Lab 1: Git Flow, Merge, and Release

Objective

- Understand and implement Git Flow, including feature, release, and hotfix branches.
- Resolve merge conflicts during pull requests.
- Simulate software versioning and release management.
- Write simple Java programs as feature implementations.

Logistics

- **Worth**: 2.5% of whole course.
- **Due Date**: Week 2 (January 17th, 2024).
- **TA Supervised**: During the tutorial session.
- **Penalty**: 5% deduction for unexcused absences.
- **Group Work**: 4 members per group, one Scrum Master.

Instructions

1. Initial Repository Setup

1. Scrum Master forks the provided repository and adds group members as contributors.
2. Create the following branches from `develop`:

- `feat/Feature1`: For the first feature.
- `feat/Feature2`: For the second feature.
- `release/1.0`: For preparing the release.
- `hotfix/1.0.1`: For fixing a post-release bug.

---

2. Detailed Task Assignments

Person 1 (`feat/Feature1`)

- Implement **User Registration**:
  - Java Class: `UserRegistration.java` validates username (5–15 characters) and password (minimum 8 characters).

- Push the branch and create a pull request to merge it into `develop`.

- Introduce a potential conflict by modifying `config.properties`.

Person 2 (`feat/Feature2`)

- Implement **User Login**:
  - Java Class: `UserLogin.java` verifies credentials against a predefined username-password map.
  - Push the branch and create a pull request to merge it into `develop`.

Person 3 (`release/1.0`)

- Create `release/1.0` from `develop`.
- Add a `RELEASE_NOTES.md` file summarizing the features and bug fixes.
- Tag the release as `v1.0.0` and merge it into `main` via pull request.

Person 4 (`hotfix/1.0.1`)

- Fix a simulated bug in the `UserRegistration` feature (e.g., username validation logic).
- Create `hotfix/1.0.1`, apply the fix, and merge it into both `main` and `develop` via pull requests.

---

3. Simulating Merge Conflicts

- Modify the shared file `config.properties` in both `feat/Feature1` and `feat/Feature2` to simulate a conflict.
- Resolve the conflict during the merge into `develop`.

---

4. Git Workflow Diagram

- Include the following in the diagram:
  - Initial `main` and `develop` branches.
  - Creation and merging of `feat/Feature1`, `feat/Feature2`, `release/1.0`, and `hotfix/1.0.1`.
  - Label each commit and node appropriately.

---

5. Deliverables

1. Forked GitHub repository link.
2. A `deliverables/` directory containing:

- `GitWorkflowDiagram.png`: A graphical representation of the Git Flow.
- `Tasks.md`: Team members' task assignments and contributions.
- `RELEASE_NOTES.md`: A summary of version `1.0` features and fixes.

Evaluation Criteria

1. **Git Workflow Diagram (0.5 points)**:

- Accurate representation of branches, merges, and commits.
- Deduct 0.1 points for each major error.

2. **Feature Implementation (0.6 points)**:

- Code in `UserRegistration.java` and `UserLogin.java` must be functional and follow specifications.

3. **Merge Conflict Resolution (1.2 points)**:

- Properly handle the conflict during the merge process.

4. **Release and Hotfix (1.2 points)**:

- Correctly tag and merge `release/1.0` into `main`.
- Apply and merge `hotfix/1.0.1` into both `main` and `develop`.

5. **Task Assignment and Contribution (0.5 points)**:

- Submit a complete `Tasks.md`.
