Experiment 07

Aim: To Understand workflow of Github For Contanize.

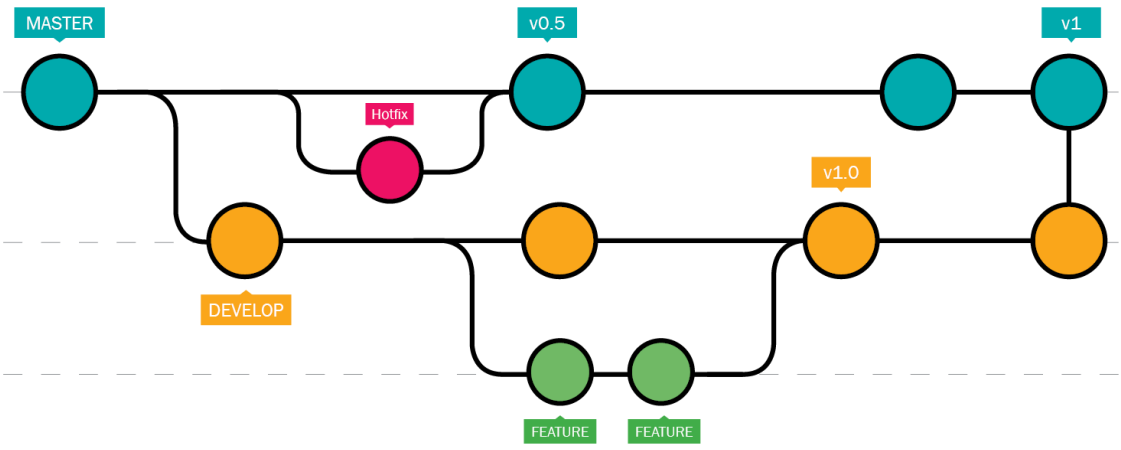
Tools: Git, Github

Theory:

GitHub is a web-based platform that uses Git, a version control system, to help developers manage and collaborate on code projects. Here are the main reasons it's widely used:

1. **Version Control**: GitHub allows developers to track changes in their code over time, making it easy to revert to previous versions if necessary.
2. **Collaboration**: Multiple developers can work on the same project simultaneously. GitHub facilitates collaboration through features like pull requests and code reviews.
3. **Open Source**: Many projects on GitHub are open source, allowing anyone to contribute to them, learn from them, or use the code in their own projects.
4. **Project Management**: GitHub provides tools for managing tasks, bugs, and feature requests, helping teams stay organized.
5. **Community**: It fosters a strong developer community where users can share ideas, get feedback, and collaborate on projects.

Overall, GitHub is essential for modern software development, streamlining workflows and improving code quality through collaboration.



Git Commands:

1. git add

- **Purpose**: Stages changes in your working directory, preparing them for the next commit.
- **Usage**: You can add specific files or all modified files.

Stage a specific file: **git add filename.txt**
Stage all modified files: **git add .**

2. git commit

- **Purpose**: Saves your staged changes to the repository with a descriptive message.
- **Usage**: Always include a message to describe the changes made.

git commit -m "Fix bug in user authentication"

3. git push

- **Purpose**: Uploads your local commits to a remote repository, such as GitHub, making them accessible to others.
- **Usage**: Typically used after **`git commit`.**

git push origin main

4. Release Tag

- **Purpose**: Tags are used to mark specific points in your commit history, often to denote releases (like version 1.0.0).
- **Usage**: Create a tag to identify a particular commit.

git tag v1.0.0

This creates a tag named `v1.0.0`. To push this tag to the remote repository, use:

git push origin v1.0.0

These commands are fundamental for managing code changes effectively in Git and GitHub!

Output:

```
$ git add .
```

```
$ git commit -m "1st commit"
[master (root-commit) f131bee] 1st commit
 1 file changed, 0 insertions(+), 0 deletions(-)
 create mode 100644 alpha.html
```

```
$ git push -u crio master
Enumerating objects: 3, done.
Counting objects: 100% (3/3), done.
Writing objects: 100% (3/3), 209 bytes | 104.00 KiB/s, done.
Total 3 (delta 0), reused 0 (delta 0), pack-reused 0
remote:
remote: Create a pull request for 'master' on GitHub by visiting:
remote:      https://github.com/akashadr/Crio/pull/new/master
remote:
To https://github.com/akashadr/Crio.git
 * [new branch]      master -> master
Branch 'master' set up to track remote branch 'master' from 'crio'.
```

```
$ git tag v0.1
fatal: tag 'v0.1' already exists
```

```
$ git tag -f v0.1
Updated tag 'v0.1' (was 15f7bf9)
```

Conclusion: Implementing the `git add`, `git commit`, `git push`, and release tag commands in the Travel Management System project is vital for an organized workflow.

1. **`git add .`**: Stages all changes, ensuring every update related to booking and user management is included.
2. **`git commit -m "Message"`**: Records changes with clear messages, enhancing collaboration and tracking project evolution.
3. **`git push origin main`**: Shares commits with the remote repository, keeping the team synchronized and up-to-date.
4. **Release Tags (`git tag vX.X.X`)**: Marks important milestones for easy reference and version control.

NAME:

ROLL NO.:

For Faculty use

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