# Module Interface Specification for Software Engineering

Team 8 – Rhythm Rangers

Ansel Chen Muhammad Jawad Mohamad-Hassan Bahsoun Matthew Baleanu Ahmed Al-Hayali

January 17, 2025

# 1 Revision History

Date	Version	Notes
Date 1	1.0	Notes
Date 2	1.1	Notes

# 2 Symbols, Abbreviations and Acronyms

See SRS Documentation at [give url —SS] [Also add any additional symbols, abbreviations or acronyms —SS]

# Contents

1	Rev	vision 1	History					
2	Symbols, Abbreviations and Acronyms							
3	Introduction							
4	Notation							
5	Module Decomposition							
3	$\mathbf{G}\mathbf{U}$	I Mod	ule					
	6.1	GUI N	Module					
	6.2	Uses						
	6.3	Syntax	x					
		6.3.1	Exported Constants					
		6.3.2	Exported Access Programs					
	6.4	Semar	ntics					
		6.4.1	State Variables					
		6.4.2	Environment Variables					
		6.4.3	Assumptions					
		6.4.4	Access Routine Semantics					
		6.4.5	Local Functions					
•	MIS	S of A	udio File Input Module					
	7.1	Audio	File Input Module					
	7.2	Uses						
	7.3	Syntax	x					
		7.3.1	Exported Constants					
		7.3.2	Exported Access Programs					
	7.4	Semar	ntics					
		7.4.1	State Variables					
		7.4.2	Environment Variables					
		7.4.3	Assumptions					
		7.4.4	Access Routine Semantics					
		7.4.5	Local Functions					
	MIS	S of Se	earch Query Module					
	8.1		n Query Module					
	8.2							
	8.3		x					
		8.3.1	Exported Constants					
		8.3.2	Exported Access Programs					

	8.4	Seman	tics					 			 				6
		8.4.1	State Variables					 			 				6
		8.4.2	Environment Variables					 			 				6
		8.4.3	Assumptions					 			 				6
		8.4.4	Access Routine Semantics					 			 				6
		8.4.5	Local Functions					 							6
9	MIS	of Cl	ient Communication Mo	dι	ıle	)									7
	9.1	Client	Communication Module					 			 				7
	9.2	Uses						 			 				7
	9.3	Syntax	·					 			 				7
		9.3.1	Exported Constants					 			 				7
		9.3.2	Exported Access Programs					 			 				7
	9.4	Seman	tics					 			 				7
		9.4.1	State Variables					 			 				7
		9.4.2	Environment Variables					 			 				7
		9.4.3	Assumptions					 			 				7
		9.4.4	Access Routine Semantics					 			 				8
		9.4.5	Local Functions					 			 				8
10	MIS	of Se	rver Communication Mo	odı	ule	9									8
	10.1	Server	Communication Module .					 			 				8
	10.2	Uses						 			 				8
	10.3	Syntax	<u>.</u>					 			 				8
		10.3.1	Exported Constants					 			 				8
		10.3.2	Exported Access Programs	3.				 			 				9
	10.4	Seman	tics					 			 				9
		10.4.1	State Variables					 			 				9
		10.4.2	Environment Variables					 			 				9
		10.4.3	Assumptions					 			 				9
		10.4.4	Access Routine Semantics					 			 				9
		10.4.5	Local Functions					 			 				9
11	MIS	of Dr	iver Module												10
	11.1	Driver	Module					 			 				10
	11.2	Uses						 			 				10
	11.3	Syntax						 			 				10
		11.3.1	Exported Constants					 			 				10
			Exported Access Programs												10
	11.4		tics												10
			State Variables												10
		11.4.2	Environment Variables					 			 				10
			Assumptions												10

			Access Routine Semantics	11 11
12	MIS	of Ter	mpo (BPM) Feature Extraction Module	11
12			(BPM) Feature Extraction Module	11
		_	(DI W) Feature Extraction Wording	11
				11
	12.5		Exported Constants	11
				11
	19.4		Exported Access Programs	11
	12.4		State Variables	11
				12
			Environment Variables	
			Assumptions	12
			Access Routine Semantics	12
		12.4.5	Local Functions	12
13	MIS	of Key	y and Scale Feature Extraction Module	12
		•	d Scale Feature Extraction Module	12
				12
				12
	10.0	•	Exported Constants	12
			Exported Access Programs	12
	13 4		tics	13
	10.1		State Variables	13
			Environment Variables	13
			Assumptions	13
			Access Routine Semantics	13
			Local Functions	13
		13.4.5	Local Functions	10
14	MIS	of Ins	trument Type Feature Extraction Module	13
	14.1	Instrun	nent Type Feature Extraction Module	13
	14.2	Uses .		13
				13
			Exported Constants	13
			Exported Access Programs	14
	14.4		tics	14
			State Variables	14
			Environment Variables	14
			Assumptions	14
			Access Routine Semantics	14
				14
		14.4.0	Local Functions	14

15 MIS of Vocal Gender Feature Extraction Module	14
15.1 MIS of Vocal Gender Feature Extraction Module	 . 14
15.2 Uses	 . 15
15.3 Syntax	 . 15
15.3.1 Exported Constants	 . 15
15.3.2 Exported Access Programs	 . 15
15.4 Semantics	
15.4.1 State Variables	
15.4.2 Environment Variables	
15.4.3 Assumptions	
15.4.4 Access Routine Semantics	
15.4.5 Local Functions	
16 MIS of Dynamic Range Feature Extraction Module	16
16.1 Dynamic Range Feature Extraction Module	
16.2 Uses	 . 16
16.3 Syntax	 . 16
16.3.1 Exported Constants	 . 16
16.3.2 Exported Access Programs	 . 16
16.4 Semantics	 . 16
16.4.1 State Variables	 . 16
16.4.2 Environment Variables	 . 16
16.4.3 Assumptions	 . 16
16.4.4 Access Routine Semantics	 . 16
16.4.5 Local Functions	 . 17
17 MIS of Instrumentalness Feature Extraction Module	17
17.1 Instrumentalness Feature Extraction Module	
17.2 Uses	
17.3 Syntax	
17.3.1 Exported Constants	
17.3.2 Exported Access Programs	
17.4 Semantics	
17.4.1 State Variables	
17.4.2 Environment Variables	
17.4.3 Assumptions	
17.4.4 Access Routine Semantics	
17.4.5 Local Functions	 . 18
18 MIS of Contour Feature Extraction Module	18
18.1 Contour Feature Extraction Module	
18.2 Uses	
18.3 Syntax	 18

		18.3.1 Exported Constants	18
		18.3.2 Exported Access Programs	18
	18.4	Semantics	18
		18.4.1 State Variables	18
		18.4.2 Environment Variables	18
		18.4.3 Assumptions	19
		18.4.4 Access Routine Semantics	19
		18.4.5 Local Functions	19
19			19
			19
		Uses	19
	19.3	Syntax	19
		19.3.1 Exported Constants	19
		19.3.2 Exported Access Programs	19
	19.4	Semantics	19
		19.4.1 State Variables	19
		19.4.2 Environment Variables	19
		19.4.3 Assumptions	20
		19.4.4 Access Routine Semantics	20
		19.4.5 Local Functions	20
20	MIS		20
<b>20</b>		of Genre Feature Extraction Module	<b>20</b>
<b>20</b>	20.1	of Genre Feature Extraction Module  Module	20
<b>2</b> 0	20.1 20.2	of Genre Feature Extraction Module  Module	20 20
<b>20</b>	20.1 20.2	of Genre Feature Extraction Module  Module	20 20 20
20	20.1 20.2	of Genre Feature Extraction Module  Module	20 20 20 20
20	20.1 20.2 20.3	of Genre Feature Extraction Module  Module	20 20 20 20 20
20	20.1 20.2 20.3	of Genre Feature Extraction Module  Module	20 20 20 20 20 21
20	20.1 20.2 20.3	of Genre Feature Extraction Module  Module	20 20 20 20 20 21 21
20	20.1 20.2 20.3	of Genre Feature Extraction Module  Module	20 20 20 20 20 21 21 21
20	20.1 20.2 20.3	of Genre Feature Extraction Module  Module	20 20 20 20 21 21 21 21
20	20.1 20.2 20.3	of Genre Feature Extraction Module  Module	20 20 20 20 21 21 21 21 21
20	20.1 20.2 20.3	of Genre Feature Extraction Module  Module	20 20 20 20 21 21 21 21
	20.1 20.2 20.3 20.4	of Genre Feature Extraction Module  Module	20 20 20 20 21 21 21 21 21
	20.1 20.2 20.3 20.4 MIS	of Genre Feature Extraction Module  Module	20 20 20 20 21 21 21 21 21 21
	20.1 20.2 20.3 20.4 MIS 21.1	of Genre Feature Extraction Module  Module	20 20 20 20 21 21 21 21 21 21 21
	20.1 20.2 20.3 20.4 MIS 21.1 21.2	of Genre Feature Extraction Module  Module	20 20 20 20 21 21 21 21 21 21 21 22
	20.1 20.2 20.3 20.4 MIS 21.1 21.2	of Genre Feature Extraction Module  Module .  Uses .  Syntax .  20.3.1 Exported Constants 20.3.2 Exported Access Programs .  Semantics .  20.4.1 State Variables .  20.4.2 Environment Variables .  20.4.3 Assumptions .  20.4.4 Access Routine Semantics .  20.4.5 Local Functions .  of Recommendation Module .  Recommendation Module .  Uses .	20 20 20 20 21 21 21 21 21 22 22 22
	20.1 20.2 20.3 20.4 MIS 21.1 21.2	of Genre Feature Extraction Module  Module	20 20 20 20 21 21 21 21 21 22 22 22
	20.1 20.2 20.3 20.4 MIS 21.1 21.2 21.3	of Genre Feature Extraction Module  Module Uses Syntax 20.3.1 Exported Constants 20.3.2 Exported Access Programs Semantics 20.4.1 State Variables 20.4.2 Environment Variables 20.4.3 Assumptions 20.4.4 Access Routine Semantics 20.4.5 Local Functions of Recommendation Module Recommendation Module Uses Syntax 21.3.1 Exported Constants	20 20 20 20 21 21 21 21 21 22 22 22 22

		21.4.2 Environment Variables	22
		21.4.3 Assumptions	
		21.4.4 Access Routine Semantics	23
		21.4.5 Local Functions	23
<b>22</b>	MIS	S of Program Results Interface Module	23
	22.1	Program Results Interface Module	23
	22.2	Uses	23
	22.3	Syntax	23
		22.3.1 Exported Constants	23
		22.3.2 Exported Access Programs	23
	22.4	Semantics	23
		22.4.1 State Variables	23
		22.4.2 Environment Variables	24
		22.4.3 Assumptions	24
		22.4.4 Access Routine Semantics	24
		22.4.5 Local Functions	24
23	App	pendix	26

# 3 Introduction

The following document details the Module Interface Specifications for [Fill in your project name and description —SS]

Complementary documents include the System Requirement Specifications and Module Guide. The full documentation and implementation can be found at .... [provide the url for your repo —SS]

# 4 Notation

[You should describe your notation. You can use what is below as a starting point. —SS]

The structure of the MIS for modules comes from Hoffman and Strooper (1995), with the addition that template modules have been adapted from Ghezzi et al. (2003). The mathematical notation comes from Chapter 3 of Hoffman and Strooper (1995). For instance, the symbol := is used for a multiple assignment statement and conditional rules follow the form  $(c_1 \Rightarrow r_1 | c_2 \Rightarrow r_2 | ... | c_n \Rightarrow r_n)$ .

The following table summarizes the primitive data types used by Software Engineering.

Data Type	Notation	Description
character	char	a single symbol or digit
integer	$\mathbb{Z}$	a number without a fractional component in $(-\infty, \infty)$
natural number	N	a number without a fractional component in $[1, \infty)$
real	$\mathbb{R}$	any number in $(-\infty, \infty)$

The specification of Software Engineering uses some derived data types: sequences, strings, and tuples. Sequences are lists filled with elements of the same data type. Strings are sequences of characters. Tuples contain a list of values, potentially of different types. In addition, Software Engineering uses functions, which are defined by the data types of their inputs and outputs. Local functions are described by giving their type signature followed by their specification.

# 5 Module Decomposition

The following table is taken directly from the Module Guide document for this project.

Level 1	Level 2
Hardware-Hiding	
	GUI Module
	Audio File Input Module
	Search Query Module
Behaviour-Hiding	Client Communication Module
	Server Communication Module
	Driver Module
	Tempo (BPM) Feature Extraction Module
	Key and Scale Feature Extraction Module
	Instrument Type Feature Extraction Module
	Vocal Gender Feature Extraction Module
	Dynamic Range Feature Extraction Module
	Instrumentalness Feature Extraction Module
	Contour Feature Extraction Module
	Mood Feature Extraction Module
	Recommendation Module
	Program Results Interface
	Database
Software Decision	Spotify API
	Deezer API
	Genre Feature Module

Table 1: Module Hierarchy

# 6 GUI Module

## 6.1 GUI Module

## 6.2 Uses

- First-Match Text Field Input Module
- URL Input module
- Audio File Input Module
- Spotify Query Search & Select

# 6.3 Syntax

## 6.3.1 Exported Constants

N/A

## 6.3.2 Exported Access Programs

Name	In	Out	Exceptions
Consolidate	Up to 4 collection(s)	Merged collection of	-
Inputs	of reference(s) to	track references	
	$\operatorname{track}(s)$		

## 6.4 Semantics

### 6.4.1 State Variables

• Data type of the collection of track reference(s)

### 6.4.2 Environment Variables

N/A

## 6.4.3 Assumptions

N/A

## 6.4.4 Access Routine Semantics

consolidate\_inputs():

• output: parses the user input and returns the songs that are sent to be processed

## 6.4.5 Local Functions

• parse\_wav\_file(file)

\_

• parse\_url(url)

\_

• parse\_text(text)

\_

# 7 MIS of Audio File Input Module

# 7.1 Audio File Input Module

User inputs an audio file to the system to analyze.

## 7.2 Uses

N/A

# 7.3 Syntax

## 7.3.1 Exported Constants

N/A

## 7.3.2 Exported Access Programs

Name	In	Out	Exceptions
On Input	Audio File	Collection of song ref-	Invalid
Button		erence(s)	File Type
Press			

## 7.4 Semantics

## 7.4.1 State Variables

• Collection of track reference(s)

### 7.4.2 Environment Variables

## 7.4.3 Assumptions

- User has a properly named Audio File.
- User audio file input is actually a song.

#### 7.4.4 Access Routine Semantics

```
[accessProg —SS]():
```

- transition: [if appropriate —SS]
- output: [if appropriate —SS]
- exception: [if appropriate —SS]

[A module without environment variables or state variables is unlikely to have a state transition. In this case a state transition can only occur if the module is changing the state of another module. —SS]

[Modules rarely have both a transition and an output. In most cases you will have one or the other. —SS]

#### 7.4.5 Local Functions

[As appropriate—SS] [These functions are for the purpose of specification. They are not necessarily something that is going to be implemented explicitly. Even if they are implemented, they are not exported; they only have local scope.—SS]

# 8 MIS of Search Query Module

# 8.1 Search Query Module

User inputs a song and that is turned into a spotify search query where the top 10 matches are available for user to select

### 8.2 Uses

N/A

# 8.3 Syntax

### 8.3.1 Exported Constants

#### 8.3.2 Exported Access Programs

Name	In	Out	Exceptions
Search	text input	top 10 matches from	_
Query		spotify query search	
Request			
Output re-	user selection	Collection containing	_
sult selec-		track reference	
tion			

### 8.4 Semantics

#### 8.4.1 State Variables

• Collection containing track reference

### 8.4.2 Environment Variables

- Spotify Client ID
- Spotify Client Secret

## 8.4.3 Assumptions

N/A

#### 8.4.4 Access Routine Semantics

[accessProg —SS]():

- transition: [if appropriate—SS]
- output: [if appropriate —SS]
- exception: [if appropriate —SS]

[A module without environment variables or state variables is unlikely to have a state transition. In this case a state transition can only occur if the module is changing the state of another module. —SS]

[Modules rarely have both a transition and an output. In most cases you will have one or the other. —SS]

#### 8.4.5 Local Functions

[As appropriate—SS] [These functions are for the purpose of specification. They are not necessarily something that is going to be implemented explicitly. Even if they are implemented, they are not exported; they only have local scope. —SS]

# 9 MIS of Client Communication Module

## 9.1 Client Communication Module

User inputs a song and that is turned into a spotify search query where the top 10 matches are available for user to select

# 9.2 Uses

N/A

## 9.3 Syntax

## 9.3.1 Exported Constants

N/A

## 9.3.2 Exported Access Programs

Name	In	Out	Exceptions
Search	text input	top 10 matches from	N/A
Query		spotify query search	
Request			
Output re-	user selection	Collection containing	N/A
sult selec-		track reference	
tion			

## 9.4 Semantics

### 9.4.1 State Variables

• Collection containing track reference

#### 9.4.2 Environment Variables

- Spotify Client ID
- Spotify Client Secret

## 9.4.3 Assumptions

#### 9.4.4 Access Routine Semantics

[accessProg —SS]():

• transition: [if appropriate —SS]

• output: [if appropriate —SS]

• exception: [if appropriate —SS]

[A module without environment variables or state variables is unlikely to have a state transition. In this case a state transition can only occur if the module is changing the state of another module. —SS]

[Modules rarely have both a transition and an output. In most cases you will have one or the other. —SS]

#### 9.4.5 Local Functions

[As appropriate—SS] [These functions are for the purpose of specification. They are not necessarily something that is going to be implemented explicitly. Even if they are implemented, they are not exported; they only have local scope. —SS]

# 10 MIS of Server Communication Module

### 10.1 Server Communication Module

User inputs a song and that is turned into a spotify search query where the top 10 matches are available for user to select

### 10.2 Uses

N/A

# 10.3 Syntax

#### 10.3.1 Exported Constants

### 10.3.2 Exported Access Programs

Name	In	Out	Exceptions
Search	text input	top 10 matches from	N/A
Query		spotify query search	
Request			
Output re-	user selection	Collection containing	N/A
sult selec-		track reference	
tion			

### 10.4 Semantics

#### 10.4.1 State Variables

• Collection containing track reference

### 10.4.2 Environment Variables

- Spotify Client ID
- Spotify Client Secret

## 10.4.3 Assumptions

N/A

#### 10.4.4 Access Routine Semantics

[accessProg —SS]():

- transition: [if appropriate —SS]
- output: [if appropriate —SS]
- exception: [if appropriate —SS]

[A module without environment variables or state variables is unlikely to have a state transition. In this case a state transition can only occur if the module is changing the state of another module. —SS]

[Modules rarely have both a transition and an output. In most cases you will have one or the other. —SS]

#### 10.4.5 Local Functions

[As appropriate—SS] [These functions are for the purpose of specification. They are not necessarily something that is going to be implemented explicitly. Even if they are implemented, they are not exported; they only have local scope. —SS]

# 11 MIS of Driver Module

## 11.1 Driver Module

User inputs a song and that is turned into a spotify search query where the top 10 matches are available for user to select

## 11.2 Uses

N/A

# 11.3 Syntax

## 11.3.1 Exported Constants

N/A

## 11.3.2 Exported Access Programs

Name	In	Out	Exceptions
Search	text input	top 10 matches from	N/A
Query		spotify query search	
Request			
Output re-	user selection	Collection containing	N/A
sult selec-		track reference	
tion			

## 11.4 Semantics

### 11.4.1 State Variables

• Collection containing track reference

### 11.4.2 Environment Variables

- Spotify Client ID
- Spotify Client Secret

## 11.4.3 Assumptions

#### 11.4.4 Access Routine Semantics

[accessProg —SS]():

• transition: [if appropriate —SS]

• output: [if appropriate —SS]

• exception: [if appropriate —SS]

[A module without environment variables or state variables is unlikely to have a state transition. In this case a state transition can only occur if the module is changing the state of another module. —SS]

[Modules rarely have both a transition and an output. In most cases you will have one or the other. —SS]

#### 11.4.5 Local Functions

[As appropriate—SS] [These functions are for the purpose of specification. They are not necessarily something that is going to be implemented explicitly. Even if they are implemented, they are not exported; they only have local scope. —SS]

# 12 MIS of Tempo (BPM) Feature Extraction Module

# 12.1 Tempo (BPM) Feature Extraction Module

### 12.2 Uses

N/A

# 12.3 Syntax

## 12.3.1 Exported Constants

N/A

## 12.3.2 Exported Access Programs

Name	In	Out	Exceptions
Extract	Audio time series	Song Tempo $\in \mathbb{R}$	-
Tempo	(np.ndarray)		

### 12.4 Semantics

#### 12.4.1 State Variables

### 12.4.2 Environment Variables

N/A

## 12.4.3 Assumptions

Valid audio file with coherent song information.

### 12.4.4 Access Routine Semantics

ExtractTempo():

• transition: N/A

• output: Song\_Tempo : = ExtractTempo(Audio\_Time\_Series)

• exception: N/A

### 12.4.5 Local Functions

N/A

# 13 MIS of Key and Scale Feature Extraction Module

# 13.1 Key and Scale Feature Extraction Module

## 13.2 Uses

N/A

# 13.3 Syntax

## 13.3.1 Exported Constants

N/A

## 13.3.2 Exported Access Programs

Name	In	Out	Exceptions
Extract	Audio time series	Song Key, Scale	_
Key &	(np.ndarray)	$\in \mathbb{Z}^2$	
Scale			

# 13.4 Semantics

### 13.4.1 State Variables

N/A

### 13.4.2 Environment Variables

N/A

## 13.4.3 Assumptions

Valid audio file with coherent song information.

#### 13.4.4 Access Routine Semantics

ExtractKeyScale():

- transition: N/A
- output: Song\_Key, Song\_Scale: = ExtractKeyScale(Audio\_Time\_Series)
- exception: N/A

## 13.4.5 Local Functions

N/A

# 14 MIS of Instrument Type Feature Extraction Module

# 14.1 Instrument Type Feature Extraction Module

## 14.2 Uses

N/A

# 14.3 Syntax

## 14.3.1 Exported Constants

### 14.3.2 Exported Access Programs

Name	In	Out	Exceptions
Extract	Audio time series	Instrument Type	_
Instrument	<pre>(np.ndarray)</pre>	$\in \mathbb{Z}^k$	
Туре			

## 14.4 Semantics

#### 14.4.1 State Variables

N/A

#### 14.4.2 Environment Variables

N/A

#### 14.4.3 Assumptions

Valid audio file with coherent song information.

#### 14.4.4 Access Routine Semantics

ExtractInstrumentType():

• transition: N/A

• output: Instrument\_Type : = ExtractInstrumentType(Audio\_Time\_Series)

• exception: N/A

#### 14.4.5 Local Functions

N/A

# 15 MIS of Vocal Gender Feature Extraction Module

## 15.1 MIS of Vocal Gender Feature Extraction Module

This feature seeks to quantify whether the voices features in the inputted audio file are largely more feminine or masculine sounding. This is represented by a float with a range between 0 and 1 where 0 means only "masculine" sound signatures are contained and 1 means only "feminine" sounds, where values in-between represent a blend.

# 15.2 Uses

N/A

# 15.3 Syntax

## 15.3.1 Exported Constants

N/A

## 15.3.2 Exported Access Programs

Name	In	Out	Exceptions
Extract	Audio time series	Vocal Gender $\in \mathbb{R}$	-
Vocal	<pre>(np.ndarray)</pre>		
Gender			

## 15.4 Semantics

## 15.4.1 State Variables

N/A

## 15.4.2 Environment Variables

N/A

## 15.4.3 Assumptions

Valid audio file with coherent song information.

## 15.4.4 Access Routine Semantics

### ExtractVocalGender():

• transition: N/A

• output: Vocal\_Gender: = ExtractVocalGender(Audio\_Time\_Series)

• exception: N/A

# 15.4.5 Local Functions

# 16 MIS of Dynamic Range Feature Extraction Module

# 16.1 Dynamic Range Feature Extraction Module

Feature extracts the range of sounds (difference between peak and through) of the audio signal.

## 16.2 Uses

N/A

# 16.3 Syntax

## 16.3.1 Exported Constants

N/A

## 16.3.2 Exported Access Programs

Name	In	Out	Exceptions
Extract	Audio time series	Dynamic Range	-
Dynamic	<pre>(np.ndarray)</pre>	$(\texttt{decibels}) \in \mathbb{R}$	
Range			

## 16.4 Semantics

#### 16.4.1 State Variables

N/A

### 16.4.2 Environment Variables

N/A

### 16.4.3 Assumptions

Valid audio file with coherent song information.

#### 16.4.4 Access Routine Semantics

ExtractDynamicRange():

• transition: N/A

• output: Dynamic\_Range : = ExtractDynamicRange(Audio\_Time\_Series)

• exception: N/A

#### 16.4.5 Local Functions

N/A

# 17 MIS of Instrumentalness Feature Extraction Module

### 17.1 Instrumentalness Feature Extraction Module

Extracts the how prominent instrumental sounds are within the song. Represented by a float variable where the range is between 0 and 1, where higher values mean more instrumental sounds and lower means less. Eg, 0 would mean an acapella piece of music, 1 would be something that purely features instruments.

## 17.2 Uses

N/A

## 17.3 Syntax

## 17.3.1 Exported Constants

N/A

## 17.3.2 Exported Access Programs

Name	In	Out	Exceptions
Extract	Audio time series	${\tt Instrumentalness} \ \in$	_
Instrument	alness	$\mathbb{R}$	
(np.ndarray)			

## 17.4 Semantics

#### 17.4.1 State Variables

N/A

## 17.4.2 Environment Variables

N/A

#### 17.4.3 Assumptions

Valid audio file with coherent song information.

### 17.4.4 Access Routine Semantics

ExtractInstrumentalness():

• transition: N/A

• output: Instrumentalness: = ExtractInstrumentalness(Audio\_Time\_Series)

• exception: N/A

### 17.4.5 Local Functions

N/A

# 18 MIS of Contour Feature Extraction Module

## 18.1 Contour Feature Extraction Module

## 18.2 Uses

N/A

# 18.3 Syntax

## 18.3.1 Exported Constants

N/A

## 18.3.2 Exported Access Programs

Name	In	Out	Exceptions
Extract	Audio time series	Contour	-
Melodic	(np.ndarray)		
Contour			

## 18.4 Semantics

### 18.4.1 State Variables

N/A

### 18.4.2 Environment Variables

## 18.4.3 Assumptions

Valid audio file with coherent song information.

### 18.4.4 Access Routine Semantics

ExtractMelodicContour():

• transition: N/A

• output: Contour : = ExtractMelodicContour(Audio\_Time\_Series)

• exception: N/A

## 18.4.5 Local Functions

N/A

# 19 MIS of Mood Feature Extraction Module

## 19.1 Mood Feature Extraction Module

### 19.2 Uses

N/A

# 19.3 Syntax

## 19.3.1 Exported Constants

N/A

## 19.3.2 Exported Access Programs

Name	In	Out	Exceptions
Extract	Audio time series	${\tt Mood} \in \mathbb{Z}$	-
Mood	<pre>(np.ndarray)</pre>		

## 19.4 Semantics

### 19.4.1 State Variables

N/A

## 19.4.2 Environment Variables

## 19.4.3 Assumptions

Valid audio file with coherent song information.

#### 19.4.4 Access Routine Semantics

## ExtractMood():

• transition: N/A

• output: Mood : = ExtractMood(Audio\_Time\_Series)

• exception: N/A

#### 19.4.5 Local Functions

N/A

## 20 MIS of Genre Feature Extraction Module

## 20.1 Module

Genre Feature Extraction Module

## 20.2 Uses

- Featurizer Module: Receives metadata from the Featurizer Module and extracts the genre attribute from it. - Metadata Structure: Utilizes the metadata structure to locate and retrieve the genre attribute.

# 20.3 Syntax

## 20.3.1 Exported Constants

None.

## 20.3.2 Exported Access Programs

Name	In	Out	Exceptions
extractGer	re metadata: Metadata	genre: String	MissingGenreException,
			Invalid-
			Meta-
			dataEx-
			ception

### 20.4 Semantics

#### 20.4.1 State Variables

- metadata: The metadata provided by the Featurizer Module, which contains the genre attribute.

#### 20.4.2 Environment Variables

None.

## 20.4.3 Assumptions

- The metadata provided by the Featurizer Module is valid and includes the genre attribute.
- The genre attribute in the metadata is correctly formatted and accessible.

#### 20.4.4 Access Routine Semantics

extractGenre(metadata: Metadata):

- Transition: Extracts the genre attribute from the provided metadata.
- Output: Returns the extracted genre as a string.
- Exceptions: MissingGenreException: Raised if the genre attribute is not found in the metadata. InvalidMetadataException: Raised if the provided metadata is improperly formatted or invalid.

#### 20.4.5 Local Functions

#### validateMetadata:

- Purpose: Ensures the provided metadata is valid and contains the necessary attributes.
- Input: metadata.
- Output: Boolean (true if valid, false otherwise).

#### retrieveGenre:

- Purpose: Locates and retrieves the genre attribute from the metadata.
- Input: metadata.
- Output: genre (String).

# 21 MIS of Recommendation Module

## 21.1 Recommendation Module

## 21.2 Uses

- Tempo (BPM) Feature Extraction Module
- Key and Scale Feature Extraction Module
- Instrument Type Feature Extraction Module
- Vocal Gender Feature Extraction Module
- Dynamic Range Feature Extraction Module
- Instrumentalness Feature Extraction Module
- Contour Feature Extraction Module
- Mood Feature Extraction Module
- Driver Module
- Spotify API

# 21.3 Syntax

## 21.3.1 Exported Constants

N/A

## 21.3.2 Exported Access Programs

Name	In	Out	Exceptions
Generate	Song_Features	Rec_Tracks	_
Recs	$( exttt{np.ndarray} \in Feature)$	${\tt np.ndarray} \in {\tt Track}$	

## 21.4 Semantics

#### 21.4.1 State Variables

N/A

### 21.4.2 Environment Variables

## 21.4.3 Assumptions

N/A

## 21.4.4 Access Routine Semantics

GenerateRecommendations():

• transition: N/A

• output: Recommended\_Songs : = GenerateRecommendations(Song\_Features)

• exception: N/A

### 21.4.5 Local Functions

N/A

# 22 MIS of Program Results Interface Module

# 22.1 Program Results Interface Module

## 22.2 Uses

• Spotify API

# 22.3 Syntax

## 22.3.1 Exported Constants

N/A

## 22.3.2 Exported Access Programs

Name	In	Out	Exceptions
Generate	Rec_Track	Tracks_Embed (Spo-	_
Spotify	$(\texttt{np.ndarray} \; \in \;$	tify Embed Element)	
Embed	Track)		
Display	Song Features	${ t Features\_Display}$	-
Features	$(\texttt{np.ndarray} \in$	(UI Image)	
	Feature)		

## 22.4 Semantics

## 22.4.1 State Variables

## 22.4.2 Environment Variables

N/A

## 22.4.3 Assumptions

N/A

## 22.4.4 Access Routine Semantics

GenerateSpotifyEmbed():

- transition: N/A
- output: Tracks\_Embed\_Widget: = GenerateSpotifyEmbed(Tracks)
- exception: N/A

DisplayFeatures():

- transition: N/A
- output: Features\_Display: = DisplayFeatures(Song\_Features)
- exception: N/A

## 22.4.5 Local Functions

# References

Carlo Ghezzi, Mehdi Jazayeri, and Dino Mandrioli. Fundamentals of Software Engineering. Prentice Hall, Upper Saddle River, NJ, USA, 2nd edition, 2003.

Daniel M. Hoffman and Paul A. Strooper. Software Design, Automated Testing, and Maintenance: A Practical Approach. International Thomson Computer Press, New York, NY, USA, 1995. URL http://citeseer.ist.psu.edu/428727.html.

# 23 Appendix

 $[{\bf Extra~information~if~required~-\!SS}]$ 

# Appendix — Reflection

## [Not required for CAS 741 projects—SS]

The information in this section will be used to evaluate the team members on the graduate attribute of Problem Analysis and Design.

The purpose of reflection questions is to give you a chance to assess your own learning and that of your group as a whole, and to find ways to improve in the future. Reflection is an important part of the learning process. Reflection is also an essential component of a successful software development process.

Reflections are most interesting and useful when they're honest, even if the stories they tell are imperfect. You will be marked based on your depth of thought and analysis, and not based on the content of the reflections themselves. Thus, for full marks we encourage you to answer openly and honestly and to avoid simply writing "what you think the evaluator wants to hear."

Please answer the following questions. Some questions can be answered on the team level, but where appropriate, each team member should write their own response:

- 1. What went well while writing this deliverable?
- 2. What pain points did you experience during this deliverable, and how did you resolve them?
- 3. Which of your design decisions stemmed from speaking to your client(s) or a proxy (e.g. your peers, stakeholders, potential users)? For those that were not, why, and where did they come from?
- 4. While creating the design doc, what parts of your other documents (e.g. requirements, hazard analysis, etc), it any, needed to be changed, and why?
- 5. What are the limitations of your solution? Put another way, given unlimited resources, what could you do to make the project better? (LO\_ProbSolutions)
- 6. Give a brief overview of other design solutions you considered. What are the benefits and tradeoffs of those other designs compared with the chosen design? From all the potential options, why did you select the documented design? (LO\_Explores)