

**REAL-TIME ALGORITHMIC MUSIC COMPOSITION APPLICATION**

**BY**

**ALISA YAP YI HUI**

**A REPORT**

**SUBMITTED TO**

**Universiti Tunku Abdul Rahman**

**in partial fulfillment of the requirements**

**for the degree of**

**BACHELOR OF COMPUTER SCIENCE (HONOURS)**

**Faculty of Information and Communication Technology**

**(Kampar Campus)**

**JAN 2022**

## REPORT STATUS DECLARATION FORM

Title: REAL-TIME ALGORITHMIC MUSIC COMPOSITION  
APPLICATION

Academic Session: JAN 2022

I ALISA YAP YI HUI  
(CAPITAL LETTER)

declare that I allow this Final Year Project Report to be kept in  
Universiti Tunku Abdul Rahman Library subject to the regulations as follows:

1. The dissertation is a property of the Library.
2. The Library is allowed to make copies of this dissertation for academic purposes.

Verified by,



(Author's signature)



Online Signature during COVID-19  
SOONG HOONG CHENG (Lecturer)  
FACULTY OF ICT  
UNIVERSITI TUNKU ABDUL RAHMAN (UTAR)  
PERAK CAMPUS, JLN UNIVERSITI, BANDAR BARAT,  
31900 KAMPAR.

(Supervisor's signature)

Address:

288A-2-10, JALAN  
THEAN TEIK, AIR ITAM,  
11500 PULAU PINANG.

TS. SOONG HOONG CHENG

Supervisor's name

Date: 22 APRIL 2022

Date: 22 APRIL 2022

<b>Universiti Tunku Abdul Rahman</b>			
Form Title : <b>Sample of Submission Sheet for FYP/Dissertation/Thesis</b>			
Form Number: <b>FM-IAD-004</b>	Rev No.: <b>0</b>	Effective Date: <b>21 JUNE 2011</b>	Page No.: <b>1 of 1</b>

**FACULTY OF INFORMATION AND COMMUNICATION TECHNOLOGY**

**UNIVERSITI TUNKU ABDUL RAHMAN**

Date: 22/04/2022

**SUBMISSION OF FINAL YEAR PROJECT /DISSERTATION/THESIS**

It is hereby certified that Alisa Yap Yi Hui (ID No: 19ACB01195 ) has completed this final year project entitled “ **Real-Time Algorithmic Music Composition Application**” under the supervision of Ts Soong Hoong Cheng (Supervisor) from the Department of **Digital Economy Technology** , Faculty of **Information and Communication Technology** .

I understand that University will upload softcopy of my final year project in pdf format into UTAR Institutional Repository, which may be made accessible to UTAR community and public.

Yours truly,



\_\_\_\_\_  
(Alisa Yap Yi Hui)

## DECLARATION OF ORIGINALITY

I declare that this report entitled “**REAL-TIME ALGORITHMIC MUSIC COMPOSITION APPLICATION**” is my own work except as cited in the references. The report has not been accepted for any degree and is not being submitted concurrently in candidature for any degree or other award.

Signature :  \_\_\_\_\_

Name : ALISA YAP YI HUI

Date : 22 APRIL 2022

## **ACKNOWLEDGEMENTS**

First and foremost, I would like to express my sincere thanks and appreciation towards my supervisor, Ts Soong Hoong Cheng for his invaluable guidance and tutelage throughout my final year project. When in doubt, his guidance and suggestions have been able to inspire and point me towards the right direction.

Besides that, I also wish to express my gratitude towards my academic advisor, Dr Goh Chuan Meng, for his kind support and advice throughout this course, and my friends and family for their love, support and words of encouragement, which have given me the drive and motivation to work harder towards achieving my goals.

## **ABSTRACT**

This project is about the study of evolutionary music, and focuses on the development of an algorithmic music composer using the Java programming language. The motivation of this project is to provide a solution to problems including the high cost and time consumption for composing music, the complexity of music composition, and the limitations of using copyrighted music. The developed system shall be able generated music based on a genetic algorithm, where users can define basic parameters including the instrument used to play the music, tempo, number of generated notes, and the number of measures to generate. In addition, the system also utilises JavaFx and jFugue for its graphical user interface and music programming respectively. Due to time constraints, the development of this system uses a form of RAD development model, namely the phased development model. Besides generating music, the developed system includes various functionalities including saving the music into an audio file, editing the generated music, playing music, as well as loading and viewing the music previously generated and saved by the users. The music produced through this system are relatively simple, and is suited to be applied in personal projects which require such audio accompaniment. Lastly, the deliverable of this project will be a desktop application for users to compose and generate music in real-time with minimal effort.

# TABLE OF CONTENTS

<b>TITLE PAGE</b>	<b>i</b>
<b>REPORT STATUS DECLARATION FORM</b>	<b>ii</b>
<b>FYP THESIS SUBMISSION FORM</b>	<b>iii</b>
<b>DECLARATION OF ORIGINALITY</b>	<b>iv</b>
<b>ACKNOWLEDGEMENTS</b>	<b>v</b>
<b>ABSTRACT</b>	<b>vi</b>
<b>TABLE OF CONTENTS</b>	<b>vii</b>
<b>LIST OF FIGURES</b>	<b>x</b>
<b>LIST OF TABLES</b>	<b>xi</b>
<b>LIST OF SYMBOLS</b>	<b>xii</b>
<b>LIST OF ABBREVIATIONS</b>	<b>xiii</b>
<b>CHAPTER 1 INTRODUCTION</b>	<b>1</b>
1.1 Problem Statement and Motivation	1
1.2 Objectives	5
1.3 Project Scope and Direction	7
1.4 Contributions	8
1.5 Report Organization	9
<b>CHAPTER 2 LITERATURE REVIEW</b>	<b>10</b>
2.1 Methods for Algorithmic Composition	10
2.1.1 Stochastic Methods	10
2.1.2 Rule-Based Models	12
2.1.3 Artificial Intelligence	13
2.1.4 Comparison of Algorithmic Composition Methods	15
2.2 Music Composition	16
2.2.1 The Circle of Fifths	16
2.2.2 The Golden Ratio and its Application in Music	17
2.3 Critical Remarks of Previous Works	18
2.3.1 Computoser	18
	vii

2.3.2	Ecret Music	19
2.3.3	ArtSong	21
2.3.4	Comparison of Existing Systems	23
<b>CHAPTER 3 PROPOSED METHOD/APPROACH</b>		<b>24</b>
3.1	System Design Diagram/Equation	24
3.1.1	System Architecture Diagram	24
3.1.2	Use Case Diagram and Description	25
3.1.3	Activity Diagram	26
<b>CHAPTER 4 SYSTEM DESIGN</b>		<b>33</b>
4.1	System Flow	33
4.1.1	Music Generation	33
4.1.2	Chord Generation	37
4.2	User Requirements	38
4.3	System Specifications	38
4.4	System User Interface Design	39
<b>CHAPTER 5 SYSTEM IMPLEMENTATION</b>		<b>40</b>
5.1	System Methodology	40
5.2	Project Workflow in Phased Development Model	41
5.3	Project Timeline	42
5.4	Technologies and Tools Involved	43
5.5	Implementation Outcome	45
<b>CHAPTER 6 SYSTEM EVALUATION AND DISCUSSION</b>		<b>49</b>
6.1	Testing Setup and Remark	49
6.2	Project Challenges	57
6.3	Objectives Evaluation	57



<b>CHAPTER 7 CONCLUSION AND RECOMMENDATION</b>	<b>58</b>
7.1 Conclusion	58
7.2 Recommendation	59
<b>REFERENCES</b>	<b>61</b>
<b>WEEKLY LOG</b>	<b>64</b>
<b>POSTER</b>	<b>77</b>
<b>PAPER PUBLISHED</b>	<b>78</b>
<b>PLAGIARISM CHECK RESULT</b>	<b>79</b>
<b>FYP2 CHECKLIST</b>	<b>83</b>

## LIST OF FIGURES

Figure Number	Title	Page
Figure 2.1	The Markov Chain	11
Figure 2.2	Genetic Algorithm Flowchart	14
Figure 2.3	The Circle of Fifths	16
Figure 2.4	The Fibonacci Spiral	17
Figure 2.5	Preferences Configuration in <i>Computoser</i>	18
Figure 2.6	Music Player in <i>Computoser</i>	19
Figure 2.7	<i>Ecrett Music</i> User Interface	20
Figure 2.8	Adjusting Tempo, Volume and Duration in <i>Ecrett Music</i>	20
Figure 2.9	Property Editor in <i>ArtSong</i>	21
Figure 2.10	<i>ArtSong</i> User Interface	22
Figure 2.11	<i>ArtSong</i> MIDI Note Editor	22
Figure 3.1	System Architecture Diagram	24
Figure 3.2	Use Case Diagram	25
Figure 3.3	Activity Diagram for Generate Chords	26
Figure 3.4	Activity Diagram for Generate Music	27
Figure 3.5	Activity Diagram for Save Music	28
Figure 3.6	Activity Diagram for Play Music	29
Figure 3.7	Activity Diagram for Edit Music	29
Figure 3.8	Activity Diagram for Load Music	30
Figure 3.9	Activity Diagram for Delete Music	31
Figure 3.10	Activity Diagram for View Music Library	32
Figure 4.1	Flowchart for Music Generation	33
Figure 4.2	Flowchart of Genetic Algorithm	34
Figure 4.3	Crossover and Mutation	35
Figure 4.4	User Interface Design for Basic Music Generation and Editing	39
Figure 4.5	User Interface Design for Music Library	39
Figure 5.1	System Development Life Cycle	40

Figure 5.2	Phased Development Model	41
Figure 5.3	Gantt Chart for Final Year Project 1	43
Figure 5.4	Gantt Chart for Final Year Project 2	43
Figure 5.5	Software Involved	45
Figure 5.6	Screenshot of the System (Generate Music)	46
Figure 5.7	Screenshot of the System (Generate Chords)	46
Figure 5.8	Screenshot of the System (Music Editor)	47
Figure 5.9	Screenshot of the System (Music Library)	48
Figure 6.1	Load from Music Library	55
Figure 6.2	Loaded to Editor	56
Figure 6.3	Saved Files in System	56
Figure 6.4	Opening in Windows Media Player	56

## LIST OF TABLES

<b>Table Number</b>	<b>Title</b>	<b>Page</b>
Table 1.1	Objectives and the Corresponding Problems to be Solved	7
Table 2.1	Comparison of Algorithmic Composition Methodologies	15
Table 2.2	Comparison of the Reviewed Systems	23
Table 4.1	Genetic Algorithm Parameters and Implementation Details	34
Table 4.2	Music Parameters and Description: Generate Music	35
Table 4.3	Evolution Method and Example Output	36
Table 4.4	Probability Distribution of Chord Succession in Classical Music	37
Table 4.5	Music Parameters and Description: Generate Chords	37
Table 4.6	System Specifications	38
Table 5.1	Laptop Specifications	50
Table 6.1	Verification Plan P1 (Music/Chord Generation)	49
Table 6.2	Verification Plan P2 (Save Music)	50
Table 6.3	Verification Plan P3 (Load Music)	51
Table 6.4	Verification Plan P4 (Edit Music)	52
Table 6.5	Verification Plan P5 (Play Music)	53
Table 6.6	Verification Plan P6 (View Music Library)	53
Table 6.7	Verification Plan P6 (Delete Music)	54

## LIST OF SYMBOLS

$\Phi$       Phi

## LIST OF ABBREVIATIONS

<i>API</i>	Application Programming Interface
<i>AI</i>	Artificial Intelligence
<i>ANN</i>	Artificial Neural Network
<i>GA</i>	Genetic Algorithm
<i>GP</i>	Genetic Programming
<i>GUI</i>	Graphical User Interface
<i>HSA</i>	Harmony Search Algorithm
<i>IDE</i>	Integrated Development Environment
<i>MIDI</i>	Musical Instrument Digital Interface
<i>ML</i>	Machine Learning
<i>RAD</i>	Rapid Application Development
<i>UML</i>	Unified Modeling Language
<i>SDLC</i>	System Development Life Cycle

# Chapter 1

## Introduction

The deliverable of this project will be a music composition application where users can use it to generate original music, edit it and export it as an audio file in real-time. The following will provide an insight of the project through its background, problem statements and motivation, scope, objectives, and contributions.

### 1.1 Problem Statement and Motivation

Music, derived from the Greek word *mousike*, is a form of art that expresses emotion and ideas through the arrangement of various sounds in time to produce a composition, all while taking the four important elements of melody, harmony, rhythm, and dynamics into consideration [1]. Music has a wide range of applications, ranging from entertainment purposes to religious rituals and representations of the cultural traditions and identities of societies. Perhaps the most widely known implementation of music today is through its manifestation in the various forms of media we are exposed to, as it is vastly used in the film and video game industry as a medium to evoke a certain feeling or emotion from the audience. In such applications, music has an important role in constructing the tone of a scene and is commonly seen as a foundation to set up the emotional atmosphere [2].

The digital revolution has contributed vastly to the adoption of digital computers and electronics in many individuals to enhance productivity and allow better work-flow management. In the past few decades, most people have integrated the use of computerised systems to simplify the procedures required in their field of work. This includes the musical field as well. The idea of algorithmic composition has been around for centuries where the adoption of algorithms in music, i.e. a predefined set of instructions that are executed step-by-step as a solution to a particular problem, were mainly principles designed for manually predictable styles of composition. Because of this, the introduction of computerised systems has opened doors to the creation of automated musical compositions through the implementation of these algorithms.

Furthermore, the creative process can also be divided into two distinct groups: ideas formulated from sheer genius and the other, which is through incremental revisions or hard work. While the first type of creativity is seemingly abstract, the latter has the potential to be integrated in systems due to its algorithmic nature [3]. This point can be further explained by such: Any given musician can only compose a piece of classical music only if they know what a classical piece sounds like to begin with. On the same token, a fundamental characteristic of the creation of art also includes varying degrees of a search and natural selection process to filter out the best ideas until a satisfactory outcome or product is achieved. Dawkins describes this phenomenon as a form of “Universal Darwinism”, in which the creative process of generating new intellectual ideas is actually derived from the process of iterative refinement of existing competing ideas [4]. This is rather agreeable as oftentimes we also find ourselves referring and seeking out the works of others to gain inspiration and develop ideas of our own. Therefore, it would be possible to simplify the refinement and optimisation process of music composition by applying genetic algorithms (GA) for the selection and mutation of any given note sequences, in a way similar to the nature of natural genetics. This leads us to the field of evolutionary computation, which is a global optimisation technique that is inspired by biological evolution.

Having said so, the development of evolutionary music dates back to the late 20th century, one of which was where Gibson and Byrne had been able to produce and combine musical fragments through the GA they developed [5]. The fitness of the produced music was then evaluated on a trained artificial neural network (ANN) from samples of music composed by human musicians. Besides that, other early examples of using evolutionary computation techniques to generate music include the system *GeNotator*, a computer-assisted composition system capable of constructing musical “grammar”, which is a set of predefined rules to define the composition boundaries of the generated music [4] and *Vox Populi*, a music composer with interactive evolutionary computation [6].

Additionally, much research has been done regarding evolutionary music in the past decades. Papers published in more recent years consist of those conducted by Matic, who used GA for their music composition [7]. While a vast majority implement the use



of genetic programming (GP) and GAs in evolutionary music development, others like Geem and Choy integrate a metaheuristic optimization method of using harmony search algorithms (HSA) into their systems [8]. Nevertheless, both approaches share the similarity of being optimisation search algorithms, which is a popular technique for implementing evolutionary computation.

In addition, alike many forms of art, music could actually be interpreted with mathematical concepts and geometrical elements. Some well-known concepts of mathematics being applied in music also include the golden ratio, symmetry and musical tessellations during compositions. Moreover, musical notes, keys, beats, and may other aspects of music may also be represented by a number, which allows us to be able to perform operations on them by mapping each to a numerical value.

Lastly, music creation is a process which requires skill and a concrete understanding on the topic, and the whole process of doing so from scratch may require a lot of time and effort. Therefore, the intention of this project is to employ a mathematical approach to produce an algorithmic music composition application capable of generating unique pieces of music in real-time, so that the task of music composition may be simplified by only requiring minimal human intervention.

Therefore, the motivation of this project is to develop an algorithmic music composer so that music composition can become an easier task for people as there are several existing problems revolving music composition, including its high time consumption and cost, the complexity of the process, and the limitations of using copyrighted music. The problem statements can be further elaborated as follows:

**I. The production of music is resource intensive.**

Even with adequate field experience and music composition skills, one has to be willing to invest time and patience in order to successfully craft a musical piece. Additionally, depending on the composer's intent of including which instruments to be used for the music they have written, an assortment of instruments will also be needed to play the written music. This stands to be an

issue because the low production scale of musical instruments, as well as the intricate craftsmanship required in the making of such instruments, are mainly the reason behind their high cost.

While one may argue that many systems today have been developed to replicate and mimic the tunes produced by various instruments, it should be noted that most professional music composition software does not come cheap either. For these reasons, to produce a good piece of music usually comes with a decent amount of time investment and a hefty price tag. This would be a problem if a composer or client seeking for an original-made musical piece only has a limited time frame and budget to work with.

## **II. Music composition is a complex process.**

Music composition is complex process which requires inspiration, creativity, and the formulation and structuring of ideas. It is a necessity for people who are new to the musical field to learn the fundamentals of music such as music theory, how to read, analyse and write musical scores, and familiarise themselves with the sounds of different musical instruments before they are able to partake in music composition. In addition, it also takes well-cultivated skills and talent to be able to compose a good piece of music, resulting in the requirement of a huge effort and a long period of time before competency in music composition could be achieved.

As such, due to the complex nature of creating music, existing music composition systems are also ridden with many features and functionalities that may be out of the league of intermediate composers. This would be an issue if the person intending to produce a musical piece has no intention of entering the field to become an expert, but only wants to compose a short piece for a one-time project. Yet, they may not have the connections or means to outsource this task to a skilled individual. People who fall into this category include independent creators or indie game developers who work alone or in a small team, and are usually not backed up by a company. These people may be specialised in a whole different field as compared to music, and the task of

composing original music as an accompaniment for their videos or games may turn out to be a challenging feat.

### **III. Limitations of using music produced by others.**

Intellectual property like musical works are often copyrighted by its owner(s), in order to protect their rights of having ownership over that musical piece. Because of this, it may be necessary to pay a certain fee or royalty in order to use the copyrighted works of others legally. It is also crucial to know which tracks are licensed under copyright as copyright infringement is not to be taken lightly.

Besides that, other limitations of using already existing music produced by others also include being subjected to fair use policies and having to turn to royalty-free alternatives. Even so, it is also time consuming and difficult for content creators to find a piece of music that is able to act as a perfect complementation to convey the exact feeling or mood of their work among many available options, as compared to composing the music themselves.

## **1.2 Objectives**

The main objective of this project is to develop an algorithmic music composer that can produce music in real time. On this basis, it can be further divided into several sub-objectives, which are listed below:

- I. To develop a real-time algorithmic music composition application which is capable of reducing production overhead and requires minimal human intervention.
- II. To simplify the music creation process and create a system with a simple and intuitive user interface for ease of user operation.
- III. To compose new and original melodies for users through the use of genetic algorithms.

Each objective is explained in further detail as follows, and the corresponding problems which they intend to solve are listed in Table 1.1:

- I. To develop a cost-free and time saving music composition system. This project's purpose is to make music composition more time efficient and cost-effective for potential music composers by producing a free to use, quick, and efficient way to compose algorithmic music. This could be achieved by using the Java programming language and JFugue API, all of which are open-source, and adds little to no cost on the development process. Moreover, the proposed system shall also be able to generate music by its own in real-time, which greatly reduces the skill and time needed for music composition.
- II. Create a simple and intuitive user interface, and minimise the musical knowledge required for composition. The proposed system will implement a user-friendly interface and provide its users the critical features needed for algorithmic composition. Users would only have to provide the parameters of the key, speed, the layers of sound, and instrument type in order to be able to compose music. After that, the rest of the composition task will be left to the system, requiring minimal human intervention and musical skill for composition.
- III. Enable to system to generate original pieces of music. To solve the problem of lacking the inspiration to create original music and the difficulty of obtaining specifically tailored music from composers or from already existing sources, an algorithmic music composer that can generate unique musical pieces shall be developed. The music generator will utilise a genetic algorithm to select, mutate and perform crossovers to the randomly generated sequence of notes, so that each of the musical pieces produced is ensured to be original. Users shall also be able to ne the parameters, so that the music output can meet their needs.

*Table 1.1 Objectives and the Corresponding Problems to be Solved*

<b>Objectives</b>	<b>Problems</b>
<b>To develop a real-time algorithmic music composition application which is capable of reducing production overhead and requires minimal human intervention.</b>	The production of music is resource intensive. <ul style="list-style-type: none"> <li>• Time-consuming.</li> <li>• High cost.</li> </ul>
<b>To simplify the music creation process and create a system with a simple and intuitive user interface for ease of user operation.</b>	Music composition is a complex process. <ul style="list-style-type: none"> <li>• Requires knowledge, skill and talent to compose music.</li> <li>• Design of systems are complex due to the complicated process of composing music.</li> </ul>
<b>To compose new and original melodies for users through the use of genetic algorithms.</b>	Limitations of using music produced by others. <ul style="list-style-type: none"> <li>• Being subjected to copyright and fair-use policies.</li> <li>• Hard to find good open-source alternatives that fit user's needs.</li> </ul>

### 1.3 Project Scope and Direction

The project scope will comprise of the necessary features to be included in the system for it to be able to produce a musical sequence based on user's input parameters, and other functionalities like editing, evaluating, saving and exporting music files, thus equipping the system with a full set of functionalities that would overcome the limitations of the reviewed methods and systems for their lack thereof.

#### I. Generate music.

The system shall be able to generate a random sequence of notes and retrieve user input for the parameters on how to evolve and mutate the randomly generated sequence through a genetic algorithm. This includes the key,

duration of the generated music, number of layers, and the musical instrument(s) which the music would be played in. Once the system has these basic parameters, it shall then be able to generate a unique musical piece for the user in real time.

II. Edit music.

The system shall allow users to edit the music generated by providing the options to modify each input parameter after the music has been produced, such as making changes to the speed from high to low. Therefore, the system should be able to display the settings for the current musical piece, provide a GUI for the user to freely key in and adjust their input parameters, play, and listen to the music which has been produced.

III. Evaluate music.

The system shall have the functionality to perform a fitness evaluation so that it is able to determine how well the melody is for the music being generated, and reiterate the musical piece through the genetic algorithm to produce finer melodies if it falls below the expectations of the fitness before the final piece of music is produced as an output.

IV. Save and export music.

The system shall be able to allow users to save their configuration and the generated music into a MIDI file, so that the users of the system may load the file to make further edits to their configuration, or export the musical file to be used or listened to outside of the system boundaries.

## 1.4 Contributions

The novelty of this project is to produce an algorithmic music composition application utilising mathematical concepts in the Java programming language. The expected outcome would be a system which is able to simplify the music composition process, even for users with little musical background, as most of the generative work is done by the system itself. The system will enable users to generate music based on their given

set of input parameters, evaluate it, and allow them to adjust the parameters of the produced piece by editing it through a well-structured GUI. Moreover, users would also be able to export the music into an audio file. Compared to traditional composition methods where the user is required to write the music, refine, and play it using a musical instrument as a complete set of processes before they are able to construct a complete musical piece that could be listened to, this system would provide a contribution by tremendously saving composer's time through simplifying and automating the whole composition and refinement process. This would also reduce the skill and effort needed for users to create music, and its capability to generate musical pieces at random could also act as a source of inspiration for advanced composers to produce more complex compositions of higher originality.

### **1.5 Report Organization**

This report is organised into 7 chapters: Chapter 1 Introduction, Chapter 2 Literature Review, Chapter 3 System Methodology/Approach, Chapter 4 System Design, Chapter 5 System Implementation, Chapter 6 System Evaluation and Discussion, and Chapter 7 Conclusion and Recommendation.

The first chapter is the introduction of this project which includes the problem statement and motivation, the objectives, scope and direction, contributions as well as the report organisation. The second chapter comprises of the literature review, in which three methods for algorithmic composition will be studied, the methods of music composition, and finally three existing systems, then conduct a comparison among them. Next, chapter three will be about the proposed method and approach taken to develop the system, containing the architecture diagram, user case and activity diagrams. Chapter four will describe the system design in detail, which includes the system flow, user requirements, system specifications, user interface design, performance definition and verification plan design. The fifth chapter will be about the system implementation. This includes the methodology and development model, timeline, technologies and tools involved, and implementation outcome. The sixth chapter contains details about the system evaluation and discussion, comprising of the testing setup, challenges faced, evaluation of the objectives and a concluding remark. Lastly, a conclusion and recommendation will be made in the final chapter.

# Chapter 2

## Literature Review

This chapter will review three existing methodologies implemented for algorithmic composition, the existing algorithmic music composition systems for their strengths, limitations, and how they can be resolved, as well as the musical aspects of this project along with the ways it can be represented by mathematics.

### 2.1 Methods for Algorithmic Composition

#### 2.1.1 Stochastic Methods

The stochastic approach for algorithmic composition was introduced by Iannis Xenakis in 1954. This non-deterministic algorithm for music generation incorporates both the concepts of probability distributions and randomness for music generation by randomly selecting notes based on its probability distribution on which musical note is most likely to follow after it. Additionally, conceptual complexity can be introduced to this model through the use of statistical theory, Markov chains and chaos theory [9].

Several works that have utilised this model are those by [10] who applied an adaptive scheme in addition to their use of the stochastic Markov chain, which enabled the dynamical change of the initial constructed graph topology, and [11] who came up with a rule-based approach to be used alongside the stochastic process for introducing randomness in compositions. The stochastic method is considered to be one of the simplest methods for algorithmic composition [9], and one of the most popular approaches to implement is through the use of Markov chains.

However, this model yielded very poor results according to Basseto and Neto [10]. This is because by determining the next node based solely on its predecessor, only linear grammars could be produced by first-order Markov chains. Higher-order chains had to be constructed to enable the generation of notes to consider a number of past states for the music composer to produce better results. Even so, doing this would quickly complicate the original solution and cause the relationships to the formal grammars to



become less noticeable. Therefore, while the stochastic technique is good at formalising the general rules or tendencies, its limitations lies in its inaccuracy for the management of structural details [11].

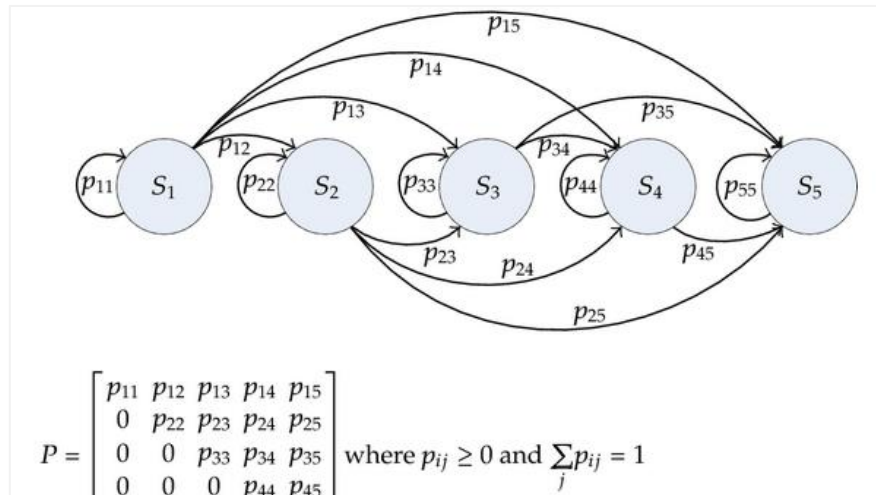


Figure 2.1 The Markov Chain

Figure 2.1 depicts the structure of a Markov chain, where each vertex could be used to represent a musical note, and each edge represents the probability of having the note it points to be followed after it. Below are the general steps for implementing a stochastic Markov chain within a system:

1. Select a corpus of notes.
2. Find out the probability distribution for the notes, which note is likely to follow after it.
3. Select the first note to be used for composition.
4. Based on the probability distribution, determine the next note to be played by selecting it randomly.
5. Repeat step 4 until a sequence of notes are generated.
6. End.

### Strengths:

- Basic stochastic models are easy to implement compared to other methods.
- The probability distribution makes it a good candidate for formalising the general rules and tendencies in music.

**Limitations:**

- Models of higher complexity have to be constructed to prevent linear grammars that result in poor melodies.
- It is inaccurate for managing the structural details of music.

**2.1.2 Rule-Based Models**

Rule-based algorithmic composition systems integrate the use of constraints and predefined formal grammars, which are a set of principles to be followed during the music generation. This method requires a deep understanding on music theory and concepts in order for the developer to model the musical grammar for the system [9]. Instead of generating music from randomness, rule-based models lean towards to approach of defining the potential actions at the initiation stage for the system to compose music.

Anders developed his knowledge-based music composition system to be able to model music theories, and the system was based on a collection of compositional rules, which included rules revolving around the rhythmic, harmonic structure, melodic and counterpoints of music composition [12]. However, the construction of rule-based systems is effortful as music composition is complex by nature, and its lack of randomness also makes it lacklustre for generating unique music compositions [13]. On the other hand, [4] had incorporated the use of musical grammar with evolutionary techniques to construct his compositions to overcome this limitation, resulting in a hybrid compositional environment that enabled its users to specify their grammar for composition [4].

**Strengths:**

- Constructs a set of composition rules based on original music theory and concepts.
- Users of rule-based systems have fuller control over what type of music is generated.

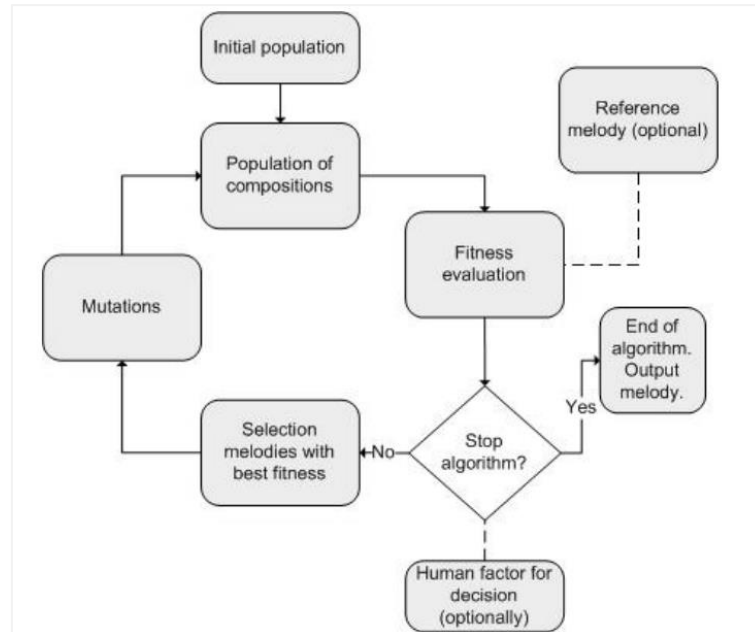
**Limitations:**

- The developer of the system has to have a deep understanding on music to construct the musical grammar.
- The variety of musical output may be limited as compared to other algorithmic composition methods.

**2.1.3 Artificial Intelligence**

The last of these three methods of algorithmic composition is through the use of artificial intelligence (AI). Somewhat related to rule-based systems in the way which it constructs music based on pre-defined grammar, the critical point of difference between AI and rule-based systems is that they are able to learn from existing music and construct its own set of rules to compose music [9]. Frameworks of this kind may include the use of machine learning (ML) methods like ANNs. However, this method has its own limitations as well, as ML based models are specifically trained on a set of data, outputs of systems which utilise AI may produce a musical output that more or less would resemble the training set [13].

Furthermore, a branch of using AI to generate music is through genetic programming (GP). In addition of constructing the grammar for composition, GP in algorithmic composition further enables the system to construct the musical material to be used for the composition as well [9]. Hence, this resolves the earlier problem of having similar outputs as the trained music data. Research related to this topic include [6] which developed the system *Vox Populi*, that utilised interactive evolutionary computation to allow users to determine the fitness functions of the algorithmic music composer, and [7]. Both incorporated the use of GA for their music composition systems. In GA implementations, each individual is assigned a fitness value which is used to determine its quality. Fitness functions act as a criterion for the comparison between individuals. Well-fitting individuals are selected and rerun through the iterative process of mutations and crossovers to produce better offspring, and the process is continued until a desired composition is produced [7].



*Figure 2.2 Genetic Algorithm Flowchart*

Figure 2.2 shows the flowchart of a genetic algorithm, and a general implementation of GA [7] can be listed as the following steps:

1. Randomly generate a population from the initial pool.
2. Evaluate the fitness of each individual in the population.
3. If the produced melody meets the fitness requirement, end.  
Else, proceed with step 4.
4. Select the best melodies and perform crossover and mutations.
5. Repeat step 3 until a solution is met.

#### **Strengths:**

- Produces a system that is capable of learning from music data and create its own ruleset for generating compositions.
- GP enables the generation of both composition rules and music material for composition.

#### **Limitations:**

- Systems that utilise ML methods may produce outputs that resemble the data it is trained on.

### 2.1.4 Comparison of Algorithmic Composition Methods

The following table shows the comparison table of the strengths and limitations for each of the described models for algorithmic composition:

*Table 2.1 Comparison of Algorithmic Composition Methodologies*

	<b>2.1.1 Stochastic Method</b>	<b>2.1.2 Rule-Based Models</b>	<b>2.1.3 Artificial Intelligence</b>
<b>Strengths</b>	Easy to implement. good at formalising the general rules or tendencies in music by providing the probability distribution.	Sets up a set of musical grammar for composition, hence compositions remain true to original music concepts. Users will have more control over music output.	Produces a learning-based system that creates its own rules for composition. GP enables the generation of both composition rules and music material for composition.
<b>Limitations</b>	Inaccurate for managing structural details, and higher complexity models have to be constructed for better music output.	Requires a deeper understanding on music theory and concepts to construct grammar. It also imposes a restriction on the diversity of musical output.	ML methods may result in outputs that highly resemble the training set.

Additionally, it had been brought to attention that many researchers mentioned in the prior sections have integrated the use more than one method so that their developed system could overcome the limitations of each model and provide better results. Examples of this include [4] and [11]. Both of the systems developed by them applied a mixture of a rule-based and AI or stochastic model, as the non-deterministic nature of these two methods were able to widen the variety of music produced compared to solely using a rule-based approach.

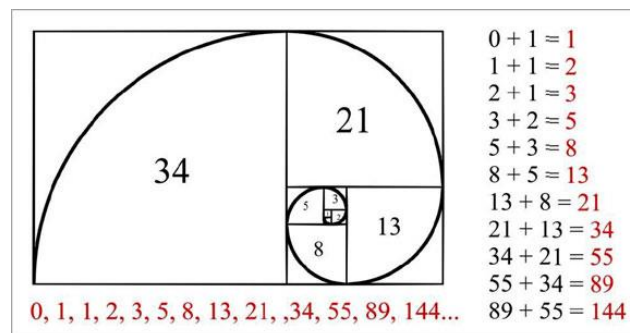


### 2.3.2 The Golden Ratio and its Application in Music

Also known as the divine proportion, the golden ratio is a unique mathematical concept which is commonly found in nature. The golden ratio is also widely applied in geometry, architecture, and various forms of art due to the pleasant aesthetics created by its balance and harmony [15].

For us to obtain this ratio, a line is divided into two parts  $a$  and  $b$ , such that  $a < b$ , and  $\frac{a}{b} = \frac{a+b}{a}$

Expanding this function provides us the two possible solutions of  $\frac{1+\sqrt{5}}{2}$  and  $\frac{1-\sqrt{5}}{2}$ , which results in two irrational numbers that approximate to 1.618 and -0.618 respectively. 1.168 is what's known as the golden number, and it is denoted by the Greek letter Phi ( $\Phi$ ) such that  $1: \Phi$ . (Mann, 2019) On top of that, Phi is also related closely to the Fibonacci sequence, a series of numbers where each subsequent number is the sum of its two preceding numbers, as shown: 0, 1, 1, 2, 3, 5, 8, 13, 21, 34, and so on. If we take the ratio of every two subsequent terms from this sequence, we'd notice that as the terms get larger, we gradually obtain a limit ratio that resembles  $\Phi$ .



*Figure 2.4 The Fibonacci Spiral*

The Fibonacci sequence and golden ratio is apparent in music through its presence in the design of musical instruments, composition techniques, and the octave, which is the base unit of melody and harmony. Examples of the application of the Fibonacci sequence include the increment of bar phrases, and also for the progression of notes and the modulation of keys to give the listener of the composition a sense of growth [16]. To illustrate an example, each key could be mapped in the way such that the keys of C, D, E, F, G are interpreted as 1, 2, 3, 4, 5, so when this is integrated this into music composition, the progression of keys could be 1, 1, 2, 3, 5 where the duration of C is twice as long as the others.

## 2.2 Critical Remarks of Previous Works

### 2.2.1 Computoser

*Computoser* [17] is a free-to-use algorithmic music composer where users can make a series of configurations for the music they want to compose. First, the system retrieves the user's input parameters such as the mood, tempo, instrument among many others, as shown in Figure 2.3. Once the user has provided the configuration, the system will then automatically compose a random piece of music based on it. *Computoser* states that it uses an AI algorithm for its computer-generated music, and every track produced by the system is algorithmically generated.

Users are also able to hear the composed music through the provided music player shown in Figure 2.4, where they can play, pause or adjust the volume output. Clicking on the “previous” and “next” buttons enables the user to navigate between different tracks composed by the same parameters. However, *Computoser* does not have the functionality of allowing its users to make any modifications to the music apart from their initial configuration. Besides that, users are also able to save their tracks as MIDI, MP3 or a MusicXML file and share it to social media platforms like Twitter and Facebook.

**Configure your preferences for the next tracks**

**Mood**  
 Any  Major (happy)  Minor (sad)

**Tempo**  
 Any  Very slow  Slow  Medium  Fast  Very fast

**Accompaniment (chords)**  
 Optional  Yes  No chords

**Instrument**  
 Piano

**Scale**  
 Major

**Classical**  
 Optional  Yes

**Electronic-like**  
 Optional  Yes  No

**Drums**  
 Optional  Yes  No drums

**More dissonant**  
 Optional  Yes

Play

Figure 2.5 Preferences Configuration in *Computoser*





*Figure 2.6 Music Player in Computoser*

### **Strengths:**

- Users are able to specify a wide range of parameters for what kind of music they wish for the system to produce, so the produced music would be closer to what they expect.
- Offers a variety of file formats like MIDI, MP3 and MusicXML for the music to be saved in, and even allows users to share the composed track to social media platforms.
- Allows users to effortlessly navigate between tracks.

### **Limitations:**

- Users are unable to control the layers and duration of the composed music.
- Further edits to the composition cannot be made beyond the initial configuration.

### **2.2.2 Ecret Music**

*Ecret Music* [18] is a web-based music composer which allows its users to create royalty-free music through a simple user interface. Its website claims that it uses AI for its music composition, and users can choose from the scene, mood and genre selections provided for the music they want to create, as shown in Figure 2.5. After the user has selected their preferred configurations, clicking on the “create music” button will then prompt the system to generate a random piece of music based on it.

Besides that, the buttons shown on the right further allow users to manage their created music by adding them as favourites, view their download history, preview how user’s videos will go with the music, and view their creation history. Once the system has generated a composition, users are then able to further edit the music by adding or removing each “block” of music, adjust the tempo, volume of each music layer, and the music duration by clicking on the

dropdown list on the top right corner of the music editor. Then, the modifications can be made accordingly as shown in Figure 2.6.

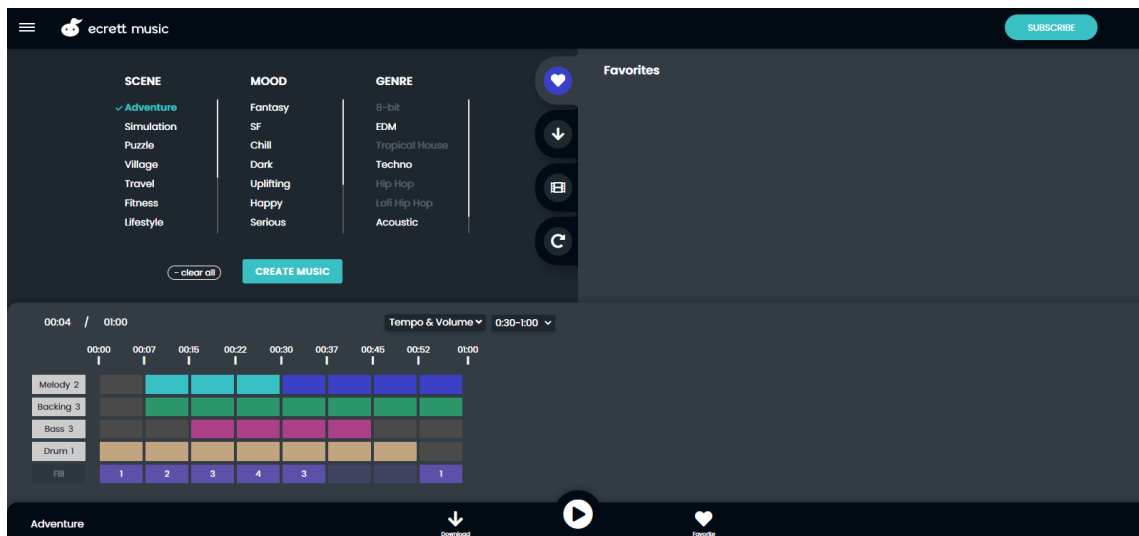


Figure 2.7 Ecrett Music User Interface

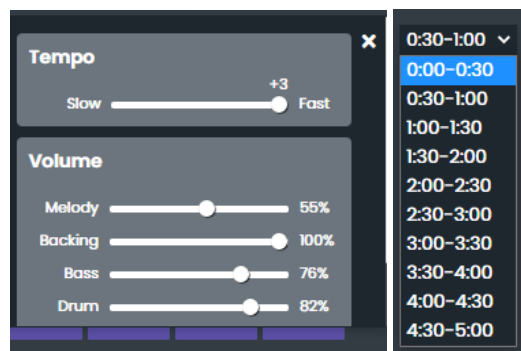


Figure 2.8 Adjusting Tempo, Volume and Duration in Ecrett Music

**Strengths:**

- Allows users to edit the music in detail, with options to adjust the tempo, volume and duration of the musical piece.
- Users are able to save their preferred pieces as “favourites”, and the system shows the history of created pieces so that users could manage the created songs more easily.
- Uses simple terms like “Adventure”, “Sad” and “Happy” to represent the type of music that would be generated, saving users’ time for defining the composition parameters.

**Limitations:**

- The music output has five layers by default, and users have to individually remove each block if they want to remove a layer.
- Users are unable to specify which musical instrument they want the music to be played in.

**2.2.2 ArtSong**

*ArtSong* [19] is an algorithmic music composer developed for the Windows operating system. Users can control the parameters of the generated music in the property editor as shown in Figure 2.7, where they can adjust a range of the elements like the length, beats, track count and scale of the music they want. Additionally, users may also modify the seed of the random generator to experiment with the different compositions the system may provide. After that, clicking on the “Compose” button will cause the system to compose a musical piece based on the input parameters provided by the user.

Figure 2.8 shows the user interface of *ArtSong*, users may edit the composed music through the workspace, and add or remove components from the left sidebar. Moreover, the system also allows users to modify the notes of the generated composition through its MIDI note editor as shown in Figure 2.9, where users can use it to easily add, edit or delete any of the generated notes.

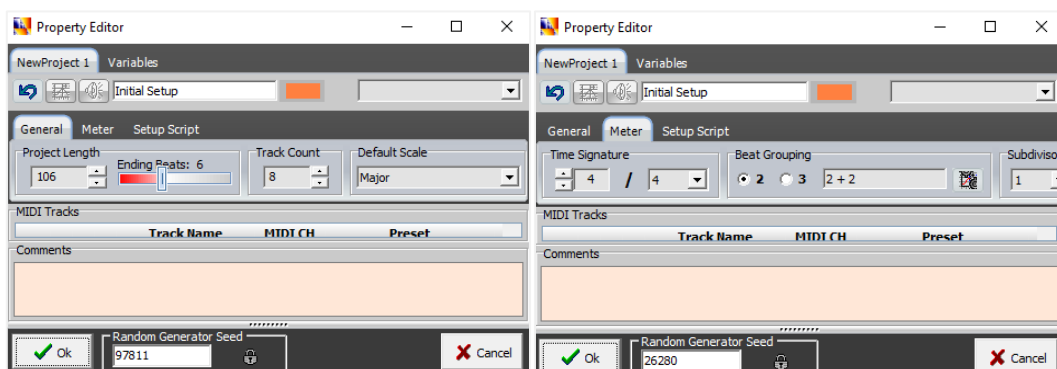


Figure 2.9 Property Editor in *ArtSong*

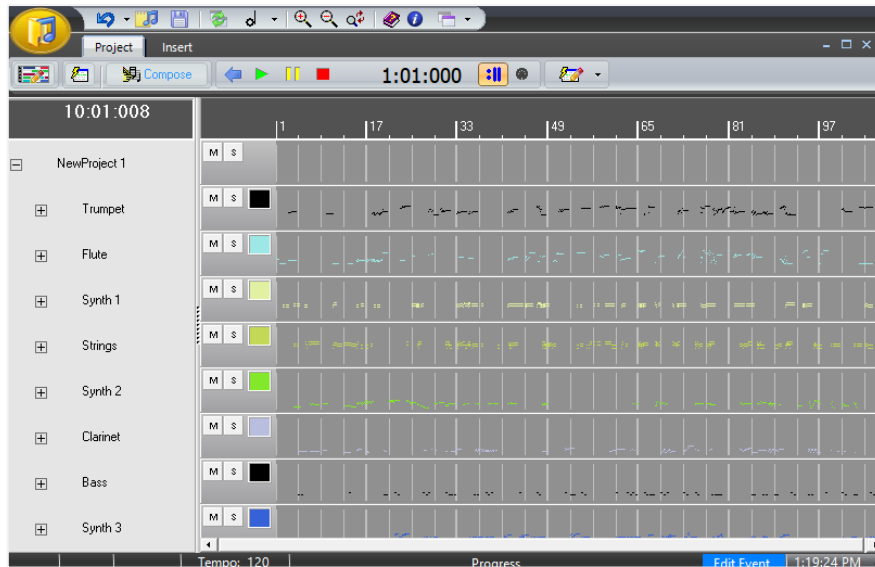


Figure 2.10 ArtSong User Interface

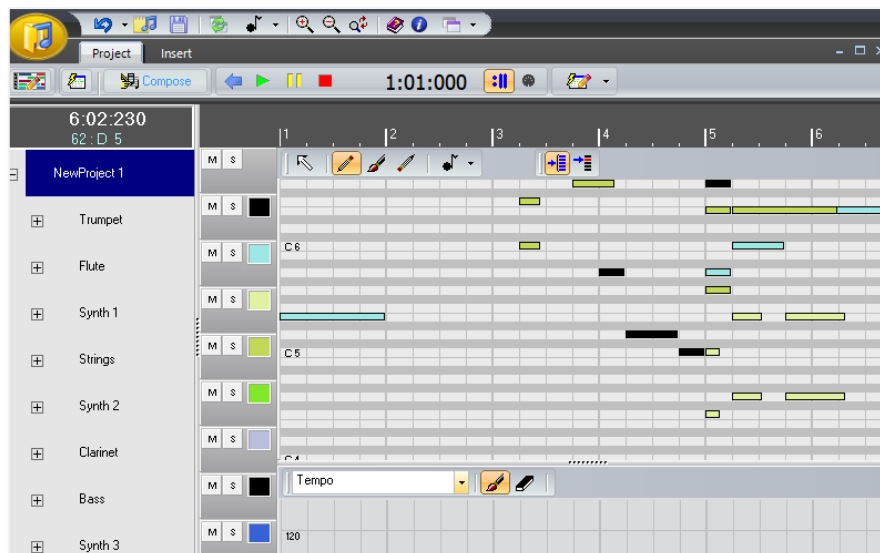


Figure 2.11 ArtSong MIDI Note Editor

**Strengths:**

- Users can save their work as a project file to continue where they left off at a later time.
- Users can edit the algorithmically composed music in great detail, and can even add, edit and delete the notes of the generated music through the system's note editor.
- The seed of the random generator can be modified to produce different music outputs.

**Limitations:**

- The system's complexity may require users to spend some time to get used to its features and how to navigate the system.

- The system only allows users to save the music as a MIDI file.

The table below shows the comparison of the features provided by each of the

#### 2.2.4 Comparison of Existing Methods

The following table shows the comparison table of the algorithmic composition systems from section 2.2.1 to 2.2.3. Most of the limitations mentioned in the previous sections can be resolved by adding additional features within the system that provides that functionality. Additionally, the weakness of the system's complexity as mentioned in section 2.2.3 could be solved by providing only the crucial features needed for users to create music and letting the system handle the finer details by itself.

*Table 2.2 Comparison of Existing Systems*

	<b>2.2.1 Computoser</b>	<b>2.2.2 Ecrett Music</b>	<b>2.2.3 ArtSong</b>
<b>Music editor</b>	No	Yes	Yes
<b>Specify duration</b>	No	Yes	Yes
<b>Instrument selection</b>	Yes	No	Yes
<b>Add and remove music layers</b>	No	Yes	Yes
<b>Add music as favourite</b>	No	Yes	No
<b>View/Navigate history of generated compositions.</b>	Yes	Yes	No
<b>Save as audio file</b>	Yes, as MIDI, MP3 and MusicXML.	Yes, as MP3.	Yes, as MIDI.
<b>Save as project file</b>	No	No	Yes

## Chapter 3

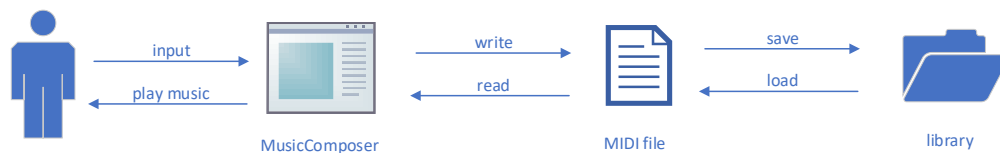
# System Methodology/Approach

This chapter explains the approach taken to develop the system. Unified Modelling Language (UML) diagrams such as a use case diagram is used to show the use cases users are able to perform in the system, complete with activity diagrams to show the control flow and decision paths of all activities. For further system design details, refer to Chapter 4.

### 3.1 System Design Diagram/Equation

#### 3.1.1 System Architecture Diagram

Figure 3.1 shows a simple diagram of how a user interacts with the system. Users provide inputs to the system, and after the music is generated by the system, it writes the music into a MIDI (.midi) file, and saves it into the library folder. When a file is loaded, the midi file requested is retrieved from the library folder and read by the system. Then, the music will be played to the user.



*Figure 3.1 System Architecture Diagram*

### 3.1.2 Use Case Diagram and Description

The use case diagram for the algorithmic music composer is shown as follows:

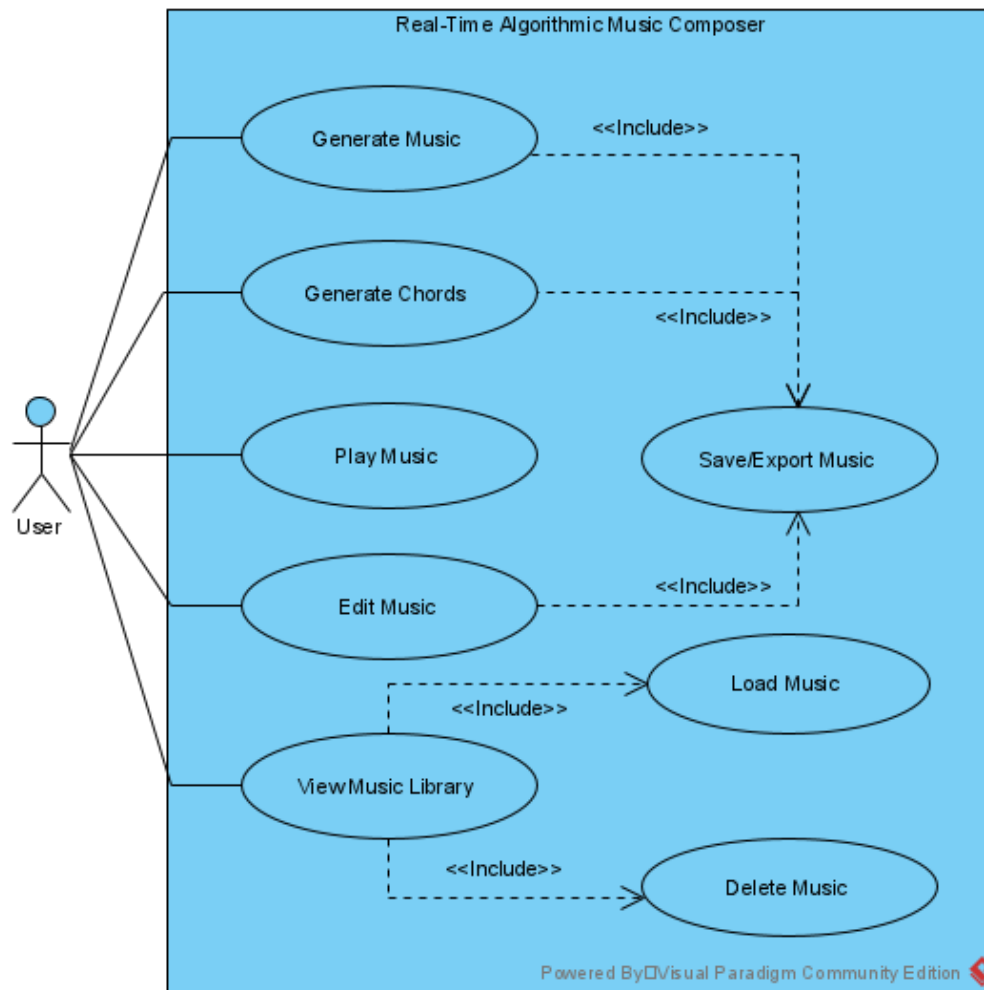


Figure 3.2 Use Case Diagram

As shown in the diagram above the sole actor of the system is the user. Users are able to perform various actions such as generating music, generating chord progressions, playing music, editing music and viewing their music library to load or delete music files. When music is generated or after a user finishes editing a piece of music, they are also able to save the music into a MIDI file.

### 3.1.3 Activity Diagram

The activity diagrams for each activity in the use case diagram are shown as follows:

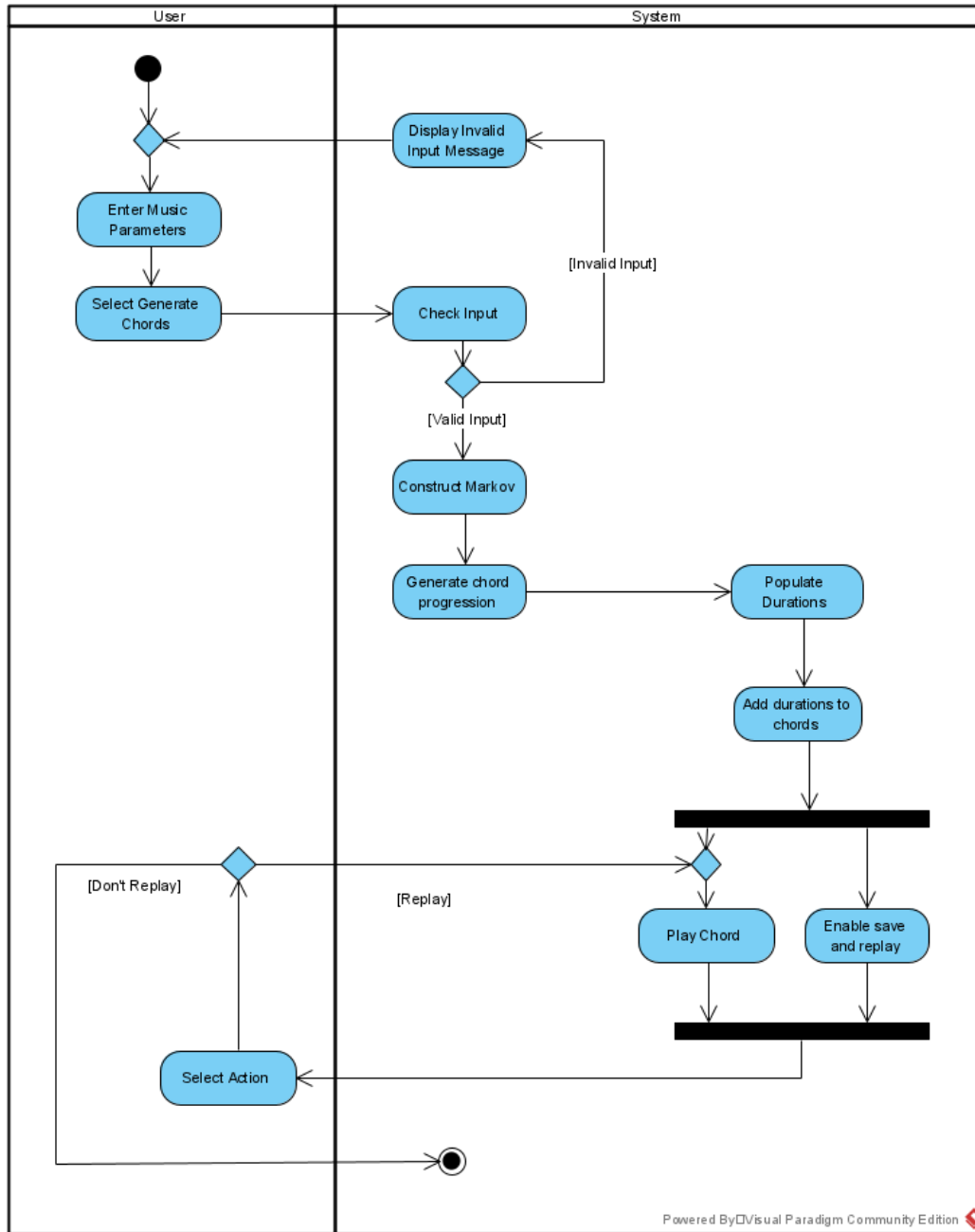


Figure 3.3 Activity Diagram for Generate Chords



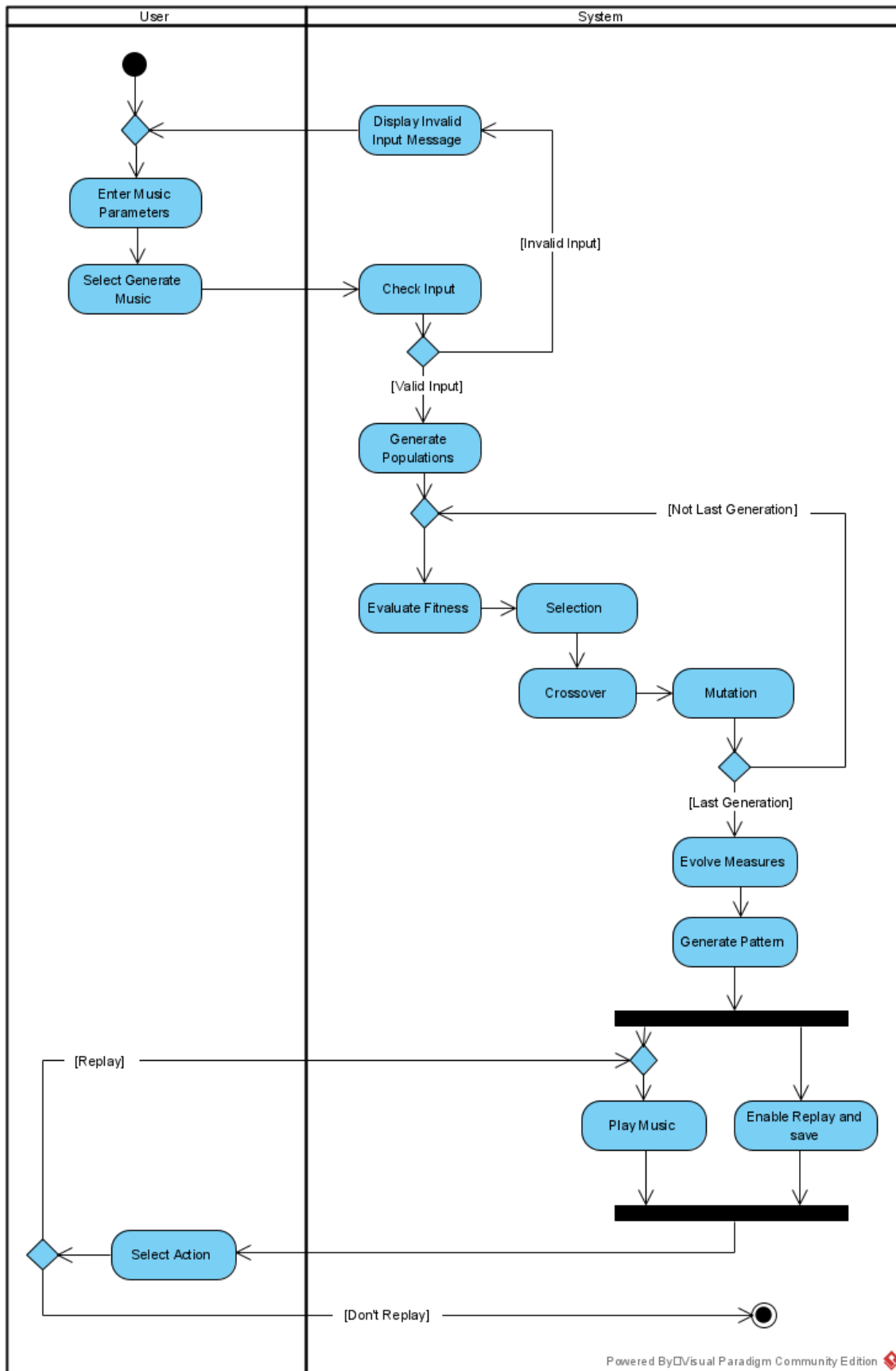


Figure 3.4 Activity Diagram for Generate Music

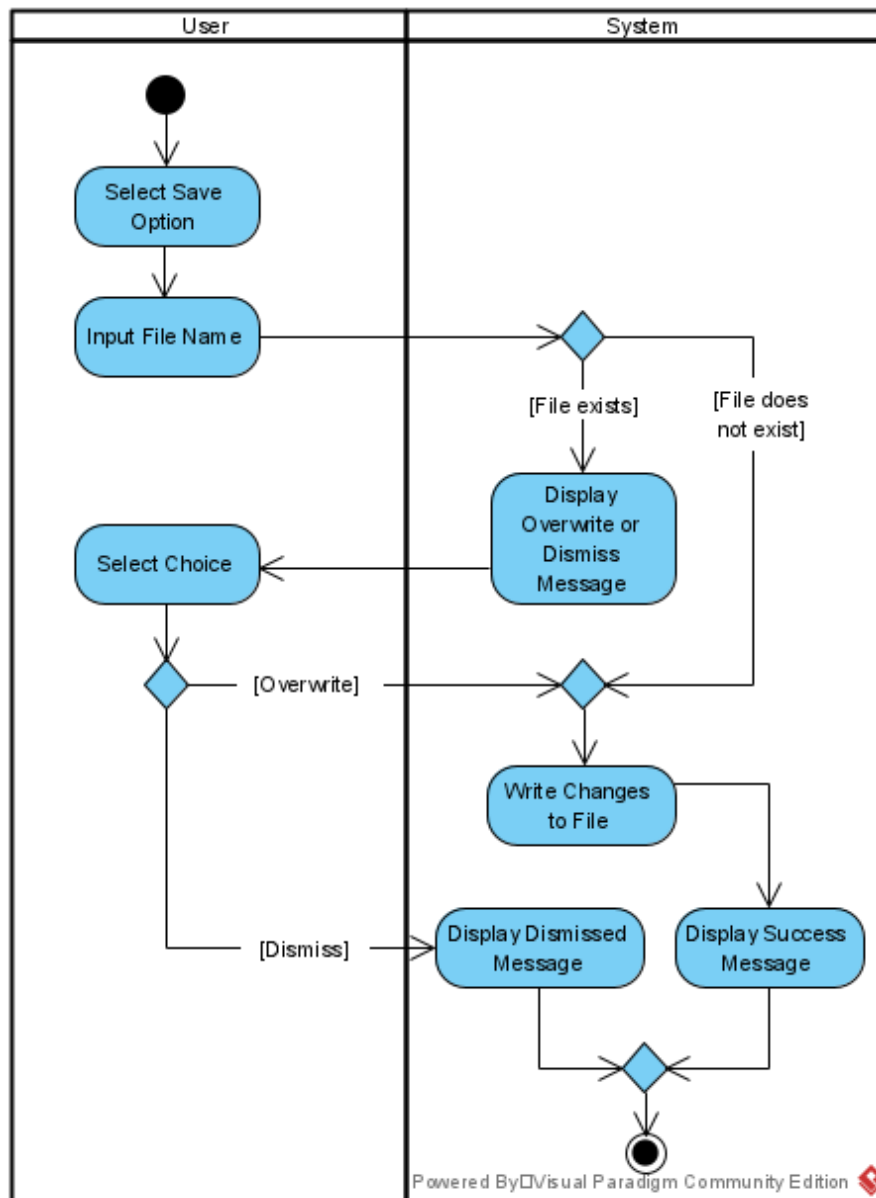


Figure 3.5 Activity Diagram for Save Music

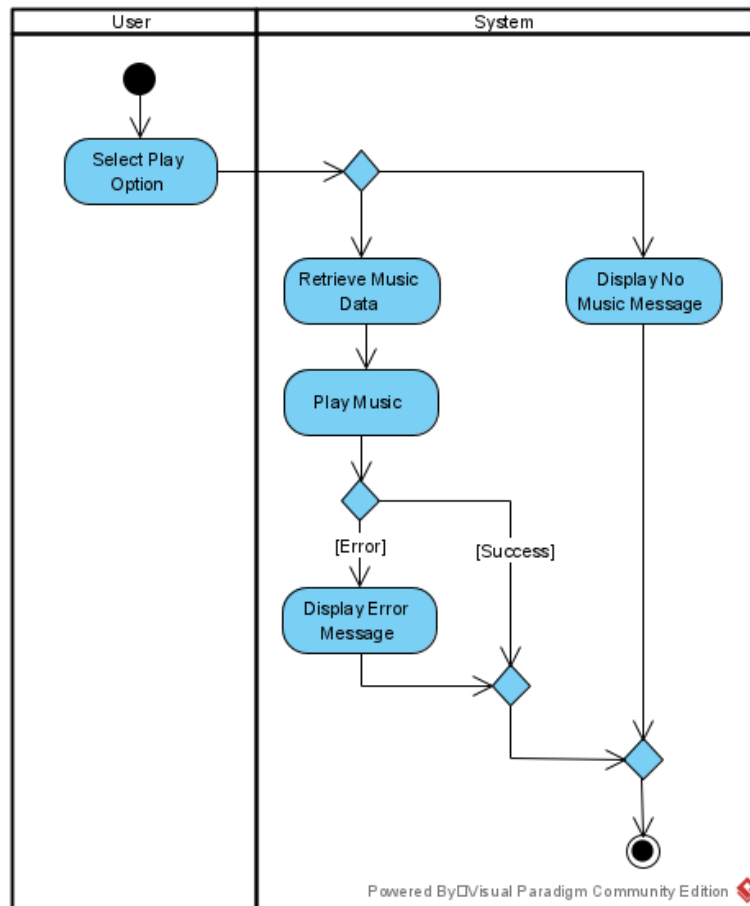


Figure 3.6 Activity Diagram for Play Music

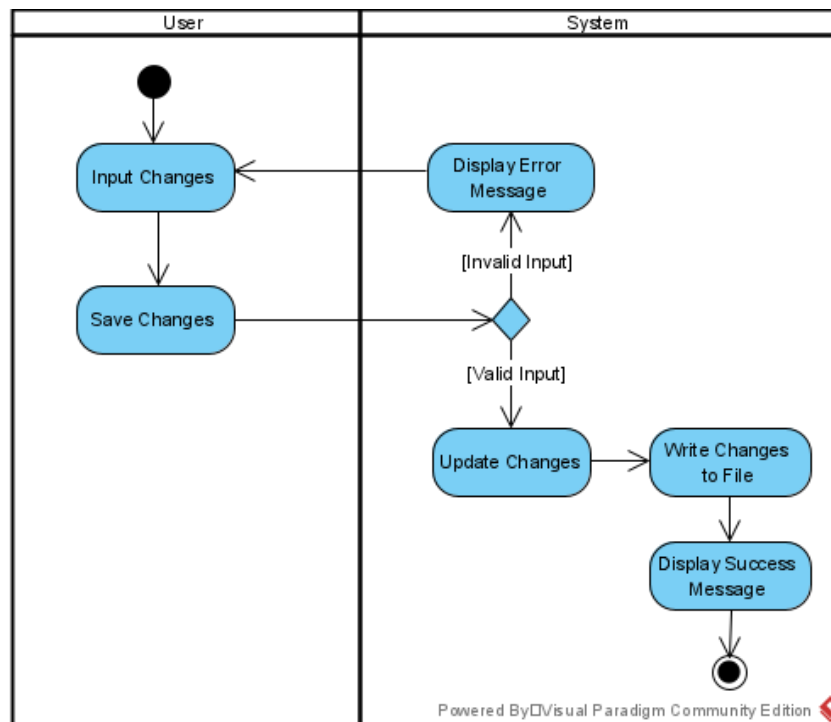


Figure 3.7 Activity Diagram for Edit Music

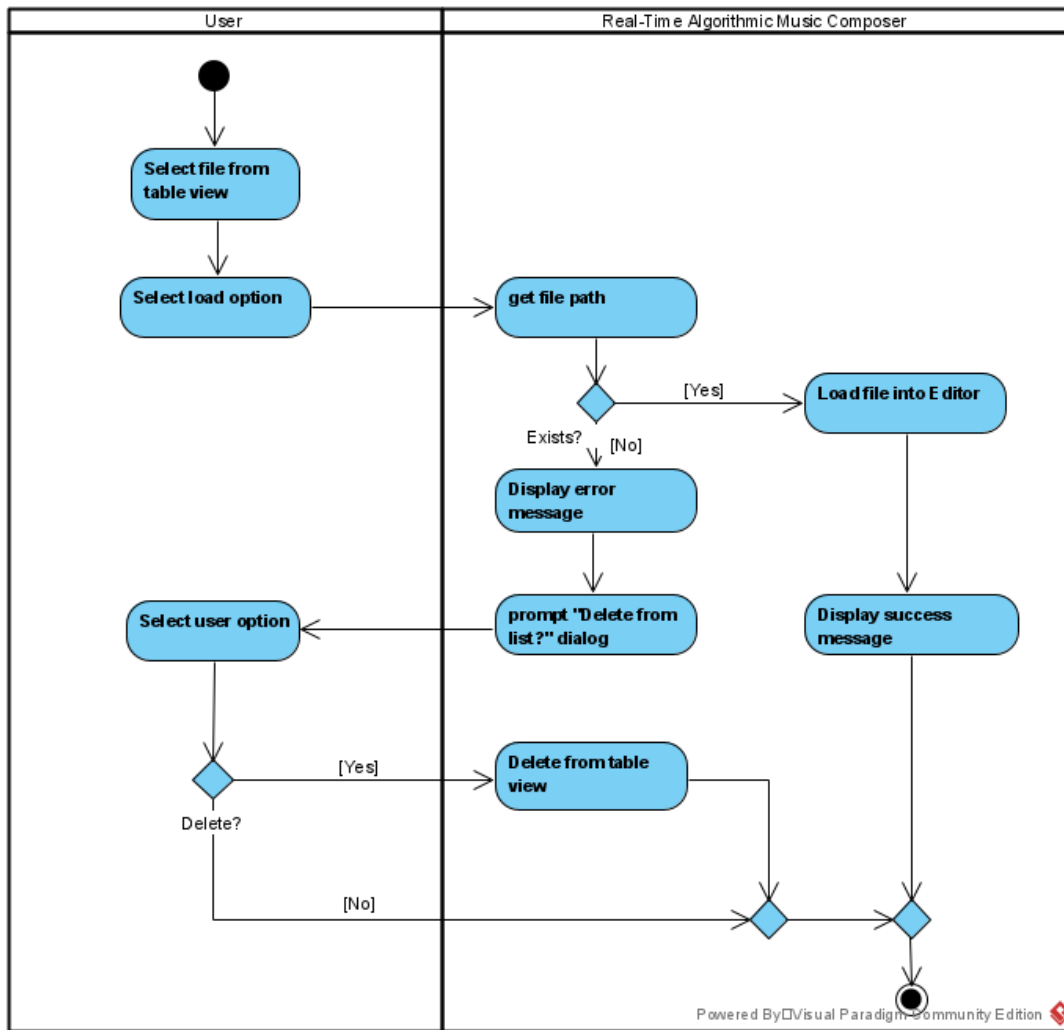


Figure 3.8 Activity Diagram for Load Music

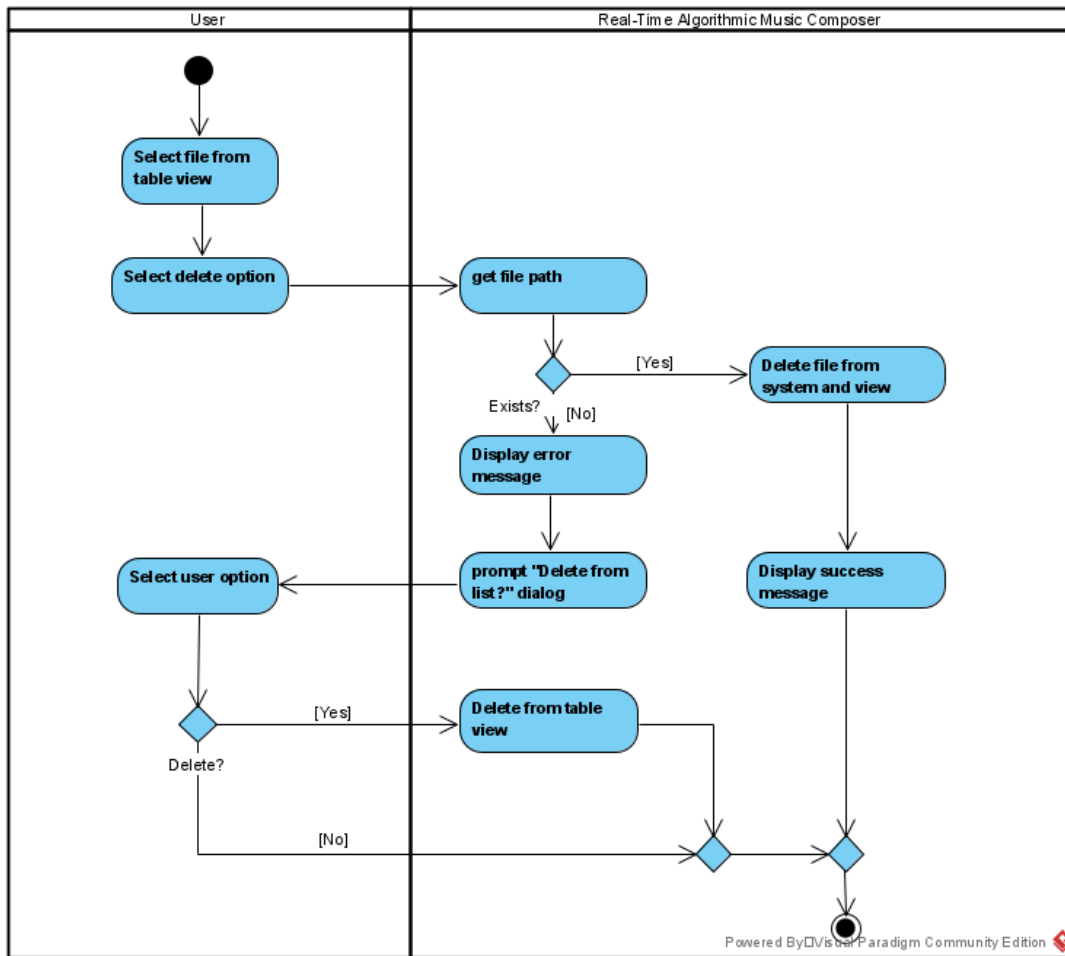


Figure 3.9 Activity Diagram for Delete Music

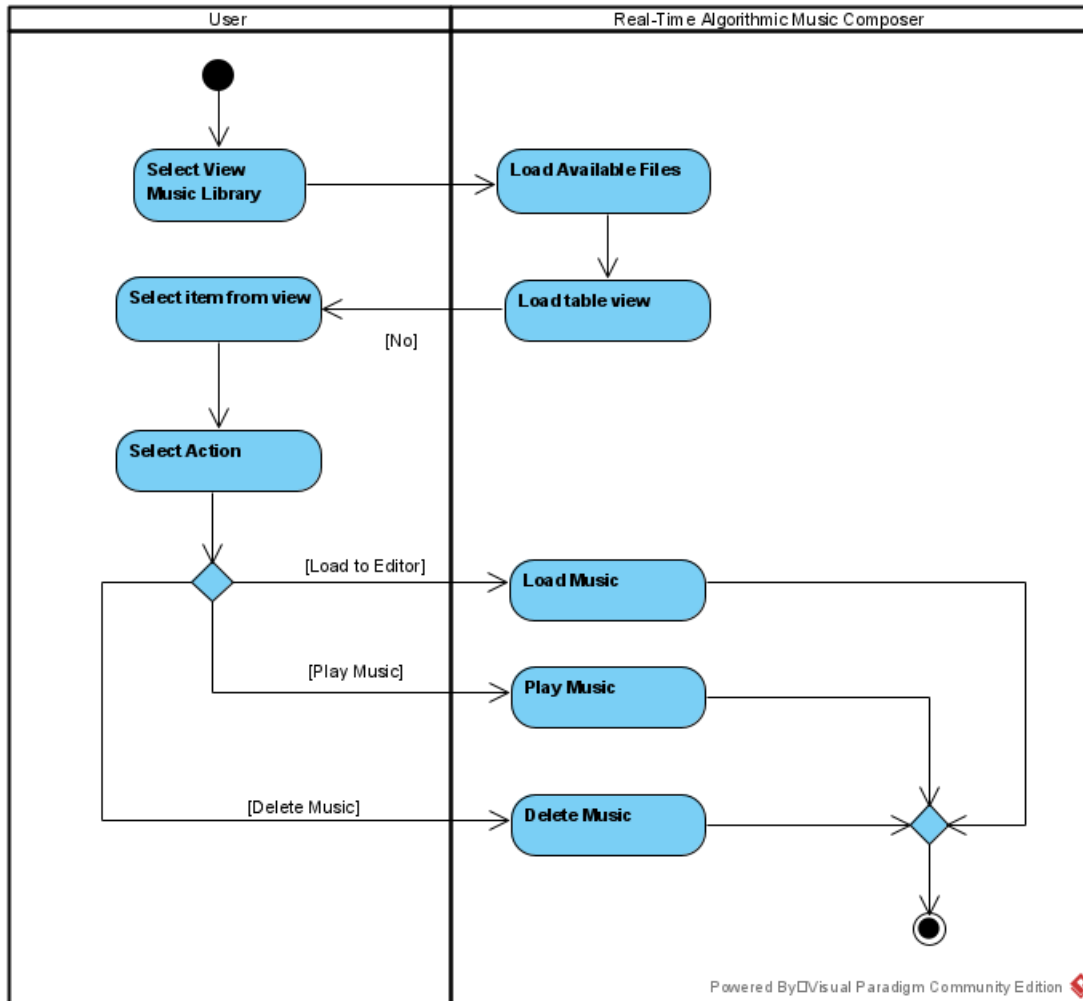


Figure 3.10 Activity Diagram for View Music Library

# Chapter 4

## System Design

This chapter will discuss the design of the system. The developed system will utilise the genetic algorithm and Markov chain to generate music and chord progressions. The following sections will explain in detail about what each parameter will be used for and the methods used for evolving the generated sequences of notes.

### 4.1 System Flow

#### 4.1.1 Music Generation

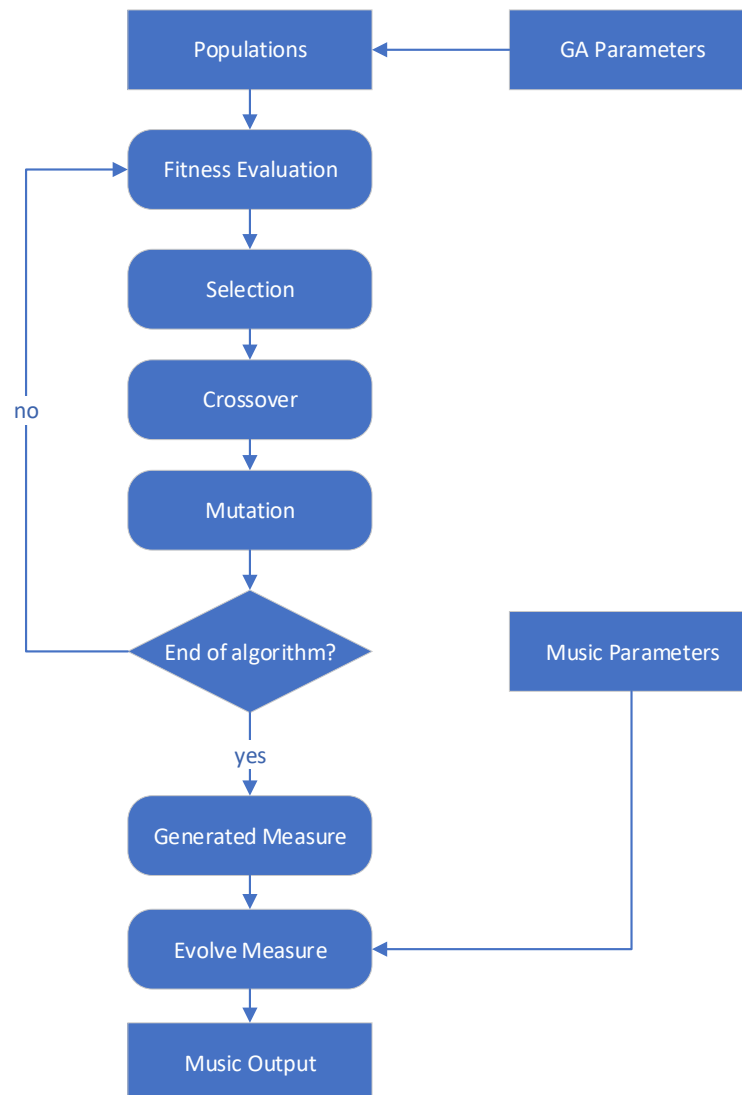


Figure 4.1 Flowchart for Music Generation

The flow for music generation is shown in the figure above. First, the system receives the GA parameters provided by the user, and determines the number of notes to generate, and how many generations to be run. After that, populations go through the steps of fitness evaluation, selection, crossover and mutation, where random notes will be chosen to perform the crossovers or mutation. If the current iteration is not the last generation, the process will be reiterated, else, if it is, then a generated measure will be obtained, and the system will use this sequence of notes to create the music based on the music parameters provided by the user. Finally, the music output is produced. Figure 3.2 shows the general flow of a genetic algorithm, whereas table 3.1 shows the implementation details of each parameter.

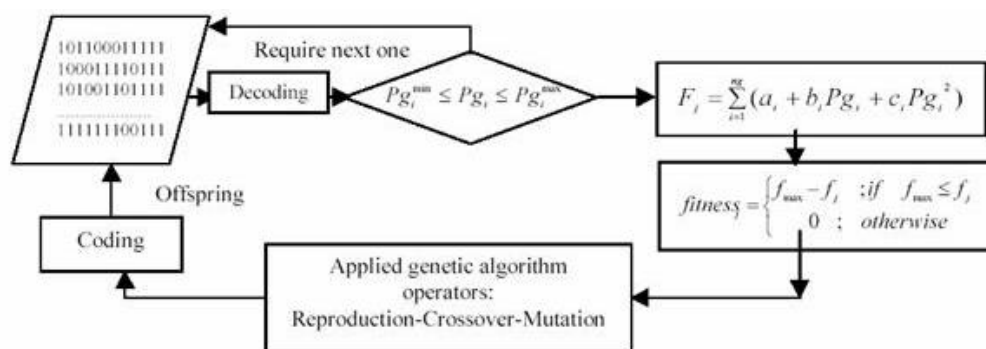


Figure 4.2 Flowchart of Genetic Algorithm

Table 4.1 Genetic Algorithm Parameters and Implementation Details

Parameter	Implementation Details
<b>Populations</b>	The number of populations (sequence of notes) that are to be generated. A total of $n$ sequences of notes (members) will be generated in a population. The default number of populations is set to 1000. Population is generated based on the other GA parameters provided + selected key and number of notes to generate.
<b>Fitness Evaluation</b>	For fitness evaluation, fitness is increased if the distance between prev. note and cur. note is equal or less than 5 notes is of the user selected octave (e.g. note E5 = 64, octave is C (60 to 71) and 64 is within 60 to 71) Fitness is decreased if the distance of two notes exceeds 12.



<b>Selection</b>	Selection is done by adding members of current generation with 1st and 2nd highest fitness to be included for the next generation.
<b>Crossover rate</b>	Crossover is performed on parents randomly selected through a roulette selection to form a total of $n$ children. Default rate set to 0.85. Decision on which patents are chosen to perform crossover on is decided using a roulette selection, so that members with higher fitness have more chances to be selected.
<b>Mutation rate</b>	The rate of how often a note will be mutated into another. Default rate set to 0.005.

Figure 4.2 shows an example on the resulting output of crossover and mutation.

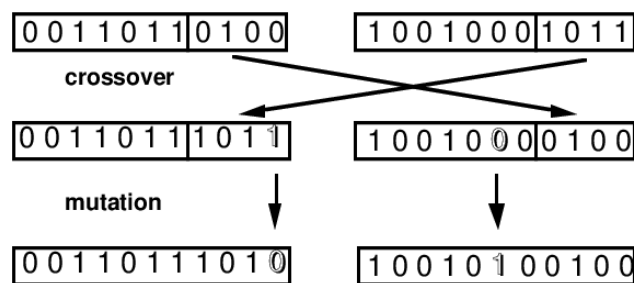


Figure 4.3 Crossover and Mutation

As mentioned, the program will sort the population based on fitness. After that, it increments the generation counter and loops through the process again. When the program is done running through 500 generations, it performs next step, which is to extend the produced sequence according to the music parameters. Music Parameters is determined from user input. The table below shows the parameters that the system is able to take in and generate music based on them.

Table 4.2 Music Parameters and Description: Generate Music

Parameter	Type	Description
<b>Instrument</b>	integer	The instrument number that represents a given instrument in JFugue. For example, the predefined value for PIANO is 0.
<b>Tempo</b>	integer	Determines the speed the notes are played.

<b>Measure Count</b>	integer	The number of measures to be generated.
<b>Note Length</b>	integer	The number of notes to be generated in one measure.
<b>Key</b>	string	The music will be generated in either C, C#, D or D# key depending on user selection.
<b>Volume</b>	integer	Sets the volume of the music to be played in. Adjusted by using the MIDI command.
<b>Voice</b>	string	Determines how many music layers are used for music generation. The system generates music using either two or four layers, depending on how many instruments the user selects.

Note durations are assigned to each note randomly after undergoing the following process, which randomly selects 1 of the 4 methods to extend the generated sequence:

1. Clone the sequence.
2. Reverse the sequence by half.
3. Reverse the whole sequence.
4. Mutate a note in the sequence.

*Table 4.3 Evolution Method and Example Output*

<b>Method</b>	<b>Example (Before and After)</b>
<b>Clone the sequence</b>	C5h D5q E5q D5q C5h D5q E5q D5q
<b>Reverse the sequence by half</b>	C5h D5q E5q D5q E5q D5q C5h D5q
<b>Reverse the whole sequence</b>	C5h D5q E5q D5q D5q E5q D5q C5h
<b>Mutate a note in the sequence</b>	C5h D5q E5q D5q C5h D5q E5q E5q

### 4.1.2 Chord Generation

For the generation of chord progressions, the following probability distribution will be used to construct a Markov chain. The probability distribution in Figure 3.3 was obtained through analysing a dataset of Classical music [20].

*Figure 4.4 Probability Distribution of Chord Succession in Classical Music*

		S e c o n d    C h o r d						
		I	ii	iii	IV	V	vi	vii
F i r s t  C h o r d	I		15%	1%	28%	41%	9%	6%
	ii	1%		1%	0%	72%	1%	26%
	iii	3%	3%		53%	6%	32%	3%
	IV	22%	13%	0%		39%	2%	23%
	V	83%	1%	0%	7%		9%	0%
	vi	12%	30%	5%	11%	33%		9%
	vii	90%	0%	1%	2%	4%	3%	

*Table 4.5 Music Parameters and Description: Generate Chords*

Parameter	Type	Description
<b>Instrument</b>	integer	The instrument number that represents a given instrument in JFugue. For example, the predefined value for PIANO is 0.
<b>Tempo</b>	integer	Determines the speed the notes are played.
<b>Key</b>	string	The music will be generated in either C, C#, D or D# key depending on user selection.
<b>Order</b>	Boolean	Fixed or Random. Determines the order of the chords played. Recommended to set as fixed.
<b>Durations</b>	Boolean	Fixed or Random. Determines the duration each chord is played. Fixed chords play an eighth note long. Random durations may include a whole note, half note, quarter note or eighth note.

## 4.2 User Requirements

### Functional Requirements

1. Users shall be able to use the system to generate music.
2. Users shall be able to provide the parameters for the generated music, including instrument type, tempo, measure count and notes generated per measure.
3. Users shall be able to listen to the generated music.
4. Users shall be able to save the music as a music file type (.mid, .wav, etc.).
5. Users shall be able to load and edit the piece of generated music.
6. Users shall be able to view the history of their previously generated music.

### Non-Functional Requirements

1. The system shall be able to produce musical output in real-time.
2. The system shall be able to run in a Windows environment.
3. The system shall provide users with an easily navigable and user-friendly graphical interface.

## 4.3 System Specifications

The system should be able to run with the minimum requirements as shown below:

*Table 4.6 System Specifications*

Description	Specifications
<b>Processor</b>	Intel Core i7-3520M or higher
<b>Operating System</b>	Windows 10 64-bit
<b>Memory</b>	4GB of system memory
<b>Storage</b>	150MB of available space

### 4.5 System User Interface Design

The designed user interface for the system is shown below. Generally, main controls will be displayed in the centre, while in the bottom part, there will be the controls like generate, play, and save. Besides that, there will also be a left navigation bar for users to navigate between the system's functionalities. The two following diagrams show the basic design templates the system will use.

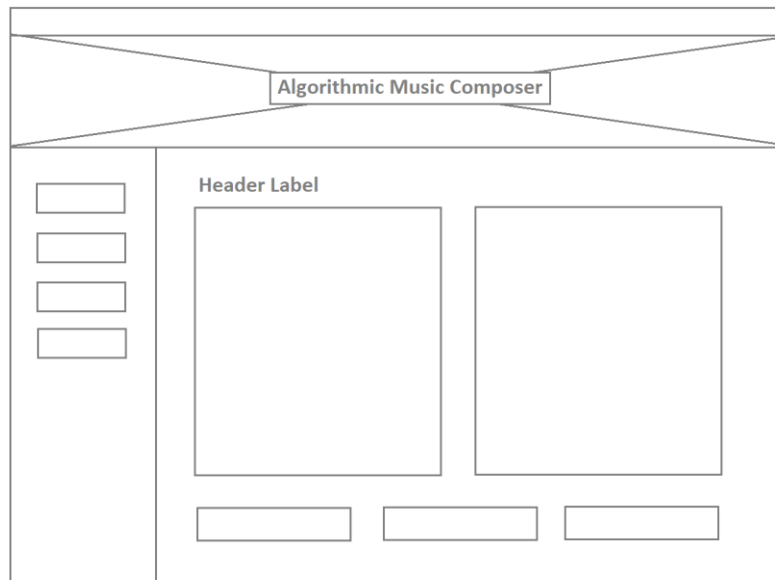


Figure 4.4 User Interface Design for Basic Music Generation and Editing

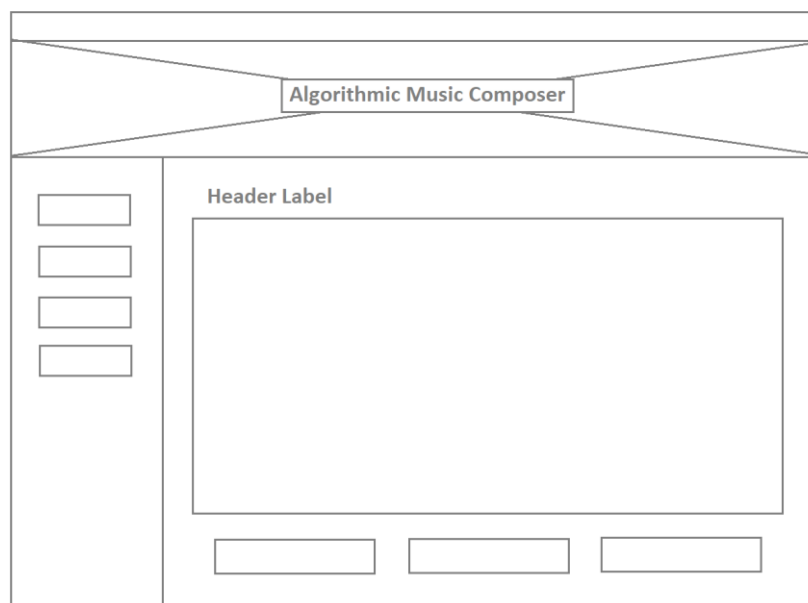


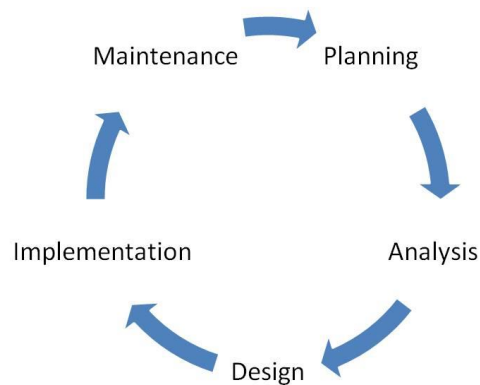
Figure 4.5 User Interface Design for Music Library

# Chapter 5

## System Implementation

This project will implement a rapid application development (RAD) based methodology for the development process. Besides that, the application will be built using the Java programming language whilst integrating the use of Java APIs. In this chapter, the implementation details such as methodology, technologies and tools involved, project timeline, and challenges faced during implementation will be discussed.

### 5.1 System Methodology



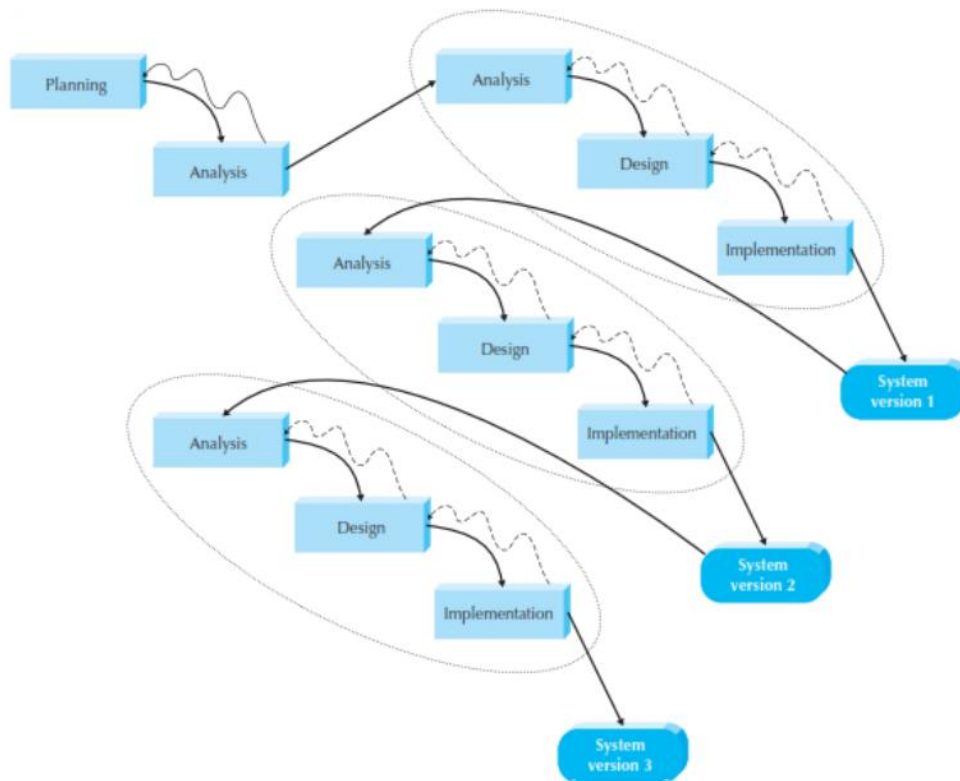
*Figure 5.1 System Development Life Cycle*

A system development lifecycle (SDLC) consists of several phases that each contributes to the process of building a computerised system. Although there have been several versions of the SDLC which derived from each other over time, one of the interpretations which is most commonly agreed upon consist of the five phases of planning, analysis, design, implementation and maintenance [21].

With that in mind, a methodology refers to an organised, systematic way for implementing the SDLC. For this project, the phased development model is selected to build this system, as it provides a higher degree of flexibility as compared to structured designs like the waterfall development model. The phased development model falls under the category of rapid application development (RAD), in which the system development is done through frequent

iterations of analysis, design and implementation before the final version of the system is produced. On top of that, Improvements and modifications to the system's design is also made through continuous feedback.

## 5.2 Project Workflow in Phased Development Model



*Figure 5.2 Phased Development Model*

Figure 5.2 shows the process of a phased development model. As shown, the whole system is broken down into numerous versions that would be developed in a sequential manner.

### Planning

Firstly, the scope of the existing problems will be determined, and the solutions for the problems would be mapped out during the planning phase. Besides that, other aspects to consider during this phase also includes the resources needed to execute the project.

### Analysis

Next, the analysis phase consists of determining the system's users and its functional requirements. The user's needs are analysed and gathered, so that a design for the requirements

can be formulated to be implemented within the system. As a result, the overall system concept will be identified during this phase.

### **Design**

The third phase is the design phase, which refers to the construction the algorithms and operations required for the system to be able to meet its functional requirements. Additionally, this also includes the architectural and interface design for the system.

### **Implementation**

In the fourth stage, which is the implementation phase, the design of the system will be implemented by coding it. During this stage, the system will be programmed and tested before the system is launched to the users. Any potential bugs will be identified and fixed accordingly.

### **Reiteration of the Analysis, Design and Implementation Phases**

By implementing this development model, the analysis, design, and implementation phases are looped several times before the final version of the system is launched. The most fundamental requirements to the overall system would be prioritised in the first iteration of the analysis phase, and would therefore be included in the first version of the system. The benefit of using this development model is that additional analysis could be done based on the requirements we have made previously, so that new ideas can be produced, combined and be implemented along with previous ones to keep improving the system in its later versions.

### **Maintenance**

Lastly, following the launch of the final system version, maintenance is then done by monitoring the system's performance to ensure that the system is up and running as intended.

## **5.3 Project Timeline**

For Project 1, the tasks that are required to be done are defined according to the report chapters, and they can be found in the Gantt chart plotted below. Tasks include planning, analysis, implementation and the related write-up for the report, as well as other tasks like final checking and submission. Besides that, for project 2, tasks include continuing to develop the rest of the system's functionalities, final testing, debugging and implementing other



refinements to the system, and finally, last checking and submission. Details about each task is listed in the Gantt charts below:

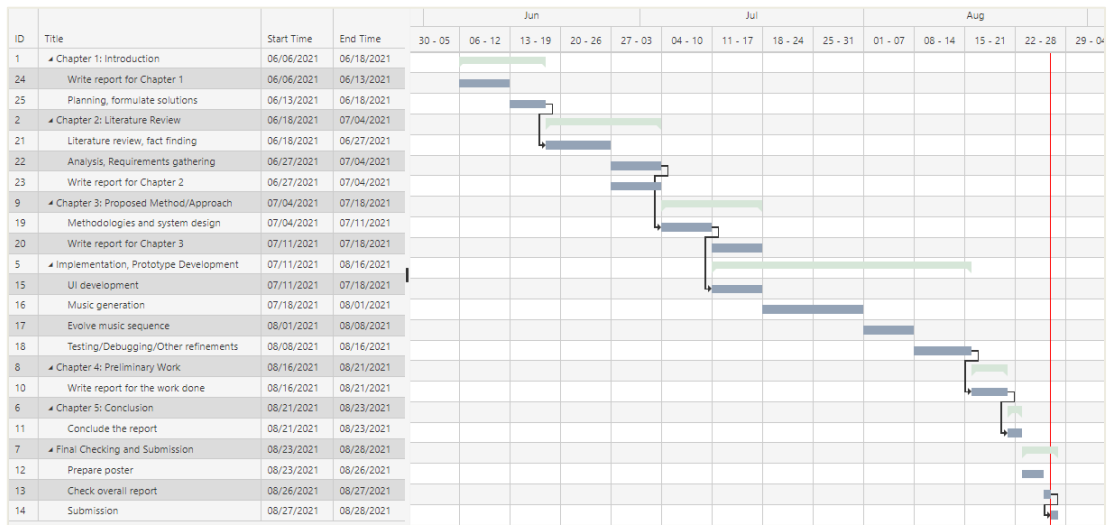


Figure 5.3 Gantt Chart for Final Year Project 1

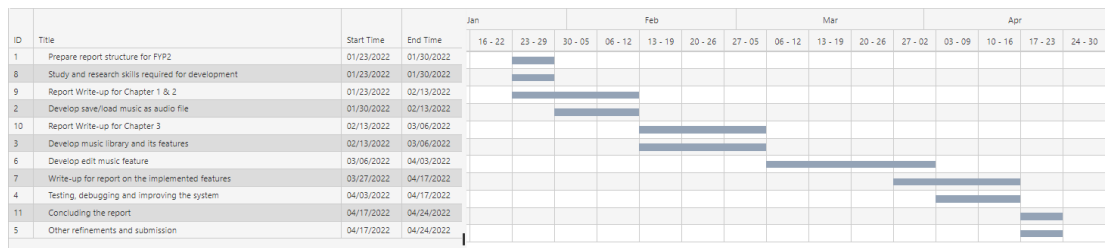


Figure 5.4 Gantt Chart for Final Year Project 2

### 5.4 Technologies and Tools Involved

The Java programming language will be used for the development of this system. This is because Java provides various advantages such as its Object-Oriented capabilities that can enable us to reuse code, which could save us a lot of development time. Moreover, Java is also relatively simple to understand, write and debug as compared to other languages like C++, partially due to its automatic allocation of memory and garbage collection, whereas C++ requires the programmer to manually do so.

Besides that, Java also has many available application programming interfaces (APIs) like JavaFX for GUI programming, and JFugue, which could help in programming music [22]. Compared to Swing, JavaFX has more consistent updates and MVC support. Moreover, it also

provides the ability to markup UIs with FXML and also CSS theming. Besides that, JFugue's API provides a wide range of music programming capabilities that can simplify the programming process tremendously. Additionally, JFugue also has a proper guide on how to use the API, and documentation of the API could also be found online for reference. Lastly, the application will be developed in, and be primarily developed for, a Windows environment. With that being said, the development of the system will be done in the Eclipse IDE, an open-source integrated development environment compatible with Windows, Mac and Linux devices. Its auto-formatting capabilities, customisability, code completion, increased readability and other useful features is able to enhance the development process by reducing the time and effort required.

Therefore, based on the advantages mentioned, by using the mentioned technologies, it would allow us to significantly reduce the time and cost needed to develop the system.

In conclusion, the hardware and software used for this project is listed below:

### 1. Hardware Involved:

- Laptop

*Table 5.1 Specifications of laptop*

Description	Specifications
<b>Model</b>	Dell Latitude E6430
<b>Processor</b>	Intel Core i7-3520M CPU @ 2.90GHz
<b>Operating System</b>	Windows 10 64-bit
<b>Memory</b>	12GB DDR3 RAM
<b>Storage</b>	Apacer AS340 480GB SSD

### 2. Software Involved:

- Java 11.0.4
- Java APIs – jFugue 5.0.9, JavaFX
- Eclipse IDE



*Figure 5.5 Software Involved*

### **1.7 Implementation Outcome**

This section will show the results of the developed system. The generate music feature is shown in Figure 5.6. By using the GUI provided, users can generate random music after providing the required parameters. Users have a range of 10 different instrument to choose from. This includes "Piano", "Accordion", "Acoustic Bass", "Electric Piano", "Flute", "Guitar", "Trumpet", "Trombone", "Violin", "Steel Drums", and "Taiko Drum. For the second instrument selection, it can be set as "None" a single instrument is preferred.

Keys users are able to select form are "C", "C#", "D" and "D#". Other fields that may be set by the user are the tempo, measure count and note length. A volume slider is also provided for users to freely adjust the volume of the music to be generated. Genetic algorithm parameters are predefined and are set in their corresponding text fields upon launch of the program. However, users may also adjust it according to their preferences.

In Figure 5.7, users will be able to generate chord progressions. Similar to the parameters that are required for music generation, users may provide their preferred instrument, key, tempo and measure count. While generating chords, users may also select the option to generate them in fixed or random order, and whether they would like every chord to be played in uniform duration or a mix of different durations.

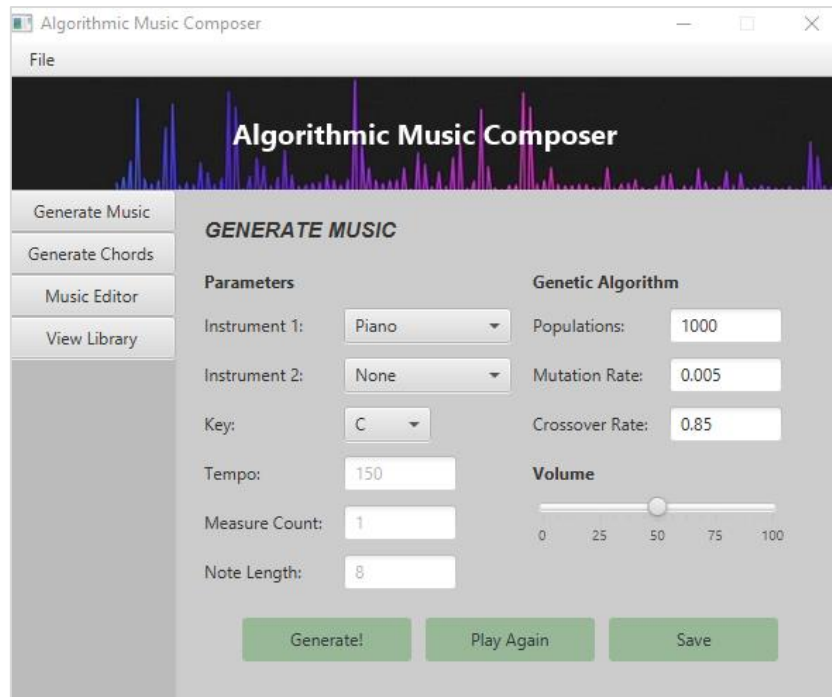


Figure 5.6 Screenshot of the System (Generate Music)

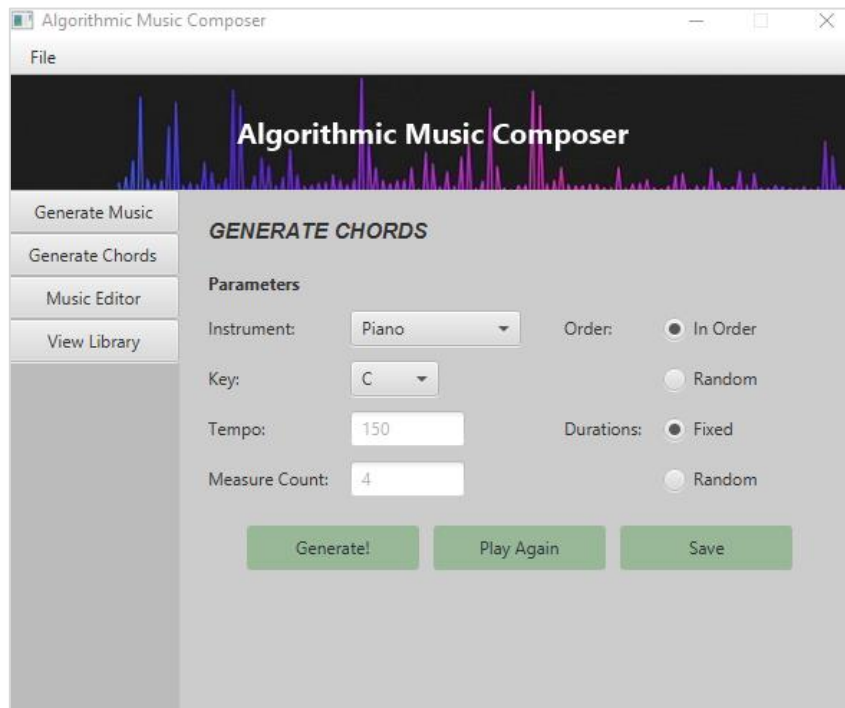
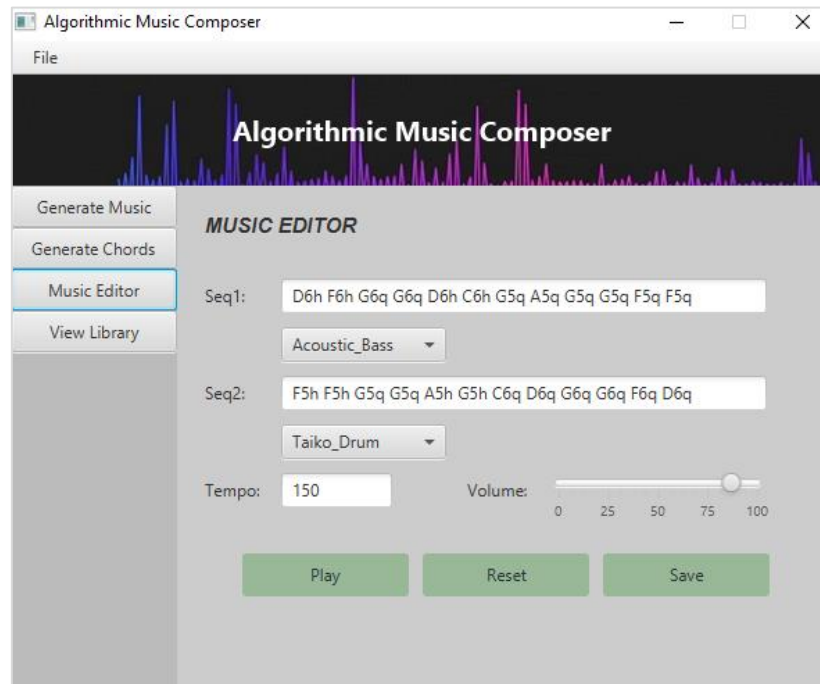


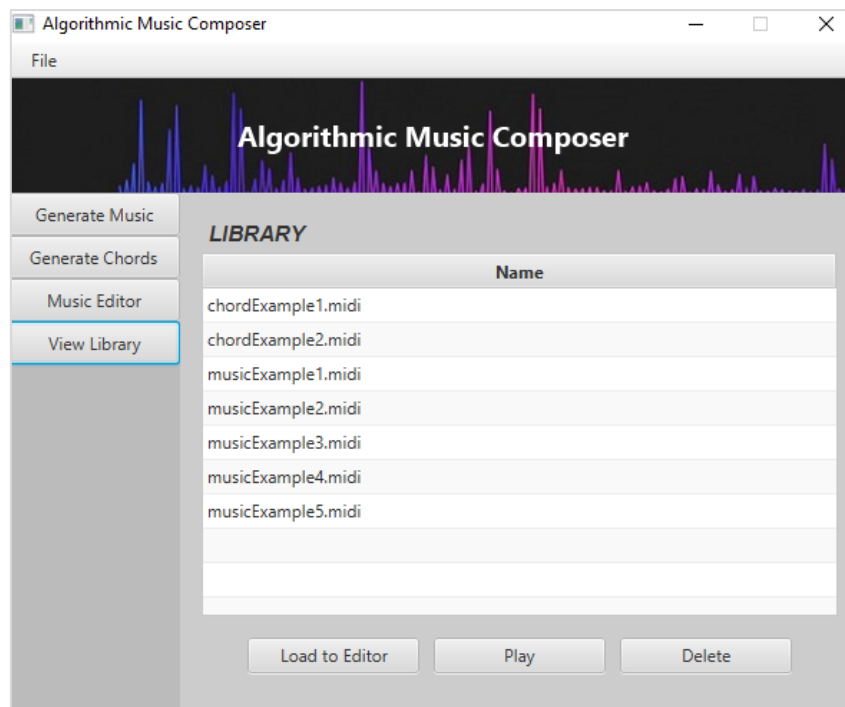
Figure 5.7 Screenshot of the System (Generate Chords)

The music editor is shown in Figure 5.8 below. The fields of Seq1 and Seq2 display the sequence of notes generated by the system. These fields are non-editable within the system, but users may use third-party MIDI editors to modify the notes in a higher-level view as compared to the string notation shown. On the other hand, users may adjust the tempo and volume within the system, and save the edited file as a new MIDI file or overwrite the previously saved file.



*Figure 5.8 Screenshot of the System (Music Editor)*

Figure 5.9 shows a table view of the music files the user has generated. By clicking an item from the view, users may load the selected file to the music editor, play the music or delete it. Multithreading was applied while developing the system so that users are able to navigate the system simultaneously as music is being played.



*Figure 5.9 Screenshot of the System (Music Library)*

## Chapter 6

# System Evaluation and Discussion

This chapter will be about the evaluation of the results obtained after rounds of testing, discussion about the challenges faced while developing the system, as well as evaluating how well the system managed to meet the project's objectives.

### 6.1 Testing Setup and Result

The system performance will be evaluated on accuracy, efficiency and the speed of computation. First and foremost, the system should be able to accurately perform the functionalities as requested by its users. Besides that, the quality of the music produced by the system should also be consistent and according to the parameters configured by the user. Generated music should be harmonic and pleasant to listen to.

Besides that, the system should also be efficient in the way such that user input should be minimal in order to compose music. The system should also utilise the available resources. Computation resources shall have to be focused on to avoid underutilisations and reduction of computational speed, and be able to return composed musical pieces based on the user input in real time

Below shows the verification plans for the scenarios of Music Generation, Save Music, Load Music, Edit Music, Play Music, View Music Library, Chord Generation and Delete Music.

#### 1. Music/Chord Generation

*Table 4.7 Verification Plan P1 (Music/Chord Generation)*

Procedure Number	P1
Method	Testing
Purpose/Scope	To ensure that the composer can successfully generate music based on user's input parameters.
Special Conditions/Limitations	Users may provide an invalid input, such as a non-integer value for a field that requires an integer.

Acceptance Criteria	The system correctly generates a piece of music according to the input given.
Prerequisites	None
Procedures	<ol style="list-style-type: none"> <li>2. Launch the application.</li> <li>3. Select instrument.</li> <li>4. Input the corresponding values into the tempo, measures to generate, notes generated fields or any other which applicable</li> <li>5. Click on the generate button. <ol style="list-style-type: none"> <li>a. If fields are correct, proceed with step 5.</li> <li>b. Else if fields are empty or incorrect, ensure that a pop-up window appears alerting users to provide the correct input.</li> </ol> </li> <li>6. Observe whether the system generates the correct music.</li> </ol>
Troubleshooting	Revise written code, ensure that the correct parameters are being passed into the functions and that their data type are correct.

## 2. Save Music

*Table 4.8 Verification Plan P2 (Save Music)*

Procedure Number	P2
Method	Testing
Purpose/Scope	To ensure that the system can successfully save music.
Special Conditions/Limitations	Error may occur while saving the file.
Acceptance Criteria	The system saves the music with the correct file name and file type, and correct data is written into the file.
Prerequisites	Generate music on the system before testing.
Procedures	<ol style="list-style-type: none"> <li>1. Click on the save music button. <ol style="list-style-type: none"> <li>a. If music has not been generated, ensure that a pop-up window appears to alert the user.</li> <li>b. Else if the music is generated, continue and</li> </ol> </li> </ol>



	<p>proceed with step 3.</p> <ol style="list-style-type: none"> <li>2. Input the name of the file to be saved.</li> <li>3. Click on the save button.</li> <li>4. If a file with the same name exists, alert the user on whether they would like to cancel or overwrite. Else, skip to step 5. <ol style="list-style-type: none"> <li>a. If overwrite, ensure that the file is overwritten.</li> <li>b. Else if cancel, ensure that the save procedure is dismissed.</li> </ol> </li> <li>5. Ensure that the system displays success message to user upon successful save. <ol style="list-style-type: none"> <li>a. If the file failed to save, ensure that the system displays error message to the user.</li> </ol> </li> <li>6. Locate the file in the system explorer.</li> <li>7. Observe whether the file is saved in the correct file name and file type.</li> </ol>
Troubleshooting	Revise written code, ensure that the correct data is being written into the file or that the path is correct.

### 3. Load Music

*Table 4.9 Verification Plan P3 (Load Music)*

Procedure Number	P3
Method	Testing
Purpose/Scope	To ensure that the system is able to load a previously generated piece of music.
Special Conditions/Limitations	The file intended to be read may be corrupted.
Acceptance Criteria	The file is able to be successfully loaded into the system.
Prerequisites	Ensure that there is a save file or related file type on the system.
Procedures	<ol style="list-style-type: none"> <li>1. Navigate the music library.</li> <li>2. Locate the file to be loaded.</li> </ol>

	<ol style="list-style-type: none"> <li>3. Click on the load button. <ol style="list-style-type: none"> <li>a. If the file has been loaded successfully, ensure that the system displays success message to user.</li> <li>b. Else if the file failed to load, ensure that the system displays error message to the user.</li> </ol> </li> <li>4. Observe whether the file has been successfully loaded into the system.</li> </ol>
Troubleshooting	Revise written code, ensure that the correct file is being selected for read.

#### 4. Edit Music

*Table 4.10 Verification Plan P4 (Edit Music)*

Procedure Number	P4
Method	Testing
Purpose/Scope	To detect whether the generated music can be edited.
Special Conditions/Limitations	-
Acceptance Criteria	The system saves the edited music with the updated parameters.
Prerequisites	Ensure that there is a generated piece of music, or that a file has been loaded into the system for the user to edit.
Procedures	<ol style="list-style-type: none"> <li>1. Edit the fields of the music parameters.</li> <li>2. Execute Verification Plan P2 (Save File).</li> <li>3. Execute Verification Plan P3 (Load File).</li> <li>4. Observe whether the parameters have been updates to the newest values.</li> </ol>
Troubleshooting	Revise written code, ensure that the correct data is being passed and that it overwrites the previous values.

## 5. Play Music

*Table 4.11 Verification Plan P5 (Edit Music)*

Procedure Number	P5
Method	Testing
Purpose/Scope	To detect whether the generated music can be edited.
Special Conditions/Limitations	Wrong file type provided.
Acceptance Criteria	The system is able to play the music.
Prerequisites	Ensure that the system has either generated a piece of music, or that a file has been loaded and could be played.
Procedures	<ol style="list-style-type: none"> <li>1. Click on the play music button.</li> <li>2. Observe whether music is being played.</li> </ol>
Troubleshooting	Revise written code or ensure that that the correct file is being read.

## 6. View Music Library

*Table 4.12 Verification Plan P6 (View Music Library)*

Procedure Number	P6
Method	Testing
Purpose/Scope	To ensure that the music files are being displayed in the system's music library.
Special Conditions/Limitations	No related files found on the user's system.
Acceptance Criteria	The system is able to play the music.
Prerequisites	-
Procedures	<ol style="list-style-type: none"> <li>1. Navigate the music library.</li> <li>2. Observe whether the system displays the list of music that has been saved by the user.</li> <li>3. Observe whether the list can be interacted with (e.g. loading and editing the selected music).</li> </ol>

Troubleshooting	Revise written code, ensure that that the paths of the files are correct.
-----------------	---

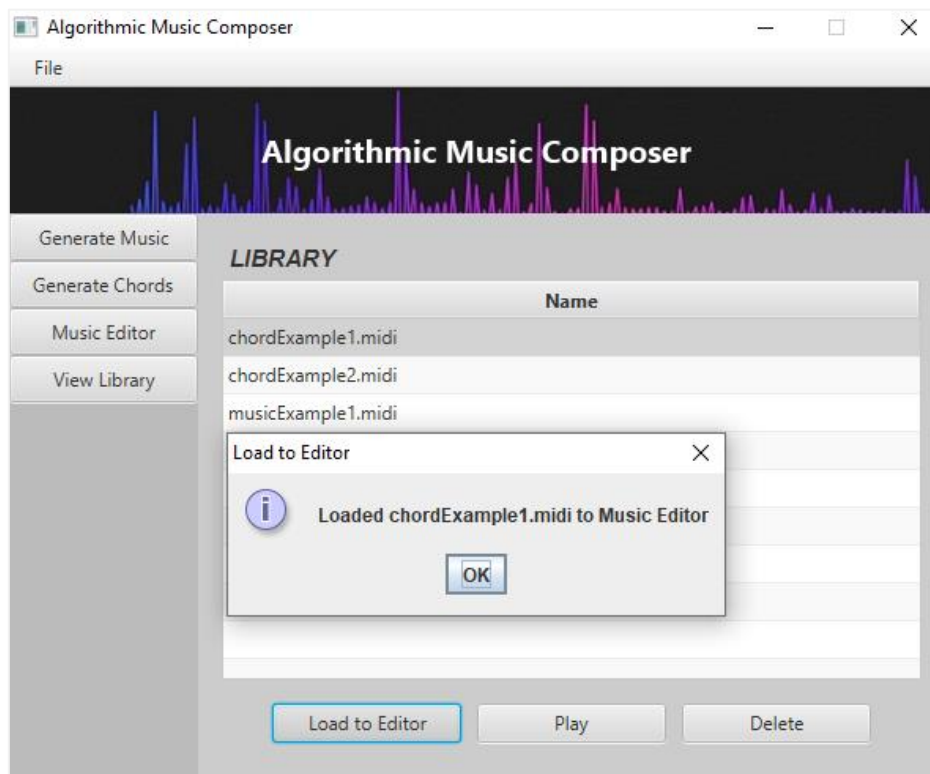
## 7. Delete Music

*Table 4.13 Verification Plan P7 (Delete Music)*

Procedure Number	P7
Method	Testing
Purpose/Scope	To ensure that the system is able to delete the midi file of a previously generated piece of music.
Special Conditions/Limitations	The file intended to be deleted may have been removed or renamed.
Acceptance Criteria	The file is able to be successfully deleted from the system.
Prerequisites	-
Procedures	<ol style="list-style-type: none"> <li>1. Navigate the music library.</li> <li>2. Locate the file to be deleted.</li> <li>3. Click on the delete button. <ol style="list-style-type: none"> <li>a. If the file has been deleted successfully, ensure that the system displays success message to user.</li> <li>b. Else if the file failed to be deleted, ensure that the system displays error message to the user.</li> </ol> </li> <li>4. Observe whether the file has been removed from the system.</li> </ol>
Troubleshooting	Revise written code, ensure that the correct file is being selected for read.

The validation plans above were used to ensure that the system provides the correct output each time. The results were correct and the system is working as intended, alerting the users with an error message each time an invalid input was provided. Besides that, music was also able to be generated by the system based on the user's input parameters without fail.

All outputs managed to be exactly according to the user's input. The diagrams below show some examples of the successful test results. The generated sequences and other parameter settings were able to be seen loaded into the Music Editor, as well as the music files which have been successfully exported as MIDI files. The generated music is playable in the developed system, and opening the file using Windows Media Player, we can also see that it is able to be played as intended:



*Figure 6.1 Load from Music Library*

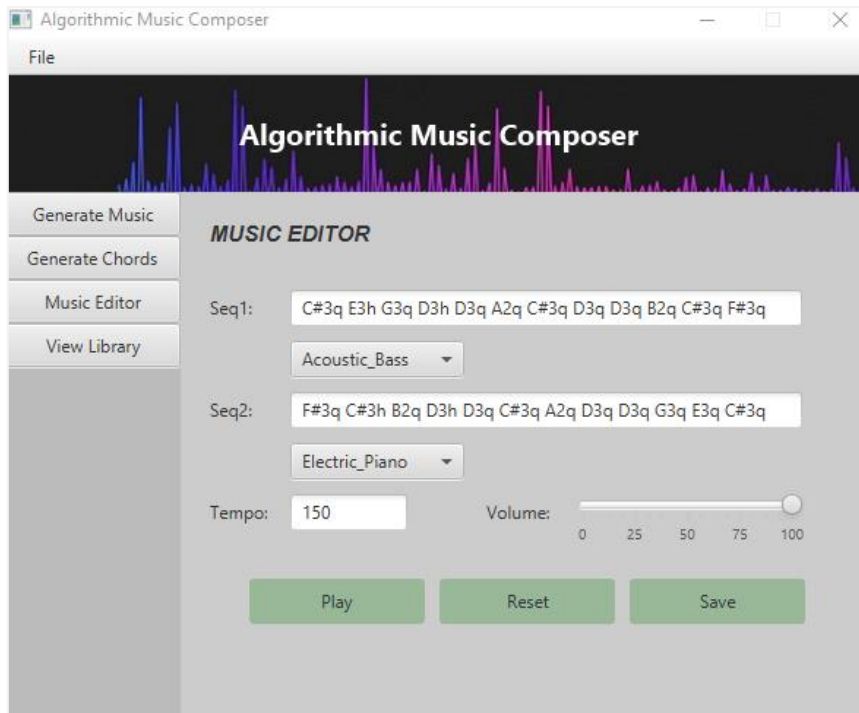


Figure 6.2 Loaded to Editor

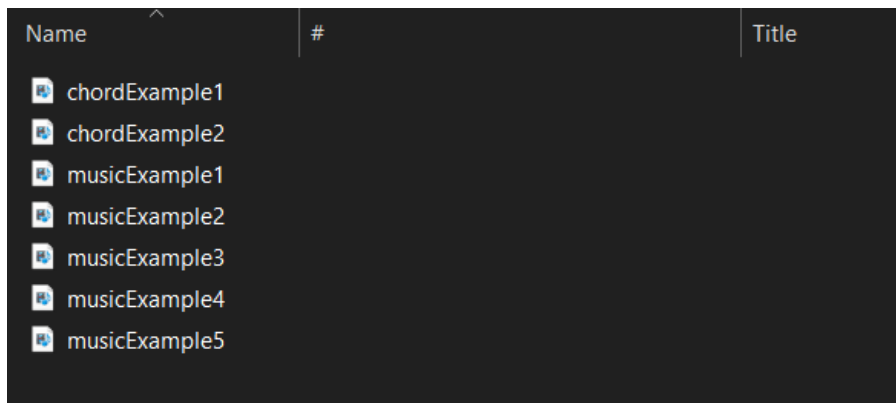


Figure 6.3 Saved Files in System



Figure 6.4 Opening in Windows Media Player

## 6.2 Project Challenges

The issues and challenges faced in the development of this project include the fitness function to be used for determining the quality of music, as unlike how humans evaluate music, the system has to have a predefined set of criteria on how to determine the quality of the produced music. If the fitness is not defined properly, the quality of the music generated by the system will not be good. Moreover, as what is considered “good music” is relatively subjective, it is also difficult to determine the system’s composition capabilities through the opinion of oneself, and therefore may require to gather the evaluations and opinions of others for proper benchmarking. Besides that, other challenges include time restrictions and the level of musical skill. Due to the unfamiliarity with music, it would be necessary to frequently refer to other sources for knowledge and information regarding this field, which could potentially be time consuming. Therefore, proper time management and planning ahead is required so that risks of falling behind schedule can be avoided.

## 6.3 Objectives Evaluation

In Chapter 1, it was stated that there are three main objectives to be achieved by this development project. From the results obtained, it can be said that the system successfully meets all objectives stated. This can be observed from:

1. The music composer returns results almost instantaneously after the user provides the required parameters and clicks on the “generate” button. This is a lot simpler and faster compared to traditional methods of composing music. With the use of this system, composing music could be as easy as one single click of a button.
2. The user only has to provide easy, simple to understand inputs to generate music. No complicated music knowledge is needed in the process of generating music. Besides that, the use of JavaFx to create a GUI also improves the usability and user-friendliness of the system.
3. The genetic algorithm was used to compose music. Moreover, users not only have many combinations of instruments to choose from, but they can also change many settings like the tempo and the key of the generated music, which are both factors that can determine the mood of a piece of music. With all these combined, there is a wide possibility of different musical pieces that can be generated from the system.

## Chapter 7

# Conclusion and Recommendation

In this final chapter, a conclusion of the report will be made, as well as recommendations for future work and improvement.

### 7.1 Conclusion

In conclusion, algorithmic music in this context is a method of composing music through computerised systems by abiding to a set of rules and principles. The problem domains for this project is that music is time and cost consuming to compose, and it is a complex procedure that requires skill, talent and good understanding on music. Besides that, using the works of existing musical pieces has its limitations and drawbacks due to copyright and the time consumed to find an exact piece that suits the user's needs. The proposed solutions to overcome the problems are by developing a cost-free and time saving composition system, integrate a user-friendly and intuitive GUI along with easy-to-understand functionalities, and enable the system to compose original pieces of music. With that being said, the objectives of the project can be summarised as follows: (1) To develop a music composition application with real-time capabilities. (2) To simplify the music creation process. (3) To generate new and original melodies.

Additionally, in chapter two, a literature review was done on three existing methods for algorithmic composition, which are the stochastic, rule-based and AI methods. AI methodologies further branches into the use of GP, which enabled systems to generate both the composition material and ruleset. Each method has its own strengths and limitations, which could be resolved by using hybrid methods that incorporate more than one methodology. Besides that, a review was done on three of the existing systems, which were *Computoser*, *Ecret Music*, and *ArtSong*. Each system's strengths and limitations, and solutions for it were then evaluated based on their functionalities. Lastly, this chapter also includes the findings on music composition techniques, consisting of the circle of fifths and application of the golden ratio in music.



In chapter three, the approach that will be taken to develop the system is described using a system architecture diagram and use case diagram. The sole actor of the system is the “user”. Use cases of this system include generating music, generating chord, playing music, editing music and viewing the music library. By using the music library, users may also load and delete music when the file is selected from the table view, and generated pieces may be saved as MIDI files for future playback or to applied in scenarios which require the use of music. Activity diagrams are also drawn for each use case.

Furthermore, in chapter four, the system design was explained with the use of a flowchart to illustrate the process that will be taken for music generation. The genetic algorithm’s parameters and implementation details were explained, as well as the music parameters that would be provided by the users of the system. The evolution methods of sequences were also elaborated in this chapter. Besides that, for chord generation, the system uses a probability distribution obtained through a dataset of classical music. User requirements, system specifications and the user interface design are also presented in this chapter.

Based on chapter five, this project will utilise a RAD based methodology of phased development, and the technologies that will be involved in this project include a laptop for development, the Java programming language, as it has many advantages including the APIs that could be used along it, namely JavaFX and JFugue, which will be used for GUI development and music programming respectively. A project timeline for both Project 1 and Project 2 were also provided. Finally, the implementation outcome was described, and screenshots of the system were also included.

In chapter six, the system was evaluated using a verification plan. A plan was written for each use case to test out the functionalities of each of them and to ensure that the system is working as intended. After testing out the system, it had been confirmed that the system is not bugged. Similar to chapter five, screenshots of the system were also provided to show the output received. Challenges faced during the development of this project is that computer-generated music may not be as good as music created by humans, and that it is hard to define what “good” music is. Furthermore, the project is also restricted by time.

In short, the novelty of this project is to produce a simple algorithmic music composition application for novice composers with limited music knowledge. As proven from our test results, the system is able to successfully generate and plays music, therefore being able to save the composer's time by automating the whole composition and refinement process. Lastly, although the fitness function is currently able to let the genetic algorithm produce sequence of notes that sound harmonious together, there could still be adjustments to further improve the quality of the produced music.

## **7.2 Recommendation**

A final recommendation that could be made to improve the system is to define a fitness function that is more obviously defined and measurable. Different musical concepts and methods of music composition can be studied so that the fitness function may be improved. Besides that, a functionality for users to upload music files to be passed into the genetic algorithm and produce an output from them could also be implemented as enhancement.

## REFERENCES

- [1] Mid-America Piano LLC. (2012) The Four Elements of Music – Melody, Harmony, Rhythm, And Dynamics. [Online] Available: <https://pianonotes.piano4u.com/index.php/2012/07/the-four-elements-of-music-melody-harmony-rhythm-and-dynamics/>
- [2] Anara Publishing. (2021) The Importance of Music in Video Games. [Online] Available: <https://www.anarapublishing.com/the-importance-of-music-in-video-games/>
- [3] B. L. Jacob, “Algorithmic Composition as a Model of Creativity,” *Organised Sound*, vol. 1, no. 3, pp. 157–165, 1996. Available: <https://www.cambridge.org/core/journals/organised-sound/article/abs/algorithmic-composition-as-a-model-of-creativity/8C72F68D73DD7E57E1D6C5546CE68D0E>
- [4] K. Thywissen, “GeNotator: An Environment for Exploring the Application of Evolutionary Techniques in Computer-Assisted Composition,” *Organised Sound*, vol. 4, no. 2, pp. 127–133, 1999. Available: <https://ieeexplore.ieee.org/abstract/document/140338>
- [5] P. M. Gibson and J. A. Byrne, "NEUROGEN, Musical Composition Using Genetic Algorithms and Cooperating Neural Networks," *1991 Second International Conference on Artificial Neural Networks*, 1991, pp. 309-313.
- [6] A. Moroni, J. Manzolli, F. V. Zuben, and R. Gudwin., “Vox Populi: An Interactive Evolutionary System for Algorithmic Music Composition.” *Leonardo Music Journal*, vol. 10, 2000, pp. 49–54, 2000, Available: [https://www.researchgate.net/publication/240678026\\_Vox\\_Populi\\_An\\_Interactive\\_Evolutionary\\_System\\_for\\_Algorithmic\\_Music\\_Composition](https://www.researchgate.net/publication/240678026_Vox_Populi_An_Interactive_Evolutionary_System_for_Algorithmic_Music_Composition)
- [7] D. Matic (2010) A Genetic Algorithm for Composing Music. [Online] Available: <http://elib.mi.sanu.ac.rs/files/journals/yjor/39/yujorn39p157-177.pdf>

[8] Z. W. Geem, J.Y., (2007). Music Composition Using Harmony Search Algorithm. [Online] Available:

[https://www.academia.edu/462855/Music\\_composition\\_using\\_harmony\\_search\\_algorithm](https://www.academia.edu/462855/Music_composition_using_harmony_search_algorithm)

[9] John A. Maurer. (1999) A Brief History of Algorithmic Composition [Online] Available: <http://ccrma.stanford.edu/~blackrse/algorithm.html>

[10] B. A. Basseto, J. J. Neto, "A Stochastic Musical Composer Based on Adaptive Algorithms." *Anais do XIX Congresso Nacional da Sociedade Brasileira de Computação*. SBC-99, Vol. 3, pp.105-13, 1999. Available:

<http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.525.5681&rep=rep1&type=pdf>

[11] O. Sandred, , M. Laurson, M, Kuuskankare. "Revisiting the Illiac Suite – A Rule-Based Approach to Stochastic Processes" *Sonic Ideas/Ideas Sonicas*. 2. 42-46, 2004. Available:

[https://www.researchgate.net/publication/260791942\\_Revisiting\\_the\\_Illiac\\_Suite\\_-\\_A\\_rule-based\\_approach\\_to\\_stochastic\\_processes](https://www.researchgate.net/publication/260791942_Revisiting_the_Illiac_Suite_-_A_rule-based_approach_to_stochastic_processes)

[12] T. Anders. "Teaching Rule - Based Algorithmic Composition: The PWGL Library Cluster Rules" *Journal of Pedagogic Development* 6 (1), 2016, Available:

<https://uobrep.openrepository.com/handle/10547/603545>

[13] S, Yelkenci. (2017) Algorithmic Music Composition Using Linear Algebra [Online] Available: <https://pqdtopen.proquest.com/doc/1926772724.html?FMT=AI>

[14] S. Malgaonkar, Y. B. Nag, R. Devadiga and T. Hirave, "An AI based intelligent music composing algorithm: Concord," *2013 International Conference on Advances in Technology and Engineering (ICATE)*, 2013, pp. 1-6, Available:

[https://www.researchgate.net/publication/261488857\\_An\\_AI\\_based\\_intelligent\\_music\\_composing\\_algorithm\\_Concord](https://www.researchgate.net/publication/261488857_An_AI_based_intelligent_music_composing_algorithm_Concord)

[15] A. Mann, (2019) Phi: The Golden Ratio. [Online] Available:

<https://www.livescience.com/37704-phi-golden-ratio.html>

- [16] R. Gend, (2014) The Fibonacci Sequence and the Golden Ratio in Music. [Online] Available: <http://ww.nntdm.net/papers/nntdm-20/NNTDM-20-1-72-77.pdf>
- [17] Computoser. (n.d.) Listen to unique, computer-generated music... [Online] Available: <http://computoser.com/>
- [18] Ecret Music. (2018) Royalty Free Music for Creators [Online] Available: <https://ecrettmusic.com/>
- [19] ArtSong. (n.d.) ArtSong.org - Algorithmic Composition Software. [Online] Available: <https://www.artsong.org/>
- [20] University of Massachusetts Amherst with the Five College Consortium. (n.d.) Music 110 Fundamentals of Theory [Online] Available: <https://fundamentalsofmusictheory.umasscreate.net/unit-18/>
- [21] R, Bernstein. (2017) 5 System Development Life Cycle Phases. [Online] Available: <https://online.concordia.edu/computer-science/system-development-life-cycle-phases/>
- [22] JFugue (n.d.) The Complete Guide to JFugue. [Online] Available: <http://www.jfugue.org/4/jfbmrkklprpp/TheCompleteGuideToJFugue-v1.pdf>

# FINAL YEAR PROJECT WEEKLY REPORT

(Project II)

<b>Trimester, Year: Y3S3</b>	<b>Study week no.: 1</b>
<b>Student Name &amp; ID: ALISA YAP YI HUI 19ACB01195</b>	
<b>Supervisor: TS SOONG HOONG CHENG</b>	
<b>Project Title: REAL-TIME ALGORITHMIC MUSIC COMPOSITION APPLICATION</b>	

## 1. WORK DONE

[Please write the details of the work done in the last fortnight.]

Reviewed and recollected progress from FYP1.

## 2. WORK TO BE DONE

Continue coding the rest of the project. List out the work that has to be done. Load/Saving music, user input validation checking, music editor functionality, music library and etc.

## 3. PROBLEMS ENCOUNTERED

Have to recall If anything has been missed out from FYP1.

## 4. SELF EVALUATION OF THE PROGRESS

So far so good.

  
Online Signature during COVID-19  
SOONG HOONG CHENG (Lecturer)  
FACULTY OF ICT  
UNIVERSITI TUNKU ABDUL RAHMAN (UTAR)  
PERAK CAMPUS, JLN UNIVERSITI, BANDAR BARAT,  
31900 KAMPAR.

Supervisor's signature



Student's signature

# FINAL YEAR PROJECT WEEKLY REPORT

(Project II)

<b>Trimester, Year: Y3S3</b>	<b>Study week no.: 2</b>
<b>Student Name &amp; ID: ALISA YAP YI HUI 19ACB01195</b>	
<b>Supervisor: TS SOONG HOONG CHENG</b>	
<b>Project Title: REAL-TIME ALGORITHMIC MUSIC COMPOSITION APPLICATION</b>	

## 1. WORK DONE

[Please write the details of the work done in the last fortnight.]

Coded Load/Save function. Users can now load saved midi files from system and save the generated music into midi files.

## 2. WORK TO BE DONE

Implement music library for users to view their saved files within the application.

## 3. PROBLEMS ENCOUNTERED

The alert dialog is appearing twice when saved is clicked.

## 4. SELF EVALUATION OF THE PROGRESS

So far so good. Need some time debugging.

  
Online Signature during COVID-19  
SOONG HOONG CHENG (Lecturer)  
FACULTY OF ICT  
UNIVERSITI TUNKU ABDUL RAHMAN (UTAR)  
PERAK CAMPUS, JLN UNIVERSITI, BANDAR BARAT,  
31900 KAMPAR.

Supervisor's signature



Student's signature

# FINAL YEAR PROJECT WEEKLY REPORT

(Project II)

<b>Trimester, Year: Y3S3</b>	<b>Study week no.: 3</b>
<b>Student Name &amp; ID: ALISA YAP YI HUI 19ACB01195</b>	
<b>Supervisor: TS SOONG HOONG CHENG</b>	
<b>Project Title: REAL-TIME ALGORITHMIC MUSIC COMPOSITION APPLICATION</b>	

## 1. WORK DONE

[Please write the details of the work done in the last fortnight.]

Debug previous errors. Implemented the music library functionality to list out midi files in the application. Allow users to play or delete any selected file from the table display.

## 2. WORK TO BE DONE

Implement load to editor and music editor functionalities.

## 3. PROBLEMS ENCOUNTERED

None.

## 4. SELF EVALUATION OF THE PROGRESS

So far so good.

  
Online Signature during COVID-19  
SOONG HOONG CHENG (Lecturer)  
FACULTY OF ICT  
UNIVERSITI TUNKU ABDUL RAHMAN (UTAR)  
PERAK CAMPUS, JLN UNIVERSITI, BANDAR BARAT,  
31900 KAMPAR.

Supervisor's signature



Student's signature



# FINAL YEAR PROJECT WEEKLY REPORT

(Project II)

<b>Trimester, Year: Y3S3</b>	<b>Study week no.: 4</b>
<b>Student Name &amp; ID: ALISA YAP YI HUI 19ACB01195</b>	
<b>Supervisor: TS SOONG HOONG CHENG</b>	
<b>Project Title: REAL-TIME ALGORITHMIC MUSIC COMPOSITION APPLICATION</b>	

## 1. WORK DONE

[Please write the details of the work done in the last fortnight.]

Implemented the music editor. Users can load files into the editor and edit the instrument and tempo. They are able to reset the settings back to the original file and save the changes.

## 2. WORK TO BE DONE

Add more settings for users to enhance the generated music. Allow users to overwrite save files if the save file already exists.

## 3. PROBLEMS ENCOUNTERED

None.

## 4. SELF EVALUATION OF THE PROGRESS

Progress is faster than planned.

  
Online Signature during COVID-19  
SOONG HOONG CHENG (Lecturer)  
FACULTY OF ICT  
UNIVERSITI TUNKU ABDUL RAHMAN (UTAR)  
PERAK CAMPUS, JLN UNIVERSITI, BANDAR BARAT,  
31900 KAMPAR.

Supervisor's signature



Student's signature

# FINAL YEAR PROJECT WEEKLY REPORT

(Project II)

<b>Trimester, Year: Y3S3</b>	<b>Study week no.: 5</b>
<b>Student Name &amp; ID: ALISA YAP YI HUI 19ACB01195</b>	
<b>Supervisor: TS SOONG HOONG CHENG</b>	
<b>Project Title: REAL-TIME ALGORITHMIC MUSIC COMPOSITION APPLICATION</b>	

## 1. WORK DONE

[Please write the details of the work done in the last fortnight.]

Added slider for volume adjustment. Modified previously written code to accommodate the changes made.

## 2. WORK TO BE DONE

Add more settings for users to enhance the generated music. Add error alert message to inform user if a file they try to load or delete does not exist.

## 3. PROBLEMS ENCOUNTERED

Music generation code has to be modified for users to be able to modify the volume. Difficult to find information on JFugue online to solve the problem faced of volume being unable to be adjusted, so a lot of time was spent debugging.

## 4. SELF EVALUATION OF THE PROGRESS

So far so good.

  
Online Signature during COVID-19  
SOONG HOONG CHENG (Lecturer)  
FACULTY OF ICT  
UNIVERSITI TUNKU ABDUL RAHMAN (UTAR)  
PERAK CAMPUS, JLN UNIVERSITI, BANDAR BARAT,  
31900 KAMPAR.

Supervisor's signature



Student's signature

# FINAL YEAR PROJECT WEEKLY REPORT

(Project II)

<b>Trimester, Year: Y3S3</b>	<b>Study week no.: 6</b>
<b>Student Name &amp; ID: ALISA YAP YI HUI 19ACB01195</b>	
<b>Supervisor: TS SOONG HOONG CHENG</b>	
<b>Project Title: REAL-TIME ALGORITHMIC MUSIC COMPOSITION APPLICATION</b>	

## 1. WORK DONE

[Please write the details of the work done in the last fortnight.]

Added another instrument choice for users to select. Users can now choose up to two instruments to be played. Modified previous code to accommodate the changes.

## 2. WORK TO BE DONE

Add error alert message to inform user if a file they try to load or delete does not exist.

## 3. PROBLEMS ENCOUNTERED

None.

## 4. SELF EVALUATION OF THE PROGRESS

So far so good.

  
Online Signature during COVID-19  
SOONG HOONG CHENG (Lecturer)  
FACULTY OF ICT  
UNIVERSITI TUNKU ABDUL RAHMAN (UTAR)  
PERAK CAMPUS, JLN UNIVERSITI, BANDAR BARAT,  
31900 KAMPAR.

Supervisor's signature



Student's signature

# FINAL YEAR PROJECT WEEKLY REPORT

(Project II)

<b>Trimester, Year: Y3S3</b>	<b>Study week no.: 7</b>
<b>Student Name &amp; ID: ALISA YAP YI HUI 19ACB01195</b>	
<b>Supervisor: TS SOONG HOONG CHENG</b>	
<b>Project Title: REAL-TIME ALGORITHMIC MUSIC COMPOSITION APPLICATION</b>	

## 1. WORK DONE

[Please write the details of the work done in the last fortnight.]

Added error alert message to inform user if a file they try to load or delete does not exist. Check if user input is correct for other functionalities including play and save music.

## 2. WORK TO BE DONE

Allow users to randomly generate music instead of always providing input. Start to prepare FYP2 report.

## 3. PROBLEMS ENCOUNTERED

None.

## 4. SELF EVALUATION OF THE PROGRESS

So far so good.

  
Online Signature during COVID-19  
SOONG HOONG CHENG (Lecturer)  
FACULTY OF ICT  
UNIVERSITI TUNKU ABDUL RAHMAN (UTAR)  
PERAK CAMPUS, JLN UNIVERSITI, BANDAR BARAT,  
31900 KAMPAR.

Supervisor's signature



Student's signature

# FINAL YEAR PROJECT WEEKLY REPORT

(Project II)

<b>Trimester, Year: Y3S3</b>	<b>Study week no.: 8</b>
<b>Student Name &amp; ID: ALISA YAP YI HUI 19ACB01195</b>	
<b>Supervisor: TS SOONG HOONG CHENG</b>	
<b>Project Title: REAL-TIME ALGORITHMIC MUSIC COMPOSITION APPLICATION</b>	

## 1. WORK DONE

[Please write the details of the work done in the last fortnight.]

Added error alert message to inform user if a file they try to load or delete does not exist. Check if user input is correct for other functionalities including play and save music.

## 2. WORK TO BE DONE

Allow users to randomly generate music instead of always providing input. Start to prepare report.

## 3. PROBLEMS ENCOUNTERED

None.

## 4. SELF EVALUATION OF THE PROGRESS

So far so good.

  
Online Signature during COVID-19  
SOONG HOONG CHENG (Lecturer)  
FACULTY OF ICT  
UNIVERSITI TUNKU ABDUL RAHMAN (UTAR)  
PERAK CAMPUS, JLN UNIVERSITI, BANDAR BARAT,  
31900 KAMPAR.

Supervisor's signature



Student's signature

# FINAL YEAR PROJECT WEEKLY REPORT

(Project II)

<b>Trimester, Year: Y3S3</b>	<b>Study week no.: 9</b>
<b>Student Name &amp; ID: ALISA YAP YI HUI 19ACB01195</b>	
<b>Supervisor: TS SOONG HOONG CHENG</b>	
<b>Project Title: REAL-TIME ALGORITHMIC MUSIC COMPOSITION APPLICATION</b>	

## 1. WORK DONE

[Please write the details of the work done in the last fortnight.]

Added random function to randomly set input values for generating music. Prepared FYP2 report template, migrate FYP1 content to the report.

## 2. WORK TO BE DONE

Perform testing to ensure that the system is working as intended. Continue to write the report.

## 3. PROBLEMS ENCOUNTERED

None.

## 4. SELF EVALUATION OF THE PROGRESS

So far so good.

  
Online Signature during COVID-19  
SOONG HOONG CHENG (Lecturer)  
FACULTY OF ICT  
UNIVERSITI TUNKU ABDUL RAHMAN (UTAR)  
PERAK CAMPUS, JLN UNIVERSITI, BANDAR BARAT,  
31900 KAMPAR.

Supervisor's signature



Student's signature

# FINAL YEAR PROJECT WEEKLY REPORT

(Project II)

<b>Trimester, Year: Y3S3</b>	<b>Study week no.: 10</b>
<b>Student Name &amp; ID: ALISA YAP YI HUI 19ACB01195</b>	
<b>Supervisor: TS SOONG HOONG CHENG</b>	
<b>Project Title: REAL-TIME ALGORITHMIC MUSIC COMPOSITION APPLICATION</b>	

## 1. WORK DONE

[Please write the details of the work done in the last fortnight.]

Tested some of the system's functionalities and debugged some minor errors. Wrote the report up to chapter 2 and the passed test cases.

## 2. WORK TO BE DONE

Continue to test and debug the system, continue writing the report.

## 3. PROBLEMS ENCOUNTERED

None.

## 4. SELF EVALUATION OF THE PROGRESS

So far so good.

  
Online Signature during COVID-19  
SOONG HOONG CHENG (Lecturer)  
FACULTY OF ICT  
UNIVERSITI TUNKU ABDUL RAHMAN (UTAR)  
PERAK CAMPUS, JLN UNIVERSITI, BANDAR BARAT,  
31900 KAMPAR.

Supervisor's signature



Student's signature

# FINAL YEAR PROJECT WEEKLY REPORT

(Project II)

<b>Trimester, Year: Y3S3</b>	<b>Study week no.: 11</b>
<b>Student Name &amp; ID: ALISA YAP YI HUI 19ACB01195</b>	
<b>Supervisor: TS SOONG HOONG CHENG</b>	
<b>Project Title: REAL-TIME ALGORITHMIC MUSIC COMPOSITION APPLICATION</b>	

## 1. WORK DONE

[Please write the details of the work done in the last fortnight.]

Wrote the report up to chapter 4 and chapter 6 which is about the testing performed the last two weeks.

## 2. WORK TO BE DONE

Continue to write the report. Take screenshots of the system as example for chapter 5.

## 3. PROBLEMS ENCOUNTERED

None.

## 4. SELF EVALUATION OF THE PROGRESS

So far so good.

  
Online Signature during COVID-19  
SOONG HOONG CHENG (Lecturer)  
FACULTY OF ICT  
UNIVERSITI TUNKU ABDUL RAHMAN (UTAR)  
PERAK CAMPUS, JLN UNIVERSITI, BANDAR BARAT,  
31900 KAMPAR.

Supervisor's signature



Student's signature



# FINAL YEAR PROJECT WEEKLY REPORT

(Project II)

<b>Trimester, Year: Y3S3</b>	<b>Study week no.: 12</b>
<b>Student Name &amp; ID: ALISA YAP YI HUI 19ACB01195</b>	
<b>Supervisor: TS SOONG HOONG CHENG</b>	
<b>Project Title: REAL-TIME ALGORITHMIC MUSIC COMPOSITION APPLICATION</b>	

## 1. WORK DONE

[Please write the details of the work done in the last fortnight.]

Added contents of chapter 5, completed chapter 6.

## 2. WORK TO BE DONE

Check if report is complete, write the conclusion and check formatting.

## 3. PROBLEMS ENCOUNTERED

None.

## 4. SELF EVALUATION OF THE PROGRESS

So far so good.

  
Online Signature during COVID-19  
SOONG HOONG CHENG (Lecturer)  
FACULTY OF ICT  
UNIVERSITI TUNKU ABDUL RAHMAN (UTAR)  
PERAK CAMPUS, JLN UNIVERSITI, BANDAR BARAT,  
31000 KAMPAR.

\_\_\_\_\_  
Supervisor's signature



\_\_\_\_\_  
Student's signature

# FINAL YEAR PROJECT WEEKLY REPORT

(Project II)

<b>Trimester, Year: Y3S3</b>	<b>Study week no.: 13</b>
<b>Student Name &amp; ID: ALISA YAP YI HUI 19ACB01195</b>	
<b>Supervisor: TS SOONG HOONG CHENG</b>	
<b>Project Title: REAL-TIME ALGORITHMIC MUSIC COMPOSITION APPLICATION</b>	

## 1. WORK DONE

[Please write the details of the work done in the last fortnight.]

Finalised the report.

## 2. WORK TO BE DONE

Submit the report to FYP portal.

## 3. PROBLEMS ENCOUNTERED

None.

## 4. SELF EVALUATION OF THE PROGRESS

No problems faced.

  
Online Signature during COVID-19  
SOONG HOONG CHENG (Lecturer)  
FACULTY OF ICT  
UNIVERSITI TUNKU ABDUL RAHMAN (UTAR)  
PERAK CAMPUS, JLN UNIVERSITI, BANDAR BARAT,  
31000 KAMPAR.

Supervisor's signature



Student's signature

# APPENDICES

## POSTER

**UTAR**  
UNIVERSITI TUNKU ABDUL RAHMAN

Faculty of Information and Communication Technology  
**REAL-TIME ALGORITHMIC MUSIC  
COMPOSITION APPLICATION**

### II: INTRODUCTION

This project is about the study of evolutionary music, and focuses on the development of an algorithmic music composer using the Java. Users will be able to generate random pieces of music with minimal input required.

---

### ♪ METHODOLOGIES

Generates music based on a genetic algorithm.

In addition, the system also utilises JavaFx and jFugue for its graphical user interface and music programming respectively.

Due to time constraints, the development of this system uses a form of RAD development model, namely the phased development model.

---

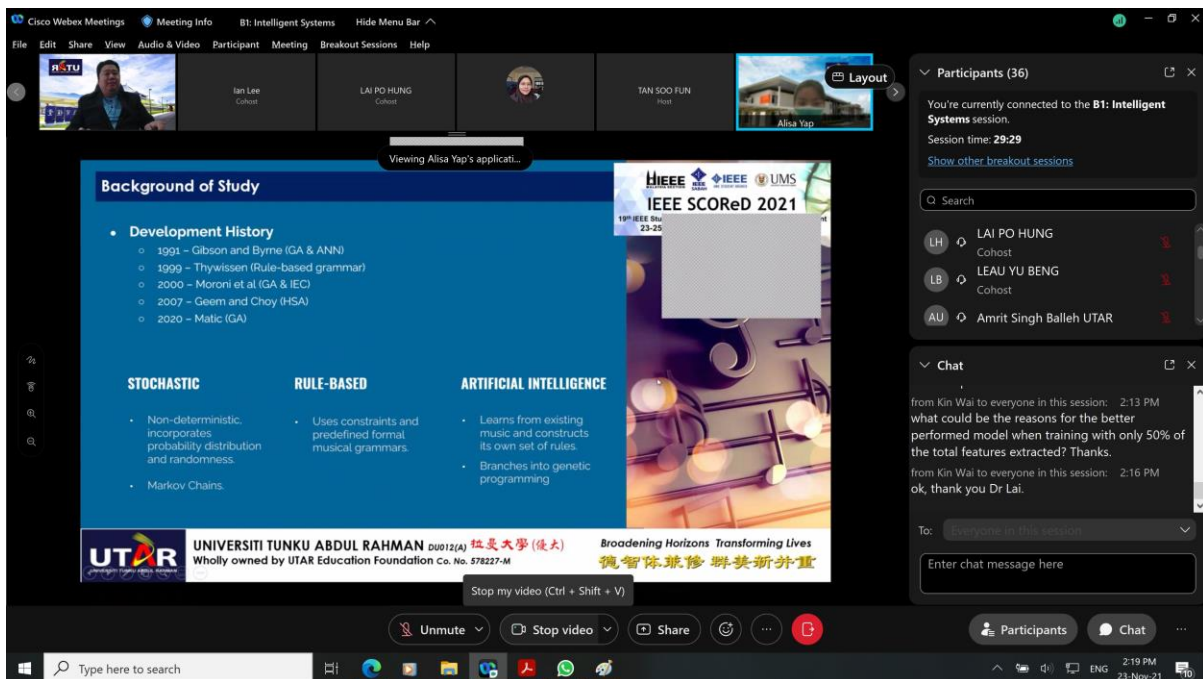
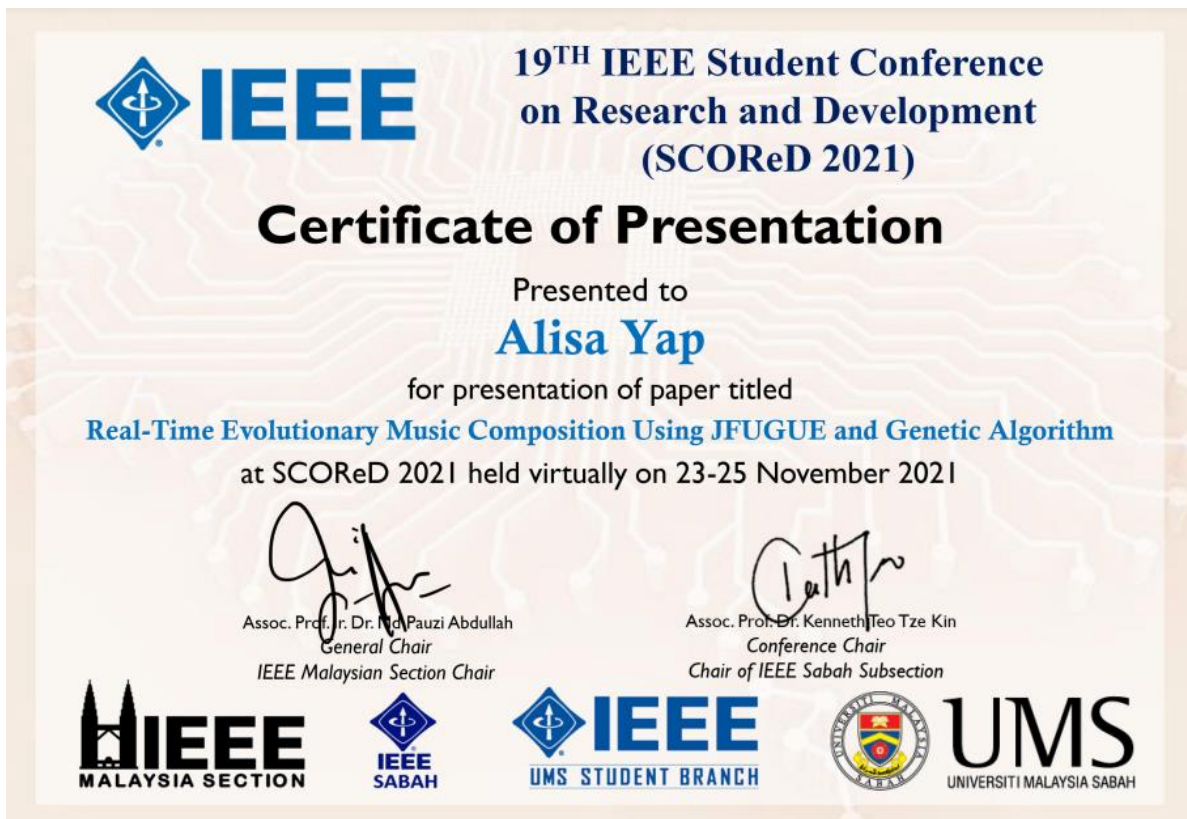
<h3>♪ DISCUSSION</h3> <p>Music produced through this system are relatively simple, and is suited to be applied in personal projects which require such audio accompaniment.</p> <p>Deliverable of this project – a desktop application for users to compose and generate music in real-time with minimal effort.</p>	<h3>CONCLUSION</h3> <p>System successfully produces and plays music based on the input provided by the users.</p> <p>However, there is still notable room for improvement on the quality of music generated by the system, which could be further improved during final year project 2.</p>
--	---

---

Bachelor of Computer Science (Hons)  
Final Year Project

Project by: Alisa Yap Yi Hui  
Project Supervisor: Ts Soong Hoong Cheng

## PAPER PUBLISHED



## PLAGARISM CHECK RESULT

19ACB01195\_FYP2

### ORIGINALITY REPORT

<b>3%</b> SIMILARITY INDEX	<b>2%</b> INTERNET SOURCES	<b>0%</b> PUBLICATIONS	<b>1%</b> STUDENT PAPERS
-------------------------------	-------------------------------	---------------------------	-----------------------------

### PRIMARY SOURCES

<b>1</b>	<b>fict.utar.edu.my</b> Internet Source	<1%
<b>2</b>	<b>Submitted to Universiti Teknologi MARA</b> Student Paper	<1%
<b>3</b>	<b>Submitted to The Robert Gordon University</b> Student Paper	<1%
<b>4</b>	<b>Submitted to University of Nottingham</b> Student Paper	<1%
<b>5</b>	<b>en.wikipedia.org</b> Internet Source	<1%
<b>6</b>	<b>epdf.pub</b> Internet Source	<1%
<b>7</b>	<b>Submitted to University of Northumbria at Newcastle</b> Student Paper	<1%
<b>8</b>	<b>Submitted to Institute of Research &amp; Postgraduate Studies, Universiti Kuala Lumpur</b> Student Paper	<1%

9	<a href="http://ir.lib.uth.gr">ir.lib.uth.gr</a> Internet Source	<1 %
10	Submitted to The University of Manchester Student Paper	<1 %
11	Submitted to University of Leeds Student Paper	<1 %
12	<a href="http://2eac3a3b-5e23-43c7-b33c-f17ad8fd3011.filesusr.com">2eac3a3b-5e23-43c7-b33c-f17ad8fd3011.filesusr.com</a> Internet Source	<1 %
13	Nakano, Reiichiro Christian S., Argel Bandala, Gerard Ely Faelden, Jose Martin Maningo, and Elmer P. Dadios. "A genetic algorithm approach to swarm centroid tracking in quadrotor unmanned aerial vehicles", 2014 International Conference on Humanoid Nanotechnology Information Technology Communication and Control Environment and Management (HNICEM), 2014. Publication	<1 %
14	<a href="http://netapps.internetworks.my">netapps.internetworks.my</a> Internet Source	<1 %
15	<a href="http://dspace.daffodilvarsity.edu.bd:8080">dspace.daffodilvarsity.edu.bd:8080</a> Internet Source	<1 %
16	<a href="http://www.studymode.com">www.studymode.com</a> Internet Source	<1 %

17	Du, H.. "Constrained H <sup>2</sup> approximation of multiple input-output delay systems using genetic algorithm", ISA Transactions, 200704 Publication	<1 %
18	Jonathan Bendor, Daniel Diermeier, David A. Siegel, Michael Ting. "A Behavioral Theory of Elections", Walter de Gruyter GmbH, 2011 Publication	<1 %
19	docplayer.net Internet Source	<1 %
20	docshare.tips Internet Source	<1 %
21	hdl.handle.net Internet Source	<1 %
22	uir.unisa.ac.za Internet Source	<1 %

Exclude quotes  On  
Exclude bibliography  On

Exclude matches  < 5 words

<b>Universiti Tunku Abdul Rahman</b>			
<b>Form Title : Supervisor's Comments on Originality Report Generated by Turnitin for Submission of Final Year Project Report (for Undergraduate Programmes)</b>			
Form Number: FM-IAD-005	Rev No.: 0	Effective Date: 01/10/2013	Page No.: 1 of 1




**FACULTY OF INFORMATION AND COMMUNICATION TECHNOLOGY**

<b>Full Name(s) of Candidate(s)</b>	Alisa Yap Yi Hui
<b>ID Number(s)</b>	19ACB01195
<b>Programme / Course</b>	CS
<b>Title of Final Year Project</b>	Real-Time Algorithmic Music Composition Application

<b>Similarity</b>	<b>Supervisor's Comments (Compulsory if parameters of originality exceeds the limits approved by UTAR)</b>
<b>Overall similarity index: <u>3</u> %</b>  <b>Similarity by source</b> Internet Sources: <u>2</u> % Publications: <u>0</u> % Student Papers: <u>1</u> %	
<b>Number of individual sources listed of more than 3% similarity: <u>0</u></b>	
<b>Parameters of originality required and limits approved by UTAR are as Follows:</b> (i) Overall similarity index is 20% and below, and (ii) Matching of individual sources listed must be less than 3% each, and (iii) Matching texts in continuous block must not exceed 8 words <i>Note: Parameters (i) – (ii) shall exclude quotes, bibliography and text matches which are less than 8 words.</i>	

Note Supervisor/Candidate(s) is/are required to provide softcopy of full set of the originality report to Faculty/Institute

***Based on the above results, I hereby declare that I am satisfied with the originality of the Final Year Project Report submitted by my student(s) as named above.***

  
Online Signature during COVID-19  
**SOONG HOONG CHENG (Lecturer)**  
FACULTY OF ICT  
UNIVERSITI TUNKU ABDUL RAHMAN (UTAR)  
PERAK CAMPUS, JLN UNIVERSITI, BANDAR BARAT,  
31900 KAMPAR.

\_\_\_\_\_  
Signature of Supervisor

Name: Ts Soong Hoong Cheng

Date: 22 April 2022

\_\_\_\_\_  
Signature of Co-Supervisor

Name: \_\_\_\_\_

Date: \_\_\_\_\_





**UNIVERSITI TUNKU ABDUL RAHMAN**

**FACULTY OF INFORMATION & COMMUNICATION TECHNOLOGY  
(KAMPAR CAMPUS)**

**CHECKLIST FOR FYP2 THESIS SUBMISSION**

Student Id	19ACB01195
Student Name	ALISA YAP YI HUI
Supervisor Name	TS SOONG HOONG CHENG

TICK (√)	DOCUMENT ITEMS
	Your report must include all the items below. Put a tick on the left column after you have checked your report with respect to the corresponding item.
	Front Plastic Cover (for hardcopy)
√	Title Page
√	Signed Report Status Declaration Form
√	Signed FYP Thesis Submission Form
√	Signed form of the Declaration of Originality
√	Acknowledgement
√	Abstract
√	Table of Contents
√	List of Figures (if applicable)
√	List of Tables (if applicable)
√	List of Symbols (if applicable)
√	List of Abbreviations (if applicable)
√	Chapters / Content
√	Bibliography (or References)
√	All references in bibliography are cited in the thesis, especially in the chapter of literature review
√	Appendices (if applicable)
√	Weekly Log
√	Poster
√	Signed Turnitin Report (Plagiarism Check Result - Form Number: FM-IAD-005)
√	I agree 5 marks will be deducted due to incorrect format, declare wrongly the ticked of these items, and/or any dispute happening for these items in this report.

\*Include this form (checklist) in the thesis (Bind together as the last page)

I, the author, have checked and confirmed all the items listed in the table are included in my report.

(Signature of Student)

Date: 22 April 2022