PES UNIVERSITY
AUGUST – DECEMBER 2022 SEMESTER 5
SOFTWARE ENGINEERING LAB TASKS
Case Study - 3

MusicWreck

VERSION 1.0

Prepared By-

PES1UG20CS331 - RENITA KURIAN

PES1UG20CS334 – RICHA SHAHI

PES1UG20CS337 – RIMJHIM SINGH

PES1UG20CS344 – RIYA JHA

Problem Statement – 3:

Configuration management helps maintain various versions of the project and eases promotion of new releases. With respect to your project, answer the below questions and explore the mentioned scenarios with git

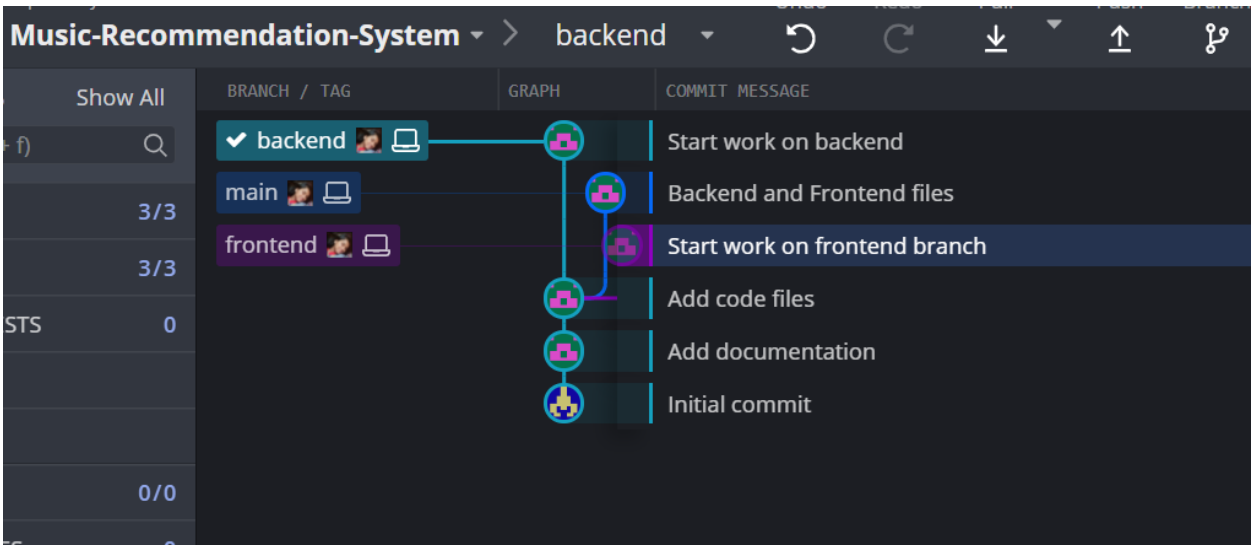
- Build a table to separate all of your files on the project repository into configuration items

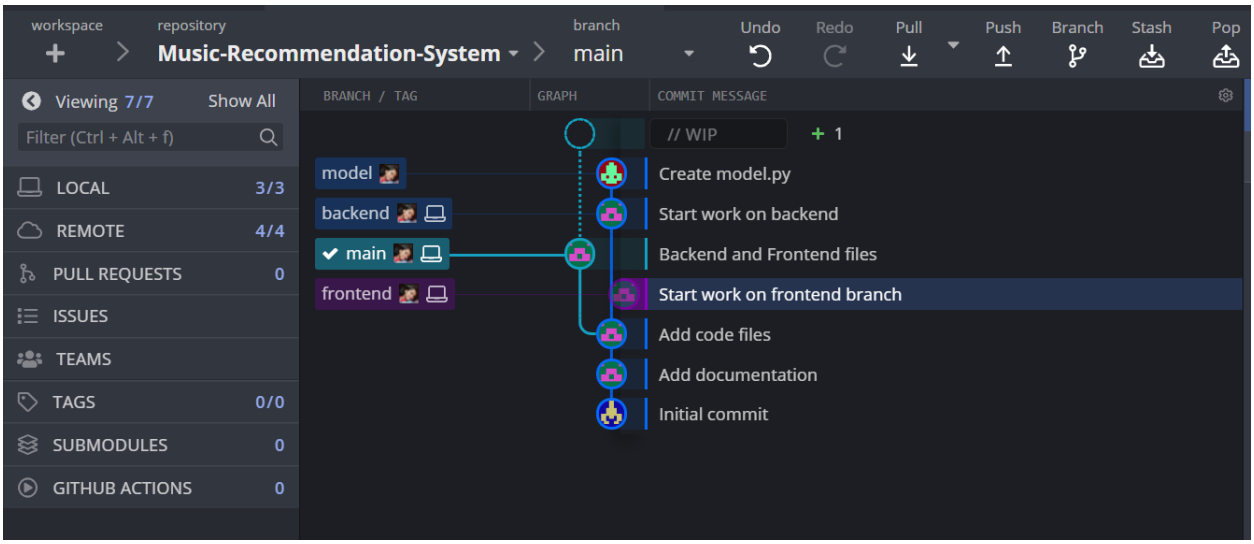
File or Record	Definition	File Type	Version
Spotify.csv	Dataset for training and testing MI models extracted from Spotify Developers API	Comma Separated Values	V1.0
Model.ipynb	Jupyter Notebook for data analysis and visualization	.ipynb file	V1.0.1
Frontend.py	Frontend file having embedded design files such as html and css	Flask frontend	V1.0.1
Backend.py	Backend file deployed on WSGI development server	Flask backend	V1.0.1
Error.txt	Error Log File	Text file	

SRS	System Requirement Specification for requirement Engineering	Docx file	V1.0.1
.gitignore	Configuration file	.gitignore with versions specification for every tool	

- Each team member should add/modify any of the configuration items on a separate branch

  Team members working on different branches-

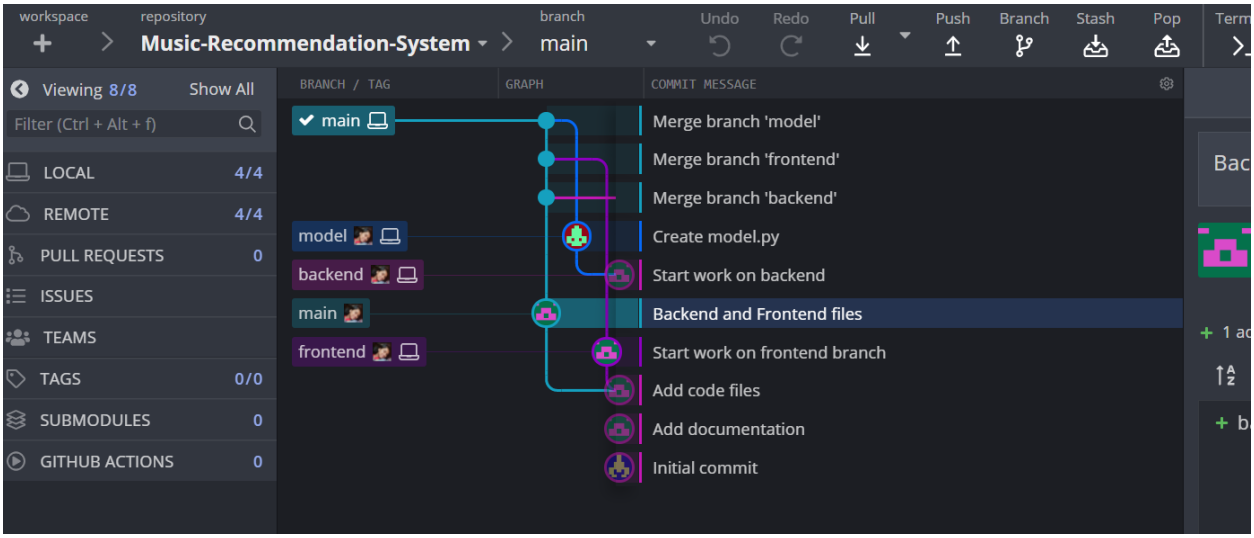





```
Chairman APGB@DESKTOP-B1LK69A MINGW64 /d/Documents/PESU/SE/PROJECT/Music-Recommendation-System (backend)
$ git branch
* backend
  frontend
  main
```

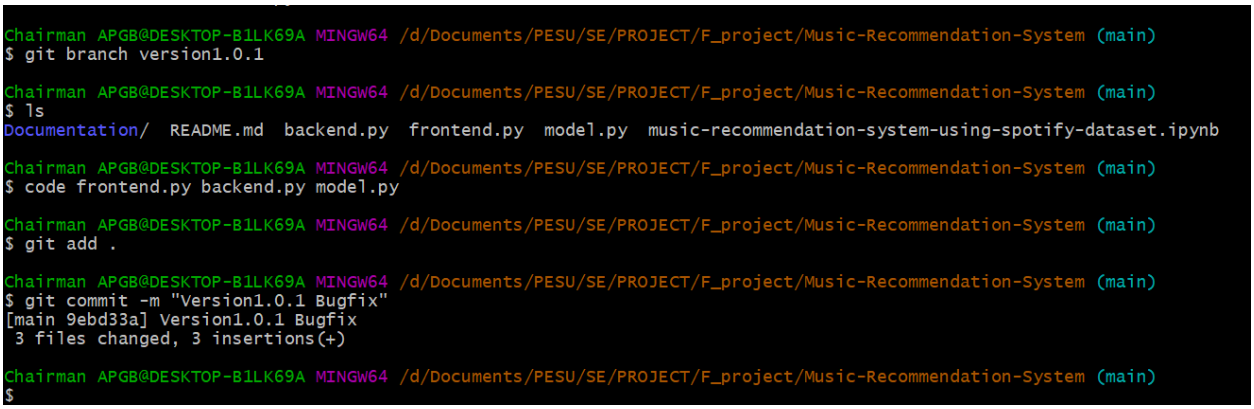
Merging branches to main-

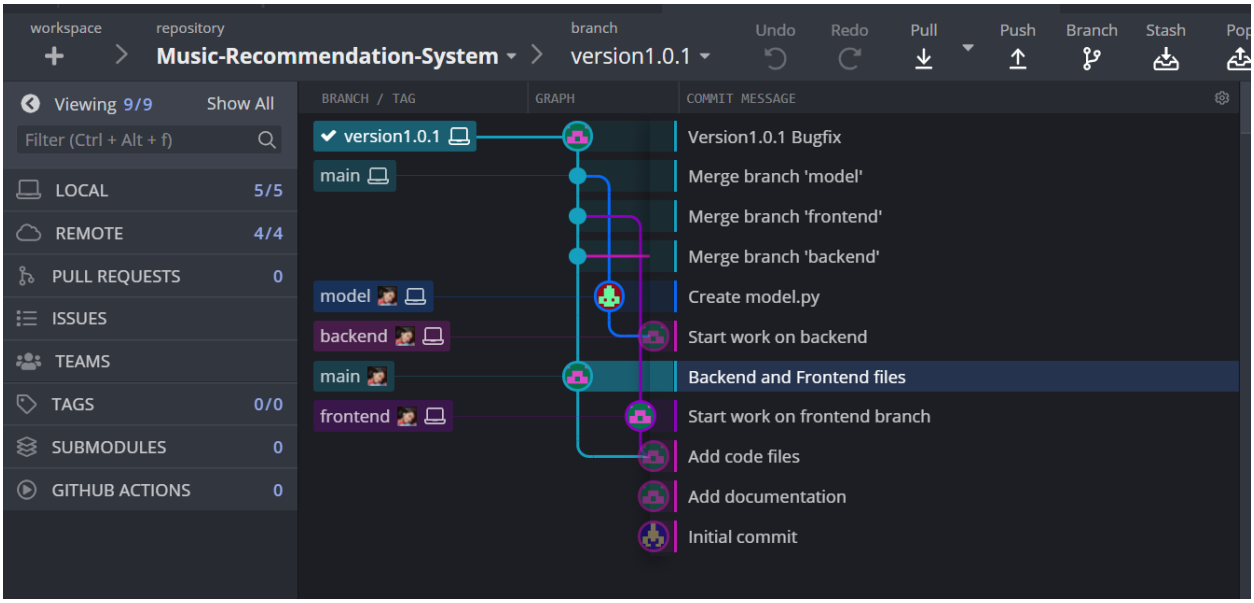
```
Chairman APGB@DESKTOP-B1LK69A MINGW64 /d/Documents/PESU/SE/PROJECT/F_project/Music-Recommendation-System (main)
$ git merge backend
Merge made by the 'recursive' strategy.

Chairman APGB@DESKTOP-B1LK69A MINGW64 /d/Documents/PESU/SE/PROJECT/F_project/Music-Recommendation-System (main)
$ git merge frontend
Merge made by the 'recursive' strategy.
 frontend.py | 0
 1 file changed, 0 insertions(+), 0 deletions(-)
 create mode 100644 frontend.py

Chairman APGB@DESKTOP-B1LK69A MINGW64 /d/Documents/PESU/SE/PROJECT/F_project/Music-Recommendation-System (main)
$ git merge model
Merge made by the 'recursive' strategy.
 model.py | 1 +
 1 file changed, 1 insertion(+)
 create mode 100644 model.py

Chairman APGB@DESKTOP-B1LK69A MINGW64 /d/Documents/PESU/SE/PROJECT/F_project/Music-Recommendation-System (main)
$
```



- Assuming your main branch to be the baseline, make further modifications on a different branch to indicate version of the project

  Working on different versions

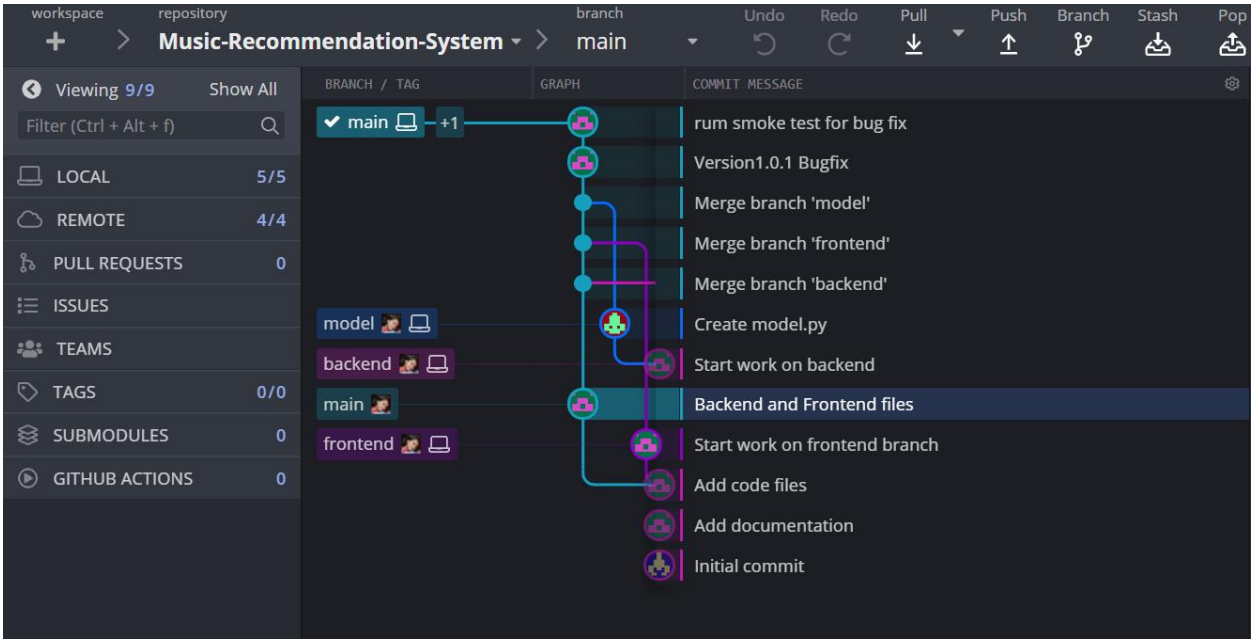




- Promote a branch to the baseline by merging

  Merging into main branch

```
Chairman APGB@DESKTOP-B1LK69A MINGW64 /d/Documents/PESU/SE/PROJECT/F_project/Music-Recommendation-System (master)
$ git branch -m version1.0.1

Chairman APGB@DESKTOP-B1LK69A MINGW64 /d/Documents/PESU/SE/PROJECT/F_project/Music-Recommendation-System (version1.0.1)
$ git add module.py
fatal: pathspec 'module.py' did not match any files

Chairman APGB@DESKTOP-B1LK69A MINGW64 /d/Documents/PESU/SE/PROJECT/F_project/Music-Recommendation-System (version1.0.1)
$ touch module.py

Chairman APGB@DESKTOP-B1LK69A MINGW64 /d/Documents/PESU/SE/PROJECT/F_project/Music-Recommendation-System (version1.0.1)
$ git add module.py

Chairman APGB@DESKTOP-B1LK69A MINGW64 /d/Documents/PESU/SE/PROJECT/F_project/Music-Recommendation-System (version1.0.1)
$ git commit "rum smoke test for bug fix"
error: pathspec 'rum smoke test for bug fix' did not match any file(s) known to git

Chairman APGB@DESKTOP-B1LK69A MINGW64 /d/Documents/PESU/SE/PROJECT/F_project/Music-Recommendation-System (version1.0.1)
$ git commit -m "rum smoke test for bug fix"
[version1.0.1 3b680da] rum smoke test for bug fix
 1 file changed, 0 insertions(+), 0 deletions(-)
 create mode 100644 module.py

Chairman APGB@DESKTOP-B1LK69A MINGW64 /d/Documents/PESU/SE/PROJECT/F_project/Music-Recommendation-System (version1.0.1)
$ git switch main
Switched to branch 'main'

Chairman APGB@DESKTOP-B1LK69A MINGW64 /d/Documents/PESU/SE/PROJECT/F_project/Music-Recommendation-System (main)
$ git merge version1.0.1
Updating 2883fdc..3b680da
Fast-forward
 backend.py  | 1 +
 frontend.py | 1 +
 model.py    | 1 +
 module.py   | 0
 4 files changed, 3 insertions(+)
 create mode 100644 module.py

Chairman APGB@DESKTOP-B1LK69A MINGW64 /d/Documents/PESU/SE/PROJECT/F_project/Music-Recommendation-System (main)
$
```


workspace
repository
Music-Recommendation-System
branch
main
Undo
Redo
Pull
Push
Branch
Stash
Pop
Viewing 9/9
Show All
Filter (Ctrl + Alt + f)
LOCAL 5/5
REMOTE 4/4
PULL REQUESTS 0
ISSUES
TEAMS
TAGS 0/0
SUBMODULES 0
GITHUB ACTIONS 0
BRANCH / TAG
GRAPH
COMMIT MESSAGE
main +1
model
backend
main
frontend
rum smoke test for bug fix
Version1.0.1 Bugfix
Merge branch 'model'
Merge branch 'frontend'
Merge branch 'backend'
Create model.py
Start work on backend
Backend and Frontend files
Start work on frontend branch
Add code files
Add documentation
Initial commit

Problem Statement – 4:

Let's assume you have received funding to launch your project as a start-up. Being at the nascent stage of development processes, you have been tagged under the "Initial" maturity level. Your task is to brainstorm and come up with at least 2-3 new functionality or ways to improve the quality of your project and attain higher levels of maturity according to the CMM model.

ANSWER

Documentation is a way for engineers and programmers to describe their product and process they used in
creating it, in a formal writing. It includes the technical manuals, online versions of manuals etc.
Start-ups are vulnerable to legal problems related to many different fields, such as protecting their intellectual property, having the correct terms and conditions etc. To avoid such mishaps to occur, documentation is crucial for our project as it will :

1. Reduce liability and mitigate risks from unforseen events
2. Resolve disputes
3. Outline the details of wealth distribution
4. Clearly define the specific roles of our partners, employees, advisors, Board of Directors and other company stakeholders
5. Delineate the standard operating procedures and processes for the corporation for efficient and optimal business execution
6. Define the business objectives and execution strategy.
7. Improve customer education
8. Improve operations - Producing informative documentation internally will allow us to delegate more effectively.
9. Increase the collective knowledge of everyone that you work with.

Implementing Scrum will help our team to work together. It'll encourage us to learn through experiences, self-organize while working on a problem and reflect on the wins and losses to continuously improve.

- It helps to structure around the learning process.

- In managing the changing requirements of the projects. It'll help in building the project faster and efficiently while increasing the productivity of each member.
- With the sprint-based model, we can receive feedback from stakeholders after each sprint, if there are any problems or changes, the scrum team of the company can easily and quickly adjust product goals during future sprints to provide more valuable iterations.
- The quality of the product improves drastically with lower costs
- This way, the customers/stakeholders are satisfied by the product which results in the employee's satisfaction.

Project Management tools like Jira can be used to plan and manage project.
It will also help to keep track of company resources. Proper scheduling will improve productivity and efficiency which would in turn increase the overall output of the company in the long run. It can also be used to keep track of, schedule and resolve bugs or issues that come up. Upcoming features can be planned along with deadlines for deployment. Project management will also ensure better use of company resources by providing estimates and keeping an account of resources already used or in use. For example, potential features such as like/unlike a song can be added to Jira and can be planned for later implementation.

Version control softwares like Git can be used to keep track of changes. It will ensure that each developer has access to the project while maintaining different versions. Different versions of the project can be stored using branches. Each team can work on a feature independent of other teams. It will also enable developers to backup code. It also allows developers to see exactly what was changed and when. Version control makes collaboration between developers and teams easier. For example, developer A might be working on a bug while developer B might want to add new songs to the database and re-train the model. A and B would create a branch and work on their respective branches. Once done, they

would merge the branch with modified code to the main branch. This way B does not have to wait for A to finish.

Software testing identifies issues or defects in the source code so they can be fixed before deployment. Test cases can be developed so that project requirements are met and the product works without issues. Quality testing ensures that a superior product in delivered.

Perfomance metrics can be established to track growth and progress of the product. Metrics can be based on no. of users, size, complexity etc.

Security of product can be improved to make sure that user information cannot be accessed by third parties without proper authorization. This would ensure that details like user's spotify account info is not made vulnerable.