

2018-10-20

System Analysis and Design Report

Team 3

Melody Maker

# SYSTEM ANALYSIS AND DESIGN REPORT

1. *The System Analysis and Design (“SA&D”) phase signifies the commencement of system implementation. The objectives of this phase are:*

*i) to investigate and understand the user and technical requirements; ii) to specify and design the new system; and iii) to detail the implementation requirements in terms of cost, effort and time.*

1. *The SA&D Report will be produced to document the findings and recommendations of this phase.*
2. *A sample template of the SA&D Report with sample content is provided in the following pages.*
3. *Notes for using the template are written in “*italic*” text enclosed in pointed brackets “< >”, while sample contents are written in “***bold italic***” and can be replaced by project-specific information or removed to suit specific project needs. After all changes are made, all notes should be removed and font of all “*italic*” text should be changed to black.*

Contents

[GUIDE TEMPLATE 1](#_Toc513841934)

[*1*  *SYSTEM ANALYSIS AND DESIGN REPORT* 2](#_Toc513841935)

[1 MANAGEMENT SUMMARY (NEWLY ADDED SECTION) 1](#_Toc513841936)

[1.1 APPROVAL SOUGHT 1](#_Toc513841937)

[*Current Environment Description* 6](#_Toc513841938)

[*Current system description* 6](#_Toc513841939)

[*Current System Overview* 6](#_Toc513841940)

[*High level system overview* 6](#_Toc513841941)

[*Current System Hardware, Software and Network* 7](#_Toc513841942)

[*Minimum Cubase requirements* 7](#_Toc513841943)

[*CURRENT PROBLEMS AND ISSUES IN COMPOSING MUSIC* 7](#_Toc513841944)

[*CURRENT PROBLEMS AND ISSUES IN DEVELOPMENT* 7](#_Toc513841945)

[**SYSTEM SPECIFICATION** 17](#_Toc513841946)

[FUNCTIONAL SPECIFICATION 17](#_Toc513841947)

[Required System Overview 17](#_Toc513841948)

[Application Architecture 18](#_Toc513841949)

[Security 18](#_Toc513841950)

[SYSTEM DESIGN 18](#_Toc513841951)

[Common Frameworks 18](#_Toc513841952)

[Design 19](#_Toc513841953)

[User Experience Design 19](#_Toc513841954)

[ Application Data Model 19](#_Toc513841955)

[4.3.2 Data Model 25](#_Toc513841956)

# MANAGEMENT SUMMARY (NEWLY ADDED SECTION)

## 1.1 APPROVAL SOUGHT

This SAD document aims to seek approval from Belgium Campus to proceed to the next stage of system implementation and Integration. We as a team believe that our project has massive potential value for the field of music.

## 1.2 SYSTEM OBJECTIVES

Our main objectives are:

* Create a test and training dataset to use as input for the teaching of our artificial neural network.
* Create a database that stores paths/storage locations of different music genres of music in .midi format.
* Create a melody focused artificial neural network that uses the database as input for its unsupervised learning.
* Create a new way of producing musical melodies.

## 1.3 BACKGROUND

Reasons for Project

* This project does not serve any business-related reason for being, it is purely an attempt to create new material for the musical field and spur on the imagination of composers and musicians worldwide. We might find a new way of looking at the creation of a melody, as our project aims to develop melodies that are pleasing to the ear. E.g. to focus on using strong semitones between notes of a melody.

##### 1.3.1 Current System Description

Currently the only way of composing music is by imagination.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **User** |  | **Channel** |  | **Imagination** |  | **External System** |

**Using imagination to create music**



Find a combination of notes that sound pleasing



Add bassline + drumbeat



Record song or compose song in a music library like Cubase

Music Composer

Record Label

Give song to public

Figure 1.3.1

## PROPOSED SYSTEM

##### 1.4.1 System Overview

The proposed IT system will enable users to generate unique music after selecting parameters such as genre.

Diagram**:**

Input .midi + parameter

Convert to .csv

Add to database

Train ANN on ..csv file data

Compose melody

Reads parameters



User

Figure 1.4.1

##### 1.4.2 System Functions

List of Future Business Processes:

|  |  |
| --- | --- |
| **Process ID** | **Business Process Title** |
| **BP-001** | **Input .csv files - .midi musical pieces converted to .csv (train the ANN)** |
| **BP-002** | **Specify genre** |
| **BP-003** | **Train model on genre data** |
| **BP-003** | **Record output** |

Table 1.4.2

##### 1.4.3 Technical System Architecture

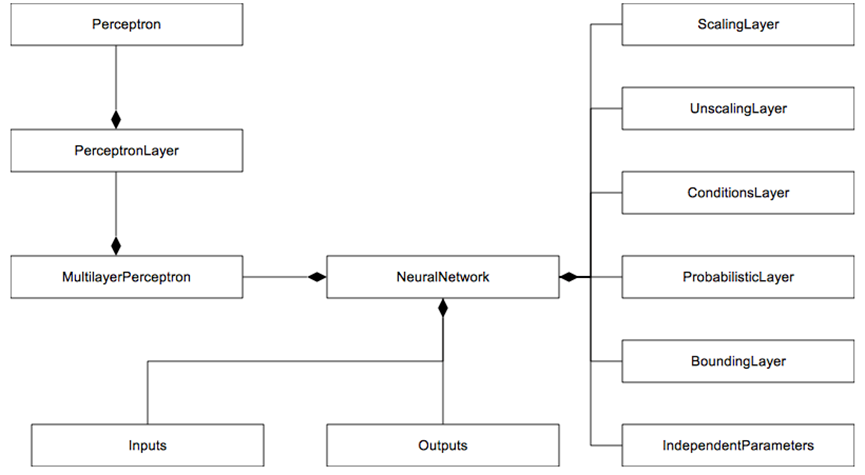
****

Figure 1.4.3

## 1.5 RESOURCE IMPLICATIONS

This is a detailed breakdown of our system developmental process. Divided into roles for easy management

**Detailed breakdown for System Implementation & Integration (SI&I) services in man-days:**

Table 1.5

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ***Stage*** | ***Program***  ***Manager*** | ***Data Analysts*** | ***Database***  Admin | ***Programmer*** |
| **Stage 2 - Physical Design and Program Development** | 90 | **20** | **80** | **120** |
| **Stage 3 - Acceptance Testing** | **10** | **10** | **30** | **30** |
| **Stage 4 - Data Conversion** | **10** | **120** | **30** | **20** |
| **Stage 5 - Documentation and Rollout** | **20** | **20** | **30** | **5** |
| **Stage 6 - Updates** | **5** | **0** | **10** | **5** |
| **Total** | **140** | **170** | **180** | **180** |

## 1.6 COSTS AND BENEFITS

|  |
| --- |
| ***Benefits***   * No real cost except for the Udemy course fee * No Annual Cost |
| ***Development Cost: R2400***   * Udemy Courses: R2400.00 |
| ***Operational Cost: R0.00***   * Hardware: R0.00 * Software: R0.00 * Operational: R0.0 |
| ***Total Cost: R2400.00*** |

Table 1.6

##### 1.6.1 Costs

There is no cost in creating a neural network nor storing it on a Github repository, thus after development no more capital in needed.

|  |  |
| --- | --- |
| ***Non-Recurrent Cost*** | ***Cost (R)*** |
| **Hardware & Software** | **2400.00** |
| **Service implementation** | **0 .00** |
| **IT-Staff** | **0 .00** |
| **Total** | **2400 .00** |

Table 1.6.1

##### 1.6.2 Benefits

Our project holds value for the field of music in a way that is unmeasurable. We seek not profit, but new ways of looking at music and a better understanding of why certain combinations of notes sounds more pleasing to the ear than others.

## 1.7 IMPLEMENTATION PLAN SUMMARY

Implementation Strategy

1. Prepare the infrastructure.
2. Coordinate with the organizations involved in implementation.
3. Install the production solution.
4. Convert the data
5. Perform final verification in production
6. Implement new processes and procedures.
7. Monitor the solution.

Implementation Schedule

|  |  |  |
| --- | --- | --- |
| **Deliverable** | **Description** | **Planned Date** |
| Milestone 1  Final project proposal document submission | This includes the final proposal that will be assessed before commencement of the project. It is the first item of the development process that will be used to guide all other items on a basic level. | 2018-03-21 |
| Milestone 2  Planning document submission | Much like the proposal, the planning document will be used to guide the development of all other items, however, it will be used on a more in-depth scale. It provides guides and steps in a predefined manner that allows the configuration of all items to be executed as harmoniously as possible. | 2018-04-13 |
| Milestone 3  System Analysis and Design document submission | Here we identify, break down and assess all major and minor aspects of the development of the system, the environment, effects and the functionality of the system itself. | 2018-05-04 |
| Milestone 4  Technical 1: Database Design | This is technical aspect of the system. The database is one of the most important physical components of the system. Once this has been correctly developed, it can also be used for Class creation which is the next item. | 2018-05-25 |
| Milestone 5  Technical 2: Class Design | This item can be based on the elements found in the database design. It is a technical item that defines the basis of code development of the system. | 2018-06-14 |
| Milestone 6  Technical 3 : Object Behaviour Model | This technical item works with thee classes and database as a baseline. However, it is also the baseline for coding the system. It will be assessed and defined before coding commences for the purpose of simplifying the intense complicated nature of coding. | 2018-07-04 |
| Milestone 7  Technical 4 : Coding | This is the item that configures all previous items. It is the physical compilation of all planning up until this point. The system is physically created here. | 2018-08-04 |
| Test plan document submission | Here we will identify our approach to testing the application. | 2018-09-14 |
| Milestone 8  Testing | This is where we will implement our testing strategy that we have identified in the test plan document for our application. | 2018-10-22 |
| Milestone 9  Implementation | This is where the implementation process of our application will be executed. | 2018-10-29 |
| Project Submission | The project is in final form and must be ready to be summited on the given date. | 2018-11-01 |

# 2. Current Environment Description

Currently the only way of composing music is by imagination.

## 2.1 Current System Overview

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **User** |  | **Channel** |  | **Imagination** |  | **External System** |

**Using imagination to compose music**



Find a combination of notes that sound pleasing



Add bassline + drumbeat



Record song or compose song in a music library like Cubase

Music Composer

Record label

Give song to public

Figure 2.1

## 2.2High level system overview

|  |  |
| --- | --- |
| **Subsystem** | **Function** |
| **<List the subsystems>** | **<List the functions>** |
| **Using imagination to compose music** | * Find a pleasing combination of notes. * Add bassline + drumbeat * Record or compose song in a music library such as Cubase |

Table 2.2

## 2.3 Current System Hardware, Software and Network

## Minimum Cubase requirements

|  |
| --- |
| Windows 7, 8, 10 |
| 64-Bit Windows 7 / 8.x / 10 |
| 64-bit Intel or AMD multi-core processor (Intel i5 or faster recommended) |
| 4 GB RAM (8 GB or more recommended) |
| 18 GB free HD space |
| 1366 x 768 display resolution (1920 x 1080 recommended) |
| Graphics card with DirectX 10 and WDDM 1.1 support (Windows only) |
| USB port for USB-eLicenser (license management) |
| OS-compatible audio hardware\* |
| An internet connection is required for activation, account setup and personal/product registration. Additional downloads may be required for the installation. |

*No other software is required to compose music. An instrument or two might help.*

## 2.1 CURRENT PROBLEMS AND ISSUES IN COMPOSING MUSIC

* *Composing good melodic music takes time and experience.*
* *Even composers can run out of ideas for new melodies.*

|  |  |
| --- | --- |
| No. | Description |
| <Unique number> | <Description> |
| 1 | Sorting our .csv files into a database. |
| 2 | Converting .midi to .csv without doing it manually. |
| 3 | Visualizing the optimal output via MathLib – a python addon |

Table 2.1

# 3 REQUIREMENTS SPECIFICATION

## 3.1 USER REQUIREMENTS DOCUMENT

##### 3.1.1 Proposed System Overview

The purpose of the proposed system is to create an *artificial* neural network that we can train to recognize/analyse sheet music – Looking at the types of notes used in what way and the pauses between notes. The neuralnet can then generate its own unique music. It could be capable of producing music indistinguishable from a human composer.

###### Description of Proposed **Neural Network**

Input .midi + parameter

Convert to .csv

Add to database

Train ANN on ..csv file data

Compose melody

Reads parameters



User

Figure 3.1.1.1

###### System User Profile

System User Profile:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **No.** | **User Role** | **Responsibilities** | **Branch/**  **Division/**  **Section/ Unit** | **Staff**  **Post/Rank** | **Stakeholder Group** |
| **1** | **User(Musician)** | **Output Specifications** | **Production** | **Musician** | **Creation/Patent Group** |

Table 3.1.1.2

##### 3.1.2 Future Business Process

###### 3.1.2.1 List of Future Business Processes

List of Future Business Processes:

|  |  |
| --- | --- |
| **Process ID** | **Business Process Title** |
| **BP-001** | **Input .csv files - .midi musical pieces converted to .csv (train the ANN)** |
| **BP-002** | **Specify genre** |
| **BP-003** | **Train model on genre data** |
| **BP-003** | **Record output** |

Table 3.1.2.1

###### Business process

**Business process**

User

System

Start

Open system

Pick genre

Creates melody

(

processing

End

Generate .csv output

Give input .midi

Train Model

-

(

Is not

done

)

(

Is done

-

)

Reads parameters – Pick a key

Figure 3.1.2.2

Narratives:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Task No.** | **Actor** | **Task Name and Description** | **Input** | **Output** |
| 1 | User | Input the sheet music into the system to teach it and let it identify notes and spaces | Sheet music | data |
| 2 | system | Reads notes, identifies them and records the data | specifications | Learned information |
| 3 | user | Input specifications on what the output melody should sound like | specifications | Learned information |
| 4 | system | Creation of melody based on previously learned notes and rhythms | Learned information | Melody |

Table 3.1.2.2

Other information:

|  |
| --- |
| **Glossary N/A** |
| **References N/A** |
| **Assumptions:** |
| 1. It is assumed all input sheet music has been patented by someone else. 2. It is assumed that the User does not want to recreate the exact same sound as any of the inputs 3. The sheet music inputs are without any errors or musical mistakes or ‘no-go’s’ |
| 1. **Business Rules** |
| 1. The user will be the owner of the output melody 2. The system should not output the same melody as an input due to ownership. |

##### 3.1.3 Functional Requirements

###### 3.1.3.1 List of Functional Requirements

Priority Key:

* MUST (M),
* SHOULD (S),

|  |  |  |  |
| --- | --- | --- | --- |
| **Req. ID** | **Requirement Title** | **Target Users** | **Priority** |
| REQ-SYS-001 | Music note identification | System | M |
| REQ-SYS-002 | Note spaces measurement | System | M |
| REQ-BOR-001 | Visual Sheet | User | C |
| REQ-BOR-002 | Music output | User | M |
| REQ-BOR-003 | Melody description | User | C |

* COULD (C),
* WON’T (W)

List of Functional Requirements:

Table 3.1.3.1

###### 3.1.3.2 REQ-SYS-001 Music note identification

|  |  |
| --- | --- |
| **Item** | **Description** |
| Requirement ID | REQ-SYS-001 |
| Requirement Title | Music note identification |
| Priority | Must |
| Functional Requirement Description | * The system must be able to identify a note on sheet in order to know what it sounds like so that it can be recorded |
| Frequency of Use | During every music sheet input |
| Acceptance Criteria | Only predefined notes are to be recognised |
| Related Business Process | Refer to BP-001. |

Table 3.1.3.2

###### 3.1.3.3 REQ-BOR-001 Note spaces measurement

|  |  |
| --- | --- |
| **Item** | **Description** |
| Requirement ID | REQ-SYS-002 |
| Requirement Title | Note spaces measurement |
| Priority | Must |
| Functional Requirement Description | * The system must be able to identify the spaces between and the position of a note on a sheet in order to know what pitch it sounds like so that it can be recorded |
| Frequency of Use | Every sheet music input |
| Acceptance Criteria | Follows the same number of lines as a standard sheet |
| Related Business Process | Refer to BP-001. |

Requirements Description:

Table 3.1.3.3

###### 3.1.3.4 REQ-BOR-002 Visual Sheet

Requirements Description:

|  |  |
| --- | --- |
| **Item** | **Description** |
| Requirement ID | REQ-SYS-003 |
| Requirement Title | Visual Sheet |
| Priority | Could |
| Functional Requirement Description | * The system can show the sheet visually after it is inputted in order to confirm its accuracy |
| Frequency of Use | After every music sheet input |
| Acceptance Criteria | It should be identical to the input |
| Related Business Process | Refer to BP-001. |

Table 3.1.3.4

###### 3.1.3.4 REQ-SYS-004 Music output

Requirements Description:

|  |  |
| --- | --- |
| **Item** | **Description** |
| Requirement ID | REQ-SYS-004 |
| Requirement Title | Music output |
| Priority | Must |
| Functional Requirement Description | * The system must be able to output a melody based on all input notes tht it has learned from |
| Frequency of Use | User defined |
| Acceptance Criteria | It must sound elegant, based on predefined criteria of a ‘catchy song’ |
| Related Business Process | Refer to BP-002. |

###### 3.1.3.5 REQ-SYS-005 Melody description

Requirements Description:

|  |  |
| --- | --- |
| **Item** | **Description** |
| Requirement ID | REQ-SYS-005 |
| Requirement Title | Melody Description |
| Priority | Could |
| Functional Requirement Description | * The system can provide a description of the output, highlighting how it was created, what tempo or genre it is, etc. |
| Frequency of Use | After every output |
| Acceptance Criteria | All details of the melody should be shown |
| Related Business Process | Refer to BP-002. |

##### 3.1.4 Non-functional Requirements

Categories:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Req. ID** | **Category** | **Requirement Title** | **Target Users** | **Priority** |
| REQ-ACS1 | Audit , Control & Security | System Security | User | M |
| REQ-ACS2 | Audit , Control & Security | Backup and Recovery Requirements | User | M |
| REQ-ACS3 | Audit , Control & Security | Disaster Recovery Requirements | User | M |
| REQ-USR1 | Usability | General Usability Requirements | User | S |
| REQ-SLT1 | Service Level Targets | System Availability | User | S |
| REQ-SLT2 | Service Level Targets | System Performance | User | S |

* audit, control and security,
* global business rules,
* data requirements,
* usability requirements,

|  |  |
| --- | --- |
| **Item** | **Description** |
| Requirement ID | REQ-ACS1 |
| Category | Audit , Control & Security |
| Requirement title | System Security |
| Priority | Must (except for the owner of the project) |
| Non-functional requirement description | The project should not be accessed by anyone who is not working on it. |

* service level targets

###### 3.1.4.1 List of Non-functional Requirements

###### 3.1.4.2 REQ-ACS1 System Security

Requirements Description:

###### 3.1.4.3 REQ-ACS2 Backup and Recovery Requirements

Requirements Description:

|  |  |
| --- | --- |
| **Item** | **Description** |
| Requirement ID | REQ-ACS2 |
| Category | Audit , Control & Security |
| Requirement title | Backup and Recovery Requirements |
| Priority | Must |
| Non-functional requirement description | The project must be able to back up, auto-save and recover projects that can be lost due to unforeseen shutdowns. |

###### 3.1.4.4 REQ-ACS3 Disaster Recovery Requirements

Requirements Description:

|  |  |
| --- | --- |
| **Item** | **Description** |
| Requirement ID | REQ-ACS3 |
| Category | Audit , Control & Security |
| Requirement title | Disaster Recovery Requirements |
| Priority | Must |
| Non-functional requirement description | The project must be able to recover projects that can be lost due to unforeseen shutdowns, losses or system damages. |

###### 3.1.4.5 REQ-USR1 General Usability Requirements

Requirements Description:

|  |  |
| --- | --- |
| **Item** | **Description** |
| Requirement ID | REQ-USR1 |
| Category | Usability |
| Requirement title | General Usability Requirements |
| Priority | Should |
| Non-functional requirement description | The project should be user friendly and be able to accommodate for first time users |

###### 3.1.4.6 REQ-SLT1 System Availability

Requirements Description:

|  |  |
| --- | --- |
| **Item** | **Description** |
| Requirement ID | REQ-SLT1 |
| Category | Service Level Targets |
| Requirement title | System Availability |
| Priority | Should |
| Non-functional requirement description | The project should be available to everyone in the target market. Access should be as easy as possible without compromising business processes. |

###### 3.1.4.7 REQ-SLT1 System Performance

Requirements Description:

|  |  |
| --- | --- |
| **Item** | **Description** |
| Requirement ID | REQ-SLT1 |
| Category | Service Level Targets |
| Requirement title | System Performance |
| Priority | Should |
| Non-functional requirement description | The project should be as efficient and effective as possible. It should work as fast as possible and simplify all components for ease of use. |

## 3.2 TECHNICAL REQUIREMENTS

##### 3.2.1 List of Technical Requirements

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Req. ID** | **Requirement Title** | **Priority** | **Category** | **Responsible**  **Team**  **Member(s)** |
| TR-SBR-001 | Computer Hard drive save | S | System Backup and Recovery Requirements | Programmers/Systems  Analyst(s)/Project Manager(s) |
| TR-SBR-002 | Backup and recovery | S | System Backup and Recovery Requirements | Programmers/Systems  Analyst(s)/Project Manager(s) |
| TR-QRC-001 | QR Code Standards | M | QR Code requirements to follow ISO/IEC 18004:2006 standards | Programmers/Systems  Analyst(s)/Project Manager(s) |
| TR-DC-001 | Images into text | M | Data Conversion | Programmers/Systems  Analyst(s)/Project Manager(s) |
| TR-UE-001 | User Interface | S | User Experience/Interface Requirements | Design architect |

##### 3.2.2.1 TR-SBR-001 Computer Hard drive save

Technical requirements description:

|  |  |
| --- | --- |
| **Item** | **Description** |
| Requirement ID | TR-SBR-001 |
| Requirement title | Computer Hard drive save |
| Priority | M |
| Category | System Backup and Recovery Requirements |
| Technical requirement description | System logs and all changes should be systematically saved and backed up onto the computer hardrive. |

##### 3.2.2.2TR-SBR-002 Backup and Recovery

Technical requirements description:

|  |  |
| --- | --- |
| **Item** | **Description** |
| Requirement ID | TR-SBR-002 |
| Requirement title | Backup and recovery |
| Priority | S |
| Category | System Backup and Recovery Requirements |
| Technical requirement description | System should be able to recover changes and items after an unplanned shutdown. |

##### 3.2..2.3 TR-QRC-003 QR Code Standards

Technical requirements description:

|  |  |
| --- | --- |
| **Item** | **Description** |
| Requirement ID | TR-QRC-001 |
| Requirement title | QR Code Standards |
| Priority | M |
| Category | QR Code requirements to follow ISO/IEC 18004:2006 standards |
| Technical requirement description | System must be coded following the predefined coding standards set out in 2006 by the ISO/IEC |

##### 3.2.2.4 TR-DC-001 QR Images into Text

Technical requirements description:

|  |  |
| --- | --- |
| **Item** | **Description** |
| Requirement ID | TR-DC-001 |
| Requirement title | Images into Text |
| Priority | M |
| Category | Data Conversion |
| Technical requirement description | System must be able to input images of sheet music and then output it as text or graphical symbols to be able to show the melody in sheet form. |

##### 3.2.2.5 TR-DC-001 QR User Interface

Technical requirements description:

|  |  |
| --- | --- |
| **Item** | **Description** |
| Requirement ID | TR-UE-001 |
| Requirement title | User Interface |
| Priority | M |
| Category | User Experience/Interface Requirements |
| Technical requirement description | System user interface design must be both visually functional but also have an appealing look to it. It cannot have colour schemes that hurt the eyes and the layout must be understandable for the user. |

F**urthermore, these are the standards and specifications that the system will strive to adopt:**

**○ System Backup and Recovery Requirements**

* + - Backup arrangements
    - Recovery procedures requirement under various system failures

**○ Disaster Recovery Requirements**

* + - Minimum service level under disaster
    - Off-site backup arrangement
    - Recovery procedure
    - Time required to recover upon disaster

○ **Privacy Requirements**

* + - Protection of personal data from unauthorised disclosure e.g. protection on personal identification document number.

○ **Technical Support Requirements**

* + - Software and hardware support levels
    - Equipment maintenance and repair cycles
    - Test/diagnostic equipment

○ **Interface Requirements**

* + - User groups
    - Content presentation
    - Application navigation

○ **Maintainability, Control and System Management Requirements**

* + - System failure(s)
    - Operational readiness and success
    - System effectiveness evaluation and improvement

○ **Testing**

* + - Design stage testing procedure

○ **Data Conversion**

* + - Data conversion process
    - Data cleansing
    - Verification program

○ **User Experience**

* + - Overall experience and satisfaction when a user is using a product or system. Details within the user interface functionality, behaviour, and design

# 4.1 SYSTEM SPECIFICATION

## 4.1.1FUNCTIONAL SPECIFICATION

##### 4.1.1.1 Required System Overview

|  |  |  |
| --- | --- | --- |
| *Business Needs* | *Major Features* | *System Related Functions* |
| *Music Pattern Learning* | *Learn melody patterns from provided sheet music and compose music from what the application has learned* | *Sheet Music Reading, Analysis, Analysis Storage, Music Composition* |

##### 4.1.1.2 MMFI-01 Read Midi

|  |  |
| --- | --- |
| *Item* | *Description* |
| *Function ID* | *MMFI-01* |
| *Function Name* | *Read Midi* |
| *Category* | *Data Access layer* |
| *Function Description* | *This function reads the midi file and extracts the data stored in the file for later analysis* |
| *Mode* | *batch* |
| *Frequency* | *When invoked* |
| *Special Service Level Requirements* | *N/A* |
| *Data Integration and Conversions* | *N/A* |
| *Business Rules* | *Check if valid midi file* |

##### 4.1.1.3 MMFI-02 Analyse Data

|  |  |
| --- | --- |
| *Item* | *Description* |
| *Function ID* | *MMFI-03* |
| *Function Name* | *Compose Music* |
| *Category* | *Business Layer* |
| *Function Description* | *This function will make use of the neural network to compose sheet music in a chosen key and create a midi file with the sheet music.* |
| *Mode* | *batch* |
| *Frequency* | *When invoked* |
| *Special Service Level Requirements* | *N/A* |
| *Data Integration and Conversions* | *N/A* |
| *Business Rules* |  |

|  |  |
| --- | --- |
| *Item* | *Description* |
| *Function ID* | *MMFI-02* |
| *Function Name* | *Analyse Data* |
| *Category* | *Business Layer* |
| *Function Description* | *This Function takes the raw data that was gathered in the function ‘Read Midi” and does an analysis of the pattern and creates a neural mapping or updates the existing one.* |
| *Mode* | *Batch* |
| *Frequency* | *When invoked* |
| *Special Service Level Requirements* | *N/A* |
| *Data Integration and Conversions* | *N/A* |
| *Business Rules* | *If a neural mapping exists the application should use it and add what it learns from new sheet music to improve on what it has learned from previous experiences.* |

## 4.2 ARCHITECTURE DESIGN

##### 4.2.1 Application Architecture

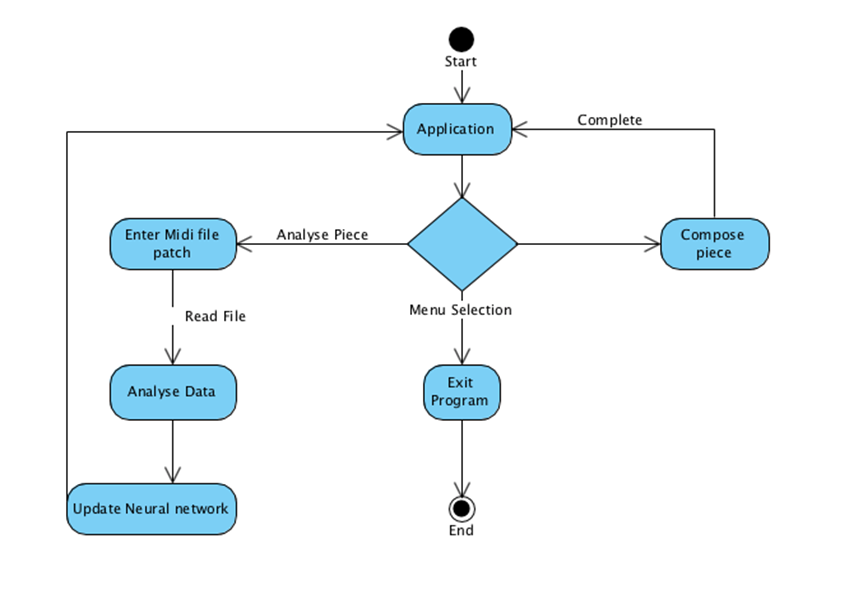
###### 4.2.1.1 Security

Our neural network is opensource and freely available on Github, thus no security threats are applicable to this project.

The availability of the information of the system will be given according to the roles the people play in the actual development of the system.

The artificial neural network voltage security monitoring and control is used. The neural network uses its association mechanism, the inherent parallel information processing nature of the neural network, which provides the capability of fast computation, enables the neural network approach to meet the demands of real time monitoring control.

###### 4.2.1.2 SYSTEM DESIGN

******

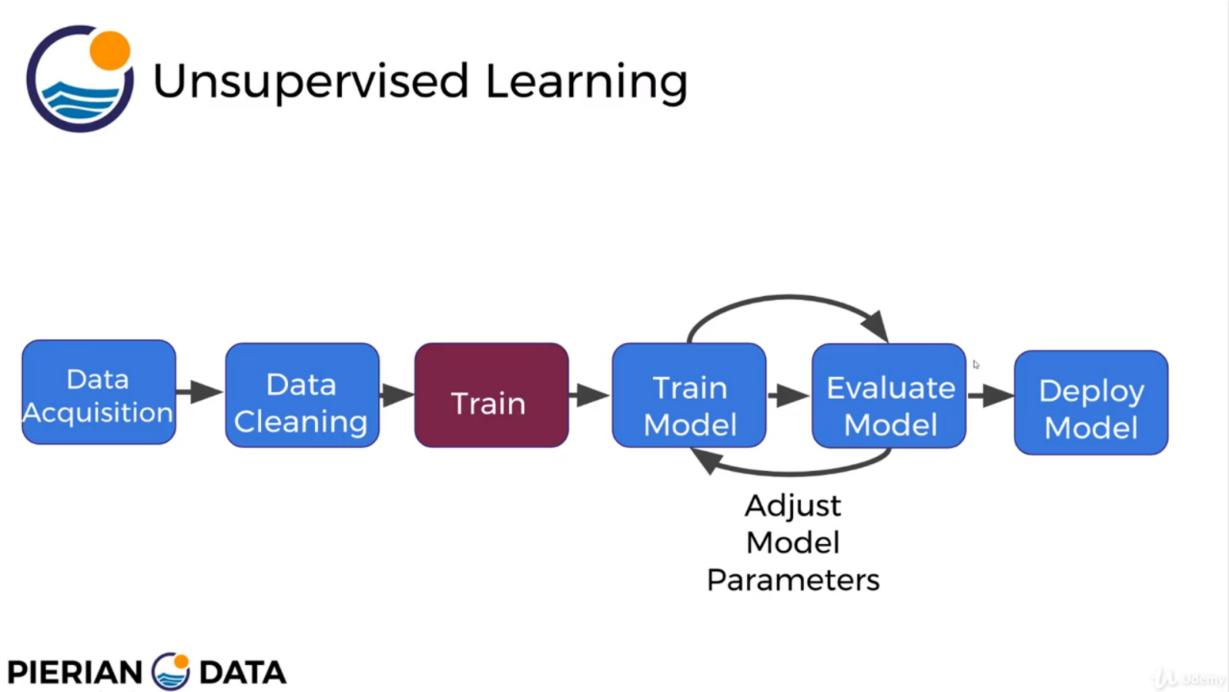
***Figure 4.2.1.2***

## 4.3 User Experience Design

*The project will use a text interface with the options:*

1. *Select genre*
2. *Train model – using input from our database.*
3. *Compose piece*
4. *Exit*

##### 4.3.1 Application Data Model

**

##### 4.3.2 Integration

* Numpy will be handeling out data managment
* Mathlib will handle our data visualization
* Sci-kit learn will handle preprocessing and data scaling

*Data management*

*Machine learning*

*Database*

*Numpy*

*.CSV*

*Output*

*MathLib*

This is how our program will be integrated

*Python data management*

*Deep learning system*

*Machine learning system*

*Anaconda*

*Matlib*

*Sci-kit learn*

*Numpy*

***Figure 4.3.2***

## 4.4 Data Architecture

##### 4.4.1.Data Source

|  |  |  |  |
| --- | --- | --- | --- |
| ***Data Source*** | ***Description*** | ***Type of data*** | ***Frequency of access*** |
| .CSV Database | A database for storing our csv input files | Musical pieces | Every time MelodyMaker wishes to compose a musical piece |
| Github Repository | Where our project is stored safely and open to the public | Program files | With every edit of the source files |

##### 4.4.2Data Store Architecture

Our Database will contain a core entity ‘Genre’ which will be used to reference all other entities which are all musical genres such as Blues, rock or classical. After converting our .midi files to .csv files we will sort them into folders and store their location as records inside each musical genre entity.

Conceptual data model diagram.

*Classical*

*Genre*

*Rock*

*1*

*1*

*\**

*Blues*

*1*

*\**

*\**

Core data entities:

|  |  |
| --- | --- |
| *Entity* | *Entity Description* |
| Genre | Main Entity – All other tables are referenced here. |
| Blues | Musical genre – All Blues csv files will be stored here |
| Rock | Musical genre – All Rock csv files will be stored here |
| … | Other music genres |

***Figure 4.4.2***

|  |  |  |
| --- | --- | --- |
| ***Data Element to be retained and archived*** | ***Archive Method and Frequency*** | ***Data Retention Policy*** |
| .csv file | Once off | Xx0000\_policy\_description |

###### 4.4.3.1Data Retention and Archive

The only data we retain are the .csv files we use to train our ANN model

###### 4.4.3.2 Data Conversion Architecture

|  |  |  |  |
| --- | --- | --- | --- |
| ***Data*** | ***Source*** | ***Target*** | ***Anticipated Volume*** |
| <Data> | <Input source> | <Input target> | <Input anticipated volume of data> |
| .csv files | .midi musical pieces | Database | 1gb |

We will be converting .midi musical pieces into .csv files so that we can use them to train our ANN model

## 4.5 Application

##### 4.5.1 Design Application

###### 4.5.1.1 Describe Common Frameworks

This is how our framework will run. A perceptron is a artificial neuron that receives input then transfers the input through multiple hidden layers and finally gives an output.

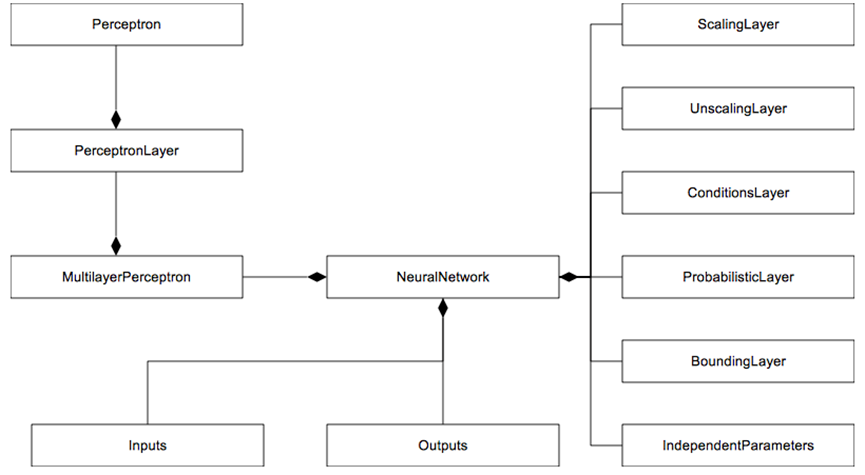
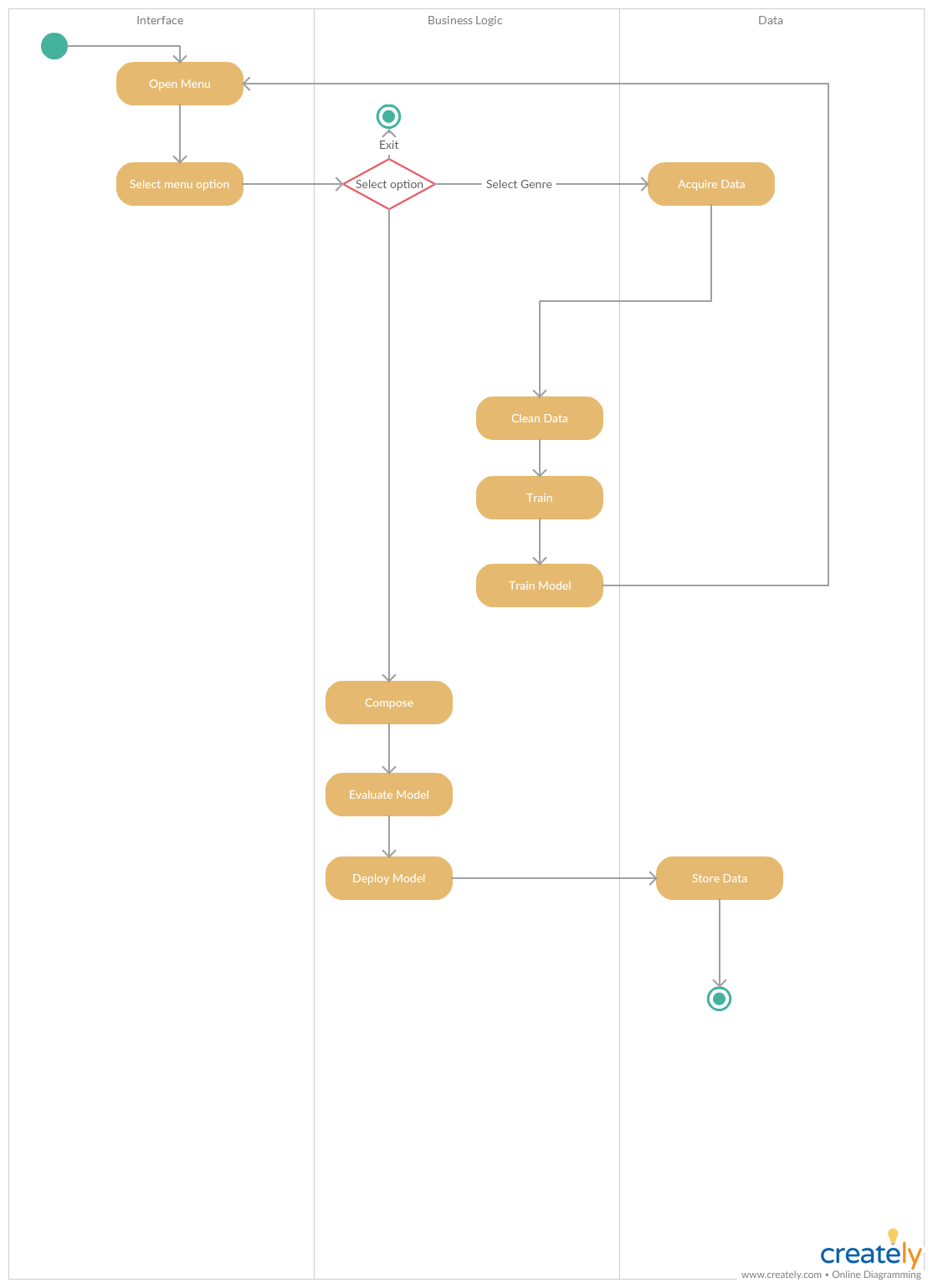


Figure 4.5.1.1

###### 4.5.1.2 Activity diagram

This diagram shows how our system will run. Beginning at the menu and ending in output data being stored

*Figure 4.5.1.2*

*Parameters given*

*Submitted*

*\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_*

*Do*

*/*

*Waitingl*

*Fetch .csv files with desired parameter*

*-*

*\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_*

*Do*

*/*

*Waiting*

*Do*

*/*

*Process preapproval*

*Training set approved*

*\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_*

*Do*

*/*

*Get training set*

*Do*

*/*

*process approval*

*Training set Rejected*

*\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_*

*Do*

*/*

*output log*

*Train model*

*\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_*

*Do*

*/*

*Data updating*

*[*

*Pre*

*-*

*approval*

*=*

*Yes*

*]*

*[*

*Pre*

*-*

*approval*

*=*

*No*

*]*

*[*

*Approval*

*=*

*No*

*]*

*[*

*Approval*

*=*

*Yes*

*]*

*Compose music*

*Application*

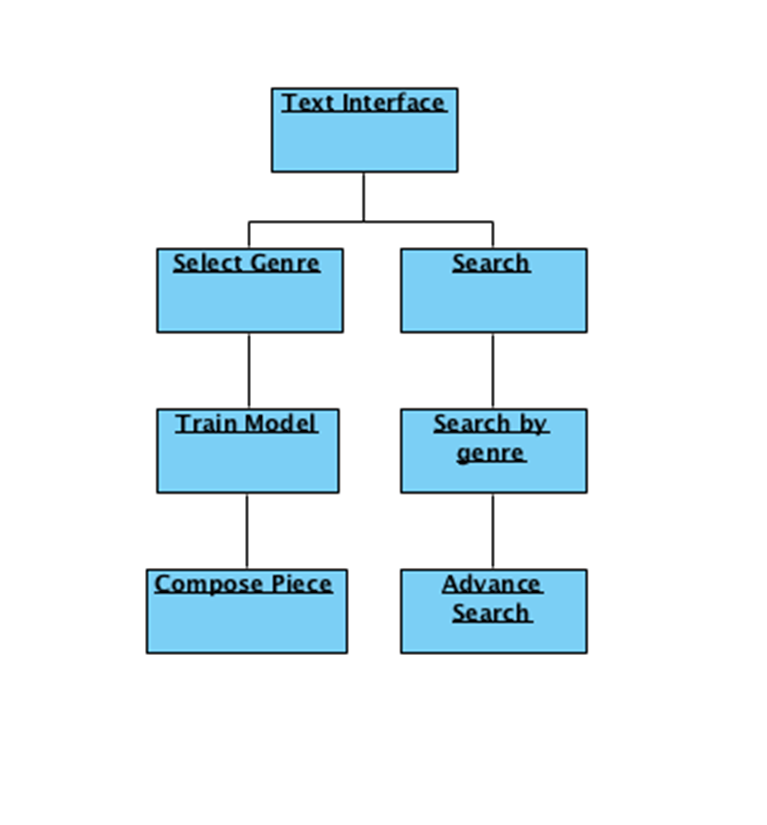
*Closure*

*Figure 4.5.1.3*

Business rules description:

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| ***Rule#*** | ***Rule*** | ***Rule***  ***Attributes*** | ***Rule***  ***Conditions*** | ***Rule***  ***Actions*** | ***Rule***  ***Priority*** | ***Rule***  ***Validity*** | ***Dependency*** |
| <Input number> | <Input name defined> | <Document the attributes and operation performed > | <Describe the conditions that the rule checks  > | <Input actions> | <Input the priority of the rule> | <Input the constraints under which the rule is valid > | <Describe the dependency on another rule> |
| 1 | Genre requested | csvID  GenreID | Data sent matching genre | Perform check whether Data = Genre | 1 | Rule is applicable  Only when spesified | N/A |

###### 4.5.1.4 User Interface



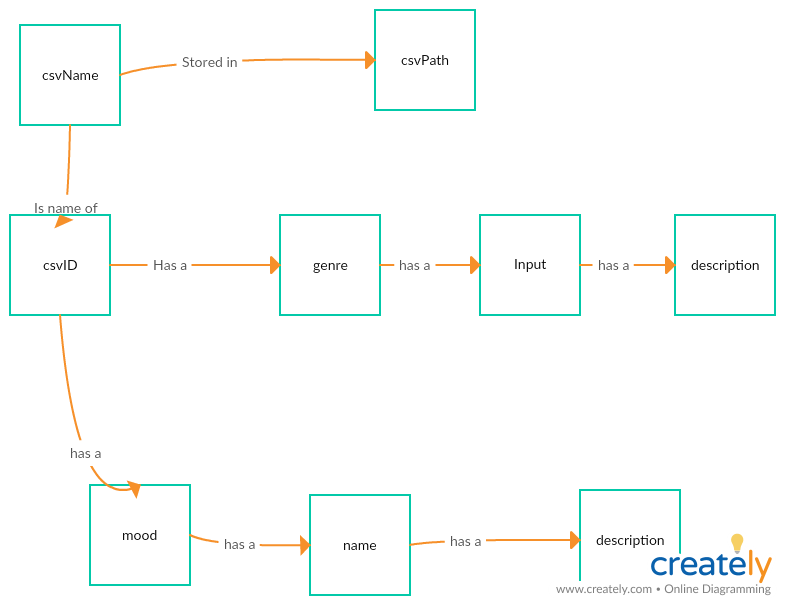
*Figure 4.5.1.4*

## 4.5.2 Data Model

##### 4.5.2.1Logical Data Model

Logical data entity description:

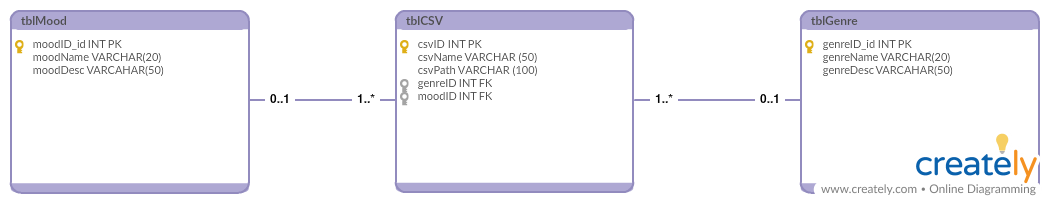
|  |  |
| --- | --- |
| ***Logical Data Entity*** | ***Logical Data Entity Description*** |
| **csvID** | **Unique identifier for csv files** |
| **csvName** | **Name of csv file** |
| **csvPath** | **Where file is stores** |
| **genreID** | **Unique identifier for genres. Connects genre table with CSV table** |
| **genreName** | **Name of genre** |
| **genreDiscription** | **Give a detailed explanation of the genre** |
| **moodID** | **Unique identifier for genres. Connects Mood table with CSV table** |
| **moodName** | **Name of mood** |
| **moodDescription** | **Give a detailed explanation of the mood** |

**Figure 4.5.2.1**

###### 4.5.2.2 Physical Data Model

Physical data entity description:

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| ***Table Name*** | ***Field Name*** | ***Field Format*** | ***Field Length*** | ***Description*** | ***Mandatory*** | ***Primary Key*** | ***Foreign Key*** |
| <Input> | <Input> | <Input> | <Input> | <Input> | <Input Y/N> | <Input  Y/N> | <Input> |
| tblCSV | csvID | int | n/a | Unique identifier for csv files | Y | y | N/A |
| tblCSV | csvName | varchar | 50 | Name of csv file | y | n | n/a |
| tblCSV | csvPath | varchar | 100 | Where file is stores | y | n | n/a |
| tblCSV | genreID | int | n/a | Foreign key to tblGenre | y | n | y |
| tblCSV | moodID | int | n/a | Foreign key to tblmood | y | n | y |
| tblGenre | genreID | int | n/a | Unique identifier for genres | y | y | n/a |
| tblGenre | genreName | varchar | 20 | Name for genres | y | n | n/a |
| tblGenre | genreDesc | varchar | 50 | Give a detailed explanation of the genre | y | n | n/a |
| tblMood | moodID | int | n/a | Unique identifier for mood | y | y | n/a |
| tblMood | moodName | varchar | 20 | Name for moods | y | n | n/a |
| tblMood | moodDesc | varchar | 50 | Give a detailed explanation of the mood | y | n | n/a |



*Figure 4.5.2.2*

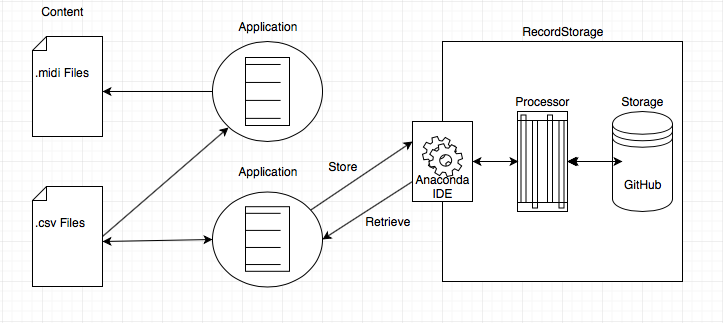
# 5 TECHNICAL SYSTEM OPTION

## 5.1 TECHNICAL SYSTEM ARCHITECTURE

##### 5.1.1 Network Architecture

Our application won’t require a network Architectture.

##### 5.1.2 Storage Architecture

The storage architecture consists of a database filled with the paths of the csv files which the program needs to analyse and preprosess to produce, the music/melody required. The application then uses the preprossed files to produce an output csv file which will then connect to another program which will convert it into a .medi file.

*Figure 5.1.2*

##### 5.1.3 Platform Architecture

GitHub] is a web-based hosting service for version control using git. GitHub is designed for collaborating on coding projects. Nonetheless, it is also a potentially great resource for researchers to make their data publicly available. Specifically you can use it to:

* store data in the cloud for future use (for free),
* track changes,
* make data publicly available for replication,

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ***Environment*** | ***Machine*** | ***Hardware*** | ***Description*** | ***Software*** |
| Music  Generation | Personal Computer | Minimum 3 GB disk space to download and install Anaconda. | Application | * 32- or 64-bit computer. * Windows, macOS or Linux. * Python 2.7, 3.4, 3.5 or 3.6. * pycosat. * PyYaml. * Requests. * Anaconda IDE * Tensorflow |

The neural network will be trained to recognize/analyze melodies in sheet music -. midi files and learn from them. We will be using Tensorflow on the Anaconda IDE to develop our system. After we have trained the neural net to create its own melodies we can then add other musical components such as a drum beat and a bass line. We will use a Long Short-Term Memory network. They are a type of Recurrent Neural Network that can efficiently learn via gradient descent. Using a gating mechanism, LSTMs are able to recognise and encode long-term patterns. LSTMs are extremely useful to solve problems where the network has to remember information for a long period of time as is the case in music generation.

MelodyMaker’s boundaries will be the input data received from users that used the method previously. The melodies from data that has already been stored will form the interface that will enable the program to create new songs and melodies.

The program will look at aspects such as repetition and structure and use this to refine the new melodies that are to be written. Using these repetition and structure repetitions the program will be able to ensure that the melodies ”sound” harmonious and that it is audible to the listener.

## 5.2 SIZING MODEL

##### 5.2.1 Data Storage

MIDI files are structured into chunks, a single header chunk followed by one or more track chunks. Each chunk consists of:

* A 4-byte chunk type (ascii)
* A 4-byte length (32 bits, msb first)
* length bytes of data

There are two types of chunks:

* Header Chunks, which have a chunk type of "MThd"

The data part of a header chunk contains three 16-bit fields. These fields specify the format, number of tracks, and timing for the MIDI file.

The length of the header chunk is 6-bytes. However software, which reads MIDI files, is required to honor the length field, even if it is greater than expected. Any unexpected data must be ignored.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ***Header Chunk*** | | | | |
| ***Chunk Type*** | ***Length*** | ***Data*** | | |
| 4 bytes  (ascii) | 4 bytes  (32-bit binary) | Length*(= 6 bytes)* | | |
| *16-bit* | *16-bit* | *16-bit* |
| *MThd* | Length | Format | Tracks | Division |

* Track Chunks, which have a chunk type of "MTrk"

The data part of a track chunk contains one or more delta time pairs. The delta time is not optional, but zero is a valid delta-time.

All functions in the system had been considered in the sizing analysis.

• The sizing model will cater for the projected 2-year growth of the system.

• 1 day = 8 working hours; 1 month = 30 working days.

• Transaction volume for Year 0 is determined based on that of existing system.

|  |  |  |
| --- | --- | --- |
| ***Track Chunk*** | | |
| ***Type*** | ***Length*** | ***Data*** |
| 4 bytes  (ascii) | 4 bytes  (32-bit binary) | Length bytes   (binary data) |
| MTrk | Length | Delta time, Event |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| ***Entity***  ***Name*** | ***Annual Growth Rate (%)*** | ***Record Length (byte)*** | ***No. of Records*** | ***Record Storage (MB)*** | | |
| *Yr 0* | *Yr 1* | *Yr 2* |
| Midi.files | 10.00% | 4 | 3000 | 0.12 | 0.13 | 0.15 |

###### 5.1.2.1.2 Record storage

###### 5.1.2.1.3 Transaction Rate

###### 5.1.2.1.3 Data Access

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ***Function***  ***Name*** | ***Entities*** | ***Avg. No. of***  ***Records***  ***Accessed*** | ***Yr 2***  ***Transaction***  ***Rate***  ***(Hourly***  ***Peak)*** | ***Retrieve*** |
| Access Application | USER | 300 | 4.8 | 1350 |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| ***Function***  ***Name*** | ***Transaction***  ***Volume of***  ***Year 0*** | ***Annual***  ***Growth***  ***Rate***  ***(%)*** | ***Mode*** | ***Transaction Rate (Hourly Peak)*** | | |
| ***Yr 0*** | ***Yr 1*** | ***Yr 2*** |
| Search Music Genre | 4,620,691 | 5% | Online | 2,406.61 | 2,526.94 | 2,653.28 |

## 5.3 COST / BENEFIT EVALUATION

|  |
| --- |
| ***Benefits***   * No Benefits   ***IF we should sell***   * Increased Sale: R30 000 |
| ***Development Cost: R2400***   * Udemy Courses: R2400.00 |
| ***Operational Cost: R0.00***   * Hardware: R0.00 * Software: R0.00 * Operational: R0.0 |
| ***Total Cost: R2400.00*** |

## 5.4 Impact analysis

##### 5.4.1 Summary on system change/enhancement

**Current State:** The neural network will be trained to recognize/analyse melodies in sheet music -. midi files and learn from them.

**Future State:** After we have trained the neural net to create its own melodies we can then add other musical components such as a drum beat and a bass line.

##### 5.4.2 Effect on organisation levels

The application’s quality will be maximised throughout the neural network with the implementation of components such as a drum beat and a bass line. Implementation of the components will result in improvement of the MelodyMaker which will increase the application’s quality.

##### 5.4.3 Significant changes in user operating procedures

UserWill be able togenerating high quality music for their personal use.

##### 5.4.4 Implementation considerations

In addition to the identified impacts that will result from the deployment of the new systems, the transitional implementation period will also bring about Project changes. A amount effort and time will be required to facilitate the implementation of the new systems.

##### 5.4.5 Change management

Prior to system deployment, User acceptance test is required to assess if the functions developed within the application fulfil and work as specified within the functional specification document. Team members are required to participate in the UAT and test the functions to make sure the functions work as specified.

**Resulting Impact**: Due to the additional effort, the project team will have less time in performing their normal daily operational tasks.

**Recommended Solution:** Extending the daily time spent on developing the project to maximize our application’s efficiency.

##### 5.4.6 Data Migration

To ensure we reach our quality goal, an extraordinary amount of midi files will need be required. This will increase the time of the process, transferring data between file formats.

**Resulting Impact**: Due to the additional effort, the project team will have less time in performing their normal daily operational tasks.

**Recommended Solution:** Extending the daily time spent on developing the project to maximize the time spent on transferring data between file formats.

##### 5.4.7 Risk Analysis

With extensive changes required in implementing the music generation application, there are certain risks involved in completing the project. To ensure success in the system implementation, potential risks and the associated mitigation solutions are identified.

**Potential Risk**: Blues music incorporates more complex rhythmic and harmonic structure, adding additional components will add complexity, making learning inherently more challenging.

**Recommended Solution:** Extend the training and fine-tuning of the network , to ensure our model is able to learn to compose music that are both syntactically correct and also able to fool humans.

## 5.5 IMPLEMENTATION PLAN

##### 5.5.1Implementation Strategy

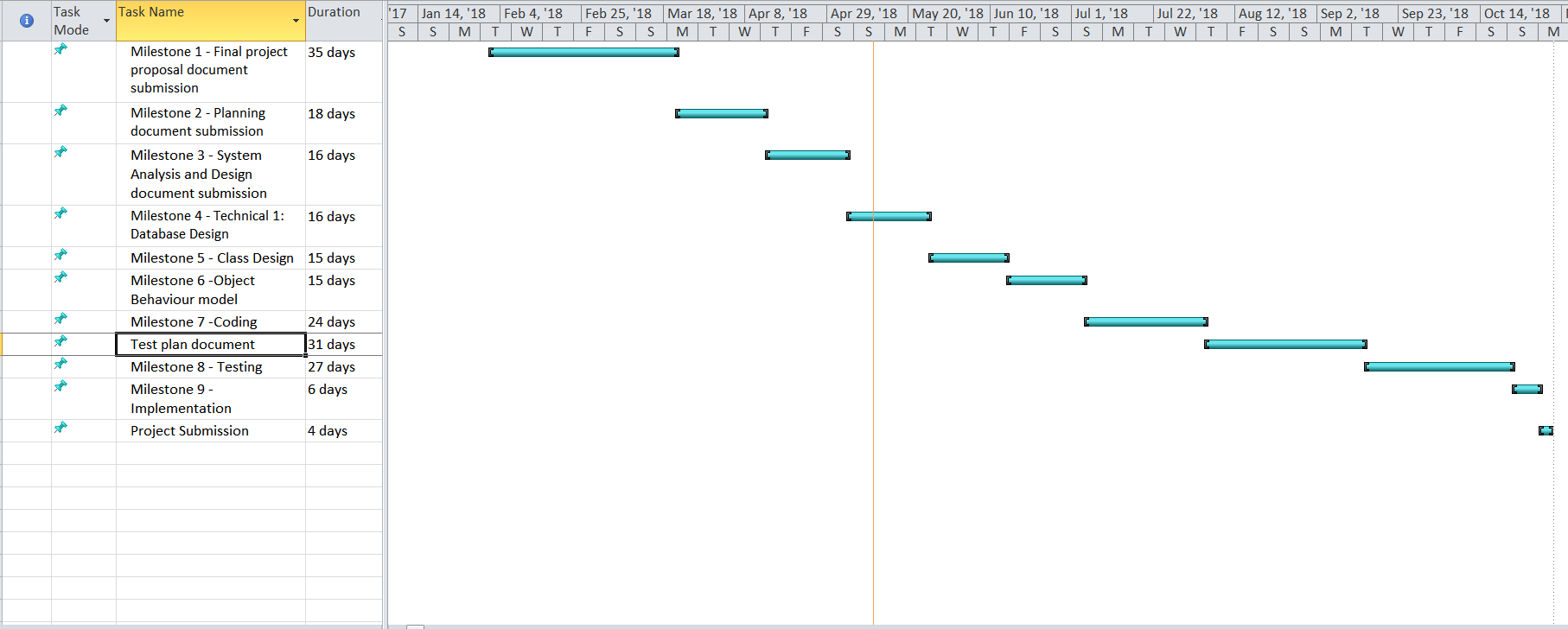
1. **Prepare the infrastructure.** This strategy includes a review of hardware, software, communications, etc. When you are ready for implementation, the production infrastructure needs to be in place.
2. **Coordinate with the organizations involved in implementation.** Part of the implementation work is to coordinate the with team members that have a role to play.
3. **Install the production solution.** Here our solution will be moved from development to test. If there are major changes to a current solution, we may have a lot less flexibility in terms of when the new solution moves to production, since the solution might need to be brought down for a period of time. We have to make sure all of your production components are implemented successfully, including new hardware, databases, and program code.
4. **Convert the data.** Changing our data from one format to another, this needs to take place once the infrastructure and the solution are implemented.
5. **Perform final verification in production.** Implementation of our testing method will take place to ensure everything is working as we expect. This may involve a combination of development and client personnel. The first check is just to make sure everything is up and appears okay. The second check is to actually push data around in the solution, to make sure that the solution is operating, as it should.
6. **Implement new processes and procedures.** Implementation of the new solution that requires success of our application will take place here. These changes will be implemented at the same time that the actual solution is deployed.
7. **Monitor the solution.** The project team will spend some period of time monitoring the implemented solution. If there are problems that come up immediately after implementation, the project team will address and fix them.

##### *5.5.2* Implementation Schedule

|  |  |  |
| --- | --- | --- |
| **Deliverable** | **Description** | **Planned Date** |
| Milestone 1  Final project proposal document submission | This includes the final proposal that will be assessed before commencement of the project. It is the first item of the development process that will be used to guide all other items on a basic level. | 2018-03-21 |
| Milestone 2  Planning document submission | Much like the proposal, the planning document will be used to guide the development of all other items, however, it will be used on a more in-depth scale. It provides guides and steps in a predefined manner that allows the configuration of all items to be executed as harmoniously as possible. | 2018-04-13 |
| Milestone 3  System Analysis and Design document submission | Here we identify, break down and assess all major and minor aspects of the development of the system, the environment, effects and the functionality of the system itself. | 2018-05-04 |
| Milestone 4  Technical 1: Database Design | This is technical aspect of the system. The database is one of the most important physical components of the system. Once this has been correctly developed, it can also be used for Class creation which is the next item. | 2018-05-25 |
| Milestone 5  Technical 2 : Class Design | This item can be based on the elements found in the database design. It is a technical item that defines the basis of code development of the system. | 2018-06-14 |
| Milestone 6  Technical 3 : Object Behaviour Model | This technical item works with thee classes and database as a baseline. However, it is also the baseline for coding the system. It will be assessed and defined before coding commences for the purpose of simplifying the intense complicated nature of coding. | 2018-07-04 |
| Milestone 7  Technical 4 : Coding | This is the item that configures all previous items. It is the physical compilation of all planning up until this point. The system is physically created here. | 2018-08-04 |
| Test plan document submission | Here we will identify our approach to testing the application. | 2018-09-14 |
| Milestone 8  Testing | This is where we will implement our testing strategy that we have identified in the test plan document for our application. | 2018-10-22 |
| Milestone 9  Implementation | This is where the implementation process of our application will be executed. | 2018-10-29 |
| Project Submission | The project is in final form and must be ready to be summited on the given date. | 2018-11-01 |

***System Analysis & Design Report Technical System Option***

<Gantt Chart Diagram.> **Sample:**



*53*