**MidiFind**

**Group 20**

Joe Crozier

Tyler Post

Vicky Bilbily

Version Number: ***WTF***

April 10, 2015

Software Engineering 2XB3: Binding Theory to Practice

Department of Computing and Software, McMaster University

# Revisions

April 10, 2015

Proofread by all members in the group. No further changes were made.

April 9, 2015

Added an internal review and evaluation of our design. Proofread and updated formatting.

April 7, 2015

Completed module descriptions and described the interface and semantics of each module implemented into the project. Finished uses relationship and documentation on implementation of the project.

April 2, 2015

Final design document constructed, title page, table of contents, and contributions recorded. Wrote executive summary and started on module descriptions.

*By virtue of submitting this document we electronically sign and date that the work being submitted by all the individuals in the group is their exclusive work as a group and we consent to make available the application developed through [CS] or [SE]-2XB3 project, the reports, presentations, and assignments (not including my name and student number) for future teaching purposes.*

Joe Crozier – 1311502 – Project Leader

*Project Specifications, GUI*

Tyler Post – 1302109 – Team Member

*Database, Design Documentation*

Vicky Bilbily – 1317465 – Log Admin

*Algorithms, Project Log Administrator*

# Contributions

|  |  |  |  |
| --- | --- | --- | --- |
| Name | Role |  | Contribution |
| Joe Crozier  *Project Leader* | Project Specifications  Front End GUI |  | * Requirements Specifications Document * Project Presentation Slides * Created project graphics and artwork using Adobe Photoshop * Produced application using PyGame Library * Combined back-end parsing framework with search and sorting algorithms |
| Tyler Post  *Team Member* | File Parsing  Design Specification Document |  | * Developed an algorithm to decompose a midi file into a musical contour * Managed development of a database that can be efficiently searched * Produced original draft of Design Specification Document according to instructions received in SE 2AA4 |
| Vicky Bilbily  *Log Admin* | Searching and Sorting Algorithms  Project Log Administrator |  | * Implemented Knuth Morris Pratt string searching algorithm to effectively find matching contours * Optimized three way quicksort in Python for mido song objects * Maintained living documentation of project and all deliverables * Organized and managed group meetings and schedules * Modified and finalized Design Specification Document |

# Executive Summary

MidiFind is a melody recognition tool designed to allow a user to easily input a melody and identify the track’s name and artist. Nearly anyone who enjoys music has experienced the frustration of knowing the melody of a song, but not the song’s actual name. MidiFind uses musical contours to identify different tracks, selecting from a dataset of 28,000 popular songs spanning all decades of music. A musical contour is a means of simplifying the melody of a song by expressing each note as a comparison to the previous note, stating whether the note is above (U), below (D), or the same (S) as the previous note. Musical contours are an excellent tool for identifying melodies, as they allow entire songs to be approximated with simple strings. By using musical contours rather than individual notes, the user does not have to input songs in their original keys, nor does their interpretation need to be exactly correct.

Table of Contents

[Revisions 2](#_Toc416125211)

[Contributions 3](#_Toc416125212)

[Executive Summary 4](#_Toc416125213)

[Design Overview 6](#_Toc416125214)

[Module Descriptions 6](#_Toc416125215)

[Class Descriptions 6](#_Toc416125216)

[Requirements Trace-Back 6](#_Toc416125217)

[Internal Design Review 6](#_Toc416125218)

# Design Overview

UML diagram showing static representation of your application classes

Why we have decomposed the application into these classes

# Module Descriptions

**FileParser**

This module is responsible for creating musical contours for each midi track in our database and writing the results to a CSV. Doing a live search of each midi file is impractical and leads to very large run times. This module only needs to be run once, in order to gather the required information from the song to compose a contour.

FileParser uses the mido library to extract information about each note played in a song. Mido returns the value, velocity, and time of each note in the song, so it can be transcribed into a string contour. FileParser also handles the formatting of title and artist names, and finally creates a csv where each row contains a songs artist, name, and musical contour.

Description of the classes we have used in our application

Include two UML state machine diagrams for two most interesting classes in your implementation

# Class Descriptions

Fucking mis and mid

Include a view of the uses relationship

# Requirements Trace-Back

# Internal Design Review