RISKS

This document details the potential risks and issues that could arise during the creation of QM-Lab™, as well as some possible solutions. **Please add a risk name in bold all caps, with a description of the risk, as well as a possible solution if applicable.**

RISK NAME: This is a description of the risk. It lists tools/technologies/people etc that the risk concerns. It includes a description of what would happen if the risk occurred, explains why that would be bad, and gives some vague estimate of how likely it is to happen.

- Finally it should describe a possible solution, if possible. Some risks may not have an obvious solution.

REALTIME API TOO HARD TO USE: The Google realtime API is our current weapon of choice, we will be using it to enable interaction across clients in real time. It could be difficult to use, which generally means that although we can make progress with it, it may be slow progress, or it could be error prone, or just very picky about what you try to do with it. This would be bad because it would slow us down, take more time to debug, and take more energy that could have been spent elsewhere.

- A possible solution would be to use some other API such as Operational Transform.

UNDO FUNCTIONALITY STORAGE SPACE: If we implement an undo function for the QM_Lab "drag" feature, assuming the feature updates the location in real-time with the mouse movement, the potential storage size of the "undo" stack could get very large, very quickly. Since Google's Realtime API sets a hard limit of 10MB of collaborative storage per document, any undo functionality we include that involves the "drag" feature could quickly overload the file/document. This would be bad, because Google's Realtime API stops working when the user gets to this limit.

- Possible solutions include storing our own undo stack that only is updated on mouse-up events, but this too could take up Google's storage space if we're not careful.

GOOGLE PRIVACY STORAGE: Certain portions of the Google Realtime API store information using Google's systems.

- Depending on how important privacy is, and where this information is stored (for example, Google has data storage within Ireland), we may need to disclaim that our information does not meet those requirements, or perhaps this is already included with Google's account setup.

TEAM LEAD DROP: We have some members of the team that are leads for certain important portions of the development work that we do. One of the team leads could drop or change his/her mind about being that lead, which would leave us without someone to manage and facilitate that particular area. This would be bad because that person probably had detailed knowledge of the progress of the team and what its plans were.

- An obvious solution is to have one of team members take over as lead for that team.

DEPENDENCE ON REALTIME API: We are super dependent upon the realtime api at this point. It seems like a little sideproject of a small team at Google. What if they decide it isn't worth continuing to maintain? We could be scrambling to make an alternative work if Google just decides to shutdown the api. Also since it's a free thing they have literally no obligation to keep it open, provide customer service, etc… Seems fairly unlikely to happen in the next 3 months, but possible down the line. But then it's osgoods problem.

- A partial solution would be to design the software to be as independant as possible. This could be done by using generic methods for updating, creating, etc in the data model. Then it would be easier to replace just the code in those standard methods if we would have to transition to node or some other platform.

UNKNOWN RATE LIMITING: While google states that the only limits on the api are a document size of 10MB. However, they might decide to block our api key if we get like a lot of users (1k+) and they start noticing unusually high amounts of bandwidth/servertime going to this obscure api they published a couple of years ago.

- Contingency procedures would be the same as for DEPENDANCE ON REALTIME API.

HACKERS: Other teams in the past have been compromised, what if we also become compromised? This could destroy our entire project. It could make us look like fools. Maybe osgood would take pity on us and estimate a high grade that we wouldn't have actually gotten?

- The solution: Do not get hacked. Keep back-ups. Be extra secure.

FIND NEW BETTER TOOL: We could find a new tool that is way more useful and effective than what we're currently using, that invalidates our current work requiring a restart.

- By this point we probably shouldn't switch.

JOINTJS HIGH LEVEL : Abstraction provided by joint.js could prevent easy creation of low level functionality, especially with Google Realtime API.

- Theoretically we could implement it ourselves because joint.js is open-source. If we tried really hard we could probably figure out a hack-y way of doing it.

JOINTJS INTERACTION WITH GOOGLE REALTIME API: May be unforeseen complications even though base functionality is proven.

- More spike prototypes and back-up tools.

API KNOWLEDGE ISSUES: Only a handful of people are knowledgeable about the API's were using neither API has much documentation that we know about and can easily find.

- Trial and error, use a bunch of functions until we find one that works. Spread the knowledge. Don't be afraid to ask.

COMMUNICATION ISSUES: The team leads may not communicate often enough with each other, and with the teams. This may result in redundant functionality or lack of task completion. For example the design team could create things that the implementation team can't create, because of lack of communication.

- Frequent meetings, scheduled communication, and religious use of communication channels such as Slack and emails.

COMMUNICATION ISSUES: The team leads may not communicate often enough with each other, and with the teams. This may result in redundant functionality or lack of task completion. For example the design team could create things that the implementation team can't create, because of lack of communication.

- Frequent meetings, scheduled communication, and religious use of communication channels such as Slack and emails.

LACK OF FOCUS: Lack of structure in meetings, we don't often have much of an agenda. Even if we have one, we may not stick to it. Also we may spend a lot of time just hanging out talking.

- Have a meeting lead, that keeps everyone on track. Use agenda's and plan the meeting topics, and stick to it.

TASK FIXATION: We may spend too much time on trivial things, and just kind of lose track of the big picture, we can get caught down in the weeds. Perfectionism can be an issue.

- Triage by the triage team, or the team leads. Better tracking of the big picture. Structured assignments for people and delegation. People take up a specific feature and really keep track of responsibilities as well as time spent on tasks.

SCHEDULING CONFLICTS: We each have different class schedules, as well as our own responsibilities and things we need to do in our own personal schedules. It's unlikely that the entire team will all happen to have a free spot in their schedule that everyone can meet during.

- Smaller meetings with less people, usage of communication channels like Slack, Skype and email. Clear coordination of meeting times so people can adjust their schedule accordingly.

TEAM MEMBER CHANGES: Members may switch between different parts of the team. Every time someone switches from one team to another, they take all of that knowledge and expertise with them, as well as require being caught up by the new team. Requires a replacement on the old team potentially. Since they're new to the other team, they may do things in a different manner, slowing the team down.

- It's not like the old team can't just ask about things they're wondering about. If someone switches into a new team, be more watchful of their work and make sure to help them get up to speed properly so they're not out of the loop.

TEAM CONFLICT: Members could become stressed and resentment could build up between members or even entire teams (testing vs implementation). Everyone is on a full schedule, and possibly underslept so fights and conflict could arise. Some members may get attached to ideas and this could cause arguments.

- Everyone needs to continually remember to behave professionally and be diplomatic. Frequent communication between members may help resolve misunderstandings. Conflict resolution through project manager, and possibly through the instructor.

BUILD UP OF OTHER RESPONSIBILITIES: For example the internship program, lots of people needed to create cover letters and send in applications and do interviews. Other classes have assignments, projects, and tests that require our attention as well. The perfect storm of responsibilities can build up all at the same time, making it very difficult for the team to work effectively.

- Work harder, and more efficient time allocation. Assign earlier deadlines than the officially supplied ones. Good communication of members responsibilities, so that other members can take over other responsibilities for them.