).

Selecting the appropriate scale to play over a secondary dominants involves determining whether or not the chord it resolves to is minor. For those that resolve to minor chords the appropriate scale is Mixb2b6, the 5th mode of the harmonic minor scale. For the others, the mixolydian is used. For example, in C major III7 (E7) resolves to VI (Am7) so the correct scale is E mixolydian b2b6. The corresponding notes are E, F, G#, A, B, C, D come from the "main" C major scale with the exception of G# (the 3rd of the secondary dominant). This example appears in the fragment of the "All of Me" lead sheet in story *Lead sheet (basic)*.

We have now seen 7th chords in dominant (V) and secondary dominant (I7, II7, III7, VI7, VII7) roles. In both roles, the root of the chord is a diatonic scale note. Sometimes we'll encounter dominant 7th chords that have non-diatonic roots. These arise as the result of *tritone substitution*: when a dominant chord (secondary or otherwise) is replaced by a dominant chord whose root is a tritone away. The 3rd and 7th of the original chord are the same notes as the 7th and 3rd of its tritone sub: having these important "colour tones" in common, combined with the root movement, creates a cool effect. For example, the II - V - I in C major (originally Dm7 - G7 - CMaj7) becomes Dm7 - Db7 - CMaj7. The chord roots are moving down in semitone steps: achieving this chromatic movement is one common reason for using tritone subs. Whenever we see a 7 chord with a non-diatonic root there's an excellent chance it's a tritone sub: if it also has a #11 then we're certain.

Non-Functional Requirements:

This doesn't mean that the product isn't supposed to work :-)

Each story is expected to meet the following NFRs. Rather than add them to the ACs for every story, they are collected here (though they'll evolve during the year and may not all be relevant initially) and you should check which ones apply when considering a story. Sprint reviews will include NFRs too. You should take NFRs into account when grooming stories. It is a good idea to decorate story cards (e.g. with a coloured sticker or an equivalent tag) to indicate any NFRs that are particularly relevant.

- If a note name does not include an octave specifier then the note is assumed to lie in the octave beginning on middle C (i.e. C4 – B5).
- GUI element designs should be approved by more than one team member. You are encouraged to share sketches with others, such as your scrum master, for comment before committing effort to implementation.
- Users should not see error tracebacks. Where relevant, some indication that an error has taken place is provided (simple status bar or transcript window message; alert box, …) and the details are directed elsewhere (log file, …). You are encouraged to use an appropriate logging framework. "Debugging" information should not appear on stdout.
- Performance must be adequate (details to follow as needed).
- Security should be adequate and commensurate with system parameters. (e.g. while we only have a single user it is sufficient to use OS & basic file system features).
- Undo/Redo (every action should be undoable/redoable where practicable).
- Persistence applies to all new artifacts and associations.
- Current user documentation, help text and examples are provided as appropriate. Extensive end-user documentation is unlikely to be required in early sprints; a current DSL reference is likely to be helpful in each sprint.
- Delivered software must run without modification in the standard CSSE lab environment. The application can be deployed on the target environments and run without the user having to perform extensive configuration. Additional hardware or software (e.g. MIDI devices) should not be required
- Upgrade path/compatibility. If projects created with the previous version (i.e. the one delivered in the previous sprint) of the application can not be opened in the current version then one or more of the following options should be available. Only conversions from the previous version need be considered. As the year goes on, the second option will be more strongly preferred.
- User is warned that the project can not be opened and is directed to the appropriate version of the application

- User is warned that project is from an old version and given the option of continuing or cancelling. A conversion/upgrade/import facility (stand-alone or integrated) is provided. If this is integrated then the user is able to "Save As..." in the current version format.

Note: The story numbering represents priority (backlog) order and not story ID so please refer to stories by their short names

1. Basic App

In order to be able to see my information and perform actions, as a user, I need a simple application with a text area, command entry, menu bar and other basic features.

Notes: This application is a “walking skeleton” rather than a “proper” application whose interface has been designed with usability in mind. At some point we’ll need a more sophisticated interface, but we don’t yet know exactly what we’ll need. It is likely that the next steps will include additional tabbed panes or windows so major GUI components should ideally be self-contained.

Acceptance criteria:

- Simple Swing JFrame
- Stand-alone application; can be started from the desktop level
- Close box “works” — doesn’t just close window but also terminates application.
- MenuBar with File menu and working Quit item
- Text field for entering commands. When the application starts, this field has focus.
- Button (appropriately labelled) for executing commands. See Command cycle story for details.
- The main application pane will have a (read-only) area for text transcript display. See Command cycle story for details.
- GUI elements behave nicely when the application window is resized. Scrollbars appear when appropriate (JScrollPane or similar) and fundamental geometrical relationships (e.g. prompt labels and corresponding text fields) are maintained.

2. Command cycle

So that I can access application features via the GUI, as a user, I need to be able to type in commands.

Note: The number and details of the commands will become more precisely defined in subsequent stories. They will constitute a DSL (Domain Specific Language), defined by a

(BNF-style) grammar. Individual commands will involve keywords (verbs in an imperative language), parameters and perhaps syntactically significant punctuation (e.g. colons, brackets, ...)

Acceptance criteria:

- When the application starts, the command entry field has focus.
- Pressing the command button, or the ‘Enter’ key when the command field has focus, causes the current command (i.e. the content of the text field) to be executed.
- The transcript pane is read-only. When a command is executed, it is echoed in the transcript, preceded by a command prompt (to distinguish commands from other transcript content.
- Any corresponding output will also appear in the transcript.
- The DSL includes a command “show version” which yields the current version/revision string for the application.

3. Transcript management

So that I can keep track of all the activities from a session and access them in future, as a user, I need to be able to save and reload the content of the transcript window.

I’d like to keep these in some sort of project so I don’t have to remember too many file names

Acceptance criteria:

- File menu has “Save Transcript…” item. User can choose location and file name for saved transcript.
- File menu has “Open Transcript…” item. User selects previously saved transcript using appropriate file selection dialog.
- If a transcript has been saved in the current session then the file selection dialog opens in the corresponding directory.
- The content of the selected file replaces any current transcript content

4. Midi note correspondence

So that I can express a unique pitch, as a user, I need to be able to obtain the midi note number corresponding to a note + octave and *vice versa*.

Note: From the musician's point of view, a 'note' is something like 'middle C.' Based on the 'standard' 88-note piano keyboard, middle C is in octave 4 and is denoted C4 accordingly. Almost every modern piano has 52 white keys and 36 black keys for a total of 88 keys (seven octaves plus a minor third, labelled from A0 to C8). MIDI controllers can handle 128 distinct notes/pitches. These are distinguished by note numbers: middle C (C4) is note number 60. The correspondence between mini note numbers and note plus octave notation is shown conveniently in the table at: http://www.electronics.dit.ie/staff/tscarff/Music_technology/midi/midi_note_numbers_for_octaves.htm. Be very careful though: We'll use C4 to refer to middle C, corresponding to MIDI note 60, but you'll sometimes see MIDI documentation based on an octave numbering scheme that would call middle C C5. For a discussion see https://en.wikipedia.org/wiki/Scientific_pitch_notation.

Acceptance criteria:

- The DSL includes a command that yields the note plus octave representation for a given midi note number. For example, entering **"note(60)"** results in the output **"C4"**.
- If the MIDI note number is not valid (i.e. outside the range 0..127) then an appropriate error message is printed on the transcript.
- The DSL includes a command that yields the midi note number for a given note plus octave representation. For example, entering **"midi(C4)"** results in the output **"60"**.
- If the note plus octave is not valid (i.e. does not map onto a midi number in the range 0..127) then an appropriate error message is printed on the transcript.

5. Note order (ascending)

So that I can refer to neighbouring notes, as a user, I need to be able to express the neighbouring note (i.e. a semitone away) higher than the current note.

See also: Note order (descending)

Acceptance criteria:

- The DSL includes a command that yields the name of the note a semitone higher than the given note.
- The result is simplified if necessary so that it is expressed using at most one sharp. For example, "semitone up from G" results in the output "G#", "semitone up from F#" results in the output "G" and "semitone up from B" results in the output "C".

- The precise syntax of the DSL command is negotiable (subject to PO approval) but overall the DSL should have a consistent approach to aspects such as commands, arguments and punctuation. For example, a form such as “semitones(1, G)” might be preferred.
- If an octave specifier is included in the given note then the appropriate octave specifier is included in the result. Thus, using another DSL syntax variant, “semitone higher than C4” is “C#4” and “semitone higher than B4” is “C5”.

6. Note order (descending)

So that I can refer to neighbouring notes, as a user, I need to be able to express the neighbouring note (i.e. a semitone away) lower than the current note.

See also: Note order (ascending)

Acceptance criteria:

- The DSL includes a command that yields the name of the note a semitone lower than the given note.
- The result is simplified if necessary so that it is expressed using at most one flat. For example, “semitone down from G” results in the output “Gb”, “semitone down from Eb” results in the output “D” and “semitone down from C” results in the output “B”.
- The precise syntax of the DSL command is negotiable (subject to PO approval) but overall the DSL should have a consistent approach to aspects such as commands, arguments and punctuation. For example, a form such as “semitones(-1, G)” might be preferred.
- If an octave specifier is included in the given note then the appropriate octave specifier is included in the result. Thus, using another DSL syntax variant, “semitone lower than C4” is “B3” and “semitone lower than D4” is “Db4”.

7. Enharmonic notes (simple)

So that I can refer to notes by their alternate (enharmonic) names, as a user, I need to be able to find the equivalent names for a given note.

See also: Enharmonic notes (full).
Notes: We're calling this “simple” because we’re only talking about enharmonic equivalents with at most one sharp or flat (see table). Note that where a note has a simple enharmonic equivalent (e.g. A#, Bb), the corresponding letter names are different. We will also need to consider the “full” case, where notes have 2 enharmonic equivalents. To do that requires the use of double flats and double sharps.

C	C#	D	D#	E	F	F#	G	G#	A	A#	B
B#	Db		Eb	Fb	E#	Gb		Ab		Bb	Cb

Acceptance criteria:

- The DSL includes a command to find out if a note has a simple enharmonic equivalent. For example, D doesn't but E does.
- The DSL includes a command to obtain the simple enharmonic equivalent of a given note.
- If an octave is specified then the enharmonic equivalent includes the corresponding octave.

8. Enharmonic notes(full)

So that I can refer to notes by the enharmonic equivalent most appropriate in context, as a user, I need to be able to access all enharmonic equivalents for a given note.

See also: Enharmonic notes (simple)

Note: Most notes have an equivalent with letter name lower and another with letter name higher. Double flats and double sharps are needed to express these.

Acceptance criteria:

- The DSL has a command to obtain the enharmonic equivalent whose base letter is higher than the given note. For example, **"enharmonic higherthan(D)"** would yield **"Ebb"**.
- The DSL has a command to obtain the enharmonic equivalent whose base letter is lower than the given note. For example, **"enharmonic lowerthan(D)"** would yield **"Cx"**.
- If an octave is specified then the enharmonic equivalent includes the corresponding octave.

9. Major scales

So that I can obtain the notes of a major scale, as a user, I need to be able to obtain a list of the notes for the major scale starting with the note specified.

See also: Major scale MIDI

Note: The intervals between the notes of the major scale are shown in the following table where T denotes a tone (Tone = 2 semitones) and S denotes a semitone. The notes for C major, D major and F major are included.

	1	2	3	4	5	6	7	8
Interval		T	T	S	T	T	T	S
C major	C	D	E	F	G	A	B	C
D major	D	E	F#	G	A	B	C#	D
F major	F	G	A	Bb	C	D	E	F

Acceptance criteria:

- The DSL includes a command to list the notes of the major scale starting on a given note.
- Only one octave (i.e. 8 notes) of the ascending scale is listed, starting with the note specified.
- Notes are shown as the corresponding letters, together with any sharp/flat symbols.
- If an octave is specified for the starting note then the notes listed also includes the corresponding octave(s).

10. Major scale MIDI

So that I can conveniently play major scales, as a user, I need to be able to obtain the MIDI note numbers corresponding to the notes of a major scale starting on a given note.

See also: Major scales

Acceptance criteria

- The DSL includes a command to list the MIDI note numbers of the major scale starting on a given note.
- Only one octave (i.e. 8 notes) of the ascending scale is listed, starting with the note specified.
- If no octave is specified for the starting note then the starting note is assumed to be in the octave beginning on C4. Thus, **“list scale C major”** would be equivalent to **“list scale C4 major”**.
- If an octave is specified for the starting note then the corresponding MIDI note number is used.

11. Command history

So that I can conveniently access previous DSL commands without cutting & pasting, as a ~~scrum master who shall remain nameless~~ user, I need to be able to scroll backwards and forwards through the commands from the current session.

Acceptance criteria:

- When the DSL command entry field has focus, commands are available to scroll through the previously executed commands. *Next command* and p*revious command* operations are available. Bindings to the up/down arrow keys would be suitable but others (e.g. <CTRL>-P for previous) may be agreed by negotiation.
- The previous command, if there is one, appears in the entry field. Hitting the command execution button or return key executes it as before.
- If there is no previous command (or the the end of the list has been reached) then the current command remains in the entry field.
- Similarly, the *next command* operation causes the next command in the list (if there is one) to appear in the entry field. If no next command is available then the current command remains in the entry field.

12. Tempo

So that I can control note durations, as a user, I need to be able to set the tempo.

Note: Tempo is commonly specified in beats per minute (BPM). Around 120 BPM is a gentle pace but some genres routinely employ much faster tempos. For now, we aren't concerned with fancy rhythms but we do need to control how fast the BPM clock "ticks" and how long a note will last. At 120 BPM the clock ticks twice a second: a note that lasts from one tick to the next is a crotchet (or quarter-note in the U.S.).

Acceptance criteria:

- The default BPM is 120, and this is set when the application starts. At some point we may want to support (global or per-user) preferences but that's not necessary for now.
- The DSL includes a command to obtain the current tempo
- The DSL includes a command to set the tempo to a specified BPM value. The BPM should be positive, integer and will usually be in a "sensible" range (e.g. 20 – 300).
- A range may be agreed by negotiation. Attempts to set a BPM outside the sensible range result in a warning to the user, who can override this with a command option (such as the familiar "**-f**").
- The DSL includes a command to report the duration of a crotchet at the current BPM.

13. Play note

So that I can hear what a note sounds like, as a user, I need to be able to play a given note.

Acceptance criteria:

- The DSL includes a command to play a note
- The note may be given as a MIDI note number
- The note may be given as a letter plus modifier(s) (e.g. “play(C4)”)
- If the note does not include a specified octave (e.g. “play(F#)”) then it is assumed to lie in the octave beginning on middle C (i.e. C4 – B5).
- The command is echoed to the transcript and the corresponding MIDI note is played.
- At this stage it doesn’t matter much what instrument is used, but it should be a pitched instrument. (a piano would be fine; a tom tom wouldn’t) so notes can be heard at the correct pitches.
- The note duration can be specified. If no duration is specified then the default is a crotchet at the current BPM.

14. Pitch comparison tutor

In order to improve my relative pitch recognition, as a learner, I need to be able to test my ability to compare pairs of notes.

See also: Interval recognition

Acceptance criteria:

- A separate pane is available (perhaps in a separate tab) for the pitch comparison tutor. It may include additional controls. By negotiation, it may have its own DSL command entry field or share the existing one with the remainder of the application.
- The DSL includes a command to initiate pitch comparison tests. An option to specify how many note pairs to use is available; if it is omitted then the default is 1.
- An option to specify the pitch range (lowest and highest notes) is available. The range may be specified as MIDI note numbers or as note names. If no octave is specified then the octave beginning on middle C is assumed. If the pitch range option is omitted, then the default range is one octave beginning on middle C (i.e. C4 – C5).
- The system plays two notes, selected at random, from the current range. The two notes can be the same. Each note is played using the same instrument and is a crotchet in duration. Three crotchets’ worth of silence separates the two notes.
- The user then indicates (by entering text or via GUI controls) whether the second note played is lower, higher, or the same as the first. The correct answer is then displayed

- If more than one note pair has been requested then the process repeats until the required number have been played. The user can cancel remaining pairs or choose to proceed to the next.
- When all the notes have been played, the display shows the number of pairs, the number that were incorrect, and the percentage that were correct.
- The user has the option of repeating any pairs that were incorrectly identified.

15. Play scale

So that I can hear what a scale sounds like, I need to be able to play a scale starting on a given note.

Note: For now, we (probably) only have implemented major scales. As we add other scale types, we'll also expect to be able to play them in a similar way.

Acceptance criteria:

- The DSl includes a command to play a specified scale. The starting note and type of scale are given (e.g. 'play scale D major').
- The corresponding MIDI notes are played, using the current instrument, as crotchets at the current tempo.
- By default, one octave of the scale is played but higher (integer) numbers of octaves may be specified.
- By default, the scale is played ascending but there is an option to specify descending or both ascending and descending. If both ascending and descending is specified, then the ascending scale is played and the descending scale begins on the final note of the ascending scale. Thus 'play scale C major updown' would result in the 16 notes 'C, D, E, F, G, A, B, C, C, B, A, G, F, E, D, C' being played.
- The notes played appear on the transcript as usual.

16. Intervals (basic)

So that I can understand and use intervals more effectively, as a user, I need to be able to find the correspondence between interval names and the number of semitones and be able to find the note a given interval away from a particular starting note.

Note; In order to be able to consider major scales in abstract terms, without needing to know which notes/pitches are involved, the intervals from the first note (the tonic) are given names. There's a lot more to intervals than this, but this is plenty for now. The table below shows the basic intervals of the major scale relative to the tonic.:

Interval	unison	Major	Major	Perfect	Perfect	Major	Major	Perfect

from tonic		2nd	3rd	4th	5th	6th	7th	octave
Semitones	0	2	4	5	7	9	11	12
C major	C	D	E	F	G	A	B	C

Acceptance criteria:

- The DSL contains a command that returns the number of semitones in a given interval.
- These intervals may be given: unison; major 2nd, 3rd, 6th, 7th; perfect 4th, 5th and octave. Any others (for now) result in an unknown interval message.
- The DSL contains a command that, when given a tonic (starting note) and an interval name, gives the corresponding note. The resulting note is never lower than the starting note. For example, "interval perfect 4th above G" would yield "C".
- If the starting note does not include an octave specifier than the result doesn't include one either. If the starting note does include an octave specifier then the result also does. Thus "interval major 7th above G4" would yield "F#5".

17. Play interval (basic)

So that I can hear what a given interval sounds like, as a user, I need to be able to play the notes corresponding to an interval and starting note.

See also: Intervals (basic), play note

Acceptance criteria:

- The DSL includes a command to play an interval, given the starting note and interval.
- For now, only basic intervals up to an octave of the major scale need be considered (see story Intervals (basic)). Others result in a warning message and no sound is produced.
- The interval may be given by name or as a number of semitones.
- The note may be given as a letter plus modifier(s). If not octave is specified then the first note is assumed to lie in the octave beginning on middle C (i.e. C4 – B5).
- The command is echoed to the transcript and the corresponding MIDI notes are played on the current instrument. Each note is played using the same instrument and is a crotchet in duration. Three crotchets' worth of silence separates the two notes.

18. Interval recognition tutor (basic)

So that I can improve my ability to recognise intervals, as a learner, I need a way to test myself.

See also: pitch comparison tutor

Acceptance criteria

- A separate pane is available (perhaps in a separate tab) for the interval recognition tutor. It may include additional controls. By negotiation, it may have its own DSL command entry field or share the existing one with the remainder of the application.
- The DSL includes a command to initiate interval recognition tests. An option to specify how many intervals to use is available; if it is omitted then the default is 1.
- The system plays the two notes of an interval selected at random. The two notes can be the same. Each note is played using the same instrument and is a crotchet in duration. Three crotchets' worth of silence separates the two notes.
- The user then indicates (by entering text or via GUI controls) what the interval was. The correct answer is then displayed.
- If more than one interval has been requested then the process repeats until the required number have been played. The user can cancel remaining intervals or choose to proceed to the next.
- When all the intervals have been played, the display shows the number of intervals, the number that were incorrect, and the percentage that were correct.
- The user has the option of repeating any intervals that were incorrectly identified.

19. Musical terms (tempo)

So that I can acquire and manage my musical knowledge, as a user, I need to be able to enter and recall musical terms relating to tempo.

Note: We'll begin with terms for tempos but will later extend this to other terms. Many musical terms are Italian or French: we'll assume that we won't need Cyrillic or other exotic character sets.

Acceptance criteria:

- The DSL includes a command to define a musical term. This will include the term itself (e.g. Lento), the source language (Italian), the meaning (Slowly) and the category (Tempo). The equivalent French term 'Lent' could also be included with a separate term definition.
- The DSL includes a command to obtain the meaning of a given term. For example 'meaningof presto' might yield 'very fast'
- Similar commands are provided to obtain the source language and category of a given term.

20. Project maintenance

So that I can access all the musical knowledge I've defined whenever I like, as a user, I need to be able to save all the information in the system and re-load it later.

Note: While the "knowledge base" currently includes just one category of musical terms further kinds of data will be added in later stories. We'll use terms such as "model," "system," "project" and "knowledge base" loosely and interchangeably to refer to the collective data in the system — but this excludes the transcripts.

See also: transcript management

Acceptance criteria:

- When the application is opened, no project data is present and the current (untitled) model is in a "new" state.
- The File menu has "New" item. Selecting this closes the current model and returns the application to the initial "new" state.
- File menu has "Save…" and "Save As…"items. User can choose location and filename for saved model.
- If there are any unsaved changes then the user is given the option of saving or discarding them.
- File menu has "Open…" item. User selects previously saved model using appropriate file selection dialog.
- If a model has been saved in the current session then the file selection dialog opens in the corresponding directory.
- The content of the opened model replaces any current information. In a future story we may consider merging models.

21. UX I

So that I can maximise my efficiency and minimise errors and frustration, as a user, I need a consistent and easy-to-use application.

Acceptance criteria:

- The current project name appears in the application window title
- Unsaved changes indicator. When there are unsaved changes, an asterisk appears at the beginning of the window title. Additional indicators may be provided (such as enabling/disabling Save item). Undo/redo actions may require this to be updated.

- Major menu items have corresponding short-cut keys. Details by negotiation but there should be no surprises i.e. <ctrl>-O for Open rather than <ctrl>-E because O has already been used for something else.
- Dialogs should have default option (for speed) and this should be highlighted
- Dialogs should have consistent button placement. Details by negotiation but for example 2-button dialogs might have OK on the right & Cancel on the left with OK as the default.
- When the main application window is resized/reshaped, scroll bars appear when needed. There is a minimum size, and all panes are visible when this is reached.

22. Musical terms tutor

So that I can review and improve my knowledge of musical terms, as a learner, I need to be able to test myself.

Acceptance criteria:

- A separate pane is available (perhaps in a separate tab) for the musical terms tutor. It may include additional controls. By negotiation, it may have its own DSL command entry field or share the existing one with the remainder of the application.
- The DSL includes a command to initiate musical terms tests. An option to specify how many terms to use is available; if it is omitted then the default is 1.
- The system asks about a musical term selected at random from the current project. The term may be tested in one of the following ways:
    - The definition is shown and the user types in the correct term.
    - The term is shown and the user types in the source language.
    - The term is shown and the user types in the definition
- Once the user has answered the question then the correct answer is displayed.
- If more than one term has been requested then the process repeats until the required number of questions has been asked. The user can cancel remaining questions or choose to proceed to the next.
- When all the questions have been asked, the display shows the number of questions, the number of incorrect answers, and the percentage that were correct.
- The user has the option of repeating any questions that were incorrectly identified.

23. Reporting/Recording scores

So that I can monitor my performance, as a learner, I need to be able to save information about what the tutors have tested me on and how well I did.

Note: In later stories we might want to record statistics and other information about user performance in the project so that it can be used for such purposes as selecting future questions. For now, we are only interested in information from the current session and this can be regarded as transient. It is likely that, at least initially, this information will be similar to the corresponding transcript content.

Acceptance criteria:

- When a tutor (e.g. the interval tutor) session is initiated (e.g. by a DSL command) a test record is created. The test record begins with the date and the user's name (if this is available).
- At the end of the tutor session the user is prompted to ask if the test record should be saved, in which case an appropriate file dialog is used, or discarded.
- The precise format used is not important but ideally the test record should be suitable for inclusion in other text documents.
- The test record contains an entry for each "question", including the question asked, answer given and whether it was correctly answered.
- The test record concludes with a summary including the number of questions asked, number correctly/incorrectly answered and percentage correctly answered.
- If more than one tutor session occurs during an application session then the test records for successive tests may be concatenated into a single file for the session. If the user has indicated that the records for any tutor session should not be saved then they are not included in the concatenated record.

24. DSL playback

So that I don't have to re-type commands or repeat GUI interactions, as a user, I need to be able to write a little program so I can replay the commands whenever I want.
See also: *Transcript management*

Acceptance criteria:

- The "Save Transcript…" command has an option to save only commands. The original behaviour (saving both commands and output) is also available.
- The DSL commands are saved in a plain text file. Thus, DSL "program" files may also be created with the user's preferred text editor.
- A DSL program may be loaded and subsequently executed. The commands may be executed one by one (e.g. via an "execute next command" button) or all at once (e.g. via a "Run" button)

25. Natural minor scales

So that I can obtain the notes of a natural minor scale, as a user, I need to be able to obtain a list of the notes for the natural minor scale starting with the note specified.

See also: *Major scales, Play scale*

Acceptance criteria:

- As for story *Major scale*, but substituting ‘natural minor’ for ‘major’ as appropriate
- Natural minor scales can be played as described in story *Play scale*. For example we might say something like ‘play scale D minor ascending 2 octaves’ — note the implied “natural.”

26. Scale recognition tutor

In order to improve my ability to recognise scales, as a learner, I need to be able to test my ability to recognise scale types.

See also: Reporting/Recording scores

Acceptance criteria:

- A separate pane is available (perhaps in a separate tab) for the scale recognition tutor. It may include additional controls. By negotiation, it may have its own DSL command entry field or share the existing one with the remainder of the application.
- The DSL includes a command to initiate scale recognition tests. An option to specify how many scales to use is available; if it is omitted then the default is 1. Options are available for specifying the number of octaves and whether the scale is played as ascending, descending or both. The defaults are one octave ascending.
- The system plays a scale selected at random. The starting note may be any note in the octave starting with C4 and any implemented scale type (major, minor, …) may be chosen.
- The user then indicates (by entering text or via GUI controls) what the scale type was. The correct answer is then displayed.
- If more than one scale has been requested then the process repeats until the required number have been played. The user can cancel remaining scales or choose to proceed to the next.
- When all the scales have been played, the display shows the number of scales, the number that were incorrectly identified, and the percentage that were correct.
- The user has the option of repeating any scales that were incorrectly identified.

27. Chords (triads)

So that I can study chords, as a user, I need to be able to obtain the notes for a given chord.

Note: Initially, only major and minor triads will be required. The root position (root, 3rd and 5th in that order) is assumed for this story. In later stories we'll also meet other chord types, chords with more than 3 notes, and chords where the notes appear in other orders.

Acceptance criteria:

- The DSL includes a command which yields the notes of the specified chord (root position). The command and output appear on the transcript as usual.
- Major and minor chords can be handled. For example, "show chord G minor" would give "G, Bb, D"

28. Play chords

So that I can hear what chords sound like, as a user, I need a way to play a given chord.

See also: *Chords (triads)*

Acceptance criteria:

- The DSl includes a command to play a specified chord. The starting note and type of chord are given (e.g. 'play chord D major').
- The corresponding MIDI notes are played, using the current instrument, for a duration equal to a crotchet at the current tempo.
- By default, the notes are played simultaneously but there is also an option to *arpeggiate* the chord. In an arpeggio or 'broken chord' the notes are played one after the other in order starting with the lowest (root in this case). If the chord is played as an arpeggio then the note durations are *quavers* (half the length of a crochet)

29. Chord recognition tutor

So that I can improve my ability to recognise chords, as a learner, I need a way to test myself

Acceptance criteria:

- A separate pane is available (perhaps in a separate tab) for the chord recognition tutor. It may include additional controls. By negotiation, it may have its own DSL command entry field or share the existing one with the remainder of the application.
- The DSL includes a command to initiate chord recognition tests. An option to specify how many chords to use is available; if it is omitted then the default is 1. Options are available for specifying how the chord is played: unison (simultaneously), arpeggio (one after another) or both. The default is unison.
- The system plays a chord selected at random. The starting (root) note may be any note in the octave starting with C4 and any implemented chord type (major, minor, …) may be chosen.

- If the ‘both’ option is specified then the chord is first played as an arpeggio then, after 2 crotchets’ worth of silence, in unison.
- The user then indicates (by entering text or via GUI controls) what the chord type was. The correct answer is then displayed.
- If more than one chord has been requested then the process repeats until the required number have been played. The user can cancel remaining chords or choose to proceed to the next.
- When all the chords have been played, the display shows the number of chords, the number that were incorrectly identified, and the percentage that were correct.
- The user has the option of repeating any chords that were incorrectly identified.

30. Intervals revisited

So that I can understand and use a wider range of intervals, as a user, I need to be able to find the correspondence between the interval names and the number of semitones as well as the intervals from a given starting note.

Note: This story builds on Intervals (basic) to add a wider range of intervals.
See also: Intervals (basic)
Depends on: Intervals (basic)

Acceptance criteria:

- The DSL contains a command that returns the number of semitones in a given interval.
- These intervals may be given: unison; minor 2nd, major 2nd, minor 3rd, major 3rd, perfect 4th, augmented 4th, diminished 5th, augmented 5th, minor 6th, major 6th, diminished 7th, minor 7th, major 7th and perfect octave. Any others (for now) result in an unknown interval message.
- Intervals up to 2 octaves are included. While the qualities (minor, augmented, ...) are the same for the higher notes, their numbers will be higher. For example, C4 – Db4 is a minor 2nd but C4 – Db5 is a minor 9th; C4 – A5 is a major 13th.
- The DSL contains a command that, when given a tonic (starting note) and an interval name, gives the corresponding note. The resulting note is never lower than the starting note. For example, “interval minor 10th above G” would yield “Bb”.
- If the starting note does not include an octave specifier than the result doesn’t include one either. If the starting note does include an octave specifier then the result also does.

31. Play interval revisited

So that I can hear what a given interval sounds like, as a user, I need to be able to play the notes corresponding to an interval and starting note for a wider range of intervals.

See also: Play interval (basic)

Acceptance criteria:

- The DSL includes a command to play an interval, given the starting note and interval.
- For now, only intervals implemented in story *Intervals revisited* need be considered. Others result in a warning message and no sound is produced.
- The interval may be given by name or as a number of semitones.
- The note may be given as a letter plus modifier(s). If not octave is specified then the first note is assumed to lie in the octave beginning on middle C (i.e. C4 – B5).
- The command is echoed to the transcript and the corresponding MIDI notes are played on the current instrument. Each note is played using the same instrument and is a crotchet in duration. Three crotchets' worth of silence separates the two notes.

32. Interval recognition tutor revisited

So that I can improve my ability to recognise intervals, as a learner, I need a way to test myself on a wider range of interval types.

See also: Interval recognition tutor (basic)

Acceptance criteria

- The GUI elements implemented in *Interval recognition tutor (basic)* are used
- The ACs of *Interval recognition tutor (basic)* apply here: the only difference is that more interval names are available and intervals up to 2 octaves are available.

33. Keyboard

So that I can interact with the system in a more natural way, as a user, I need a piano-style keyboard to be available.

Note:

- While future stories might involve other input modes (such as fretted instruments) this is not a high priority in the medium term. Several potential complications, such as alternate tunings, would need to be considered at that point.
- While the keyboard will be used to play notes, provide input in tutors etc., this story only involves developing the keyboard.

Acceptance criteria:

- A “piano-like” keyboard[4] is available. Its use is optional and it is not visible unless selected explicitly (e.g. via a menu item or application preferences).
- The keyboard may appear in a separate window, or in a pane in the application GUI. Details by negotiation subject to PO approval.
- The keyboard need not resemble a conventional piano keyboard exactly but should be as realistic as practicable. At this point it might be sufficient to use some kind of “buttons” with appropriate layout in two rows (i.e. corresponding to the black/white keys).
- The number of keys can be negotiated but should be sufficient to allow one octave of any scale to be played. In future stories it may be necessary to support chord or other concepts which span more than one octave.
- Normally, only single notes need be “played” at a time. However, a means of selecting several notes to be played together (i.e. a chord) is also necessary. Something like Shift-Click might be sufficient.
- There is a visual indication of which note is playing (e.g. the corresponding “key” is highlighted.
- Option to label each key with corresponding note name, or just display note name for currently-pressed key, or none of these.
- There is a display option to indicate which octave we’re in.

34. Enharmonic interval names

So that I can determine and use equivalent interval names, as a user, I need to be able to access equivalent interval names.

Note; There are more enharmonic equivalents than we need consider here (e.g. a minor 3rd is also an augmented 2nd)

Acceptance criteria:

- The DSL has a command to find out if an interval has an enharmonic equivalent (that we care about). For example, dim 5 and maj 6 do but major 3rd and minor 2nd don’t.
- The DSL has a command to obtain the enharmonic equivalent(s) of a given interval.

[4] It will just be called “keyboard” here so be careful to resolve any potential confusion with the regular keyboard.

35. Chords (4-note)

So that I can study a wider range of chords, as a user, I need to be able to obtain the notes for a given 4-note chord.

Note: The following table shows four common kinds of 4-note chords, together with examples that might occur in a song in the 'key of C' — we'll meet this concept later, but the point to note here is that it means these chords contain only notes from the C major scale. The diminished 7th is an exception: we'll look at these in more detail in future stories.

Name	I	III	V	VII	Symbol	
Major 7th	1	Major 3	Perfect 5th	Major 7th	C^{Δ}	C, E, G, B
Minor 7th	1	minor 3rd	Perfect 5th	Minor 7th	D_m^7	D, F, A, C
7th	1	Major 3rd	Perfect 5th	Minor 7th	G^7	G, B, D, F
Half diminished (minor 7 b5)	1	minor 3rd	diminished 5th	minor 7th	B^{Φ}	B, D, F, A
Major 6th	I	Major 3rd	Perfect 5th	Major 6th	C^6	C, E, G, A
Diminished 7th	I	minor 3rd	diminished 5th	diminished 7th	$C\#^{o7}$	C#, E, G, Bb

Acceptance criteria:

- The DSL includes a command which yields the notes of the specified chord (root position). The command and output appear on the transcript as usual.
- As well as major and minor triads, Major 7th, minor 7th, 7th and half-diminished chords can be handled. For example, "show chord G half dim" would give "G, Bb, Db, F"
- The chord recognition tutor includes these 4-note chords.

36. Play chords (revisited)

So that I can hear what more chords sound like, as a user, I need a way to play a given triad or 4-note chord.

See also: *Chords (triads)*

Acceptance criteria:

- The DSl includes a command to play a specified chord. The starting note and type of chord are given (e.g. ‘play chord D min7’). Triads and all currently implemented 4-note chord types may be specified.
- The ACs for story *Play chords* apply.

37. Chord recognition tutor (revisited)

So that I can improve my ability to recognise chords, as a learner, I need a way to test myself on a wider range of chords.

See also: *Chord recognition tutor*

Acceptance criteria:

- The ACs for story Chord recognition tutor apply. This story adds 4-note chords to the existing triad functionality.

38. Keyboard input

So that I can interact with the system in a more natural way, as a user, I need to be able to use a piano-style keyboard (PK) to input notes.

See also: Keyboard

Acceptance criteria:

- The PK keyboard is enabled when it can be used to enter data. These will include tutors and play commands. For example, if a tutor is asking for the notes in a chord or scale then the PK keyboard can be used as well as other current options such as the command entry.
- The user sees a visual indication that PK keyboard input is possible (i.e. enabled).

39. Swing

So that scales sound more natural, as a user, I need to be able to play them in swing style.

Notes:
Until now, scale playback has been “robotic” with each note played as a quaver at the same relentless interval.

From now on, we'll consider scales as being played in quavers, where a quaver is half the duration of a crotchet (i.e. At the same BPM, this means they will be twice as fast).
Two quavers take up a crotchet duration. In "straight" style the second quaver is played at the middle of the interval. In (default) "swing" style the crotchet interval is divided into thirds: the first quaver takes up the first ⅔ and the second quaver the remaining ⅓.
The typical default strength of the swing is a ⅔ / ⅓ split. In "light" swing we might have our crotchets split ⅝ / ⅜ so the quaver durations are more even. In "heavy" swing the split might be close to ¾ / ¼ --- you'll hear this in a classic blues/rock shuffle (e.g. https://www.youtube.com/watch?v=jlV3zeWnWZY) and milder forms in jazz styles (e.g. https://www.youtube.com/watch?v=MR1Yrkjk_cM).

Acceptance criteria:

The DSL commands for playing scales include an option for specifying straight or swing styles. The default remains straight style.
The amount of swing is controlled by a variable in the same way as tempo. The default is a ⅔:⅓ crotchet split.

40. Key signatures (major)

So that I can work with a wider range of harmony applications, as a user, I need to be able to explore the correspondence between key signatures and scales.

Note: We know (from Stories *Major scales* and *Natural minor scales*) how to find the notes that belong to major scales and their relative natural minor scales. It is inconvenient to have to repeat the sharps/flats every time a note occurs in a piece of music: the convention in sheet music is to show the prevailing sharps/flats (the *key signature*) once at the beginning of a piece and then only show variations ('accidentals') when they occur. For example, the key signature of D major has 2 sharps (F# and C#). The rules for determining key signatures are embodied in the *circle of fifths*. A key signature contains 0 – 7 sharps or flats. Each key signature contains only sharps or flats (i.e. no combinations). The order in which sharps/flats are added (FCGDAEB for sharps, BEADGCF for flats) is determined from the circle of fifths. Consequently, FCGDAEB (adding sharps, clockwise) and BEADGCF (adding flats, anticlockwise) are palindromes.

Acceptance criteria:

- The DSL includes a command that yields the key signature of a given scale. For example, a possible form might be 'show signature C minor' giving 'Bb, Eb, Ab' where the sharps/flats of the key signature appear in the correct order.

- There is an option to yield the number of sharps/flats rather than listing them. For example, a possible form might be ‘show signature -n A major’ giving ‘3#’. This is the default if no option is specified.
- The DSL includes a command that yields the name(s) of the scale(s) with a given key signature. Options are available to specify major only, minor only and both. The default is major only.
- The key signature may be expressed in terms of a number of sharps/flats. For example ‘show scale sig 2#’ would return ‘D major’. Only valid (0 – 7) numbers of sharps/flats may be specified. An appropriate warning message is displayed if an invalid number is specified.
- There is also an option to express the key signature listing the sharps/flats explicitly. For example ‘show scale sig F#,C#’ would return ‘D major’. If there is no matching scale then an appropriate warning message is displayed. This might occur because the sharps/flats are out of order (e.g. ‘show scale sig C#,F#’) or because the set given does not occur for any implemented scale (e.g. ‘show scale sig C#,G#’)

41. Key signature tutor

So that I can improve my knowledge of key signatures, as a learner, I need to be able to test my ability to associate key signatures with the corresponding scales.

Acceptance criteria:

- A separate pane is available (perhaps in a separate tab) for the key signature tutor. It may include additional controls. By negotiation, it may have its own DSL command entry field or share the existing one with the remainder of the application.
- The DSL includes a command to initiate key signature tests. An option is available to specify how many tests are required; if it is omitted then the default is 1. There are two types of test question. In the first, the system gives the key signature and the user identifies the corresponding scale(s). In the second, the system gives a scale name and the user provides the key signature.
- Options are available for specifying how key signatures are specified (number of sharps/flats and/or listing sharps/flats explicitly) and whether major, minor or both scale names are required.
- For “type 1” questions, the system displays a key signature selected at random subject to the current options. The user then gives (by entering text or via GUI controls) the corresponding scale name(s). The correct answer is then displayed.
- For “type 2” questions, the system displays a scale name selected at random subject to the current options. The user then gives (by entering text or via GUI controls) the corresponding key signature (in a form consistent with the current options). The correct answer is then displayed.

- If more than one test has been requested then the process repeats until the required number have been completed. The user can cancel remaining tests or choose to proceed to the next.
- When all the tests have been completed, the display shows the number of tests, the number that were incorrectly identified, and the percentage that were correct.
- The user has the option of repeating any tests that were incorrectly identified.

42. DSL menu & reference

So that I can improve accuracy and efficiency, as a user, I need to be able to access available DSL commands easily.

Acceptance criteria:

- The application has a GUI control (e.g. a help menu item) that displays a "DSL Reference Card."
- The reference card appears in a separate window so it is available for reference as needed. The reference card window can be dismissed when no longer required.
- The reference card content is a list of DSL commands, together with available options and parameters.
- The commands are grouped appropriately. The user can choose whether commands appear in alphabetical order or grouped by category.
- The application has a "Command" menu whose menu items are nested menus corresponding to the various categories of DSL command. For example, there might be "Play," and "List" menu items. The nested menus contain the various commands in the corresponding category. Thus the Command -> Play menu might contain "Chord...," "Note...," and "Scale…".
- When a command is selected it is copied to the (active) command entry field, together with information about the available options and parameters. Details by negotiation but something along the lines of a man page might be suitable.

43. Inversions

So that I can use effectively chords in different contexts, as a user, I need to be able to obtain the inversions of a given chord.

Note: Triads in (closed) root position, consisting of the 1st, 3rd and 5th, have already been implemented. Their 1st and 2nd inversions are essentially cyclic permutations where the bottom note is rotated to the top. Four-note chords can have 3rd inversions. Inversions are one form of 'voicing' to make smooth transitions between the various notes of successive chords. Some examples are shown in the table below.

Chord	Root position	1st inversion	2nd inversion	3rd inversion
C	C, E, G	E, G, C	G G E	
Dmin7	D, F, A, C	F, A, C, D	A, C, D, F	C, D, F, A

See also: *Chords (triads)*, *Chords (4-note)*

Acceptance criteria:

- The ACs for stories Chords (triads) and Chords (4-note) apply as the default (i.e. chord notes are shown in root position).
- An option is available to specify an inversion

44. Chord finder (basic)

So that I can work more effectively with chords, as a user, I need to be able to find the chord(s) a given set of three notes make.

See also: *Chord finder (enhanced)*

Acceptance criteria:

- The DSL includes a command that yields the names of chords corresponding to the given three notes.
- If the three notes correspond to a triad (major, minor, diminished, augmented) then this is reported. For example “find chord for C Eb G” might yield “C minor”.
- The notes may be given in any order (e.g. inversions). There is an option to specify whether or not inversions are included in the output (e.g whether ‘find chord for E, G, C’ returns ‘C major’ or ‘C major 1st inversion’) and the default is not to include inversions.
- If the given notes do not correspond to any known triad then an appropriate message is provided.

45. Chord finder (enhanced)

So that I can work even more effectively with chords, as a user, I need to be able to find the chord(s) a given set of three or four notes make.

Depends on: *Chord finder (basic)*

Note: Sometimes the same set of (3 or 4 in this case) notes make up more than one possible chord. These are *enharmonic chords* — for now, we’ll need to be aware of these cases:

- Major 6 has equivalent minor 7th (e.g. C6 = Amin7). {C E G A = A C E G}
- Half diminished has equivalent minor 6th (e.g. C half dim = Eb min6). {C Eb Gb Bb}
- Diminished 7th chords can be permuted to make enharmonically equivalent chords (C dim7 = Eb dim7 = Gb dim 7 = Bbb dim7) {C Eb Gb Bbb}
- Aug triads can be permuted to make enharmonically equivalent chords (e.g. C+ = E+ = G#+) {C E G# : E G# B# : G# B# D##}

Acceptance criteria:

- Triads (major, minor, diminished, augmented) and 4-note chords
- The DSL includes a command that yields the names of chords corresponding to the given three or four notes.
- The notes may be given in any order (e.g. inversions). There is an option to specify whether or not inversions are included in the output (e.g whether 'find chord for B, D, E, G, ' returns 'E minor' or 'E minor 2nd inversion') and the default is not to include inversions.
- If the given notes do not correspond to any known chord then an appropriate message is provided.
- Enharmonic chords (including inversions) as described in the note above are recognised.
- By default, all enharmonic chords are given. There is an option to just give one: if the given notes, in order, constitute a chord then this is the one returned; otherwise, any enharmonic equivalent may be given.

46. Chord spelling tutor

So that I can improve my knowledge of chords, as a learner, I need to be able to test my ability to spell and identify chords.

Acceptance criteria:

- A separate pane is available (perhaps in a separate tab) for the chord tutor. It may include additional controls. By negotiation, it may have its own DSL command entry field or share the existing one with the remainder of the application.
- The DSL includes a command to initiate chord spelling tests. An option is available to specify how many tests are required; if it is omitted then the default is 1. There are two types of test question. In the first, the system gives the chord name and the user identifies the corresponding notes. In the second, the system gives 3 or 4 notes and the user provides the chord name.
- Options are available to constrain the chord types to be used (e.g. only minor chords), specify whether all enharmonic chords are required or just one etc.
- For "type 1" questions, the system displays a chord name selected at random subject to the current options. The user then gives (by entering text or via GUI controls) the

corresponding notes. If an inversion is specified then the note order must correspond in order for the answer to be correct. Otherwise, the notes should be given in root position. The correct answer is then displayed.

- For “type 2” questions, the system displays 3 or 4 notes selected at random subject to the current options. By default, the notes should correspond to at least one chord. An option is available to allow random note combinations, which may not necessarily correspond to a “real” chord, to be selected in a specified percentage of tests. In this case, one of the available answers will be something along the lines of “No Chord”. The user then gives (by entering text or via GUI controls) the corresponding chord name. The correct answer is then displayed.
- If more than one test has been requested then the process repeats until the required number have been completed. The user can cancel remaining tests or choose to proceed to the next.
- When all the tests have been completed, the display shows the number of tests, the number that were incorrectly identified, and the percentage that were correct.
- The user has the option of repeating any tests that were incorrectly identified.

47. Pentatonic scales

So that I can work with a wider range of scales, as a user, I need to be able to list and play pentatonic scales.

See also: Scale Recognition tutor

Notes:

- A pentatonic scale has five notes.
- A *major* pentatonic has five notes from the corresponding major scale (1, 2, 3, 5, 6) with the octave conventionally included. Thus C major pentatonic would be C, D, E, G, A, C
- Fun fact (thanks Wikipedia!) The black keys on a piano keyboard is pentatonic scale: G♭, A♭, B♭, D♭, and E♭. Stick to those and you can bust out a great solo without hitting any “wrong” notes.
- A minor pentatonic has the tonic, minor 3rd, 4th, 5th and minor 7th. Thus C minor pentatonic is C, E♭, F, G, B♭, C.

Acceptance criteria:

- The DSL includes a command to list the notes of the major or minor pentatonic scale starting on a given note.
- Only one octave of the ascending scale is listed, starting with the note specified.
- As for the other scale types, notes are shown as the corresponding letters, together with any sharp/flat symbols and if an octave is specified for the starting note then the notes listed also includes the corresponding octave(s).

- The DSl includes a command to play a specified pentatonic scale. The starting note and type of scale are given (e.g. ‘play scale D major pentatonic’).
- The corresponding MIDI notes are played, using the current instrument, as crotchets at the current tempo.
- By default, one octave of the scale is played but higher (integer) numbers of octaves may be specified.
- By default, the scale is played ascending but there is an option to specify descending or both ascending and descending. If both ascending and descending is specified, then the ascending scale is played and the descending scale does not repeat the final note of the ascending scale. Thus ‘play scale C major pentatonic updown’ would result in the 11 notes ‘C, D, E, G, A, C, A, G, E, D, C’ being played.
- The notes played appear on the transcript as usual.
- The scale recognition tutor has an option to include pentatonic scales.

48. Blues scales

So that I can work with a wider range of scales, as a user, I need to be able to list and play blues scales.

See also: Scale Recognition tutor, Pentatonic scales, Play scale

Notes:

- A blues scale is minor pentatonic plus the “blue note”, the flattened fifth. The blue note can be thought of as the augmented 4th, diminished 5th or flattened fifth: it is the tritone of the tonic and an essential ingredient in great blues licks. Thus the C blues scale is C, E ♭ , F, F#, G, B ♭ , C.

Acceptance criteria:

- The DSL includes a command to list the notes of the blues scale starting on a given note.
- Only one octave of the ascending scale is listed, starting with the note specified.
- As for the other scale types, notes are shown as the corresponding letters, together with any sharp/flat symbols and if an octave is specified for the starting note then the notes listed also includes the corresponding octave(s).
- The DSl includes a command to play a specified blues scale. The starting note and type of scale are given (e.g. ‘play scale F blues’). The notes played appear on the transcript as usual.
- By default, the scale is played ascending but there is an option to specify descending or both ascending and descending. If both ascending and descending is specified (blues scales sound great this way), then the ascending scale is played and the descending scale does not repeat the final note of the ascending scale. Thus ‘play

scale F blues updown' would result in the 13 notes 'F, A ♭ , B ♭ , B, C, E ♭ , F, E ♭ , C, B, B ♭ , A ♭ , F' being played.

- The scale recognition tutor has an option to include blues scales.
- Blues scales sound best played in a triplet feel (the F blues updown example would be accented ***1*** 2 3, ***2*** 2 3, ***3*** 2 3, ***4*** 2 3, ***1***) so the blue note is accented in each direction. By negotiation, explore the practicality of achieving this: it may be handy i other future stories.

49. Diatonic chord function (major)

So that I can begin working with functional harmony, as a user, I need to be able to manage information about chord function.

See also: Chords (4-note)

Notes:

- The notes of a major scale are referred to as "diatonic" and the chords based on them are "right' for harmonising melodies based on the corresponding scale. Roman numerals (upper case) are used to denote the function of each chord (i.e. the chord for each scale degree): knowing the function is sufficient to tell the chord quality.
- Other chords will also turn up in practice, and these will be treated separately. These will fall into two categories; those with "wrong" qualities for diatonic roots (e.g. D^7 in C major) and those with non-diatonic roots (e.g. Db^7 in C major).
- There are three groups: *tonic* (I, III, VI), *sub-dominant* (II, IV) and *dominant* (V, VII) based on function. The chords in each group play "similar" roles and arrangers/composers/re-harmonisers will sometimes substitute a chord from the same group.

The following table summarises the description for C major.

Function/Degree	Quality	C major example	Spelling (C Major)
I	Maj7	C^{Δ}	C, E, G, B
II	Min7	D_m^7	D, F, A, C
III	Min7	E_m^7	E, G, B, D
IV	Maj7	F^{Δ}	F, A, C, E
V	7th	G^7	G, B, D, F

VI	Min7	A_m^7	A, C, E, G
VII	Min7b5	B^{Φ}	B, D, F, A

Acceptance criteria:

- The DSL includes a command to obtain the quality of the chord for a given degree. For example ‘qualityOf(II)’ might yield ‘m7’.
- The DSL includes a command to obtain the chord for a given function and key. For example, chordFunction(IV, D, major) might yield `GMaj7’.
- The DSL includes a command which, when given a chord and key, gives the corresponding function (or an appropriate message if there isn’t one). For example, ‘functionOf(F7, Bb)’ might yield `V’ and ‘functionOf(Fm7, Bb)’ might yield ‘NonFunctional’.
- An appropriate tutor (perhaps an extension of the chord tutor) is provided. Given a (major) key and a degree, the subject is asked to give the corresponding chord (e.g. “What is the VI chord of F major?” (Dmin7)). Given a chord and a (major) key the subject is asked to identify the corresponding function (e.g. “In F major, what is Gmin7?” (II)). Similarly, “In F major, what is G7?” (NonFunctional)).

50. Major scale modes

So that I can select appropriate scales to play when improvising, as a user, I need to be able to obtain the scales corresponding to diatonic chords.

Note: We normally think of playing a major scale by starting on the tonic. Modes arise when we start on other (diatonic) notes and play the scale from there. For example, the second mode of the C major scale comprises the notes D, E, F, G, A, B, C. Another way to think about this is as a “D-something” scale: this would be some minor-ish scale (since it has F and not F#) and it also has a flattened/minor 7th. This scale is called D dorian. Similarly, starting on A produces A aeolian (which we also know as the natural minor). The following table shows the scales that result from each mode of the major scale. The letters in the top row correspond to the notes of the C major (C ionian) scale. We’ll encounter these modes in future stories, together with modes of other scales.

	Mode	**1 (C)**	**2 (D)**	**3 (E)**	**4 (F)**	**5 (G)**	**6 (A)**	**7 (B)**
I	Ionian	1	2	3	4	5	6	7

II	Dorian	1	2	b3	4	5	6	b7
III	Phrygian	1	b2	b3	4	5	b6	b7
IV	Lydian	1	2	3	#4	5	6	7
V	Mixolydian	1	2	3	4	5	6	b7
VI	Aeolian	1	2	b3	4	5	b6	b7
VII	Locrian	1	b2	b3	4	b5	b6	b7

Acceptance criteria:

- The DSL has a command which, when given a key and a degree, can obtain the corresponding scale. For example, modeOf(C, 2) would return ‘D dorian’
- The DSL has a command which, when given a scale, can obtain the corresponding parent (ionian) scale. For example, ‘parentOf(A, phrygian)’ would return `F major’
- The DSL command that shows the notes of scales includes the ability to handle scales specified as modes. For example `show scale(D, phrygian)’ would give `D Eb F G A Bb C D’.
- The DSL command that plays scales includes the ability to play scales specified as modes. For example `play scale(D, phrygian)’
- The scale recognition tutor has an option to include major modes as scales to be recognised.
- A scale mode tutor (which might be an extension of another tutor) tests knowledge of scales which have the same notes (i.e. are modes of the same parent scale). For example, D dorian and B locrian have the same notes (both are modes of C major); D dorian and A mixolydian do not.

51. Instrument selection

So that I can have some variety in my playback, as a user, I want to be able to choose which instrument is used.

Note: The default MIDI playback instrument is likely to be the 0th of the available instruments, corresponding to an acoustic piano. Not all of the available instruments are suitable for our purposes. We need pitched instruments in order to play scales: unpitched percussion instruments won’t be able to do that.

Acceptance criteria:

- The DSL contains a command to show the current instrument. It is sufficient to display the instrument's name but additional information (e.g. a picture, pitch range, …) may also be provided.
- The DSL contains a command to list the available instruments.
- The DSL contains a command to allow the user to select a new instrument. Subsequent play commands will use the new instrument.
- Some form of 'undo' is available so that the user can conveniently revert to the previous instrument (e.g. if it turns out not to be pitched)

52. Melodic minor scales

So that I can work with a wider range of scales, as a user, I need to be able to list and play melodic minor scales.

See also: Scale Recognition tutor, Play scale

Notes:

- A melodic minor scale is the corresponding major scale with a flattened (minor) third. The melodic minor is sometimes called the "Jazz minor" because it is an important element of jazz harmony. Thus the C melodic minor scale is C, D, E ♭ , F, G, A, B, C.

Acceptance criteria:

- The DSL includes a command to list the notes of the melodic minor scale starting on a given note.
- Only one octave of the ascending scale is listed, starting with the note specified.
- As for the other scale types, notes are shown as the corresponding letters, together with any sharp/flat symbols and if an octave is specified for the starting note then the notes listed also includes the corresponding octave(s).
- The DSL includes a command to play a specified melodic minor scale. The starting note and type of scale are given (e.g. 'play scale F mel min'). The notes played appear on the transcript as usual.
- By default, the scale is played ascending but there is an option to specify descending or both ascending and descending.
- The scale recognition tutor has an option to include melodic minor scales.

53. Visualiser

So that the application will be more visually appealing, as a user, I want to be able to have an interesting visual display while sounds are being played.

Note: Many media players include sophisticated dynamic visualisers that respond to changes in the quantities such as the volume or frequency distribution of music being played. Our current needs are simpler. Discussing your ideas with your SM/PO during grooming is encouraged.

Acceptance criteria:

- The DSL has a command to enable/disable the visualiser.
- If more than one visualiser is available then the DSL includes commands to list them and to select one. In this case, the enable/disable command applies to the current visualiser.
- The visualiser, when enabled, appears in an appropriate GUI element (details by negotiation). For example, it might appear in the region where the transcript pane usually appears.
- The visualiser displays something 'appropriate' when notes, scales, chords etc are playing. The details of what is 'appropriate' are to be negotiated with PO. For example, chords might be accompanied by a static visualisation with an element representing each note and the quality. For scales, a more dynamic visualisation that build up as each note is added might be appropriate.
- Individual visualisers may have parameters that control properties such as speed and colours. In such cases the DSL include commands to display and set the corresponding values. An appropriate GUI element may also be provided for convenience.

54. Melodic minor modes

So that I can select appropriate scales to play when improvising in minor contexts, as a user, I need to be able to obtain the scales corresponding to diatonic chords.

See also: Major scale modes

Note: Modes of the major scale have been addressed in a previous story. Here we are interested in the modes of the melodic minor scale. The following table shows the scales that result from each mode of the melodic minor scale. The letters in the top row correspond to the notes of the C melodic minor (C minormajor) scale. We'll encounter these modes in future stories, together with modes of other scales.

	Mode	**1 (C)**	**2 (D)**	**3 (Eb)**	**4 (F)**	**5 (G)**	**6 (A)**	**7 (B)**
I	Minormajor	1	2	b3	4	5	6	7
II	Dorian b2	1	b2	b3	4	5	6	b7

III	Lydian #5	1	2	3	#4	#5	6	7
IV	Lydian dominant	1	2	3	#4	5	6	b7
V	Mixolydian b6	1	2	3	4	5	b6	b7
VI	Locrian #2	1	2	b3	4	b5	b6	b7
VII	Altered (Super Locrian)	1	b2	b3	b4	b5	b6	b7

Acceptance criteria:

- The DSL has a command which, when given a minor key and a degree, can obtain the corresponding scale. For example, modeOf(C, 2) would return 'D dorian b2'
- The DSL has a command which, when given a scale, can obtain the corresponding parent melodic minor scale. For example, 'parentOf(A, Lydian #5)' would return `F mel minor'
- The DSL command that shows the notes of scales includes the ability to handle melodic minor scales specified as modes. For example `show scale(F, Lyd dom)' would give `F G A B C D Eb F'.
- The DSL command that plays scales includes the ability to play scales specified as melodic minor modes. For example `play scale(D, alt)'
- The scale recognition tutor has an option to include melodic minor modes as scales to be recognised.
- A scale mode tutor (which might be an extension of another tutor) tests knowledge of scales which have the same notes (i.e. are modes of the same parent scale). This is also able to handle melodic minor modes. For example, D dorian b2 and B alt have the same notes (both are modes of C melodic minor); D dorian b2 and A mix b6 do not.

55. Digital patterns

So that I can develop a deeper knowledge of scales, as a user, I need to be able to work with digital patterns.

See also: Major scales, play scale, scale recognition tutor

Note: A digital pattern is just a sequence of number specifying a particular order for playing a scale (without having to specify a particular key). The familiar ascending order corresponds to the pattern 1 2 3 4 5 6 7 8. Familiarity with other patterns is important not only to avoid boredom and develop proficiency but also because they can be used "for real" in performance contexts. Common patterns include 1 3 2 4 3 5 4 6 5 7 6 8 7 9 8 and its reverse form 8 6 7 5 6

4 5 3 4 2 3 1 2 7 1: the pattern is essentially “up 2, back 1”. Another common pattern involves groups of three: 1 2 3 2 3 4 3 4 5 4 5 6 5 6 7 6 7 8 7 8 9 8.

Acceptance criteria:

- The command to play a scale has an option to specify a digital pattern. If no pattern is specified then the notes are played in order (respecting and ascending/descending options)
- Digital patterns may be specified and saved with a name to enable convenient re-use. For example, the patterns in the note above might be called “sequence, ” “thirds” and “triplets”
- Where a pattern is used (e.g. as an option) it may be specified literally or by name. Thus the commands `play scale C major -dp '1 3 2 4 3 5 4 6 5 7 6 8 7 9 8'` and `play scale C major -dp thirds` would each result in the notes C E D F E G F A G B A C B D C being played.
- Similarly, digital patterns can be used in commands which display the notes of other types of scales, scale knowledge tutors and any other relevant features by negotiation.

56. Users and profiles

So that I can collaborate more effectively with others, as a user, I need to be able to keep my materials separate from those of other people.

Note: This story is not primarily about security: the assumption is that users will be natural collaborators (e.g. members of a family) and the purpose of userids etc is primarily to prevent accidents rather than malicious damage.

Acceptance criteria:

- Individual user profiles can be created. Any user can create as many profiles as they wish.
- A userid (a short name for convenience in logging on etc) is associated with each profile.
- A password is also associated with each user profile. Ideally, the password should be non-null but there is no need to enforce this, require any particular password strength or prevent it being the same as the userid.
- If desired, other information (e.g. Full name, phone number, experience level, …) may be associated with a profile.
- A “public” profile is available (explicitly or implicitly) and is active when the application starts.
- Users may log on with individual userid/passwords and log out as desired. Only one user can be logged on at a time. When the application quits, and logged on user is automatically logged off.

- Data such as session transcripts and results for tutor sessions is associated with individual user profiles. If no user is logged on then such data is associated with the public user and everything works as previously.
- Similarly, settings such as tempo are on a per-user basis.
- A logged on user is prompted to save their data on logout. If no user is logged on the the usual warning applies to the public user.
- A knowledge base (e.g. of musical terms) is owned by, and only accessible to, the user who created it.
- Artifacts associated with individual users may be stored together, or by category (e.g. all knowledge bases in a directory) or in any appropriate manner. For now at least, file system security is sufficient and no further security features are required.

57. Visualising tutor scores

So that I can monitor my progress, as a user, I need to be able to access convenient visual representations of my scores from sessions with tutors.

See also: *Reporting/Recording scores, Users and profiles*

Note: The primary goal of this story is to support individual users rather than indicate relative performance compared to others. Consequently, visualisations will focus on data for one user at a time and any supported comparisons will be with aggregate data (e.g. from the public profile) rather than other specific individuals.

Acceptance criteria:

- The DSL and GUI support access to score visualisation functionality.
- The user can select previously saved score data for analysis and visualisation. This may be done by selecting one or more files/sessions.
- The user can choose to see available session data grouped by tutor type (scale, key signature, …). The sessions for each tutor type are then ordered chronologically.
- Alternatively, the user can choose to see the available session data in chronological order. In this case, an indication of the corresponding tutor type is also included.
- It may also be appropriate to select data from a specified time range (just a start & end date is sufficient: we don't need criteria such as "first Tuesday in the month").
- The selected data set is then displayed in an appropriate visual manner. Where more than one relevant visual representation is available (e.g. pie charts, bar charts, …) the user can select which is to be used.
- The details of the available visualisations are negotiable, but are likely to include common chart types such as bar charts, pie charts, kiviat charts, histograms and the like.
- Where appropriate, tabular information and statistical quantities (such as mean and standard deviation) may also be included.

- The visualisations will help users understand things like:
    - How much of the tutor content has been attempted
    - Level of achievement in topic areas
    - Topics with the most potential for improvement (things to put on high rotation in the practice schedule).
    - How performance has changed over time (Am I getting better? Do I forget topics after 2 weeks?, …)

58. Harmonic minor scales

So that I can work with a wider range of scales, as a user, I need to be able to list and play harmonic minor scales.

See also: *Scale Recognition tutor, Play scale, Melodic minor scales, melodic minor modes*

Notes:

- A harmonic minor scale is the corresponding natural minor scale with a raised (major) seventh. Thus the C harmonic minor scale is C, D, E♭, F, G, A♭, B, C.
- Unlike the melodic minor, only one other mode of the harmonic minor is commonly used. The 5th mode of the harmonic minor is the Mixolydian flat 2 flat 6. For example, Cmixb2b6 is C, Db, E, F, G, Ab, Bb, C. This scale is important if/when secondary dominants are under consideration.

Acceptance criteria:

- The DSL includes a command to list the notes of the harmonic minor scale starting on a given note.
- Only one octave of the ascending scale is listed, starting with the note specified.
- As for the other scale types, notes are shown as the corresponding letters, together with any sharp/flat symbols and if an octave is specified for the starting note then the notes listed also includes the corresponding octave(s).
- The DSL includes a command to play a specified harmonic minor scale. The starting note and type of scale are given (e.g. ‘play scale F harm min’). The notes played appear on the transcript as usual.
- By default, the scale is played ascending but there is an option to specify descending or both ascending and descending.
- The scale recognition tutor has an option to include harmonic minor scales.
- By negotiation, if it is convenient to include the mixb2b6 (5th mode) then this can be also done.

59. Note & rest duration

So that I can achieve a wider range of musical options, as a user, I need to be able to access notes and rests of various durations.

Notes: Thus far we have considered crotchets as the basic counting unit of length/duration for notes and silences with the corresponding time interval depending on the current tempo. Periods of silence are also called 'rests'. In more realistic music than we've encountered thus far, notes and rests also occur with durations longer and shorter than crotchets. For our purposes, we'll need to handle 2-crotchet (minim) and 4-crotchet (semibreve) durations as well as half-crotchet (quaver) and quarter-crotchet (semi-quaver) durations. We'll also need to handle "dotted" notes & rests (and also "double-dotted" ones). Here 'dot' comes from the conventional (aka fly sh*t) notation: a dotted note or rest is 1½ times its basic duration and a double-dotted one is 1¾ times its basic duration. Thus a dotted crotchet has the same length as 3 quavers, and a double-dotted crotchet is the same length as 7 semi-quavers. For more detail see e.g. https://en.wikipedia.org/wiki/Note_value.

Acceptance criteria:

- The default unit for note duration is crotchet at the current tempo. The unit duration may be set by an appropriate DSL command (e.g. for very fast or slow tempos).
- Commands that involve playing notes continue to do so at the default duration (though this may not be a crotchet as previously). Play commands have an appropriate option in order to support notes of different durations e.g. playNote(C4, quaver) at the current tempo and counting unit settings.
- Ditto via PK keyboard (need shift/alt etc for crotchet/quaver and dot)
- Similarly, the relevant commands can also play rests of different durations e.g. playRest(quaver).
- The defaults for scales etc. can be specified as desired (e.g. quavers) at the current tempo and counting unit settings.
- Where/if necessary, durations for notes entered via the PK keyboard can be controlled (e.g. with shift/alt/ctrl keys).

60. Melody phrases

So that I can work with simple melodies, as a user, I need to be able to edit and hear musical phrases.

Notes: We currently only need to deal with contexts where at most one note is playing at any given time. Another way to think about this is that at any time one "thing" (a note or a rest) is playing. If a time signature is available then we can check if a bar is valid. We would then know how many beats are in a bar, and their duration: a valid bar is "full" of notes and rests.

For example, in common time a bar containing 2 crotchets, 2 quavers and a crotchet rest is legal but a bar containing just 3 crotchets isn't.

See also: *Record & playback sounds*

Acceptance criteria:

- A DSL command is available to play a phrase given as a sequence of notes and rests
- An editor is available (via DSL and GUI) for entering and updating melody phrases.
- Melody phrases may be entered as a sequence of notes and rests using an appropriate notation.
- If a PK keyboard or other suitable input source is available then the editor may utilise it.
- Melody phrases may be saved and subsequently re-loaded. If a user is logged on then the saved content is associated with that user.
- Some (not too many!) examples are provided. These might include such things as:
    - The 2-note melody heard when we mount/unmount a USB device.
    - The whistle ringtone commonly found on Samsung phones (https://www.youtube.com/watch?v=Z5h411OcttA)
    - The classic Nokia ringtone (https://www.youtube.com/watch?v=QgjVxFcBO0c)
    - The "dit dit dit dah" motif that opens Beethoven's 5th symphony.
    - The Close Encounters of the Third Kind motif (e.g. C4 D4 C4 C3 G4)

61. Visualising scales

So that I can better understand the relationships between scales, as a user, I need to be able to visually examine and compare scales.

Note: In earlier stories (such as *major scales*, *Major scale modes* and *Melodic minor modes*) tables were used for this sort of thing. This story is about finding effective ways to highlight patterns, differences etc in order to help us learn and recall. For example, we can think of the modes of the major scale as two "families", those with a minor 3rd (Dorian, Phrygian, Aeolian, Locrian and those with a major 3rd (Ionian, Lydian, Mixolydian). The family members differ only in (groan!) minor ways, making them easier to remember: Lydian is just Ionian with a sharp 4, mixolydian is just Ionian with a flat 7. Similarly, phrygian is just aeolian with a flat 2 and so on. Similarly, other relationships can be identified: relative major/minor, natural/melodic/harmonic minor and so on. Visualisation might be in terms of the PK keyboard, a layout based on even semitones or other relevant visual models.

Acceptance criteria:

- The user can select a scale for display. The corresponding notes are then highlighted on the PK keyboard.

- The user can select two scales for display. The corresponding notes of both scales are then highlighted on the PK keyboard.
- The display should highlight the similarities and differences between the two. There will be some notes in both scales, some in one but not the other and *vice versa*.
- When two scales are displayed together, the user can toggle the display (e.g. by pressing/releasing a key) to see the differences clearly as the display flips between the scales.

62. Personas/User groups

So that the application will be a pleasure to use, as a user, I need its interface to meet my usability and domain-specific needs.

Note: The deliverables from this story are primarily designs and will need to be approved by PO before their implementation is included in future stories.

Acceptance criteria:

- One or more target user groups are identified and named.
- The factors/needs relating to each group are listed and described clearly.
- Factors common to more than one group are identified.
- Personas exemplifying the characteristics of members of target user groups are developed.

63. Record & playback sounds

So that I can access subsequently the sounds played by the program, as a user, I need to be able to capture them.
Note: At this stage audio file formats (e.g .WAV or .mp3) are not required. We are primarily concerned with (combinations of) notes and rests. Soon we are likely to want to play melody/scales and chords together.
See also: *Melody phrases*

Acceptance criteria:

- The DSL includes commands to start and stop recording.
- For convenience, appropriate GUI elements (e.g. menu items or toolbar icons) are provided: they in turn use the corresponding underlying DSL commands.
- When recording is activated, sounds played (e.g. via DSL commands, melody phrases or the PK keyboard) are recorded in an appropriate manner.
- The DSL includes a command to (re-)play recorded sounds. If no sound has been recorded then an appropriate message appears on the transcript. A corresponding GUI element is provided, and this is disabled when nothing has been recorded.

- The DSL includes commands to pause and resume recording & playing. A corresponding GUI element is provided and is enabled/disabled as appropriate.
- While recording is paused, DSL commands (including those that generate sounds) may be carried out but result in no further change to the recording. If recording is resumed then subsequent sounds are recorded; recording may be stopped from a paused state.
- The DSL includes a command to save recorded sounds to a file. A corresponding GUI element (such as a menu item) is provided as a convenient short-cut. If no recorded sound is available then a suitable warning is given and nothing is saved.
- If the command to start recording is given, and an unsaved recording exists then the user is warned and given the option to save or discard it.
- The DSL includes a command to play previously-saved recorded sounds from a file. A GUI shortcut is provided for convenience. During play, commands such as pause and stop may be used as for recordings not previously saved.

64. Practice planning and self-rating

So that I can become even more awesome, as a musician, I need to be able to manage my practice time efficiently and effectively.

Note: Practising anything (a musical instrument, sport, card tricks, …) requires discipline and planning. We don't want to waste time on things we can already do; we want to challenge ourselves but not attempt things beyond our reach so we get discouraged; we want variety so we don't get bored; we want to be able to see the progress we have made. For our purposes, a practice schedule involves a sequence of activities (play a scale, play a chord, play an interval, answer some questions, …) that are already available in the system.

Acceptance criteria:

- The DSL & GUI provide a feature for designing practice schedules. If story *Users* has been implemented, practice schedules are associated with individual users.
- The user can select activities to be included in a schedule. Playing a nominated scale and playing a nominated chord are essential: further activities are negotiable. The number of occurrences of each activity may be specified (e.g. 10 major scales, 10 melodic minor modes, 5 chords and 5 general knowledge questions). The activity order may be specified (e.g. the chords first, then the scales, then …) or the user can allow the system to determine the order at random.
- A difficulty level may be set for each schedule. The default is the Goldilocks value (not too easy, not too hard) on the corresponding difficulty scale.
- A practice schedule can be saved. Saved practice schedules can be re-loaded, edited and saved again.

- The DSL & GUI provide a feature for generating (instantiating) practice session content from practice schedules. This involves choosing examples of each activity type. For the example schedule above, this might involve playing the G major scale, the C# Lydian dominant scale, the BbMaj7 chord, answering a question about the meaning of a term and so on.
- Selection is initially random. If ratings are available (see below) then these are used to select an appropriate mix of activities. For example, a medium difficulty session might include a few easy or hard activities while mainly consisting of medium difficulty activities.
- The DSL & GUI provide a feature for running the generated practice session.
- The user is prompted to carry out each activity in turn (e.g. “Play C major scale).
- For some activities (such as general knowledge questions) the user response will determine whether or not they were achieved. Whether or not the user “knows” these facts is recorded in an appropriate way.
- For others (e.g. playing scales) the user will indicate a difficulty/achievement/confidence rating. The ratings are chosen from an ordinal scale (easy, medium, hard, impossible; gold, silver, bronze, aluminium; …).
- When all the activities have been completed, the session results may be saved. The saved results include the ratings and whether the user answered question-style activities correctly.
- The saved results determine the selection of activities in future instantiations of schedules.

65. GUI/Usability design

So that I can maximise my efficiency and minimise errors and frustration, as a user, I need a the application to be designed for consistency and ease of use.

See also: *Personas/User groups*

Note: Quite a few features have been added to the application of the last few months. You are possibly the most experienced users of your own application: this has pros (e.g. You know all the features well) and cons (e.g. you don’t have the perspective of a new user). While the DSL engine remains a core architectural feature, end users don’t interact with it directly so often as richer GUI features evolve. Perhaps your application is evolving to meet the needs of a target user community (adult learners, young children, blind people, …) as exemplified by personas. Perhaps the application has been given to real users. The deliverables from this story are primarily designs and will need to be approved by PO before their implementation is included in future stories.

Acceptance criteria:

- The display of DSL command entry fields is a user option. In any case, corresponding DSL commands are generated and appear on the transcript.
- Similarly, the transcript need not be visible at all times and may be optionally displayed/hidden by the user.
- GUI is appropriate for the application's target user groups/personas

66. Half-whole & whole-half tone scales

So that I can work with a wider range of musical styles, as a user, I need to be able to work with half-whole scales.

Note: The notes of a half-whole scale are the root and the notes obtained by going up a semitone, then a tone, then a semitone, then another tone, repeating this pattern until the next occurrence of the root (12 semitones away). This generates an 8-note scale (sometimes referred to as 'octatonic' or 'diminished' scales) so the spelling rules seen in earlier stories can't be used fully. For example C half-whole scale consists of:

0	1	3	4	6	7	9	10	12
C	Db	Eb	E	F#	G	A	Bb	C

Interestingly, there are only three distinct patterns. To see this, consider any note as the start of whole tone, end of a whole tone, or middle of a whole tone interval. For example, C can be the start of a half-step (as in C whole-half above), the end of a half-step (as in D half-whole: D, Eb, F ,F#, G#, A, B, C) or the middle of a whole step (C# half-whole: C#, D, E, F, G, Ab, Bb, B, C#). Even more interestingly, the whole-half scales are obtained from the same three sequences: they begin with a tone interval and correspond to simply starting one note further along. For the above example of the C half-whole scale, the Db whole-half scale is (Db, Eb, E, F#, G, A, Bb, C, Db). These scales have some important applications.
Half-whole scales sound good with dominant chords: this is because they contain the root, 3rd, 5th and flattened 7th. Thus we might bust out C half-whole when playing over C7 or its big brother C13b9. Whole-half tone scales contain the notes of the corresponding diminished chord so C whole-half would sound cool over Cdim7 or its inversions.

Acceptance criteria

- The DSL includes a command to list the notes of the half-whole and whole-half scales starting on a given note.
- Only one octave of the ascending scale is listed, starting with the note specified.

- As for the other scale types, notes are shown as the corresponding letters, together with any sharp/flat symbols and if an octave is specified for the starting note then the notes listed also includes the corresponding octave(s).
- The DSL includes a command to play a specified half-whole or whole-half scale. The starting note and type of scale are given (e.g. 'play scale F hw'). The notes played appear on the transcript as usual.
- By default, the scale is played ascending but there is an option to specify descending or both ascending and descending.
- The scale recognition tutor has an option to include half-whole and whole-half scales.
- Any other feature involving scales can also include half-whole and whole-half scales.

67. Whole tone scales

So that I can work with a fuller range of musical options, as a user, I need to be able to work with whole tone scales.

Note: Whole tone scales contain 6 notes a tone apart. There are only two distinct patterns: any whole tone scale is obtained by starting at the appropriate place in one of them.

Semitones above root	**2**	**4**	**6**	**8**	**10**	**12**
C	D	E	F#	G#	A#	C
B	Db	Eb	F	G	A	B

Whole tone scales contain the root, 3rd, sharpened 5th and flattened 7th: they sound good over augmented chords.

Acceptance criteria

- The DSL includes a command to list the notes of the whole-tone scale starting on a given note.
- Only one octave of the ascending scale is listed, starting with the note specified.
- As for the other scale types, notes are shown as the corresponding letters, together with any sharp/flat symbols and if an octave is specified for the starting note then the notes listed also includes the corresponding octave(s).
- The DSL includes a command to play a specified whole-tone scale. The starting note and type of scale are given (e.g. 'play scale F wt'). The notes played appear on the transcript as usual.
- By default, the scale is played ascending but there is an option to specify descending or both ascending and descending.
- The scale recognition tutor has an option to include whole-tone scales.
- Any other feature involving scales can also include whole-tone scales.

68. Scale chord combination

So that I can explore the principles of harmony, as a user, I need to be able to see and play a chord + scale combination together so in order to study the fit.

See also: *Chord scale match*

Note: Some aspects of the relationship between scales and chords have been encountered in previous stories. Given a major scale, we can construct a family of diatonic chords. Since every chord note is in the corresponding scale they will always sound "right' when played concurrently with the scale and any melody containing only notes drawn from it. For example (see story *Lead sheet (basic)*) the first 2 bars of *All of me* has a melody line containing only notes from C major scale over a Cmaj7 chord. Similarly, (see story *Major scale modes*) modes tell us other scales that fit together. For example, D dorian will "work" over Dmin7 chords and G mixolydian will be a great option over G7. Taking this a step further, we could potentially use any scale that has the chord tones in it.

Acceptance criteria:

- The DSL has a command (also accessible via a suitable GUI action) to select/specify a chord and a scale. Various operations may then be performed on this chord/scale combination.
- Display: This option lists the notes of the scale alongside those of the chord. Precise details are negotiable. One very basic option would be some tabular format like this:

Bm7b5	B		D		F		A
B locrian	B	C	D	E	F	G	A

- The display indicates/highlights chord notes that are not in the scale. These notes will clash --- maybe in a good way ('out' notes) maybe not. For example (assuming key of C), the V chord would normally involve G7 + G mixolydian. A tritone substitution to bII7 would replace this with Db7 + Db lydian dominant. The major 3rd and flat 7 that define the sound of the G7 are also in Db7 but the Db lydian dominant scale introduces some 'crunchy' notes to the mix.
- Play: The chord and scale are played concurrently. Options are available to specify:
    - the scale pattern as ascending, descending, both or some digital pattern
    - the number and duration of chords. For example, the chord may be repeated with each scale note and have the same duration as a scale note. One useful option would be to have the chord played once and sustained for the duration of the scale. Another would be to have the chord repeated every 4 scale notes.

    - the instruments to be used for the chord and scale. For example, it may be nice to have an instrument with good sustain (e.g. an organ) for the longer lasting chords and a lower sustain instrument (e.g. piano) for the scale. The default instrument is used for both unless overridden.
    - The number of repetitions. The default is one. A number (positive integer in a sensible range) of repetitions may be given. There is also a loop option which continues to repeat the pattern until interrupted.
- By negotiation, scale + chord combination material may be included in tutors

69. Chord scale match

So that my composition and soloing will be better, as a user, I need to be able to find appropriate scales to play.

See also: *Scale chord combination*

Notes: An appropriate scale is one that includes the notes of the corresponding chord. There may be more than one scale that satisfies this requirement. Most music involves chord progressions where the chords change often (e.g. in common time you'll often see combinations of 2 chords per bar, 1 chord per bar and 1 chord per 2 bars). In a major key, it is straightforward to find one appropriate scale for diatonic chords (see stories *Diatonic chord function*, *major scale modes*) and secondary dominants (see story *Secondary dominants*). The lydian dominant scale (see story *Melodic minor modes*) is used over non-diatonic dominants. This works because the sharpened 4th (#4 or #11) in the scale is the root of the original (before TT sub) chord. There may be more than one possible scale match for a given chord. For example, both C half-whole and C mixolydian would work over C7.

Acceptance criteria:

- The DSL includes a command (also accessible via GUI) to initiate a chord match. The user can select or specify a chord.
- The system will suggest possible scale matches, each of which contains all the chord notes. For example, 'scaleMatch(G7)' might give 'G mix, G hw'. If no matches are available then this is reported.
- The user can select any of the matches to be played as described in *Scale chord combination*.

70: Microphone input (Karaoke mode)

So that I can more effectively test my knowledge, as a user, I need to be able to input audio data to the system tutors.

Note: If microphone input is available then the user can sing (or whistle or play an instrument such as a trumpet) notes, intervals, arpeggios and scales in response to questions from tutors. Identifying the pitch of the audio input allows comparison with the correct response. A scoring system based on the accuracy of the pitch could also be developed.

Acceptance criteria:

- Tutors, such as the interval tutor, can be configured to respond to input from a microphone if one is available.
- Some question types may need modification in this context. Other new question types may be provided to take advantage of audio responses.
- The user can sing (or whistle or …) responses. The input is analysed to determine the predominant pitch for comparison with correct responses.
- An answer may be assessed as correct if it is within an acceptable tolerance from the correct one. The tolerance can be configured (e.g. beginners or non-singers are likely to require more generous tolerances than experts).

71. Tap tempo

So that I can more conveniently set the tempo, as a user, I need to be able to demonstrate the tempo I want.
See also: *Tempo*

Acceptance criteria:

- An option is available (DSL and GUI) to allow the tempo to be set by tapping.
- When this option is enabled, the user can set the tempo by tapping several times on a designated key. This may be a key on the "regular" keyboard or on the PK keyboard (in which case tapping will involve clicking with a mouse or similar device).
- By default, the taps are interpreted as crotchets. For very fast or slow tempos it may be necessary to interpret taps as longer (e.g. minim) or shorter (e.g. quaver) durations: an appropriate DSL and GUI option is available for this.
- When the taps are sufficiently regular that the tapped tempo can be detected the system tempo is set to the corresponding value and the new value is displayed (via the transcript and appropriate GUI element).
- The precise definition of sufficiently regular is negotiable: most humans will not be able to tap in an exactly regular sequence. It will typically involve a sliding window of

3 or 4 taps where the intervals between taps are sufficiently close. An optional sensitivity property may be provided.

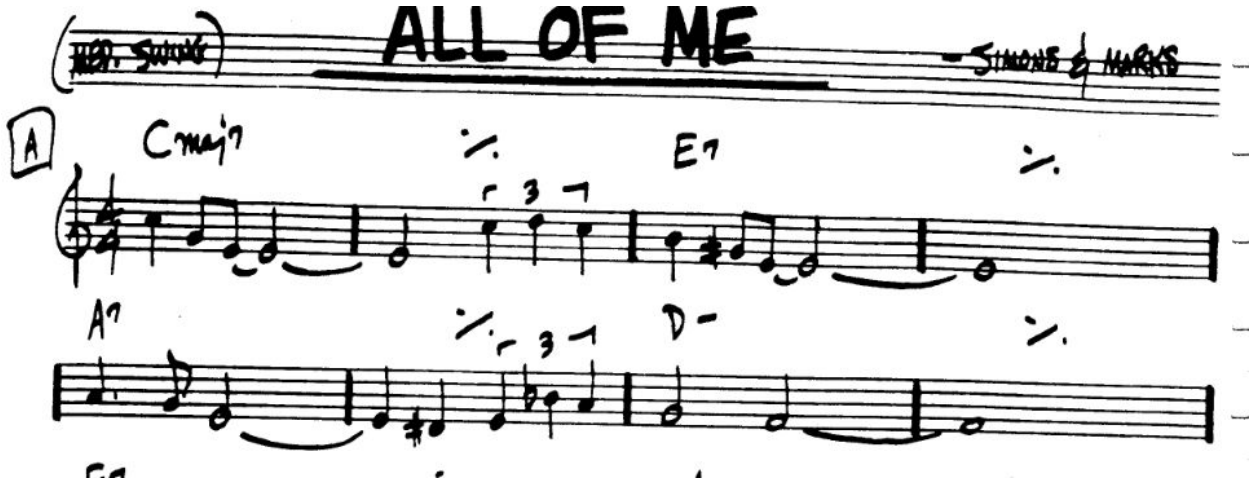
- If microphone input is available, then this may also be used to set the tempo. Suitable DSL and GUI options are available to enable "audio tapping" and the user will generate "taps" by clapping or making suitable sounds.

72. Lead sheet (basic)

So that I can write and understand music, as a user, I need to be able to specify the basic structure of a piece.

Note: For now we assume that all bars are the same length, only one key signature applies and only one time signature applies. We also assume that there is no pickup/anacrusis (lead-in notes before the first beat of bar 1) so all bars are complete. In later stories we may relax these assumptions.

Here's part of a "real" lead sheet:



Note that this one has more than just chord symbols: there's a style (medium swing), title, composer name, treble clef, time signature, key signature (no sharps or flats so it's C major) and melody notes as well. Initially, we will ignore the melody and just show the chords and bars like this:

CM7 | CM7 | E7 | E7 |
A7 | A7 | Dm7 | Dm7|

This simplified form is still very useful. For example, we can play the chords while a musician reads the (simplified) lead sheet and improvises a solo.

Acceptance criteria:

- The DSL includes a command to specify basic lead sheet detail. For convenience, the GUI also provides one or more dialogs to specify the relevant information.
- Details include:
  - Title, author
  - Number of bars (if this is not able to be inferred from the content)
  - Time signature
  - Key signature
  - Style/tempo directions (e.g. 'medium swing' in the example above)
- The lead sheet can be displayed in a suitable format. If it is broken up into lines then it is preferable to have equal length lines with 4 or 8 bars on each. This isn't always achievable, but helps the real-time reader/player considerably.
- Chords may be added, modified or deleted. There can be one chord per bar and this is quite common. In time signatures with even numbers of beats per bar (e.g. common time) then it is also common to have 2 chords per bar. For example, we might see
  | CM7 Am7 | Dmin7 G7| CM7| .
  Other combinations, such as one chord per beat, arise less frequently and can be ignored for now.
- Lead sheets may be saved and re-loaded for subsequent editing.

73. Play lead sheet (basic)

So that I can work with lead sheets in practice, as a user, I need to be able to hear what the chords sound like while I read or play along.

See also: *Lead sheet (basic)*

Acceptance criteria:

- The DSL & GUI provide the ability to play a currently-loaded simplified lead sheet.
- The chords are played in turn at the current tempo with the currently selected instrument. If there is more than one chord in a bar then the duration of the bar is divided evenly between them.
- The use can specify the behaviour when the end of the lead sheet is reached: playback can stop or loop back to the beginning to repeat a specified number of times.
- While the lead sheet is playing, the current chord is highlighted so the user can follow along.

74a: Alternate interface (basic)

So that I can access the system on more platforms, as a user, I need to be able to access it via my web browser.

Note: The DSL supports a clean separation between core functionality and user interfaces. In principle, it should be straightforward to deploy the DSL processor and related components on any Java platform. Developing additional user interfaces is then, in principle, simply a matter of using the platform of choice and connecting to the DSL engine. However, Yogi Berra's adage "In theory there's no difference between theory and practice. In practice there is." is likely to apply. A web interface (e.g. using Angular) is potentially attractive as this would also allow users convenient access to the application on a range of devices including mobile devices such as tablets and phones.

Acceptance criteria:

- A web interface is available. The precise technical details are negotiable, subject to PO approval, but simplicity is a factor. AngularJS is one likely candidate for consideration.
- As little as possible should be re-implemented. In particular, the DSL engine should be as platform-independent as practicable.
- The web interface should be convenient to use on a range of devices and form factors: phones and tablets are target platforms as well as laptops and desktop machines.
- Some features may be unsuitable or impracticable on particular platforms or form factors. Details to be negotiated with PO. The idea is to start with simple, high-value features and extend as appropriate.
- Priority should be given to features that support practice schedules (e.g. Suggestions for scales to play, with facility to self-assess achievement, flashcards or simple tutors for study aids)

74b. Mobile app (native)

In order to overcome the limitations of a web-based application, as a user, I want to be able to have a native app that I can use on my phone and tablet.

Acceptance criteria:

- The system is available as a native (Android or iOS or both) app.
- As little as possible should be re-implemented. In particular, the DSL engine should be as platform-independent as practicable.
- The app should be convenient to use on a range of phones and tablets with potentially differing form factors.

- Some features may be unsuitable or impracticable on particular platforms or form factors. Details to be negotiated with PO. The idea is to start with simple, high-value features and extend as appropriate.
- Priority should be given to features that support practice schedules (e.g. Suggestions for scales to play, with facility to self-assess achievement, flashcards or simple tutors for study aids)
- Features involving playing sounds as well as utilising the device's microphone should also be prioritised.

75. GUI/Usability improvements

So that I can maximise my efficiency and minimise errors and frustration, as a user, I need a consistent and easy-to-use application.

See also: *Personas/User groups, GUI/Usability design*

Note: This story involves implementing the designs delivered in previous stories

Acceptance criteria:

- Design is implemented and tested
- At least one user from each target user group has used the new GUI and given feedback
- By negotiation, a heuristic evaluation or other lightweight evaluation may be conducted. Specific outputs include design review proposals for future stories.

76. Full potential chords (major key)

So that I can work with richer chords, as a user, I need to be able to see and hear full potential chords.

See also: *Chords (4-note), Play chords (revisited), Chord recognition tutor (revisited), chord spelling tutor, Diatonic chord function (major)*

Note: Thus far, we have mostly considered 4-note chords constructed from stacked thirds (the "take one, leave one out" approach). If we continue beyond four notes then "bigger" chords can be built: these can contain 9ths, 11ths and 13ths. Eventually, the original tonic will be encountered so the process must terminate after 7 notes. In practice, some notes sound "bad" together in the same chord (long story, best kept for another day) or spoil the effectiveness of the common chord progressions in which they would appear (ditto). Removing these gives the (diatonic) full potential chords (FPCs). There's a lot more to "big" chords than this and we'll meet other chord types in future stories. For now we'll use C major as an example. The following table shows familiar the 4-note (7th) chords and the corresponding diatonic FPCs. There are additional chords between the 4-note chord and its

corresponding FPC but these are not shown in the table. They have the same essential quality but contain additional notes. For example, Dm9 (D, F, A, C, E) and G9 (G, B, D, F, A) would be valid diatonic chords.

		FPC(diatonic)	**1**	**3**	**5**	**7**	**9**	**4/11**	**6/13**
I	Cmaj7	Cmaj7(6/9)	C	E	G	B	D		A
II	Dm7	Dm11	D	F	A	C	E	G	
III	Em7	Em7(add4)	E	G	B	D		A	
IV	Fmaj7	Fmaj76/9(#11)	F	A	C	E	G	B	D
V	G7	G13	G	B	D	F	A		E
VI	Am7	Am11	A	C	E	G	B	D	
VII	Bm7b5	Bm7(b5)add4	B	D	F	A		E	

Acceptance criteria:

- The DSL commands implemented in story Diatonic chord function (major) need to be extended.
    - The DSL command to obtain the quality of the chord for a given degree currently returns the 4-note (7th) chord. This remains the default, but there is an option to obtain the corresponding FPC. Thus ‘qualityOf(II)’ might yield ‘m7’ and ‘qualityOf(II, fpc)’ might yield ‘m11’
    - By negotiation, an option to list all chords for the degree may be implemented. Thus, ‘qualityOf(II, all)’ might yield ‘m7, m9, m11’
    - Similarly, the DSL command to obtain the chord for a given function and key continues to return the 4-note chord by default but has an option to yield the corresponding FPC. For example, ‘chordFunction(IV, D, major, fpc)’ might yield `GMaj76/9(#11)’. By negotiation, an option to list all chords may be implemented. For example, ‘chordFunction(IV, D, major, all)’ might yield `GMaj7, GMaj9, GMaj9(#11), GMaj76/9(#11)’.
    - The DSL command which, when given a chord and key, gives the corresponding function (or an appropriate message if there isn’t one) also needs to be extended.
- Where appropriate, tutors are extended to include diatonic FPCs.
- The DSL command to play chords can accommodate chords as big as the FPCs

77. Percussion notation

So that I can compose and study backing rhythms, as a user, I need to be able to write down details of the percussion components.

Note: We only require basic notation elements at this stage. "Fancy" notation details (e.g. choking cymbals) should be omitted unless negotiated with your PO. Percussion notation is relatively simple: it essentially involves specifying which instruments are available and when to hit them. Some forms of notation (see e.g. https://en.wikipedia.org/wiki/Percussion_notation) are based on the regular stave (see domain background) while others (see e.g. https://en.wikipedia.org/wiki/Drum_tablature) involve simplified text-based notations.

Acceptance criteria:

- A percussion notation editor is available via DSL and GUI control.
- A new document is the default, but previously saved percussion notation documents may be opened.
- Percussion notation can be entered and modified using appropriate keyboard and mouse actions.
- Instruments may be specified as required.
- The notes/rests and their durations associated with playing each instrument may be specified in the agreed notation.
- The document can be saved so that it can later be re-opened and updated.

78. Play percussion

So that I can work with style-appropriate rhythm, as a user I need to be able to play percussion loops.

Note: For most purposes a short (e.g. 1 or 2 bar) rhythm can be repeated as a loop for a whole song. Examples include the bossa and skank (from Latin and reggae respectively). See also: *Percussion notation*

Acceptance criteria:

- A percussion notation document (previously created as per story *Percussion notation*) can be played (using one or more appropriate instruments).
- Playback may be paused or stopped at any point.
- The percussion piece is repeated (looped) when the end is reached. The maximum number of repetitions may be specified, otherwise the loop continues until paused/stopped.

79. Overdub melody (chords)

So that I can work with melody and scales in context, as a user, I need to be able to add them to an existing set of chords.
Note: This story involves adding a melody to an existing chord structure provided by a lead sheet. While the lead sheet chords are being played, the additional melody is recorded on top of it.
See also: *Lead sheet (basic), Play lead sheet (basic), Overdub melody (percussion)*

Acceptance criteria:

- While a lead sheet containing a set of chords is being played, the user can play a melody (e.g. via the PK keyboard).
- The melody can be discarded and another melody added in its place.
- The user can save the combined leadsheet (i.e. chords + melody)
- The saved (chords + melody) lead sheet may be re-loaded and further melodies added as desired.
- If rhythm loops are available then these may (by negotiation) also be included.

80. Overdub melody (percussion)

So that I can work with melody and scales in context, as a user, I need to be able to add them to an appropriate rhythm..
Note: This story involves adding a melody to an existing rhythm pattern provided by a percussion notation document. While the percussion loop is being played, the additional melody is recorded on top of it. It is common (though not always ideologically sound) for a tune to be performed in various styles (e.g. there are swing and bossa versions of *All of Me*, and punk versions of just about everything). So our users will find it helpful to practice playing over different rhythm loops.
See also: *Lead sheet (basic), Play lead sheet (basic), Overdub melody (chords)*

Acceptance criteria:

- While a rhythm loop is being played, the user can play a melody (e.g. via the PK keyboard).
- The melody can be discarded and another melody added in its place.
- The user can save the combined leadsheet (i.e. rhythm + melody)
- The saved (rhythm + melody) document may be re-loaded and further melodies added as desired.
- If lead sheet chords are available then these may (by negotiation) also be included.

81. Drum pad

So that I can work more effectively with percussion, as a user, I need tools to help me play and analyse percussion music.

Note: The PK keyboard gave us a convenient and natural way to interact with parts of the system involving pitched tones. The drum pad performs a similar role for percussion. In the real world, physical drum pads are used to provide MIDI input. We will model these with a GUI component having particular areas corresponding to individual percussion instruments. These may be in a regular grid or may be laid out in a more "realistic" manner. We have (from story *percussion notation*) an editor for creating percussion scores. This story builds on that.

Depends on: *Percussion notation*

Acceptance criteria:

- A drum pad is available. It may appear in a separate window, or in a pane in the application GUI. Details by negotiation subject to PO approval.
- The details of pad layout keyboard and number of instruments are negotiable but should be sufficient to support the scores produced in story *Percussion notation*.
- Each "instrument" on the pad can be configured to one of the available percussion instrument types. While the piano is technically a percussion instrument, we're thinking here of drums, woodblocks and the like.
- Some form of labelling (e.g. tool tips) is available to indicate the current mapping between instruments and pad.
- Normally, only single notes need be "played" at a time. However, a means of selecting several notes to be played together (i.e. a chord) is also necessary. Something like Shift-Click might be sufficient. If a multi-touch screen is available then it could be used for this purpose.
- There is a visual indication of which instrument is playing (e.g. the corresponding "drum" is highlighted.
- The percussion notation editor can be configured to display whatever is being played on the pad.
- New notation appears at the cursor position. The user can move the cursor in the editor. Subsequent input can be configured to either replace or overlay current notation. For example, it is possible to build up complex notation by adding one instrument at a time.
- A percussion notation file may be loaded and then played on the pad.