Our project's post-mortem revealed both hurdles and successes. A major challenge was the rush to meet deadlines, leading to disorganized GitLab management and poor task handling. This rush negatively affected our ability to track progress and maintain control over our workload.
Addressing these issues, we decided to improve our time management and project organization. We began working on new iterations immediately, aiming for a steadier workflow and better planning. This change allowed us more time to refine our GitLab tasks and keep our repository updated, reflecting our actual progress.
Despite these obstacles, our team showed good unity and communication, especially during intense work periods. These strengths were crucial in overcoming the difficulties we encountered.
An additional improvement was in our branching strategy within GitLab. Previously, our branching was inconsistent, hindering code management and collaboration. By adopting a structured branching strategy, designating specific branches for features and releases, we enhanced our development process. This approach enabled more efficient teamwork, clearer task separation, and a smoother integration and testing workflow, leading to a more organized and manageable repository.

What did you learn about team or large project development? What will you start doing, keep doing, or stop doing next time?

From the project experience, it became clear that understanding and implementing git version control, along with mastering remote repository functionality, are crucial skills for effective team collaboration in large project development. The exploration into decoupling code and adherence to SOLID principles highlighted the importance of creating maintainable and scalable software. It also underscored the significance of time management in ensuring project milestones are met within deadlines.
Moving forward, there's a commitment to continue employing the effective programming paradigms learned throughout the project. However, there's an acknowledgment of the need for flexibility in applying SOLID principles and strategies for decoupling code. While these practices are valuable for producing high-quality code, adhering to them rigidly can sometimes lead to inefficiencies. Therefore, in future projects, there will be a more balanced approach, applying these principles on a case-by-case basis to optimize both code quality and development time. This nuanced strategy aims to maintain code excellence without compromising on productivity and project timelines.

Can you draw any conclusions from what you’ve done?

Reflecting on the project, it's clear that software development is more challenging than initially thought. The key lesson is that there's a big difference between simply writing code that works and crafting code that's both high-quality and manageable. This realization highlights the complexity of software engineering, emphasizing the need for best practices like using version control with Git, writing clean code, SOLID, and organized project management.

What would you do differently, if you had the chance to start over?

If given the opportunity to start over, I would focus on improving my skills with graphical user interfaces (GUIs) and GUI builders. This would enable me to enhance the welcome screen of our project, which currently remains quite basic due to my limited knowledge in customizing GUI elements like list views and layouts. Learning these skills would allow for more sophisticated design adjustments beyond simple changes to background images and text placement, ultimately elevating the user experience.