Software Requirements Specification

for

PolyChord

Version 0.0.1 approved

Prepared by Devin Christianson, Alexandre Feren, Alexander Revello, Dawsin Blanchard

DAJ

13 February 2020

Table of Contents

Table of Contents ii

1. Introduction

1.1 Purpose

This Software Requirements Specification documents the whole of revision 0.0.1 of the PolyChord application. This includes the base framework of PolyChord, as well as it's core features, such as the keyboard interface with synth capabilities, and basic chord recognition. This document will be updated to include later revisions of PolyChord, which will expand on the above feature set, adding additional functionality that is important to our users, such as the ability to save and restore chords, download/upload MIDI files, and much more.

1.2 Document Conventions

All requirements stated in this SRS will be specified for the particular item, i,e, the priority of each specification will be explicitly stated for each requirement and will be sorted by priority. Each section will be bullet-ed, and subsections will be notated by the depth of the section with pound signs, so that a section will have 1, subsection 2, etc. Enumerations of specifications may be noted by either a - or by a corresponding numbering convention for priorities. Sections may be added removed or modified dependent on the changes in specification as development progresses.

1.3 Intended Audience and Reading Suggestions

This Software Requirements Specification is intended for developers, project managers, and marketing staff. The document is laid out in a number of multi-part sections.

The Overall Description section covers the product perspective, product functions, user types, operating environment and constraints, user documentation plans, assumptions and dependencies. The product functions, user types, operating environment and constraints, user documentation plans, and assumptions sub-sections are important reading for product managers, the product functions, operating environment and constraints, assumptions and dependencies are important reading for developers, and the product perspective, product functions, user types sub-sections are important reading for marketing staff.

The External Interface Requirements section covers the user Interfaces, hardware interfaces, and communications interfaces. All three sub-sections are important reading for project manager and developers, while the user interface sub-section is useful reading for the marketing department.

The System Features section covers the major features of PolyChord, and all its subsections are important reading for developers and project managers alike.

The Other Non-Functional Requirements section covers the performance, safety, security, and software quality requirements, as well as any business requirements that may be in play. This is important for project managers and developers.

1.4 Product Scope

PolyChord is a interactive music theory learning tool, that is designed to be more engaging and dynamic than more traditional music theory tools out there. What sets PolyChord apart is that it identifies chords as they are played, allowing the user more freedom than a traditional chord identification or learning tool. PolyChord is not intended to be a full-featured online synthesizer, nor is intended to be a fully-featured music theory learning classroom. PolyChord aims to be a supplement for these other resources, providing an easy way to re-enforce or refresh music theory know, or gain a working knowledge of basic music theory through experimentation.

1.5 References

This document depends on the Use Case model document, as that is where PolyChord's features are laid out in more detail, it terms of what the user interaction and system response looks like.

2. Overall Description

2.1 Product Perspective

PolyChord is intended to be a standalone music theory experimentation product, and is not part of a larger system, or a descendant of an existing product. The inspiration for PolyChord was the lack of music theory learning tools that were designed with the playing experience first: the majority of applications are either designed to be a reference tool, or designed to be a guided introduction to music theory. PolyChord aims to foster music theory through experimentation: the main interface for the application is a synthesizer, so it is easy and intuitive to play chords by ear and learn new chords from PolyChord's chord database.

2.2 Product Functions

The major functions the product must perform, or allow the user to perform, are as follows:

- PolyChord must allow the user to play a piano-style synth via a keyboard or MIDI device input.
- PolyChord must allow the user to play selected chord by name to be show/played.
- PolyChord must display chords by name as they are played by the user.
- PolyChord must allow the user to select a Chord progression to be show/played.
- PolyChord must display chord progression suggestions based on the chords the user is playing, and the desired "feel" selected by the user.
- PolyChord must have a tutorial for basic chords and music theory ideas.

2.3 User Classes and Characteristics

The general user classes and characteristics are as follows:

- Casual User: This group of users is likely to experiment with many of the product features, but less likely to make wide or continued use of any of them. To maximize their user experience, all features must be as intuitive and straightforward to use as possible

- Beginner/Novice level of knowledge in music theory: This group of users is the most likely to use the chord play-back feature to learn new chords, and the chord recognition feature to practice chords. To maximize their user experience, the chord playback feature must not require any prior knowledge of music theory to use, and the chord recognition feature must be responsive. This level of user may also experiment with the chord progression suggestion tool, so that tool must be able to display the chords on a "piano roll" style output, as that is the type of output a user with this level of knowledge is most likely to use.

- Intermediate level of knowledge in music theory: This group is the most likely to use the chord progression playback and progression suggestion tools, for both learning purposes as well as reference purposes, so the progressions must be based on the chords/key they are playing in, and the playback tool must provide the chords on a staff as well as "piano roll" style, as this level of user is more likely to make use of the "ledger-line" display than the "piano roll" display.

- Advanced level of knowledge in music theory: This group is most likely to use PolyChord as a reference tool, in the event that they have forgotten a chord or chord

progression, or just want to check their work. To maximize their user experience, all tools must be capable of “ledger-line” style output, as that is the type of output that they are the most comfortable with. This is also the level of user who is most likely to use a MIDI device for input, so MIDI input should be an option for all keyboard-input type tools.

2.4 Operating Environment

PolyChord is a web application, and therefore is inherently fairly operating-system agnostic, and aims to support as many browsers as possible. For features other than Midi support, universal support of all modern browsers is trivial. For Midi support, only newer versions of Chrome, and Safari support web Midi natively, with some others requiring a 3rd party extension for Midi support. Unfortunately, due to the nature of the Midi API, there is not much PolyChord can do in this regard other than fail gracefully in the face of lack of Midi support, and continue functioning normally otherwise.

2.5 Design and Implementation Constraints

There are few implementation constraints for the application as a whole, as it only has the standard requirements for a modern website, requirements that are met by all modern browsers. For Midi support specifically, there are a few implementation constraints. First, as mentioned in the previous sub-section, WebMidi is not supported on all modern browsers. Second, WebMidi has some inherent security concerns with “sysEx” webMidi calls. In using the WebMidi.js framework, the initial matter of disabling “sysEx” calls is taken for us, as it is disabled by default, though there are still further actions to mitigate this vulnerability: in order to satisfy the requirements of some modern browsers, and prevent man in the middle attacks that could erroneously enable “sysEx” calls, PolyChord must be served elusively over HTTPS.

PolyChord does plan to implement user accounts later in the development processes, so as that point approaches, research will need to be done on the rules and regulations that must be followed for the data we determine we need to collect. At this point, our goal is to collect as little information as we can from users, likely just an email to attach to the account, but since that has not yet been decided, it is hard to know exactly what regulation we will need to adhere to, though it is almost guaranteed we will at least comply with regulations such as GDPR and the California Consumer Privacy Act when handling user data.

2.6 User Documentation

PolyChord’s main documentation method is built-in tutorials, with the goal that the user-interface complexity is kept low enough that the built-in tutorials will be sufficient, and no further external documentation is needed. These tutorials will cover the usage basics for all major features of PolyChord, and a tutorial on how to enable and troubleshoot the MIDI device support that PolyChord provides. These tutorials will be both one-time tutorials when a user selects a feature for the first time, and will be able to be recalled within PolyChord’s user settings.

2.7 Assumptions and Dependencies

Assumptions

PolyChord will assume that the users all have accepted the required extensions if they are using Internet Explorer or FireFox. It also assumes that the users are generally at a basic level of music theory knowledge, and would care to learn more.

Dependencies

PolyChord depends on a few extenal libraries and databases, notably Hooktheory, Tone.js, and WebMidi.js and the WebMidi API.

HookTheory API

HookTheory is a database that has collected the chords and chord progressions of thousands of popular songs. This has been made accessible in the form of the Hooktheory API which we will be using to find the likely following chord progressions given the most recently played chord(s). This saves the effort of having to search through piles of songs on our own and doing analysis to find the likely following chord progressions given what we have.

Tone.js

PolyChord relies on Tone.js for client-side sound synthesis and playback. Tone.js is a fairly popular, widely used framework for sound synthesis and modification on the client side, and is licensed under an MIT license, meaning that is free for all types of use, but provided without any functionality guarantees. Tone.js's popularity and wide adoption is the reason it was chosen over competitors, and it's use allows for much faster, more reliable development, when compared to writing the sound synthesis logic from scratch.

WebMidi API

PolyChord relies on the WebMIDI API specification for Midi support. This accepted Midi implementation for web, and it has been widely adopted. WebMidi is supported in Chrome, and while it still does not have native support in FireFox, there are a number of extensions for FireFox, and there are plans to implement the API natively in FireFox in the future.

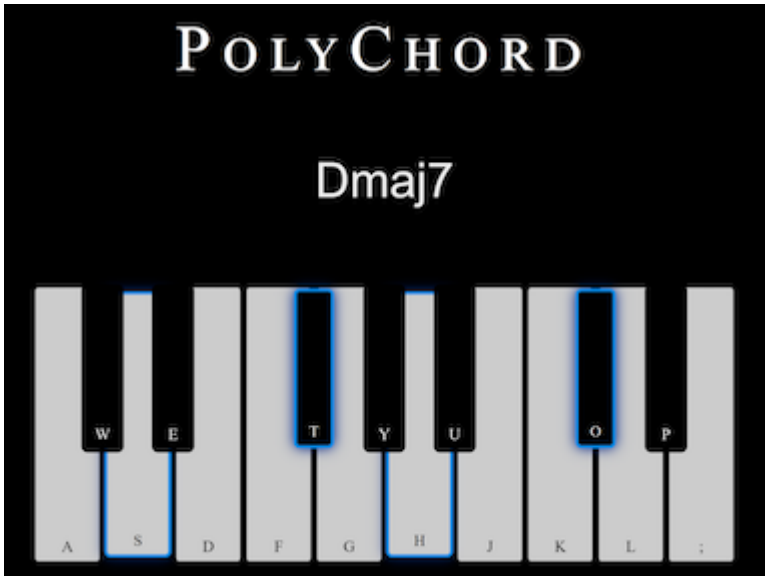
WebMidi.js

PolyChord relies on WebMidi.js for Midi support. WebMidi.js is a fairly popular framework for interacting with the WebMidi API at a higher level.

3. External Interface Requirements

3.1 User Interfaces

PolyChord will have a Keyboard that will react to input from either a MIDI Keyboard or computer keyboard. There will also be a main menu containing tutorials in music theory and a chord explorer interface, options to save a chord progression or view saved chord progressions and all other core features. There will also be a separate menu which allows for swapping between various samples such as “Piano”, or “Guitar” by the user clicking on the sample type or by typing in the corresponding sample name or index. This menu will also have an option to swap the octave that is being used by the piano in the case that they are using a computer keyboard.



3.1.1 User Interface Requirements

UI-1: The system shall react to input from a MIDI keyboard

UI-2: The system shall react to input from a computer keybaord

UI-3: The system shall contain music theory tutorials

UI-4: The system shall contain a chord explorer interface

UI-5: The system shall allow saving of a chord progression

UI-6: The system shall allow viewing of saved chord progressions

UI-7: The system shall allow the changing of the music sample by mouse

UI-8: The system shall allow the changing of the music sample by keyboard

UI-9: The system shall allow the changing of the octave with a clickable button

3.2 Hardware Interfaces

Supported Devices: MIDI Controller, primarily MIDI Keyboards and computer keyboards using keys from the top 2 rows (" a " to " ' " and " q " to " \ " to simulate a MIDI Controller. The top row will represent the black keys on a piano while the lower row will represent the white keys. PolyChord will accept input from a MIDI Keyboard or computer keyboard using the https://github.com/djipco/webmidi webmidi API. Once the input has been interpreted PolyChord will display the notes on an on-screen keyboard and play the corresponding note through the user's computer.

3.2.1 Hardware Interface Requirements

HW-1: The system shall map keys from a computer keyboard to piano keyboard

HW-2: The system shall use the webmidi API

3.3 Software Interfaces

PolyChord will be using the most recent tone.js (https://tonejs.github.io/) to play back music to the user including various types of samples and synths. It will take the inputs from the program after input has been taken in and parsed or when a user requests for a chord/chord progression to be played back to them. It handles tempo, notes, various sound samples, note durations and more that we will be using for PolyChord. The most recent webmidi.js (https://github.com/djipco/webmidi) will be used for the receiving of midi input. It works natively in Chrome, Opera, and Android, but can also work in Internet Explorer, Firefox ≤ v51, and Safari with some plugins from https://jazz-soft.net/ and https://cwilso.github.io/WebMIDIAPIShim/. This will allow for the use of any MIDI Controllers that a user may want to utilize. Polychord will also be utilizing the Hooktheory database of chord progressions that is documented at https://www.hooktheory.com/api/trends/docs to recognize what chords are likely to follow given a chord within the database and the preceding chords.

3.3.1 Software Interfaces Requirements

SOFT-1: The system shall use the tone.js library to play notes

SOFT-2: The system shall use the jazz-soft plugin to make the webmidi API work on web browsers that do not natively support it

SOFT-3: The system shall use the hooktheory chord progression database

3.4 Communications Interfaces

Polychord will work over http and https as it is a web application, and these are the standard protocols for websites at the time of this specification. As of now will not need to communicate in any way other than the connection between the user and the server. We will have an embedded survey on the website to submit bug reports so that we can improve PolyChord with help from our users. As of now, data synchronization and transfer speeds are of low priority since most of PolyChord's functionality will not require much active interaction between the user and server, and there are no large files to be loaded in with the website.

3.4.1 Communications Interface Requirements

COMM-1: The system shall support http communications

COMM-2: The system shall support https communications

COMM-3: The system shall provide a method for user feedback

4. System Features

4.1 Website

4.1.1 Description and Priority

PolyChord will provide a lightweight, multipurpose website that will work in major web browsers and a method for mobile users to use the system. With this feature, we can begin to implement other features. Priority: High

4.1.2 Stimulus/Response Sequences

Not applicable

4.1.3 Functional Requirements

SITE-1: The system shall operate on major web browsers.

SITE-2: The system shall be compatible with mobile devices.

SITE-3: The system shall adopt a color scheme that is easy to use in the dark.

SITE-4: The system shall use flask 1.1.1 with Python 3.6+ to run.

4.2 Chords

4.2.1 Description and Priority

Users will be able to play chords and have relevant information about it displayed. This is one of our primary features and can be implemented relatively early. Priority: High

4.2.2 Stimulus/Response Sequences

Stimulus: User plays a chord. Response: System looks up information about the chord and displays it.

Stimulus: User plays a chord incorrectly. Response: System looks up similar chords and displays which notes the user is not supposed to be playing.

4.2.3 Functional Requirements

CHRD-1: The system shall detect chords that the user is playing.

CHRD-2: The system shall display the name of a chord that the user is playing.

CHRD-3: The system shall display the tonic of an input chord.

CHRD-4: The system shall display played chords on a music staff.

CHRD-5: The system shall display played chords on a fret-board.

CHRD-6: The system shall display the key that a detected chord is in.

CHRD-7: The system shall suggest chords close to a chord that is incorrectly played.

CHRD-8: The system shall have a dataset of chord intervals to identify chords.

4.3 Progressions

4.3.1 Description and Priority

Users will be able to see various chord progressions based off of the chords they have played. This feature is one of our major features, but requires other features to be implemented first to function. Priority: Medium

4.3.2 Stimulus/Response Sequences

Stimulus: User plays chords. Repsonse: System finds progressions that contain those chords and displays those progressions.

Stimulus: User selects chords for playback. Response: System finds progressions that contain those chords and displays those progressions.

4.3.3 Functional Requirements

PRGSN-1: The system shall suggest chord progressions to the user based on the chords they are playing.

PRGSN-2: The system shall suggest chord progressions to the user based on selected chords.

PRGSN-3: The system shall display songs with similar chord progressions to the selected progression.

4.4 User Accounts

4.4.1 Description and Priority

Users will be able to create accounts to track statistics and save music loops and other settings. These are important to the system, but rely on all other aspects to be implemented first. Priority: Low

4.4.2 Stimulus/Response Sequences

Stimulus: User saves a chord progression. Response: System prepares a file the user can download containing information about the chord progression. Stimulus: User uploads a file they have downloaded from PolyChord. Response: Systems determines what information is in the file and changes the website to the display contained in the file.

Stimulus: User submits a bug report. Response: System forwards the report to the developers.

4.4.3 Functional Requirements

ACCT-1: The system shall allow the user to save chord progressions.

ACCT-2: The system shall allow the user to import settings.

ACCT-3: The system shall allow the user to report bugs.

ACCT-4: The system shall implement sight reading excercises.

ACCT-5: The system shall implement chord detection excercises.

ACCT-6: The system shall display chord progressions from uploaded files.

ACCT-7: The system shall give the user access to music theory lessons.

ACCT-8: The system shall display statistics for a user in training excercises.

ACCT-9: The system shall allow users to delete their accounts.

4.5 Musical Settings

4.5.1 Description and Priority

Users will be able to change the settings of aspects of the music to their liking, as well as input music notes using various methods. These features are important to the final product, but rely on many other aspects that will have to be implemented first. Priority: Medium

4.5.2 Stimulus/Response Sequences

Stimulus: The user changes a setting. Response: The system updates the display for the user and saves the current settings.

4.5.3 Functional Requirements

MSET-1: The system shall allow input from a keyboard.

MSET-2: The system shall allow input from MIDI instruments.

MSET-3: The system shall play a chosen chord progression on a loop.

MSET-4: The system shall allow a user to play melodies over a progression loop.

MSET-5: The system shall allow the user to change the ledger clef.

MSET-6: The system shall have a metronome.

MSET-7: The system shall allow the user to play looped drum patterns with their input.

MSET-8: The system shall allow the user to change the octave.

MSET-9: The system shall display notes on ledger lines.

5. Other Nonfunctional Requirements

5.1 Performance Requirements

PERF-1: The system shall recognize a chord within 2 seconds 99% of the time.

PERF-2: The system shall display chord progressions within 3 seconds 99% of the time.

PERF-3: The system shall begin a file download within 1 second of a request 99% of the time.

PERF-4: The system shall correctly identify chords 99% of the time.

5.2 Safety Requirements

No safety requirements have been identified.

5.3 Security Requirements

SEC-1: The system shall require all users to log in for all operations.

SEC-2: The system shall not place identifying information into downloaded files.

5.4 Software Quality Attributes

Availability: The system should be available during school hours, so teachers can use it and in the evening, so people home from work can use it. Correctness: The system should correctly identify chords. Portability: The system should be available on different popular operating systems and web browsers. Usability: The system should be fully functional to a user without extensive training.

5.5 Business Rules

No business rules have been identified.

6. Other Requirements

There are currently no other requirements to be specified for the SRS of PolyChord.

7. Supporting Information

7.1 Data Structures

timeSignature

Object	*Title*
int	beatsPerMeasure
int	beatNote

note

Object	*Title*
int	noteVal
int	beatDuration
double	startBeat

chord

Object	*Title*
str	name

Object	*Title*
note[]	notes

progression

Object	*Title*
timeSignature	beatsPerMeasure
int	measures
note[]	progNotes

Appendix A: To Be Determined List

1. Surveymonkey embedded form - this is a tentative method for us to allow users of PolyChord to report bugs to our team, but our method of accepting forms has not yet been finalized.