Postmortem is a document written by the developers describing **what went wrong** and **what went right** during the development. It is usually done after the release of the project. The goal of the postmortem is to reflect on the actions taken so that the team can improve its process for the next project (in the agile world, a sprint review serves a similar purpose). Postmortems are also useful to share knowledge with other developers.

Postmortem Title

The title should be related to the content. Be creative.

Introduction

For this project we delved into a condo management app where one can sign up as a condo owner or a condo user. This being our first sprint we decided to focus on implementing key functionalities such as login, sign-up, and user profile management. As a team, we aimed to establish a solid foundation for the project while adhering to agile principles and practices.

An Enterprise Resource Planning (ERP) system, such as ours, encompasses various modules and functionalities to facilitate the efficient management of resources, tasks, and operations within an organization. In our case, the ERP system caters specifically to the needs of condominium management companies and property owners, providing tools for managing property details, financial transactions, user profiles, and more.

Coming into this project the teams expectations were all a little low seeing as it was a hard start and no one knew each other yet. Once having met everything seemed to click into place and it was easy enough to decide on what languages and a teamworking system in order to integrate our project and allow it to flourish. The main languages we decided on using were the Django framework, which is a backend framework that uses python and specializes in organizing everything through an api, and the React framework, which specializes in the frontend and designing aspect of the website allowing us to make the UI in the best way possible.

During this sprint we decided to use a variety of tools including Django and React as well as MongoDB as a database network. Despite encountering some challenges and setbacks during using those tools, and trying to get everything to work properly, we emerged with valuable insights and learned a lot. The final results of this sprint marks the beginning of a successful project, with the successful implementation of core functionalities paving the way for future development and refinement.

What went wrong

For each point, describe one aspect that **did not work well** during the development. It could be technical aspects, management of the team, tools, etc. If possible, use an image that is related to the issue. For example:

- Why did it happen?
- What was the impact?
- How did you address it?
- What do you think that could have been done better?

1 - New Frameworks

During Sprint #1, some team members faced challenges due to their unfamiliarity with Django or React frameworks. This lack of prior experience slowed down development progress and introduced difficulties in understanding and implementing certain features. Despite efforts to provide several resources, such as youtube videos or blogs describing their features, the learning curve hindered efficiency and contributed to delays in sprint deliverables.

In hindsight, providing more comprehensive onboarding and training resources for team members new to Django or React frameworks could have expedited the learning process. Additionally, assigning mentors or pairing experienced team members with newcomers could have provided valuable guidance and support during the adaptation period.

Moving forward, prioritizing skill development and knowledge sharing in unfamiliar technologies will be essential to improve efficiency and ensure smoother sprint execution in future iterations as well as provide useful knowledge to team members so that they may excel in their future careers as software engineers.

2 - Risk Management Plan and Risk Analysis

During the conception phase of our risk management plan and risk analysis, we ran into a few problems. The main issues were lack of documentation provided for the RMP and Risk Analysis, and the mix of documentation online. Too many resources had the inverse effect, and made it harder to figure out certain things. Specifically, the risk impact matrix's exact format, and the residual risk column in the risk analysis. The impact this had on this deliverable was that it made it harder and a little confusing to complete as well as causing the deliverable to be more time consuming, since we redid it a few times.

We addressed this problem by using the internet to its full potential and making logical coherent decisions when in doubt. What could have been done better, possibly, was run every step by the TA ensuring that we had the right format. Although this is very time consuming and not logistically feasible.

3 - SAD

During the Software Architecture Design, the main struggle we encountered came from the vague instructions and the unfamiliarity of the SAD document. Even though an example was provided to help us understand what we were tasked to do, the instructions given to us were ambiguous. The guidelines were not clear as well. This caused confusion among us and we wasted a lot of time figuring out what was expected to be delivered. To address this issue, we had to get in contact with the TA (Stakeholder) assigned to our team to clarify our doubts. Something that could have been done better would have definitely been giving out more detailed and clear information about the task. Also, a more accurate example could have saved us a lot of time spent on deducing what is supposed to be done.

4 - Running prototype

A short delay in delivering the prototype lead to a little bit of stress at the last stage of the sprint. This also affected other aspects of the project like testing and prototyping. But it was resolved at the end with the help of everyone.

5 - Testing

In terms of testing due to the requirements not being very specific it was quite difficult to know how or what exactly to test and integration, system testing had overlaps which caused us to know exactly how to fully implement these.

This impacted how was we were able to implement test and how much coverage we could actually do. To address it we made sure to at least implement unit testing to test the backend and use Cypress to test the system which would ensure at least that the tests would be making sure the application is working.

If the type of testing and preferred technologies were specified this would have made it much easier on us. Since, test driven development requires the assistance of the client to ensure we are testing the right requirements.

What went right

For each point, describe one aspect that **did work well** during the development. It could be technical aspects, management of the team, tools, etc. If possible, use an image that is related to it. For example:

- Why did it happen?
- What was the impact?
- How did you address it?
- What do you think that could have been done better?

1 - Risk Management Plan and Risk Analysis

Regarding the risk management plan and risk analysis documents, despite running into many difficulties, we were able to create a risk management plan and risk analysis document that our team was satisfied with. We believe to have sufficiently anticipated the risks we would run into, record them with great detail and come up with good solutions or mitigation plans for each respective risks.

Additionally, our team categorized our risks adequately based on our progress, expectations and other factors. Given the scope of the project, our experience levels and expectations for the app, we think that we have satisfactorily categorized our risks. Finally, for the next sprint we can come back to the risk management logs and update the risks and see whether it has been resolved or if any further action needs to be taken.

2 - Slack questions channel

Even though requirements were not clear enough to guide us through the project, the slack question channel really helped to see that everyone else had similar concerns and ideas from the questions and answers were great help to allow us to direct our efforts.

Great idea to have a mutual question channel for all 200 students in addition to the individual team channels

3 - Team collaboration

The team was successfully able to meet on time even though the size made it a challenge with full schedules for everyone. Everyone was able to assign themselves tasks (autonomy) and was able to deliver well and on time.

Another + for the team was that even if someone wasn’t able to do their part easily, they were able to use discord for communication with responsive help from everyone since our skills are complementary

4 - Role Distribution

During our first meeting, we established who would do what and we made sure everyone knew what they would be doing. This improved the team's performance from the start. The members were able to quickly know who to contact for which part of the project which made it simple and straightforward for everyone. The roles were also redistributed once we felt like there was a bit more load in one section of the sprint deliverable/

5 - Concurrent courses

Taking the same topics needed for this sprint in the other classes at the same time as the project is going was a good help to allow us to apply the knowledge we picked up from the

class here. Concordia has a great program sequence where everything is super clear and well organized.

Conclusion

[1-3 praragraphs]
In general, what did you learn? What is the main takeaway the team got from this?
Overall, our team learned how important communication and organization was for the first sprint, since we were able to execute our tasks so efficiently and in a timely fashion. For next sprint, having more familiarity with the technologies that we have will have a greater impact on our performance as a team and allow us to implement more features for the condo management system.

Additionally, key aspects to highlight are for this sprint was how quick decisions need to be made over the sprint or else this can cause small delays and affect performance baddly.