**VCS Management Report**

I have selected GitHub for Musoplan Project as the version control management system since GitHub meets all the criteria mentioned in the organisational VCS requirements document for Globex Corporate.

As per the project development requirement, created a GitHub repository in the GitHub web account.

To work with the project in the local computer location, the repository link has copied from GitHub account and cloned it into the local project folder.

For that, I did the following steps:

* Cloned the GitHub repository link from the website.
* Open the local project folder and open the Git bash.
* Using Git command, ‘git clone <copied repository link> enter’
* Now, the local repository has created.
* Uploaded the Meeting minutes text file and VCS documentation.
* Created a branch under Main Project branch using git command:

‘git checkout -b <second branch name>’

* Added Project Design report md file in the newly created second branch using git command:

‘git add <ProjectDesign.md>’

* Added Project Coding JavaScript file in the newly created branch second branch using git command:

‘git add <ProjectCoding.js>’

* Committed and updated the change by using git command for each file:

‘git commit -m “msg” enter

* Updated files to the second branch using git command:

git push origin <second branch name>

* Checked the status of the files in the repository to check whether all the files have successfully deployed and to check for any untracked files to be committed.

Using git command: git status.

* To check the entire branch contents and activities performed,

and used git command: git log.

* Merged the completed documents to the main branch by using git commands below:

git checkout main

git merge <second branch name>

**Compliance Check with the VCS requirement document of Globex.**

☒ The Musoplan project must be done using a distributed VCS that is familiar and has good online supporting ecosystems.

☒ All development in the VCS must take place on feature branches, so the main branch is exclusively initialisation and then merges, hence created secondary branches test and code runs.

Created a new branch under main branch to coordinate project files and activities throughout the design and coding phase.

☒ All commit messages must follow the industry standard format.

Applied the format as per company standards

☒ Commit scope should be kept to a single change. Always made sure that git commit has done after every file update.

Applied the scope as single changes,

for e.g.: git commit -m “Uploading completed MusoPlanCode.js file”

git commit -m “Uploading VCSReport.md file”

☒ VCS must be configured so each commit contains the author's name and email address.

This has done and verified at the setting up and installation stage of GitHub and repository.

☒ All repositories must start with just a README.md file and no source code, do not initialise a repository with code already present.

Verified the project folder with README.md file has created at the initial stage itself.

**Issues encountered while using GIT**

Few of the issues occurred during the Git repository development and usage are below:

* Error shown: Main ahead of commits.

Solution: Checked the git status and log.

Found the uncommitted files and committed it.

* Error shown: Can’t find the file or file path

Solution: Change directory to the correct folder and ran commands again.

**Project Presentation Summary**

* How the software meets the user requirements

Musoplan Application is developed strictly according to the user requirements and VCS compliance documentations. This has been verified throughout the different stages of SDLC.

Musoplan application has the functionalities listed below as per the user requirement document.

* To add musician
* To add musician details based on instrument.
* To create troupes with musicians list.
* Function to read musician list as text file.
* Function to enter musician details by user.
* Function to display troupe details and list as text file.
* Function to display troupe details and list on screen.
* Has functionality to enter troupe summary description.
* Has functionality to enter troupe detailed description.
* Has the functionality to calculate the cost of troupe.
* How your software used encapsulation, abstraction, inheritance, and polymorphism
* Abstraction: In the terminal, user is allowed to see the prompt and instructions to enter user input and have options to enter input values. The user also gets alerts on entering wrong input values.

But the backend functionalities that are running behind these modules are hidden from the user.

* Encapsulation: After the collection of user input values, using encapsulation principle, all the sensitive user values are stored privately into class or bind to the constructor variables which will not be available as public values.
* Inheritance: Musoplan has Musician as the Parent class, it has subclasses according to musician type such as Guitarist, Bassist, Percussionist and Flautist in order to store each musicians’ details and interesting fact. For this, these child classes inherits the musician properties from the parent class.
* Polymorphism: Each child class has a function from the parent class named as “display details”. But the functionality inside each class is different from the functionality inside the parent class method.
* How you tested your software
* Performed unit testing thoroughly on each module to find and rectify any errors occurred.

Unit testing has done on:

* + User input prompt module.
  + Troupe cost calculation method
  + Musician details display method
  + Add musician functionality
  + Show troupe details and summary
  + Reading from the text file method
  + Writing to the text file method.
* How you fixed any defects discovered during testing
  + Effectively applied conditional statements with if else, while loops for data validation.
* Your response to the feedback on your VCS use report

I have updated and successfully uploaded the VCS use report after the changes requested by the client after feedback.

Updated git commands description such git log and git status and its differences.

Updated the checklist with examples from the current project.

* If your repository was forked from another repository, how would you go about having them integrate your changes to the central repository?

The repository integration part has been done by a 2-step process. The first step is to ensure that the local changes are based on the latest version of the upstream repo and are then uploaded to the GitHub repository. This is often done via a git push command. Once the local changes are online, then after selecting the GitHub Repository and click on ‘Open a Pull Request’ to create a Pull-Request for Another (Upstream) repository. This would require a Pull Request message before submitting, the other repository’s owner will verify for any changes or any feedback before being accepted / merged.

* Obtain user acceptance of your software

Successfully obtained user acceptance of the Musoplan software after the presentation.