Guidelines on Collaboration Project by Final Year CMPN on GitHub

1. Team Structure and Size

- **Group Size:** Each project group should consist of minimum 3 to maximum 4 members.
- **Group members:** It is recommended that all members of the group should be of the same batch. Exceptionally, groups across batches can be allowed but restricted to same division.
- **Roles:** Assign clear roles within the team to ensure effective division of responsibilities.
- **Team git account:** One student of the group will create a GitHub account using their vit official email-id and share login credentials of GitHub account with the all faculty in-charges (email your group login credentials to sanjeev.dwivedi@vit.edu.in, pankaj.vanwari@vit.edu.in , suja.jayachandran@vit.edu.in , swapnil.sonawane@vit.edu.in , snehal.andhare@vit.edu.in , umesh.kulkarni@vit.edu.in).

2. Problem Statement

- **Definition:** Clearly define the problem your project aims to solve. Ensure that it is specific, measurable, and relevant.
- **Scope:** Outline the scope of the problem, including the target audience or user base and the key challenges or needs that the project addresses.
- **Objectives:** Set clear objectives for what the project aims to achieve in relation to the problem statement.

3. Technologies to be Used

- **Technology Stack:** Select technologies that are appropriate for solving the problem. Preferably include all technologies under courses NLP, BDA, ML and Blockchain. Common categories include:
  - **Programming Languages:** (e.g., Python, JavaScript, Java)
  - **Frameworks/Libraries:** (e.g., React, Django, Hadoop, TensorFlow)
  - **Databases:** (e.g., MongoDB, Amazon Dynamo)
  - **Tools:** (e.g., Git for version control, Docker for containerization, Hyperledger/ Etherium for security control)
- **Justification:** Justify the choice of technologies based on their advantages, compatibility with project requirements, and team expertise.

4. Methodology

- **Approach:** Define the methodology you will use for the project. This could include:
    - **Agile:** An iterative approach with sprints and regular feedback.
    - **Waterfall:** A linear approach with distinct phases (e.g., requirements, design, implementation, testing).
    - **Hybrid:** A combination of Agile and Waterfall methods.
- **Project Phases:** Outline the key phases of the project, such as:
    - **Research and Planning:** Understand the problem and plan the project.
    - **Design:** Create design documents and prototypes.
    - **Implementation:** Develop the project based on design specifications.
    - **Testing:** Conduct testing to ensure functionality and quality.
    - **Deployment:** Deploy the project to a live environment.
    - **Evaluation:** Review the project outcomes and gather feedback.
- **Documentation:** Maintain detailed documentation throughout each phase, including design documents, user guides, and technical specifications.

5. Guidelines for Git account Management

A. Repository Setup

- **Central Repository:** Create and maintain a central repository on GitHub.
- **Access Control:** Ensure appropriate permissions for all team members and faculty in-charges.

B. Branching Strategy

- **Branch Naming:** Use descriptive names for branches (e.g., feature/user-authentication).
- **Branch Creation:** Create branches for features, bug fixes, and other significant changes. Merge them into main only after review.

C. Commit and PR Rules

- **Commit Frequency:** Commit frequently with small, manageable changes.
- **Commit Messages:** Use clear, descriptive commit messages.
- **Pull Requests (PRs):** Submit PRs for merging branches into main and ensure each PR includes a description and relevant issue links.

D. Code Reviews

- **Review Process:** Require at least one review for each PR.
- **Feedback:** Provide constructive and actionable feedback.

E. Communication and Meetings

- **Regular Meetings:** Hold weekly meetings with faculty members in lab sessions to discuss progress, issues, and next steps.
- **Status Updates:** Use project management tools or team chat for regular updates.

F. Security and Privacy

- **Sensitive Information:** Avoid committing sensitive data. Use secure methods for handling secrets.
- **Access Management:** Regularly review repository permissions.

G. Backup, Recovery and Support

- **Backup:** Regularly back up the repository and related data.

- **Recovery Plan:** Develop and test a recovery plan for data loss or repository issues.
- **Support:** Offer support for Git-related issues and questions.

6. Deliverables

- **Initial Deliverables:** Define what needs to be submitted at the beginning of the project, such as:
  - **Project Proposal:** A document outlining the problem statement, objectives, technology stack, and project plan.
  - **Project Plan:** A detailed plan with milestones, timelines, and task assignments.
- **Intermediate Deliverables:** Submit periodic updates, including:
  - **Design Documents:** Detailed designs, prototypes, or mockups.
  - **Progress Reports:** Regular updates on the status of the project, including completed tasks and any issues encountered.
- **Final Deliverables:** At the end of the project, provide:
  - **Final Report:** A comprehensive report covering the problem statement, methodology, implementation details, testing results, and project evaluation.
  - **Source Code:** The complete source code of the project organized and documented.
  - **Presentation:** A presentation summarizing the project, its objectives, outcomes, and key learnings.

- **Demo:** A working demo or deployment of the project for demonstration purposes.

These guidelines will help ensure your collaborative project is well-organized, effectively managed, and successfully executed.