1 Risk Management Plan

Group 1 Members:

<<Sushant Acharya>>
<<Kole Kenney>>
<<Ozzie Loewen>>
<<Thomas Starr-Timberlake>>
<<Justin Thomas>>

This report is covering the top five potential risks to the Commerce Transaction Web Application project. Provided is the risk exposure for each of the risks, along with a solution and some analyzing of each risk. The definition that goes along with the risk is described in each section along with the breakdown of why it would be a potential risk. Also provided is the estimates and reasoning behind the estimated risk potential. These risks are ordered from biggest potential risk to lowest potential risk based on the risk exposure estimate. However, it is also understood that these risks might not occur in this specified order or other risks may populate. There is an analysis section included at the end giving an overview about handling the situations as they occur during the production of our team's project.

1.1 Lack of Experience

This is by far the biggest risk to the project. While everyone on this team has had previous experience coding and some of us have even had experience working for more individualized clients (smaller businesses or hobbies for people). However, none of us have worked with the materials being specified in the project make up, such as Bootstrap or .Net, and with minimal experience in SQL. This risk is mitigated by the fact that this is still a learning environment with the expectation to learn from the experience and not punished for it.

Going off of the definition of risk exposure detailed in the Risk Management Lecture notes. To begin, the other risks being calculated become higher or they are no longer relevant because the project wouldn't be able to even begin. We would still be able to deliver a prototype of sorts and display a foundation of logic, but it wouldn't be in the fully operational style that the business wants. The probability that this risk will materialize has greatly decreased since the beginning of the production period. The probability that an issue sat higher at the beginning of our first iteration. During that time period the team had to do more research and studying to be able to begin coding. That number has changed since we began work as demonstrated in Table 1.

Table 1.

Iteration	Risk	Probability	Loss	Risk Exposure	Actual Loss
1	Developers understanding of the required framework.	90%	10 weeks	9 weeks	2 weeks
2	Developers understanding of the required framework.	90%	6 weeks	5.4 weeks	0.5 weeks
3	Developers understanding of the required framework.	90%	4 weeks	3.6 weeks	
4	Developers understanding of the required framework.	90%	2 weeks	1.8 weeks	
5	Developers understanding of the required framework.	90%	1 week	0.9 weeks	

The team did lose the majority of the first two weeks to be working on the coding portion of the project because it was spent learning the applications to develop the product. The calculation of the loss of production time has significantly decreased as we entered iteration 2, however it hasn't completely gone away due to the fact we haven't attempted to implement every aspect of the required framework. The six weeks is calculated from the given amount of work already completed and the projection of our current sprint. Because iteration 2 hasn't been hindered catastrophically by the material needing to be learned, the time for loss has been minimized to six weeks.

The way the team plans on mitigating the risk is to focus on what we know, and not on what we don't; find work arounds to still deliver the requested material. This may mean the code isn't as efficient as it could be if team member were working in their chosen language, but it will still be operable (done is better than perfect in some instances).

The alternative is for a group member, or group members to invest in a coding class, to learn the material and be able to better work on the project. However, this course of action would also limit time to code for the actual product, thus affecting the timeline, which is also a risk listed in section 1.3.

For iteration three through five, it is the best guess of the team how much the lack of experience will play into the development in those final iterations. It will be adjusted as iteration 3 is finished and how much of a hinderance it really becomes adding in different aspects of the requirements.

1.2 Accountability

There is the potential for team members to 'go dark' or completely drop the class. Much like the 1.1 Lack of Experience risk, that likelihood is expected to decrease as the team gets farther into the project and semester. This risk isn't as high as the Lack of Experience risk for a few reasons. The first being that the project team members are all in their final semester and this is a required class for graduation. The chances of team members dropping this class also go down as we get farther into the semester since the university incentives for staying in the class are fairly great (ex. No money back, 'W' on transcript, etc.). The second being is a sense of comradery. As we work more on the project and have more team meetings, we become more incentivized by our teammates and the feeling of community that goes along with working in a functional group. The analysis for the risk of accountability is displayed in Table 2.

Table 2.

Iteration	Risk	Probability	Loss	Risk Exposure	Actual Loss
1	Potential for team members to leave or 'go dark' or leaving.	80%	10 weeks	8 weeks	0 weeks
2	Potential for team members to leave or 'go dark' or leaving.	50%	6 weeks	3 weeks	0 weeks
3	Potential for team members to leave or 'go dark' or leaving.	50%	4 weeks	2 weeks	
4	Potential for team members to leave or 'go dark' or leaving.	50%	2 weeks	1 week	
5	Potential for team members to leave or 'go dark' or leaving.	60%	1 week	0.6 weeks	

Further analyzing the choice in numbers for the risk lead the team to decide that, at the beginning, when there were still incentives from the university for a member to drop this class with minimal repercussions, the chances of it happening were still around eighty percent (80%). It is still the team member's final semester, this is still a required class, so that did dampen the likelihood. After the first iteration had ended, and we hadn't lost a team member the chances further decreased. Since the current iteration, iteration 2, marks the halfway point for the semester, the chances of team members dropping from the project completely have drastically dropped. Fifty percent (50%) was chosen because, we are halfway through the semester, and that sense of 'team' and working together has settled in for the group. We have all made meaningful contributions so far, and believe we are heading in the right direction as far as production of the code and additional materials.

At this point, the best way to mitigate this risk is to hold team members accountable for their work and give credit where credit is due. If the members of the team feel like they are making valuable contributions, they are more likely to stick with the team and continue to contribute. This could also qualify for avoidance in the sense that our team can avoid losing a member by making them feel valuable and give them parts of the code they would like to work on.

How we would deal with losing a team member is a little different. In the unlikely event that the team should lose a member we have set a few contingencies in place. The first being commit and commit often. By uploading our code to GitHub every two days, we are likely to have the last parts of person leaving's code or other documentation. The team will then have to restructure a little bit to fill that person's lost role, but it would still be manageable. Should the team lose more than one member, it would be prudent for the team to discuss with Professor Bingham how he would advise on proceeding. With more than one person gone from a team of four, it would be tricky, but manageable to get more of the project done, but a team smaller than that, would have a hard time meeting all the requirements set out by the business. A suggested scale on what would be expected of the smaller team would most likely need to be advised since the timeline isn't really negotiable.

For iteration three through five, it is the best guess of the potential for team members accountability. We're not certain that desire to drop off will ever fully go away, however, we will have fewer weeks to do so. It might be more likely to drop off towards the end when it feels like there is less to do or that other people are on top of everything.

1.3 Timeline

The timeline for this product to be created and operable is five iterations lasting two weeks each for a total of ten weeks of programming to produce the requested product and demonstrate its use. If you asked any developer, they will always say they need more time. Every developer is likely to take the full length of time to code, we fill the time we have. If this were a project being done within Commerce, with all the resources available to a company along with having programmers who know the system and company set up, I would say this timeline is more realistic. It would be easy for the company to put more resources into producing a more competent product. The breakdown of this risk is listed in Table 3.

Table 3.

Iteration	Risk	Probability	Loss	Risk Exposure	Actual Loss
1	Timeline for project	80%	2 weeks	1.6 weeks	2 weeks
2	Timeline for project	80%	2 weeks	1.6 weeks	0.5 weeks
3	Timeline for project	80%	2 weeks	1.6 weeks	
4	Timeline for project	80%	2 weeks	1.6 weeks	
5	Timeline for project	80%	2 weeks	1.6 weeks	

The timeline for this project isn't going to change, the probability of it being a risk is fairly constant. It also involves other risks on this list. It involves the Lack of Experiences, Accountability, and Split Priorities. In the first iteration, the members had to learn how to use the framework that the Commerce requested, which was a fairly big hinderance to the timeline and what was accomplished in that sprint. We will always be trying to make up for that first sprint now. The probability that the timeline will be an issue for the project will always hover around eighty percent (80%) and leave us at about 1.6 weeks at a disadvantage when applying the other risks and their likelihood.

Factors to consider in the timeline are listed within the other risks, any one of those items will factor into the short timeline given to us and how it makes it challenging and risky for this project.

The risk can be transferred back to Commerce, in a sense. If this project were being executed for public use and the team didn't meet the guidelines, we would be fired, but it would be Commerce who would really be in trouble for not meeting their own deadline. To mitigate the risk, the team can apply many of the same principles used in the Split Priorities risk calculation listed in section 1.4 in more detail. We can plan as much as we can, build in some contingency times, take away a meeting should there be too much work that still needs to be worked on and have members report updates in the Discord channel instead. Since the project is due at the end of the semester, there really isn't time to ask for an extension.

Since this risk has the same level of risk exposure as 1.4 Split Priorities, the decision to make Timeline 1.3 rather than ranking it below split priorities wasn't easily made. Should the priorities of the team members be an issue, it will hurt the timeline of the project just as much. The timeline can be disturbed because of other factors *and* the factors listed in the split priority as well. It made the most sense to have it rank higher due to all the risks impacting the timeline.

1.4 Split Priorities

The team working on this project are students, students in their last semester at that. Every team member has other classes they are attending with additional work for those classes. This takes a pretty big chunk of coding time away from this project for Commerce. In addition to that, each team member has at least a part time job on top of their school schedule. This set up is different from a traditional work environment where your job is your main priority. Taking into account all of our many different schedules and commitments is hard but is being managed. It does pose a risk to the team however, as demonstrated in Table 4.

Table 4.

Iteration	Risk	Probability	Loss	Risk Exposure	Actual Loss
1	Scheduling conflicts and other priorities.	80%	2 weeks	1.6 weeks	2 days
2	Scheduling conflicts and other priorities.	80%	2 weeks	1.6 weeks	2 days
3	Scheduling conflicts and other priorities.	80%	2 weeks	1.6 weeks	
4	Scheduling conflicts and other priorities.	80%	2 weeks	1.6 weeks	
5	Scheduling conflicts and other priorities.	80%	2 weeks	1.6 weeks	

The table outlines that the probability of having other priorities interrupt working on the Commerce project is at eighty percent (80%). It's not 100% because at least some work was known for this class before signing up, as well as a good portion of what would be asked of each student. The unknown factors come from the other classes and their scheduling obligations. Some teachers are prepared far in advance and those assignments that might interfere or hinder coding time can be planned or worked around. Other teachers aren't as prepared and require us to make scheduling changes. This exposure is expected as the project continues, which is why it hasn't changed from the first to second iteration. It is possible that we get hit with a major requirement from another class or work keeping that expected risk at 1.6 weeks.

The way to mitigate this exposure is to plan for the unknown as best as possible. Team members can allot a certain amount of time every week as a contingency to the project. Should something come up, this excess of time can be used for project work. Since we have regular team meetings, should it become a big enough issue, the time usually spent in those meetings can be given back to the developer so they can work on this project instead. They will give an update in the Discord channel instead of during the meeting.

Holding members accountable for their work will be important during the project for this risk. If team members are open and honest about what they can accomplish, we can plan for things not working out within the project. If problems aren't acknowledged early on, we run the risk of a downward spiral of priorities, compounding the risk. This risk is listed as fourth because while it is lower, it is a constant. This is something that can only be mitigated by preparing as much as possible. There is little room for avoidance beyond just well planning but planning too much in this environment can lead to much being asked of a programmer. We can't prepare for everything, so the best we can do is have a plan for when something goes wrong.

1.5 Requirements

With any project, there is the chance of doing a demonstration for the business and having them not like what you've been working on for the last few iterations. With a demonstration (or presentation) coming up for the group, it is a possibility that we aren't working in the direction the company has wanted us to go or that what they want could change. The risk exposure for this is calculated in Table 5.

Table 5.

Iteration	Risk	Probability	Loss	Risk Exposure	Actual Loss
1	Requirements Change	0%	2 weeks	0 weeks	0 weeks
2	Requirements Change	0%	2 weeks	0 weeks	0 weeks
3	Requirements Change	50%	2 weeks	1 week	
4	Requirements Change	10%	2 weeks	0.2 weeks	
5	Requirements Change	10%	2 weeks	0.2 weeks	

The chances of there being a requirement change after the presentation, either from Commerce or taking a direction change based on what the team was able to gleam from other group presentations is a possibility for this upcoming iteration, iteration 3. The benefit of our team and the style of development we are taking is that it's more along the lines of adaptability. Should we present our product, and it is found to be inadequate, there is still the chance for us to fix the mistakes and to develop the product in a more suitable way. After the third iteration, the chance that more requirements will change is a possibility but isn't likely. Still, it would be sensible to plan for the possibility, rather than saying it just won't happen. Should we be given the requirements changes, it is likely that what Commerce or Professor Bingham will ask of us won't be too far off from what we've already created, allowing us to use much of the foundation we've already built upon. This means that there will not be too much of a loss.

We are mitigating the risk by already planning for the potential of a requirement change. This is being factored into our work and planning time going into the next sprint as well as the ones following it.

Analysis

The risks listed and detailed in this report are all entwined in some way and have a bit of dependency upon each other. Every risk affects the timeline we've been given. Should our lack of experience overwhelm us, it will set up back time wise. Experience will also play a role in our ability to prioritize differently. Since we need to learn more, we need to allot more time for that in addition to the development process. Accountability is a factor in being on time with each member's contributions and being able to accurately project where we are according to our iteration timeline.

There are many things that can't be accounted for, but as long as we give ourselves enough room for the unknowns we can try and mitigate as much of those risks as possible. There isn't much room for avoidance or transfer or risk in this project, but this is still a learning experience. We are learning and this process is designed to help us understand what will be expected of us in a real job. However, a real job will also provide us with training and wouldn't put us in charge of a project without also mitigating that as a risk.

Noted: This is the formula used to calculate the Risk Exposure and was used from Professor Bigham's notes from the lecture covering Risk Management:

This definition of risk leads to the fundamental concept of risk management which is risk exposure. Risk exposure is defined as:

$$RE = Pr(Uo) * Loss(Uo)$$

Pr(Uo) is the probability of an unsatisfactory outcome and Loss(Uo) is a measure of the loss associated with the unsatisfactory outcome.