Risk Mitigation

Contents

1 Executive Summary

The purpose of this document is to provide situations and risks that could have occurred throughout the course of our project, and provide ways in which to alleviate these problems, or prevent them. This document is broken down into four sections, 'Accepted Risks', 'Avoided Risks', 'Reduced Risks', and 'Transferred Risks'. There are very detailed instructions on how our team would mitigate the risks, and move past them, as to not make them roadblocks. There are also consequences described if these risks are not properly handled.

2 Accepted Risks

1. We don't implement everything that we planned on for the final product.
   (a) We finish the product as described, but not the additional functionality that we placed in the Parking Lot list for potential add-ons.
   (b) Mitigation:
      i. Maintain or even improve the pace of work which we are conducting.
      ii. Members of the team take on additional tasks if they have time, but not prioritizing them over the required work.
2. We cannot get proper test data.
   (a) The client is not able to provide data for specific reports, but also test data as to how forms look when data is present.
   (b) Mitigation
      i. We will create fake data to be able to give teams something to test queries and code with.

3 Avoided Risks

1. We don't finish the project.
    (a) Mitigation:
        i. Constant organization and communication with members on the team, teacher, IT vendor, and the client.
        ii. Status reports are done by Project Leaders and given to Algozzine every week.
        iii. System demonstrations every Monday during class to Algozzine or the client.
        iv. Have a plan for reduced scope.
            A. Scaled back versions of the system that are unable to completed by semesters end as communicated with the client.
        v. If features are not being finished, delegated tasks to other members of the team to take over or help.
        vi. If all else fails, remember that everyone will get an F.
2. We don't finish all the features we promised.
    (a) Mitigation:
        i. Spread work out among the various members of the team; perhaps if one team member is unable to work on/complete a feature, it can be shared or passed on to another member of the team with more available time to dedicate to the feature.
        ii. Continue to check progress weekly as a class, as well as during weekly meetings.
        iii. Assigning a team member to check on the progress of the promised features in an appropriate, timely fashion.
            A. Have a Project Manager on the team update the status of the assigned/promised/dedicated feature on Trello to keep track of tasks.
        iv. Have the system working functionally before working on aesthetics and visuals.
3. After a merge on Github, the overall code no longer works, breaks, or does not work in the intended way it is supposed.
    (a) Mitigation:
        i. Go back to previous commits and see where the issue is with the merge.
            A. Inspect different code bases that were merged and evaluate the issue with functionality.
            B. Test in isolated pieces to discover the issue and fix it.
        ii. Make sure all code is commented appropriately and also detailed. This will make it easier to read through the code and pinpoint the point of failure in a timely manner.
        iii. Notify all members of the issues and have everyone look into the issue, as it is important that the system works and that we are able to deliver the product the client.

4. The project/our code is not scalable.
    (a) Mitigation:
        i. Ensure code is commented and organized that anyone with lesser experience can understand the code.
            A. Strictly enforce this; be upfront with coders about getting this completed.
        ii. Create an API for common operations.
        iii. Ensure servers are not running at maximum capacity and can be adaptive to environments that may have hardware limitations.

5. The client wishes to have help after the semester/school year is over.
    (a) Mitigation:
        i. Communicate with the client periodically to see if any issues occur or any changes may be needed to be made.
        ii. As part of the documentation, there could be a section dedicated to problems that may occur and what to do about them. For example, if a user cannot log in, the documentation would tell them to refresh the page, contact a Super Admin, who could potentially fix the problem by altering the user's login information.
        iii. This risk could be transferred to Algozzine since he is a Super Admin and is in contact with the client more often than the class is.

4 Reduced Risks

1. After deployment, there is a bug found that was not predicted or found during testing.
    (a) Mitigation:
        i. Assign someone the task of post-launch maintenance, possibly Algozzine.
        ii. Ensure code is properly documented and commented so a future developer could easily locate, identify, and fix the problem.

2. Our documentation is poor and hard to understand.
    (a) The code is not commented, documentation is too complex or not detailed enough, demonstration/how to guides are not created.
    (b) Mitigation
        i. Require all developers to document their code.
        ii. The code must be commented and documented in an efficient way so that another developer can understand what the code does without having to have been part of the project in a timely manner.
        iii. Allow non-developer project members to read through the code documentation, to see if they understand what processes are happening in any given block of code.

3. Certain users get too much access unintentionally.
    (a) Mitigation
        i. Ensure that usernames and passwords are securely administered to each user.
        ii. Ensure certain permissions are set for each user, based on their specific admin rights and privileges defined by their user account and needs.

4. There is a data breach post-launch.
    (a) Mitigation
        i. Implement a system that monitors and detects when suspicious activity is happening on site. Such as a system log file.
        ii. Make sure there is a Firewall on site, that also can be monitored remotely, should a breach attempt occur.

5. Our system is incompatible with a user's choice browser.
    (a) Mitigation
        i. Ask the client what is their browser of choice.
        ii. Provide constant testing on the user's browser of choice, but also additional testing on other major browsers as a precaution.
        iii. Ensure the system works on multiple major browsers throughout the process of the development.

6. Our system doesn't deploy as well on the hardware at CAP Dutchess.
    (a) Mitigation
        i. Keep in contact with the IT vendor to keep up to date on their hardware and the possible compatibility it will have.
        ii. Attempt to recreate the working conditions as closely as possible on our hardware with what is available to ensure the closest possible example.
7. Users can't log in at all.
    (a) Mitigation
        i. Implement a recover password option in the event this may happen.
        ii. Allow Super Admin to change login information so that if this is to occur the Super Admin can alter the user's password.

5 Transferred Risks

1. The server or remaining drives at CAP Dutchess fail.
    (a) Mitigation:
        i. Have a detailed deployment guide so that the system could be redeployed on another server.
        ii. Make sure the latest code is saved on GitHub for cloning purposes.
2. The surveys created by our team are not used properly.
    (a) Mitigation:
        i. Include detailed guides and proper documentation on how to use the surveys.
        ii. Demonstrate to the users and train them on how they are expected to be used and give them an opportunity to ask questions about them.
3. The client does not like our system.
    (a) Mitigation:
        i. Show the client the system various times throughout the development process.
        ii. Make sure we give them an opportunity to make requests for things to be changed.
        iii. Change the system to the client's requests and likings.
4. We are informed of other necessary features after we deploy.
    (a) Mitigation
        i. The application is written in a way that allows for future enhancements and additions to the systems.
        ii. The code is commented so that if features need to be added, it is easy to understand and modify existing code.
        iii. Communication throughout the duration of the project should prevent this.
5. We lose track of our progress and tasks.
    (a) Mitigation
        i. Continually follow and adjust the project plan to accommodate additional requirements that the client may have asked for.
        ii. Project manager keeps in constant contact with team members to ensure all the members are on track for the current sprint.
        iii. Have each member demonstrate their progress and their deliverables during weekly meetings.