GitHub and GitHub Desktop Notes

1. Introduction to GitHub

- *What is GitHub?*: A platform for version control and collaborative development using Git, hosting repositories and tracking changes.
- *Key Concepts*:
  - *Repository*: Project folder with files, history, and branches.
  - *Branch*: Parallel version for isolated changes.
  - *Commit*: Snapshot of changes with a message.
  - *Pull Request (PR)*: Proposal to merge changes.
  - *Merge*: Combines changes from branches.
- *Purpose*: Enables collaboration, change tracking, and version management.

2. GitHub Desktop Overview

- *What is GitHub Desktop?*: GUI app for managing Git repositories, simplifying Git commands.
- *Features*:
  - Visual interface for cloning, branching, committing, pushing.
  - Integration with GitHub.com.
  - Displays commit history, file changes, branch status.
- *Setup*:
  1. Download from desktop.github.com.
  2. Sign in with GitHub account.
  3. Configure Git (name, email) for commits.

3. Working with Repositories

- *Cloning*: Copy remote repository locally:
  - Select "Clone a repository from the Internet" in GitHub Desktop.
  - Choose repository, save locally.
- *Creating a Repository*:
  - Initialize locally or on GitHub.com via GitHub Desktop.
  - Add files, set default branch (e.g., `main`), publish.
- *Note*: Edit files in a code editor, view changes in GitHub Desktops left panel.

4. Branching Workflow

- *Why Branch?*: Isolates changes to protect default branch (`main`).
- *Steps in GitHub Desktop*:

1. Click “Current Branch,” select “New Branch.”
2. Name branch (e.g., `feature/update-page`).
3. Make changes in the branch.
4. Switch branches as needed.

- *Best Practice*: Avoid direct commits to `main`; use feature branches.

5. Committing Changes

- *What is a Commit?*: Record of file changes with a descriptive message.
- *Steps in GitHub Desktop*:
  1. Edit files locally.
  2. View changes in left panel.
  3. Write commit message (e.g., “Added navigation bar”).
  4. Click “Commit to [branch].”
- *Best Practice*: Use clear, concise commit messages; commit all changes at once if related.

6. Pushing and Deploying

- *Pushing*: Uploads local commits to GitHub.com.
- *Steps in GitHub Desktop*:
  1. After committing, click “Push origin.”
  2. Create pull request on GitHub.com to merge into `main`.
- *Deployment*: Pushing to `main` may trigger CI/CD (e.g., GitHub Actions) for website deployment.
- *Best Practice*: Push from feature branches to avoid direct `main` changes.

7. Pull Requests and Merging

- *Pull Request Workflow*:
  - Create PR on GitHub.com to merge feature branch into `main`.
  - Review changes, discuss, resolve conflicts.
  - Merge PR after approval.
- *GitHub Desktop Role*: View PR status, open PRs, pull remote changes.
- *Best Practice*: Use PRs for code review.

8. Avoiding Common Pitfalls

- *Document Insight*: Avoid committing to `main` for safety.
- *Tips*:
  - Work in feature branches.
  - Pull remote changes before pushing.
  - Use descriptive commit messages and PR titles.
- *Conflict Resolution*: Resolve conflicts in code editor, recommit via GitHub Desktop.

9. Additional GitHub Features

- *Issues*: Track bugs or tasks.
- *GitHub Actions*: Automate workflows (e.g., testing, deployment).
- *Collaborators*: Invite team members to contribute.

10. Summary

- *GitHub*: Platform for version control, collaboration, project management.
- *GitHub Desktop*: Simplifies Git with visual interface for cloning, branching, committing, pushing.
- *Workflow*: Clone repository, create feature branch, commit, push, create PR, merge to `main`, deploy.
- *Best Practices*: Avoid `main` commits, use descriptive messages, leverage PRs.