SCROLL TO PAGE 2 FOR THE REPORT

Goals of a Post Mortem

To identify the things we did right, so that we can remember to try them again in similar situations.

To note the things that should have been done differently, so that we can refine our techniques in the future.

- **To note the things that we did wrong, and to suggest alternative approaches or safety measures that we should employ the next time we face a similar problem.**

We must remember that;

- **Everybody tries to do their best, as best they understand it.**
- **We make our decisions in stressed situations, with imperfect information.**
- **We are often called upon to carry out tasks for which we have not been trained, with what-ever tools and resources happened to be at hand.**
- **Mistakes are inevitable in such situations.**
- **The goal of this process is not to find fault with any individual or their actions. Rather it is to look at what happened and see what lessons we can learn from it.**
- **The output of this process will not be an assessment of any person or group of people, but rather an assessment of our processes, and how they can be improved.**

What we should write about:

- **What things turned out well, and why?**
- **What tools, techniques, and processes worked well, and why?**
- **What improvements would we make to them?**
- **What things turned out poorly, and why?**
- **What tools, techniques, and processes worked poorly, and why?**
- **What improvements would we make to them?**
- **What alternatives should we develop?**
- **What unexpected problems came up?**
- **What warning signs should we have noticed?**
- **What decisions could we have reasonably made differently?**
- **What still confuses or concerns us?**
- **How would we assess the success of this project?**
- **How would we assess the success of our tools, techniques, and processes?**
- **What are the few key recommendations we would make to management?**

Post-Mortem Report TOC

1. Project

A. Description
i. Project Name: Booking System
ii. Client: Homy
iii. Project Manager: Spencer Porteous
iv. Solutions Architect: Jarod Wright
v. Start Date: 27/02/2017
vi. Completion Date: 29/05/2017
B. Project Overview [Describe the project in detail.
i. Discuss the project charter
ii. What was the project success criterion?
We can analyse the success of the project by 3 key criteria. The first of the criteria is how efficient the project was. This explores how the use of time and effort spent on the project, how much of the resources were needed to fix design flaws compared to resources used to implement new functionality. The second criteria is the communication between team members and the ability to relay and simultaneously work on the same files at the same time. The last criteria is the deliverable and the quality of it.
iii. Etc.

2. Performance

A. Key Accomplishments [List and describe key project accomplishments. Explain elements that worked well and why. Consider listing them in order of importance. Be specific.]
i. What went right?
ii. What worked well?
iii. What was found to be particularly useful?
iv. Project highlights
B. Key Problem Areas [List problem areas experienced throughout the project. Be specific.]
i. What went wrong?
ii. What project processes didn't work well?
iii. What specific processes caused problems?
iv. What were the effects of key problems areas (i.e. on budget, schedule, etc.)?
v. Technical challenges
C. Risk Management [List project risks that have been mitigated and those that are still outstanding and need to be managed.]
i. Project risks that have been mitigated:
ii. Outstanding project risks that need to be managed:
D. Overall Project Assessment [Score/rank the overall project assessment according to the measures provided. A 10 indicates excellent, whereas a 1 indicates very poor.]
E. Additional Comments:
i. Other general comments about the project, project progress, etc.

3. Key Lessons Learned

A. Lessons Learned [Summarize and describe the key lessons and takeaways from the project. Be sure to include new processes or best practices that may have been developed as a result of this project and to discuss areas that could have been improved, as well as how (i.e. describe the problem and suggested solution for improvement).]
B. Post Project Tasks/Future Considerations [List and describe, in detail, all future considerations and work that needs to be done with respect to the project.]
i. Ongoing development and maintenance considerations
ii. What actions have yet to be completed and who is responsible for them?
iii. Is there anything still outstanding or that will take time to realize? (i.e. in some instances the full project deliverables will not be realized immediately)

Performance

Initially, before the development had began, as a team, we discussed the strengths and weaknesses of each team member and previous experiences working in other group driven projects. This allowed us to then later assign appropriate roles to each team member, including dedicating each member to specific functionality of the software (such as specific classes in java).

What things turned out well, and why?

What worked out well at the start of the project was allowing one team member to act as the system architect and create all the class files and consistent method names. This allowed all of us to know how the program is structured, so that we could focus our time on the method details rather than worrying about how classes and methods interact with each other.

What tools, techniques, and processes worked well, and why?

In order to keep track of the progress and address any issues, we had scheduled meeting during each week which enabled team members to inform the group if they are progressing at the rate they expected and address any problems that needed a group discussion and clarification. This process worked well; due to the consistency of the meetings, decision making was almost always done by consensus which was prudent to the development of the software.

Developing user stories played a significant role on the final product as it was a way to make sure the development was on the right direction and the requirements from the client were fully understood. This proved useful on several occasions, where the client indicated that there was a misunderstanding based on the user stories, which saved development time as the user stories were developed before implementation. This in turn allowed us to then create accurate acceptance tests that insured all the product requirements were met. Using trello as a project management and collaboration tool was a really good way for the team to

visually check the overall development progress and functionality that needed higher priority. This worked well because several functions needed to be implemented first for the other functionality to work. This allowed us to successfully plan the duration of the development and have checkpoints to make sure we are within the expected time frame.

What improvements would we make to them?

One of the key tools we used was github, our repository use was good but we only learned how to properly learned how to use branching towards the end of the project, file organisation and keeping track of what people have been working on would have been easier if it was done at the start of the project.

What things turned out poorly, and why?

Something that turned out poorly was the GUI as it could have been better if we spent more time one it. The GUI suffered as we started the project in console for the first 6 weeks of the project and then switched to a GUI and had 6 weeks to make it perfect, if we started using the GUI in the beginning we would have had another 6 weeks to work on it and make it perfect. We also struggled a lot with testing, we left junit testing till last and it really damaged the backend validation processes and caused a lot of avoidable bugs and problems.

What tools, techniques, and processes worked poorly, and why?

Something that worked poorly was our initial use of MVC pattern as components that belonged in the controller was actually in the view and plenty of other logical inconsistencies. At the start of the project we didn't fully grasp the concept of MVC and when implementing it we created these inconsistencies.

What improvements would we make to them?

As a group we followed little to no coding guidelines rather we each acted as we saw fit within our branches. This ordinarily wouldn't be an issue however as the project progressed, a couple of areas were being neglected such as commenting, javadoc and JUnit testing. These became significant issues as it became more difficult for other developers to diagnose problems in code that someone else had written. This wasn't as big of an issue in Parts A & B as the program was rough and didn't need to be polished however it caused problems in Part C where we all needed to be fixing bugs towards the end in a more QA-centric phase. Next time the group should impose rules on pull requests - to refactor, comment, document and JUnit test before merging it into the shared development branch.

What alternatives should we develop?

Moving from a basic messaging program like facebook messenger to something like slack (https://slack.com/) would have allowed our group to keep track of different parts of the program's development more easily. The way messenger is set up prevents searching for specific discussion. Segmenting discussion into frontend, backend and project discussion

would have made it easier to keep track of all discussion on the parts each group member is working on.

What unexpected problems came up?

We had a lot of issues related to the SQL schema that we never saw coming, mostly related to future parts of the project. If we knew about the Part C requirements from day 1 the sql would have been carefully divided up as to keep database integrity between businesses, and separate databases. Another problem we had was with when converting saved unix timestamps, they would convert 14 hours out of the actual entered time.

What warning signs should we have noticed?

There were two separate warning signs that our architecture wasn't effective. The first was that it was quite difficult to divide up tasks without conflicting with anyone else's class files, this caused a lot of issues throughout the whole assignment where the development branch would occasionally get broken due to merge related issues. The second warning sign emerged when we were starting to add the GUI to the application.

What decisions could we have reasonably made differently?

A much larger focus on the overall architecture and database structures right from the very start would have lessened the amount of development time spent on refactoring major portions of our code. This was exemplified by the large amounts of refactoring we had to do every time a new requirement was added to the product specification.

Developing the GUI right from the start rather than the console interface would have been easier than developing the console UI. It would have also forced us as a group to learn javafx earlier on which would have prevented the period of time where only one person in the group had learned how to use javafx and everyone else had to learn it.

What still confuses or concerns us?

One of the concerns with the final product is that we don't fully know how well it will adopt on a large scale. This is because we didn't have time to do adequate stress tests in order to ensure stable and predictable behaviour from the software under such conditions. This is rather an important requirement for this type of software.

How would we assess the success of this project?

We can analyse the success of the project by 3 key criteria. The first of the criteria is how efficient the project was. This explores how the use of time and effort spent on the project, how much of the resources were needed to fix design flaws compared to resources used to implement new functionality. The second criteria is the communication between team

members and the ability to relay and simultaneously work on the same files at the same time. The last criteria is the deliverable and the quality of it. On a scale of 1 to 10 where 10 is very successful, we would rate this project an 7/10. At the start of the project we didn't encounter many refactoring errors to fix many design flaws but as we went on we started spending more and more resources on trying to fix functionality that was either inconsistent or incomplete, this reduced our resources for new functionality and severely depleting our resource pool for the last couple of weeks. On the second success criteria, we were very active in our communication channel with bugs discovered or clarifying tasks to complete or challenging requirements, this allowed us to code less design flaws and thus removing some redundancy from our code. The deliverable was mostly working by the end of it, there is a couple of non major known shippable bugs in our code but the program works exactly as intended.

How would we assess the success of our tools, techniques, and processes?

We should assess them by their impact on overall process efficiency whether it be robustness of the system or the speed of development. Git was by far the tool that had the largest impact on our productivity and enabled our group as a whole to develop the application reasonably within the given timeframe.

What are the few key recommendations we would make to management?

Tell us the future requirements earlier (I assume this was intended for the assignment) but having to implement multiple businesses requires a lot of refactoring and redesigning the database schema was quite difficult, and brought up a lot of bugs.