**Version Control Processes**

The code base is centrally maintained on GitHub at <https://github.com/LukeRenton/CampusSafetyApp>. We adhere to best practices to ensure seamless collaboration and maintain code quality.

**Local Repositories**

Each developer will have a local clone of the repository on their development machine. To maintain consistency and avoid conflicts, developers are expected to:

* **Pull the Latest Version Regularly:** Always pull the latest changes before starting work to ensure your local copy is up-to-date.
* **Push Changes Promptly:** Push your changes to the remote repository in a timely manner to keep the codebase synchronized.
* **Use Branches for Feature Development:** Create separate branches for different features or experiments to keep the main branch stable.
* **Commit Frequently:** Make small, frequent commits with clear, descriptive messages to avoid large, difficult-to-manage changes.
* **Avoid Stale Branches:** Pull and commit often to prevent branches from becoming outdated.
* **Handle Merge Conflicts Collaboratively:** If conflicts arise, work together with other team members to resolve them efficiently.
* **Communicate Environment Differences:** Notify the team of any differences between your development environment and the production environment, especially before making changes to deployment configuration files.
* **Backup Regularly:** Regularly backup your local work to avoid losing progress, especially before significant updates or merges.

**Using Branches**

* **Branch Naming Convention:** Branch names should follow this format: <main-idea>/<description-of-idea>  
  *Example:* bug/fixing-login-failed
* **Pull Requests and Code Reviews:**
  + All pull requests to merge a branch into the master branch must be reviewed and approved by at least two other team members.
* **Prevent Stale Branches:**
  + Make frequent, small commits and regularly merge with the main branch to prevent branches from becoming stale.
  + Periodically review active branches to determine if they are still relevant. Stale branches should be merged or deleted with the agreement of both the Front End Lead and Backend Lead.
* **Feature Branch Cleanup:**
  + After a feature has been successfully merged into the master branch and deployed, the corresponding feature branch should be deleted to keep the repository clean.

**Communication and Collaboration**

Developers should maintain clear and consistent communication to ensure that branches are actively maintained, relevant, and merged in a timely manner. Effective collaboration is key to the smooth operation of our version control processes.

* **Document Major Changes:** When introducing significant changes, document them in the repository's README or another appropriate file, ensuring all team members are aware of the updates.
* **Establish Clear Deadlines:** For critical features or hotfixes, set clear deadlines for branch completion and merging to ensure timely delivery.