<u>Objective1:</u> Chalk out a Git repository structure in terms of helping in reaching maximum efficiency in terms of ease of development and collaborating.

For the teams that have hundreds of developers, branching and merging can be difficult. Bad merges and late-stage integration suck up developers time, potentially delaying future work or releases.
To reduce the pain for our teams, our branching strategy should aim to following points:

1. Optimise productivity
2. Enable parallel development
3. Allow for a set of planned, structured release
4. Provide a clear promotion path for software changes through production.
5. Support multiple versions of released software and patches.

The point of a branching strategy is to efficiently manage code changes. This workflow will impact both developer and deployment workflows.
Using a **feature branching** strategy allows developers to create a branch for a specific feature or tasks. This branch-per-issue workflow allows developers to work separately.
Example:
One team or person might be working on an intrusive bug fix deep in the code, while another might be creating a new workflow for the end user. Each team can work independently on their assigned task and merge changes back into the main branch (mainline) when they are done.

<u>Pros of feature branching:</u>

Allowing developers to experiment and work independently from the core product can keep our codebase stable. This strategy gives our teams the freedom to innovate, plus I can implement workflows for CI/CD.
It also helps developers daily segment their work. Instead of tracking an entire release, they can focus on small sets of changes. Also feature branches can de divided up according to specific groups. Each team or sub team could maintain their own branch for development. Then when they are finished, changes can be tested and reviewed before integrating them together.
NOTE: When using the feature branching strategy for a large enterprise/codebase, it is important to integrate changes across teams frequently.

<u>Cons of feature branching:</u>

The purpose of feature branch is for the branch to live as long as the feature is being deployed. This is where lot of teams struggle. Long-lived feature branches are a nightmare to merge. Because in an effort to avoid conflicts, developers may work in isolation for an extended period of time.

NOTE: Feature branching only works if developers or teams branch and merge often.

Hence, we should also integrate feature branch early in our development cycle. Then we can test and deploy more frequently to ensure there are no issues between branches.
Testing production level code helps ensure that code won't be introduced before it is ready.