# Experiment 8

**Aim:** Change specification and use any SCM Tool to make different versions

**Theory:**

Software configuration management: The traditional software configuration management (SCM) process is looked upon by practitioners as the best solution to handling changes in software projects. It identifies the functional and physical attributes of software at various points in time, and performs systematic control of changes to the identified attributes for the purpose of maintaining software integrity and traceability throughout the software development life cycle.

The SCM process further defines the need to trace changes, and the ability to verify that the final delivered software has all of the planned enhancements that are supposed to be included in the release. It identifies four procedures that must be defined for each software project to ensure that a sound SCM process is implemented. They are:

1. Configuration identification
2. Configuration control
3. Configuration status accounting
4. Configuration audits

These terms and definitions change from standard to standard, but are essentially the same.

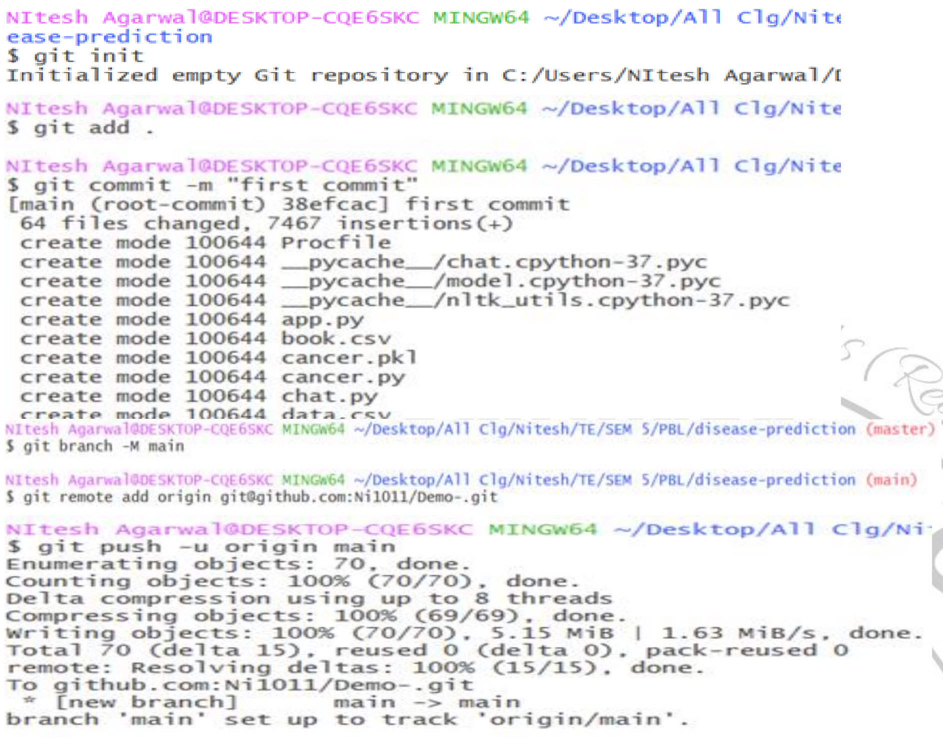
- Configuration identification is the process of identifying the attributes that define every aspect of a configuration item. A configuration item is a product (hardware and/or software) that has an end-user purpose. These attributes are recorded in configuration documentation and baselined. Baselining an attribute forces formal configuration change control processes to be effected in the event that these attributes are changed.
- Configuration change control is a set of processes and approval stages required to change a configuration item's attributes and to re-baseline them.
- Configuration status accounting is the ability to record and report on the configuration baselines associated with each configuration item at any moment of time.
- Configuration audits are broken into functional and physical configuration audits. They occur either at delivery or at the moment of effecting the change. A functional configuration audit ensures that functional and performance attributes of a configuration item are achieved, while a physical configuration audit ensures that a configuration item is installed in accordance with the requirements of its detailed design documentation.

GitHub offers all of the distributed revision control and source code management (SCM) functionality of Git as well as adding its own features. Unlike Git, which is strictly a command-line tool, GitHub provides a Web-based graphical interface and desktop as well as mobile integration. It also provides access control and several collaboration features such as bug tracking, feature requests, task management for every project.

**Implementation:**

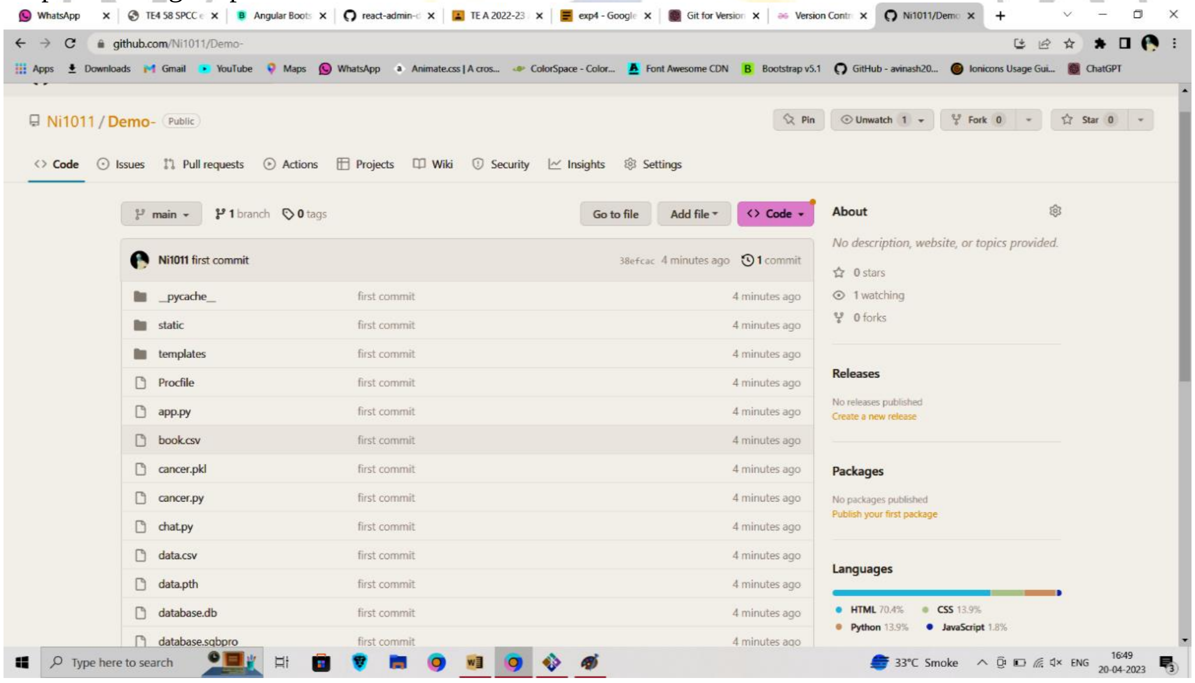
Step 1: Create a New repository in github name Demo
Step 2: Commit the project in github
git init - The git init command creates a new Git repository.
git add README.md – Guide the user
git add . – to add all the changes
git commit -m "first commit" – Commit the changes that has done
git branch -M main – Shift the branch from master to main
git remote add origin git@github.com:Ni1011/Demo-.git - Set origin to remote repo

git push -u origin main – push the change to main branch



```
NItesh Agarwal@DESKTOP-CQE6SKC MINGW64 ~/Desktop/All Clg/Nite
ease-prediction
$ git init
Initialized empty Git repository in C:/Users/NItesh Agarwal/I

NItesh Agarwal@DESKTOP-CQE6SKC MINGW64 ~/Desktop/All Clg/Nite
$ git add .

NItesh Agarwal@DESKTOP-CQE6SKC MINGW64 ~/Desktop/All Clg/Nite
$ git commit -m "first commit"
[main (root-commit) 38efcac] first commit
 64 files changed, 7467 insertions(+)
 create mode 100644 Procfile
 create mode 100644 __pycache__/chat.cpython-37.pyc
 create mode 100644 __pycache__/model.cpython-37.pyc
 create mode 100644 __pycache__/nltk_utils.cpython-37.pyc
 create mode 100644 app.py
 create mode 100644 book.csv
 create mode 100644 cancer.pkl
 create mode 100644 cancer.py
 create mode 100644 chat.py
 create mode 100644 data.csv
NItesh Agarwal@DESKTOP-CQE6SKC MINGW64 ~/Desktop/All Clg/Nitesh/TE/SEM 5/PBL/disease-prediction (master)
$ git branch -M main

NItesh Agarwal@DESKTOP-CQE6SKC MINGW64 ~/Desktop/All Clg/Nitesh/TE/SEM 5/PBL/disease-prediction (main)
$ git remote add origin git@github.com:Ni1011/Demo-.git

NItesh Agarwal@DESKTOP-CQE6SKC MINGW64 ~/Desktop/All Clg/Ni
$ git push -u origin main
Enumerating objects: 70, done.
Counting objects: 100% (70/70), done.
Delta compression using up to 8 threads
Compressing objects: 100% (69/69), done.
Writing objects: 100% (70/70), 5.15 MiB | 1.63 MiB/s, done.
Total 70 (delta 15), reused 0 (delta 0), pack-reused 0
remote: Resolving deltas: 100% (15/15), done.
To github.com:Ni1011/Demo-.git
 * [new branch]      main -> main
branch 'main' set up to track 'origin/main'.
```

Step3: Check git repo



github.com/Ni1011/Demo-

Ni1011 / Demo- Public

main · 1 branch · 0 tags

Ni1011 first commit · 38efcac 4 minutes ago · 1 commit

| | | |
|---|---|---|
| _pycache_ | first commit | 4 minutes ago |
| static | first commit | 4 minutes ago |
| templates | first commit | 4 minutes ago |
| Procfile | first commit | 4 minutes ago |
| app.py | first commit | 4 minutes ago |
| book.csv | first commit | 4 minutes ago |
| cancer.pkl | first commit | 4 minutes ago |
| cancer.py | first commit | 4 minutes ago |
| chat.py | first commit | 4 minutes ago |
| data.csv | first commit | 4 minutes ago |
| data.pth | first commit | 4 minutes ago |
| database.db | first commit | 4 minutes ago |
| database.sqbpro | first commit | 4 minutes ago |

About

No description, website, or topics provided.

0 stars
1 watching
0 forks

Releases

No releases published
Create a new release

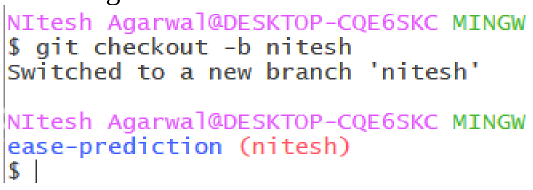
Packages

No packages published
Publish your first package

Languages

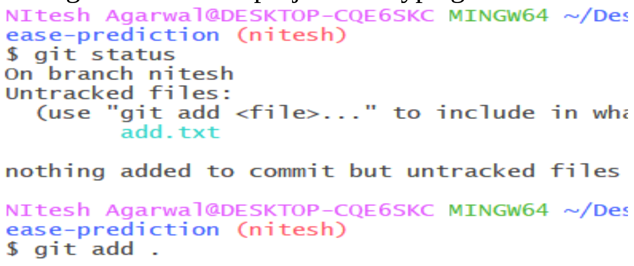
HTML 70.4% · CSS 13.9% · Python 13.9% · JavaScript 1.8%

To Change the Branch:



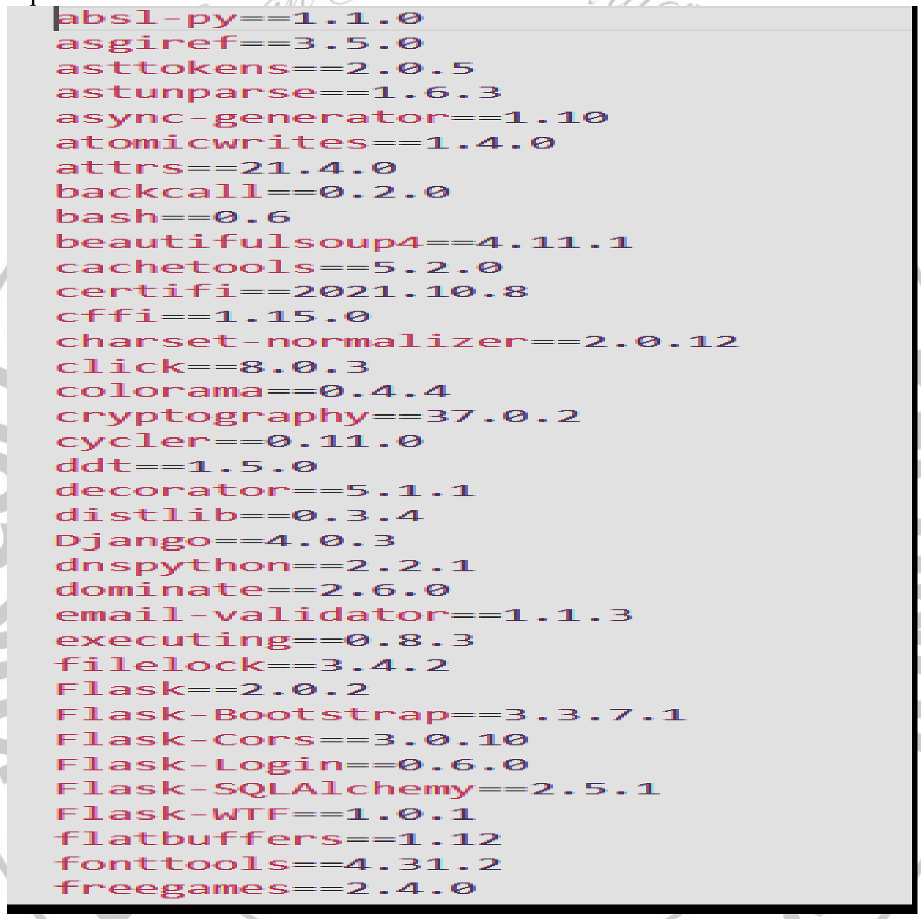
```
NItesh Agarwal@DESKTOP-CQE6SKC MINGW
$ git checkout -b nitesh
Switched to a new branch 'nitesh'

NItesh Agarwal@DESKTOP-CQE6SKC MINGW
ease-prediction (nitesh)
$ |
```

Change contained of project and type git status



```
NItesh Agarwal@DESKTOP-CQE6SKC MINGW64 ~/Des
ease-prediction (nitesh)
$ git status
On branch nitesh
Untracked files:
  (use "git add <file>..." to include in wha
        add.txt

nothing added to commit but untracked files

NItesh Agarwal@DESKTOP-CQE6SKC MINGW64 ~/Des
ease-prediction (nitesh)
$ git add .
```

Version File
$pip freeze requirement.txt



```
absl-py==1.1.0
asgiref==3.5.0
asttokens==2.0.5
astunparse==1.6.3
async-generator==1.10
atomicwrites==1.4.0
attrs==21.4.0
backcall==0.2.0
bash==0.6
beautifulsoup4==4.11.1
cachetools==5.2.0
certifi==2021.10.8
cffi==1.15.0
charset-normalizer==2.0.12
click==8.0.3
colorama==0.4.4
cryptography==37.0.2
cycler==0.11.0
ddt==1.5.0
decorator==5.1.1
distlib==0.3.4
Django==4.0.3
dnspython==2.2.1
dominate==2.6.0
email-validator==1.1.3
executing==0.8.3
filelock==3.4.2
Flask==2.0.2
Flask-Bootstrap==3.3.7.1
Flask-Cors==3.0.10
Flask-Login==0.6.0
Flask-SQLAlchemy==2.5.1
Flask-WTF==1.0.1
flatbuffers==1.12
fonttools==4.31.2
freegames==2.4.0
```

**Conclusion:**
Version control systems like Git allow developers to manage different versions of a project's source code by creating new commits and branching off separate lines of development. This helps with keeping track of changes, collaborating with others, and reverting to previous versions if necessary.

For Faculty Use

| **Correction Parameters** | **Formative Assessment [40%]** | **Timely completion of Practical [ 40%]** | **Attendance / Learning Attitude [20%]** | |
|---|---|---|---|---|
| **Marks Obtained** | | | | |