Plan of Action for Cleaning up Dev Branch

Background:

* The repository has two branches we need to merge, the main branch and the develop branch.
* The main branch is validated and the develop branch is not.
* The develop branch is ahead of main by 136 commits.
* The commits are pushed and the develop branch is using GitHub Actions to build infrastructure, so it cannot be deleted.

Goals:

* Remove the 136 commits from the develop branch, while keeping them somewhere else.
* Ensure that the develop branch is equal to the main branch.
* Add new code to the develop branch.
* Add existing code form the Yash team.

Steps:

1. Make a backup of your current repository state. This can be done by pushing your current branches to a remote repository, or by making a local copy of the repository. This step is important to ensure that you have a way to recover your code in case something goes wrong during the process.
2. Checkout your main branch using the command git checkout main. Make sure that your main branch is up-to-date with the latest commit by using the command git pull.
3. Create a new branch off of your develop branch, this will be a temporary branch where you will store the commits that you want to remove. You can create a new branch using the command **git branch temp**
4. Checkout the new branch you just created using the command git checkout temp. This will switch you to the new branch, where you can start cherry-picking commits.
5. To cherry-pick all commits from develop, use the command **git cherry-pick <starting-commit-hash>..<ending-commit-hash>** or **git cherry-pick <starting-commit-hash>^..<ending-commit-hash>**, depending on whether you want to include the starting commit or not. You can use **git log** to get the range of commits you want to cherry-pick or use **git log --oneline** to see the commits in a more readable format. On the other hand, depending on usage, you can use the command **git cherry-pick <commit-hash>** to copy each commit one by one.
6. Once you've copied all of the commits, you can switch back to the develop branch using the command **git checkout develop**.
7. Once you switched back to develop, you can take either course of action.
   1. To reset the develop branch to the same commit as the main branch, you can use the command **git reset <main-branch-commit-hash>**. You can use **git log** to find the commit hash of the main branch.
      1. Now the develop branch is at the same commit as the main branch, you can start adding new code to it.
   2. To squash the commits into one. Use the command **git rebase -i HEAD~<number-of-commits>** this will open an interactive rebase session and you will be able squash all the commits into one
      1. After squashing the commits, use the command **git reset <main-branch-commit-hash>** to reset the develop branch to the same commit as the main branch.
8. Start adding new code to the develop branch.
9. If you need to access the commits you removed, you can switch to the temporary branch you created in step 3 using the command **git checkout temp** and continue working on them.

Considerations:

* When cherry-picking multiple commits, pay attention to the order of the commits, it can cause conflicts and errors.
* It's important to note that squashing commits will change the commit history, which can make it more difficult to trace bugs and revert changes.
* The temporary branch, where you store the removed commits, will increase the repository size, but you can always delete it if you don't need it anymore.
* Additionally, you can use the command **git log --oneline --graph --all** to get a visual representation of the branches and commits, this way you can see the current stat of the branch and commits.

Additional Resources:

* [Git documentation on cherry-picking](https://git-scm.com/docs/git-cherry-pick)
* [Git documentation on branch management](https://git-scm.com/book/en/v2/Git-Branching-Basic-Branching-and-Merging)