Group 1 - Postmortem Analysis
Due: Dec 18th

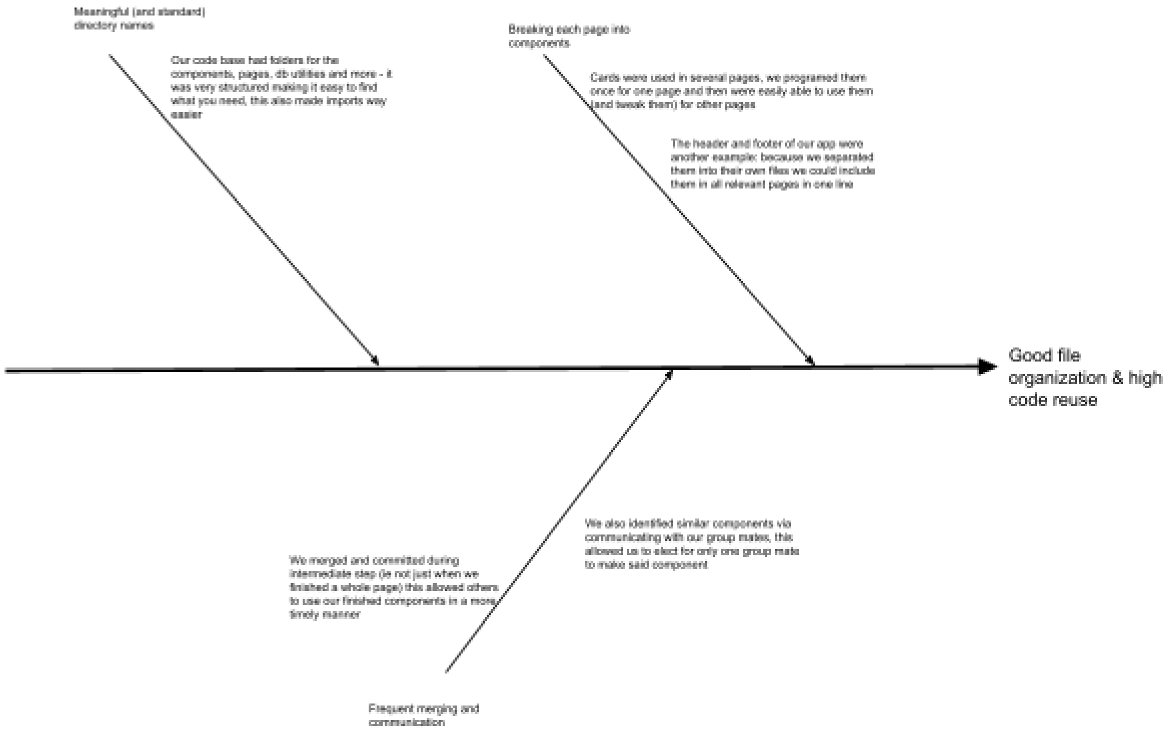
Each member of the group must add one item to each column.
We will then agree upon one from each column to create fishbone diagrams.
Focus on the process, not the people!!

What went well in our process:	**What did not go well in our process:**
1. Completing assignments in a timely manner 2. Following through with our promises to our clients 3. Consistently meeting the minimum viable project. 4. File organization and code reuse (our file system was well structured and enabled assets to be accessed across pages) 5. Communicating with each other for long periods of time, and making sure everyone was on the same page in beginning steps of development.	1. Communication while pair programming 2. Communicating with our client when making changes to solve a problem, is this change something they want? 3. Meeting outside of class to discuss progress. 4. Our app did not handle some changes well (for example we wanted to change the data type of the “finish time field” - this forced us to change code in several places and was a bit of a mess) 5. Splitting work properly. Some developers would have many more tasks than others depending on the week, which was due to having so many members in the same group.

Recommendations for change, how can we make it better?
1. While pair programming, the details of specific tasks should be discussed and distributed. Communication and status updates should be frequent. 2. Communicate more directly with our client in a timely manner (i.e. before releases) to determine if what we are working on is actually something that they want. We were lucky to have made changes that were well liked even if agreed upon afterwards. 3. Establishing a group meeting time early on to check in weekly in some manner, could help with consistent progress discussions. 4. Proactively abstract. I think we could have, at points, tried to predict what could change in the future. For example, changing how we store our data is something that we could have proactively solved. If we had more abstraction in between our data and our app we would have to change less code if we wanted to tweak what data we store. 5. Consistently planning the amount of work that developers would take on. Perhaps being more specific in what we want to implement, and sharing our ideas in the intermediate steps with each other, which would enable everyone to be on the same page and contribute to the same ideas.

Choose one thing that went well and one that didn’t from each list. Create a fishbone diagram for each. (Highlighted)

Fishbone 1: File organization and code reuse (our file system was well structured and enabled assets to be accessed across pages)



Meaningful (and standard) directory names

Our code base had folders for the components, pages, db utilities and more - it was very structured making it easy to find what you need, this also made imports way easier

Breaking each page into components

Cards were used in several pages, we programed them once for one page and then were easily able to use them (and tweak them) for other pages

The header and footer of our app were another example: because we separated them into their own files we could include them in all relevant pages in one line

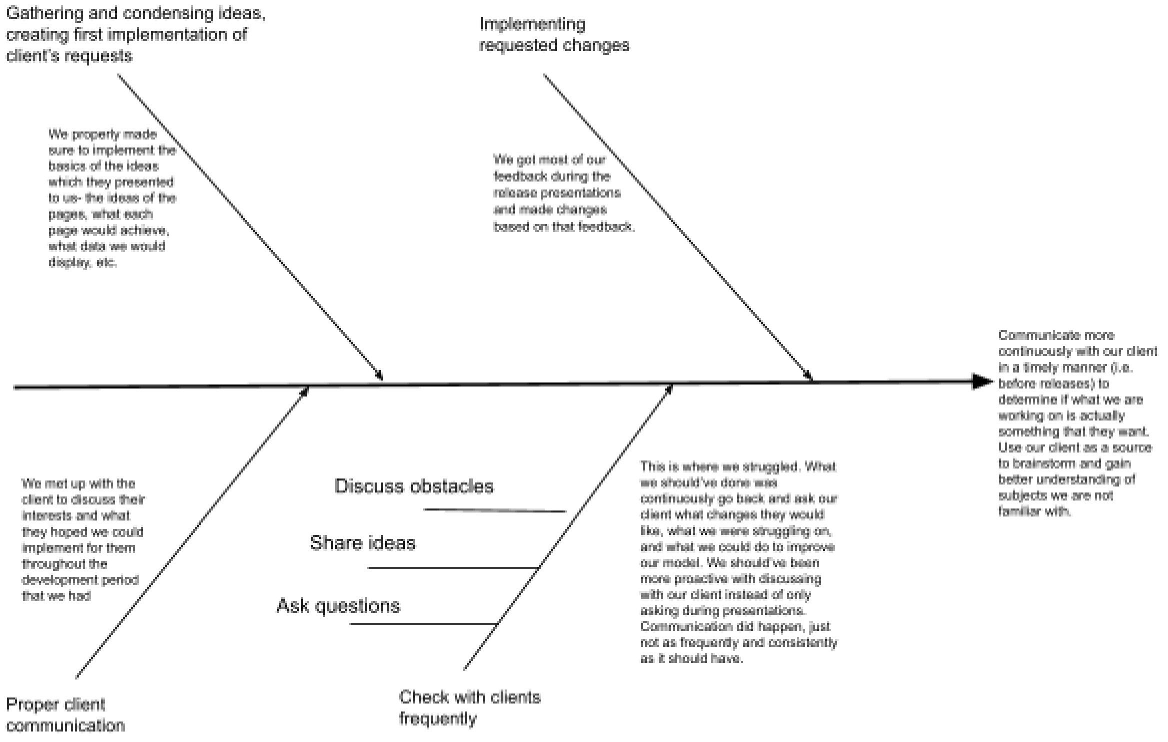
Good file organization & high code reuse

We also identified similar components via communicating with our group mates, this allowed us to elect for only one group mate to make said component

We merged and committed during intermediate step (ie not just when we finished a whole page) this allowed others to use our finished components in a more timely manner

Frequent merging and communication

Fishbone 2: Communicating with our client when making changes to solve a problem, is this change something they want?



Gathering and condensing ideas, creating first implementation of client's requests

We properly made sure to implement the basics of the ideas which they presented to us- the ideas of the pages, what each page would achieve, what data we would display, etc.

Implementing requested changes

We got most of our feedback during the release presentations and made changes based on that feedback.

Communicate more continuously with our client in a timely manner (i.e. before releases) to determine if what we are working on is actually something that they want. Use our client as a source to brainstorm and gain better understanding of subjects we are not familiar with.

We met up with the client to discuss their interests and what they hoped we could implement for them throughout the development period that we had

Proper client communication

Discuss obstacles

Share ideas

Ask questions

Check with clients frequently

This is where we struggled. What we should've done was continuously go back and ask our client what changes they would like, what we were struggling on, and what we could do to improve our model. We should've been more proactive with discussing with our client instead of only asking during presentations. Communication did happen, just not as frequently and consistently as it should have.