CHAPTER FIVE

5.1 Change Management (version control using Git)

Version control is essential for managing changes to a project over time
Git is a popular distributed version control system that allows team collaboration and tracking changes
Git provides features like branching, merging, and conflict resolution to facilitate efficient code management

5.2 Steps and Tools Used to Implement Git

To effectively manage version control and collaboration during the development of our Pharmacy Management System, we used **Git** along with **GitHub**. Below are the enhanced steps and tools applied in our project:

1. Install Git

- **Tool Used**: Git
- Downloaded and installed Git on all development machines to enable version control from the command line.

2. Initialize a Git Repository

- **Step 1**: Created a new directory for the project:

```
Mkdir pharmacy-management
Cd pharmacy-management
```

- **Step 2**: Initialized Git in the project directory:

```
Git init
```

- **Purpose**: This sets up the .git folder where Git stores all version control data.

3. Configure Git Settings (Optional but Recommended)

- Configured user information to associate commits:

```
git config --global user.name "Your Name"
git config --global user.email "your.email@example.com"
```

- Checked configuration:

- git config --list

4. Add Project Files to the Repository

- Added existing files to the Git repository:

  ```
  git add .
  ```

  or to add specific files:

  ```
  git add index.html style.css script.js
  ```

5. Commit Changes

- Committed the staged files with a meaningful message:

  ```
  git commit -m "Initial commit: Added main structure of Pharmacy Management System"
  ```

6. Connect to a Remote Repository (GitHub)

- **Tool Used**: GitHub (https://github.com/)
- Created a new repository on GitHub.
- Linked the local repo to GitHub:

  ```
  git remote add origin https://github.com/username/pharmacy-management.git
  git push -u origin master
  ```

7. Ongoing Version Control

- After each change, we used:

  ```
  git add .
  git commit -m "Describe the change"
  git push
  ```

- Used `git status` to check current status and `git log` to view commit history.

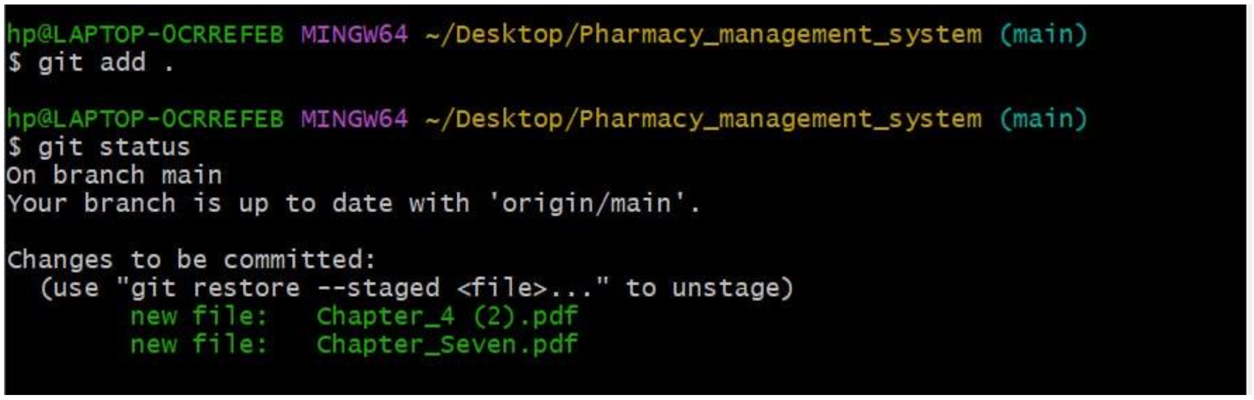
8. Collaboration and Branching

- Created feature branches for team members:

  ```
  git checkout -b feature-branch-name
  ```

- Merged features into main branch after review:
- git merge feature-branch-name

Summary
Using Git and GitHub allowed us to track changes, collaborate effectively, revert to earlier versions, and manage our project in a clean and organized way.

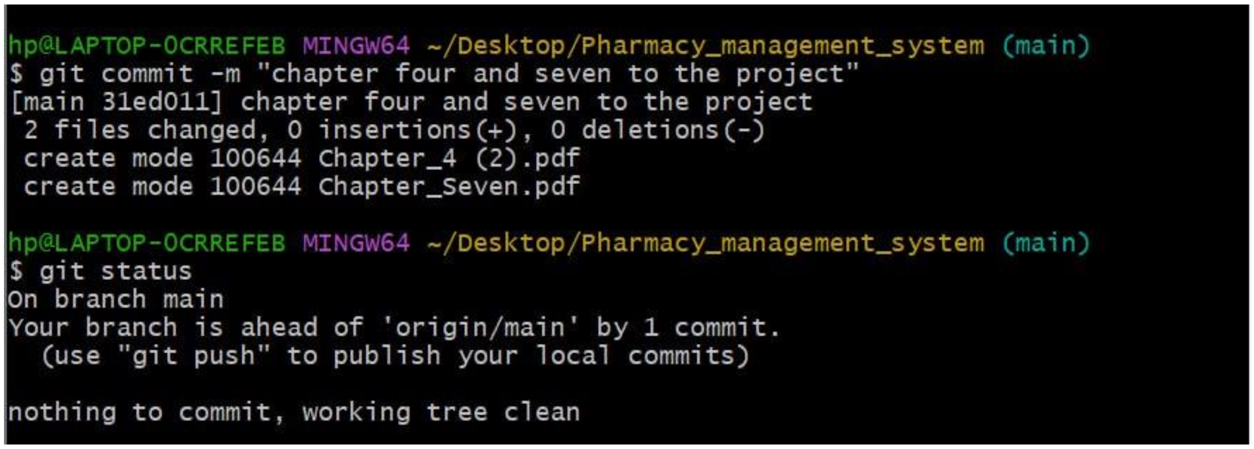


```
hp@LAPTOP-OCRREFEB MINGW64 ~/Desktop/Pharmacy_management_system (main)
$ git add .

hp@LAPTOP-OCRREFEB MINGW64 ~/Desktop/Pharmacy_management_system (main)
$ git status
On branch main
Your branch is up to date with 'origin/main'.

Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
        new file:   Chapter_4 (2).pdf
        new file:   Chapter_Seven.pdf
```

Git add



```
hp@LAPTOP-OCRREFEB MINGW64 ~/Desktop/Pharmacy_management_system (main)
$ git commit -m "chapter four and seven to the project"
[main 31ed011] chapter four and seven to the project
 2 files changed, 0 insertions(+), 0 deletions(-)
 create mode 100644 Chapter_4 (2).pdf
 create mode 100644 Chapter_Seven.pdf

hp@LAPTOP-OCRREFEB MINGW64 ~/Desktop/Pharmacy_management_system (main)
$ git status
On branch main
Your branch is ahead of 'origin/main' by 1 commit.
  (use "git push" to publish your local commits)

nothing to commit, working tree clean
```

Git commit