SOFTWARE DESIGN DOCUMENT

PROJECT NAME – MUSOPLAN

Written by – Smita Jacob

Date – 25 Aug 2021

INTRODUCTION

Globex is wanting to develop MusoPlan - a software solution that will be used by the music management team to help model and predict the success of Globex's elite musicians by identifying costs when deploying them as a troupe. This document is being created in-line with the requirements discussed with the Music Team Management Officer at Globex. It is important that this organisational software adheres to version control requirements.

SYSTEM OVERVIEW

Musoplan needs to cater to the following requirements:

i. Create a musician -

Select whether the musician is a guitarist, bassist, percussionist, or flautist

Record their name, years playing, and hourly rate

ii. Create a troupe -

Record the name, minimum duration, genre

iii Add a musician to a troupe

iv. Provide a summary description of a troupe

v. Provide a detailed description of a troupe

vi. Calculate the cost of deploying the troupe for a given number of hours

vii Read a list of troupe names to be populated from a file

viii. Write a list of the detailed descriptions for all troupes to a given filename

ARCHITECTURAL DESIGN

To implement the above requirements, this is how I have put the architected the software design

● Class descriptions and Class relationships:

Define a Super class ‘Musician’ with -

Properties:

-name

-years playing

-hourly rate

Methods:

-to display the properties listed above

Define 4 child classes: Guitarist, Bassist, Percussionist, Flautist with the following:

Properties:

-name (inherited from ‘Musician’)

-years playing (inherited from ‘Musician’)

-hourly rate (inherited from ‘Musician’)

Methods:

-to display properties (inherited from ‘Musician’)

-to display specialty (i.e., Guitarist/ Bassist/ Percussionist/ Flautist)

-to append ‘interesting fact’ of child class

eg.

“The more strings you have, the better you are

Tom is a guitarist”

So, while each child class has properties similar to the super class they have an additional property that changes value from one child class to another.

Next, we define ‘Troupe’ as a map i.e., as a collection of elements where each element is stored as a key, value pair. In this case, every time the user defines a troupe, the ‘troupe name’ gets mapped to an object with the following elements:

{

troupe's name

number of instruments it contains

genre the troupe plays

minimum duration of a booking

hourly rate for the whole troupe

}

As the user adds one troupe after another, these details get stored in memory or ‘pushed’ to an array so that they can be accessed and called using the key (troupe name). The user is prompted for Troupe values and these values are stored in the map using the ‘set’ function. When the user wants troupe related details, he/she can call for the ‘get’ function and retrieve the corresponding troupe details from the array. Once retrieved, these mappings can be displayed on the command line or written into a text file. The user can also read from this text file for retrieving troupe details.

Troupe map

Eg. [troupeA, troupeName]

instruments

genre

duration

rate

Troupe list as an array of maps/objects eg.

[troupeA, troupeB,…., troupeN,..]

● Program Inputs

Users will use the Command Line interface to access Musoplan. The program will offer interactive menu options to the user to pick and assign values to entities.

● Program Outputs

Users interact with Musoplan using the Command Line interface. Some of the outputs are displayed right on screen to the user while some outputs are written to text files for the user.

DESIGN CONSIDERATIONS

1. Assumptions and Dependencies

. Musicians are hired as part of a troupe and not individually

. A musician can only be the member of one troupe

. The hourly rates of troupe are independent of the hourly rates of the musicians in it

2. General constraints

The constraints for the Musician and Troupe entities are as follows:

Musician data constraints are

· Name should be between 3 and 30 characters

· Musician must play one of the four instruments below

· Years of playing the instrument must be non-negative

· Hourly rate must be over 50

Troupe data constraints are

· Name between 3 and 30 characters

· No more than 5 musicians

· Valid genre from (rock, jazz, pop)

· Minimum duration in hours between 0.5 and 3

3. Goals and Guidelines

The goal of the software is to give the Musician Team Officer the ability to link anywhere between 2-5 musicians to each troupe and then draw a list of troupe names or descriptions, as needed.

Once linked, the officer/user should be able to write the troupe summary/detailed description to a text file or read the list of troupe names from a text file.

The guideline is that the data entry be element by element on the Command Line interface and distributed version control be adhered to as company and industry best practices. For this, Git has been the VCS of choice.

4. Development Methods

We are following the waterfall development method for this project. This means, we would be progressing sequentially from planning, analysis, design, development, testing, implementation, and maintenance phases.