**User Acceptance Report**

In the User Acceptance meeting today, the software inputs and out[puts were displayed and they met user requirements.

***How the software used encapsulation, abstraction, inheritance, and polymorphism:***

The software uses *encapsulation* by means of Classes. For eg., the Musician class consists of properties and methods that are encapsulated and hidden from direct access. The only way to access the properties is by accessing the methods of the class.

The software uses *abstraction* in the sense that the music manager will just be accessing the software menu and getting the required outputs without knowing the inner workings of the software.

The classes Guitarist, Bassist, Percussionist and Flautist demonstrate *inheritance* when they inherit the properties and methods of the Musician class.

The classes Guitarist, Bassist, Percussionist and Flautist demonstrate *polymorphism* by way of having their own unique ‘interesting facts’.

***How I tested the software***

I tested the software in front of the user by demonstrating all the features of the software through the command line/terminal. I also displayed the text files that were written to or read from.

***How I fixed any defects discovered during testing***

The user clarified that she might need to calculate the cost of hiring the troupe as part of an independent menu, unlike how it was being calculated alongside other troupe details in the software. I have built that into the software and now the user will be able to feed the duration and rate separately to arrive at the cost of engaging the troupe.

***My response to the feedback on the VCS use report***

I have updated the VCS use report in line with the feedback. It includes the challenges I faced while using Git during the development of Musoplan software.

***User acceptance of your software***

After viewing a run-through of the software features, the music manager seems pleased with the delivery of the below requirements (as laid out in the first meeting).

1. The user is able to use the Command line for inputs and outputs. There are features to write-to and read-from text files
2. The user will be inputting specific details for musicians and troupes which will then get stored in memory so that they can be mapped to each other
3. The user will be feeding records, one property at a time
4. The user will decide how many musicians/troupe need adding, in one session. The program will allow for taking inputs as many times as indicated by user.
5. Individual musician rates are independent of troupe rates. Troupe costs are calculated purely on troupe rates and duration of engagement.

Since these requirements were met, the user has given a verbal sign-off on the software implementation on 3rd September 2021 at 12:20pm.

Report written by - Smita Jacob, Programmer, Globex small solutions department