Project Plan Document

“AutoBuild” - Automatic PC Part Builder

October 24, 2020

Saturday Solution

Connor Kobel	(015036474)
Daniel Gulland	(018648771)
Ian Ho-Sing-Loy	(026339220)
Nickolaus Marshall-Eminger	(025567241)
Sirage El-Jawhari	(018359144)
Zeinab Farhat	(017990360)

Table of Contents

1 Document Revisions

Date	Version	Changes
10/24/2020	0.1.0	Initial Draft formatting.
10/25/2020	0.2.0	Added timeline, filled in sections.
10/28/2020	0.3.0	Update format.

Legend

Team A :

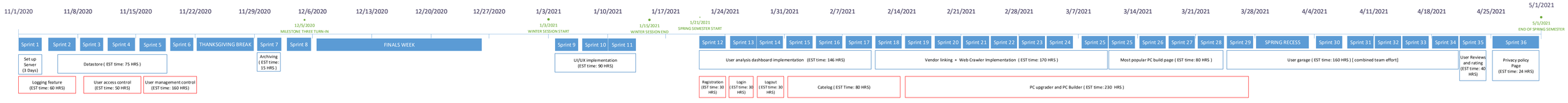
Team B :

Sprint made up of:

Sprint A

First half of week: Monday - Wednesday.
Second half of week: Thursday – Sunday

Estimated average sprint capacity: 45 HRS.



3 Exclusion

Support Page

After estimation of the amount of hours that the support page will take, the support page has been excluded from the project plan and will not be considered in the final product. The estimated time was 212 hours beyond our final deadline. This amount of time estimated along with the other features of Autobuild does not allow us enough time within our time frame of project completion to complete even the most minimal working version of the support page.

4 Risk Assessment

Risks in priority order:

1. Group members time becomes variable, with additional class work taking over developers time.
    - **Mitigation**: Reallocate resources, ensuring others cover work for the incapacitated party when able.

2. Unpredictable scenarios arise which can lower team productivity or reduce quality of work.
    - **Mitigation**: Ensure that work is being done in a timely manner. No work should be done at the last minute that can't be helped in order to account for scenarios that can't be predicted, such as this current pandemic. All work will be stored on Github to have a centralized location of our work to account for loss or corruption of data.

3. Components necessary to the completion of the project are not defined in the scope and are forgotten .
    - **Mitigation**: Identify missing components early and place them into the backlog for evaluation, placement, and grooming.

4. Elements within a component continue to grow in complexity, beyond expected project scope.
    - **Mitigation**: If a component becomes too complex we would need to start considering cutting specialized functionalities. Pruning away at the component until it could feasibly be completed.

5. Developers add "out-of-scope" functionality that they deem an improvement, despite not being in the requirements.
    - **Mitigation**: This can be mitigated by sticking to the requirements and the requested functionality until the project is completed, at which point these extras can be addressed.

6. Due to our unfamiliarity with the SCRUM methodology trying to follow the foreign guidelines has us learning and on our toes, mostly not performing the methodology correctly.
    - **Mitigation**: Learn, adjust, and grow with practice. Trying to adhere to the SCRUM methodology as closely as we can, and observing best practices when possible.

7. General time estimates are inaccurate, allocating either too much or too little time to a feature.
    - **Mitigation**: Mitigation for this comes with experience, however it is better to overestimate how long work will take (within reason) than to

underestimate and not deliver. That said, a large estimate can make developers complacent. So planning feature milestones to fit within sprints should keep work moving.

8. Lack of communication between team members.
    - **Mitigation**: During sprint meetings assert the importance of communication between team members to prevent work overlap. Allow for each team member to communicate concerns and wanted changes in a controlled setting to ensure progress is made, if necessary.

9. Lack of understanding between the developers and the client.
    - **Mitigation**: Ask questions. Reassert that your understanding of the response is unclear, either through email or “face-to-face” communication, until you are clear on what is being said to you.

10. The spring schedule for individuals will change the development teams expected capacity, as well as meeting times, and potential stress levels. This will result in reduced resources for production creating greater strain on other members and reduced productivity.
    - **Mitigation**: Team members will need to be responsible for making time to ensure their portion of work can be completed and not take on too many tasks in weeks they are too busy. Correctly accepting one’s own limitations and will lead to a more correct allocation of work and resource management.

11. Dependencies are out of order, and we later discover that a dependency is at the lower end of production order.
    - **Mitigation**:The first step is to plan out our documentation ahead of time and determine which components are likely dependent on another. Should that initial plan turn out to be incorrect, rescheduling and reallocation of resources may be necessary to catch up and readjust production.

12. Team members may become combative over project issues or personal issues.
    - **Mitigation**: Address early on, communicating concerns, and be willing to compromise. In cases of escalating tensions seek outside mediation.

13. Features and sections of the project that are intended to be in the final version are missing.
    - **Mitigation**: Address concerns with clients and stakeholders, decide what features can be pruned, which functionalities can be removed, and where they can be pruned from. An alternative is to raise team capacity by requesting more work out of people, however that will likely lead to less success and more stress.

14. Developers ignore project communications and assigned tasks.
    - **Mitigation**: Start by addressing if the developer is stuck, why are they stuck, can this be fixed, if the problem persists, assert that there are

consequences if they do not rejoin, begin pulling back from their workload, and discuss removal of the developer from the team as a final option.

15. Developers develop inaccurate expectations of how the final project should end up being, through miscommunication or lack of understanding.
    - **Mitigation**: Make sure details are clear to other members, ask if they understand what has been shared with them. Provide visual aids, examples, or other such strategies to make your ideas clear. If another's explanation is unclear, make sure to also ask for an appropriate elaboration.

16. Developer does not have enough experience in completing an assigned task.
    - **Mitigation:** Make sure to research early, and often, if something is unclear seek out resources and start planning to work on a subject in the next sprint, if you have down time. Make sure that each unfamiliar task, so as to strengthen in the case that similar tasks arise.

17. A developer could drop out, or fail the current semester, not bothering to show for the next and restarting the curriculum. This would leave the team a member short and under budget for time in the spring semester.
    - **Mitigation**: Cover what can be done, readjust project scope, and reassess maximum capacity to recalculate project end date. Prune features, potentially remove features, and salvage as much time as possible to allow for the remaining members to complete the project successfully.

18. A developer on the team actively dislikes the project, or is disinterested in the project. This creates a negative attitude which could potentially hinder the project, whether actively or passively.
    - **Mitigation**: Address any concerns regarding a person's level of disinterest. Find something within the project they may be passionate about and try to allocate more appropriate work in their field of interest.

19. Conflict between stakeholders. Conflict could arise as personalities start to clash, jokes do not go the same way, or people just simply do not get along.
    - **Mitigation**: Try to maintain a civil environment, do not let conflicts interfere with work duties, and separate the aggrieved parties as much as effectively possible without interfering with tasks. Avoid assigning oppositional members to the same team.

20. Requirements are not well defined, either functional or non-functional, and changes are proposed to address them, but those changes do not resolve the issue.
    - **Mitigation**: Work as a team to create an appropriate solution, addressing and using parts of each solution that can be salvaged and repurposed. Potentially pooling ideas and pulling from original requirements.

21. Developers misunderstand the project requirements resulting in wasted time and effort. This problem is extremely pervasive in our group, and could result in lost days of work.
    - **Mitigation**: Ask for clarification on any details that are confusing, or even slightly ambiguous. Make sure we fully understand what can be done to move forward even if the content is not fully applicable. Ask for iterative feedback to make sure you are not moving in the wrong direction. Do not wait till you are done to check if your work is correct.

22. When certain individuals spend most of their time communicating only ideas, and engaging developers, or the client, instead of advancing the project, work progress can come to a stall.
    - **Mitigation**: Address that the individual is communicating too frequently and that work progress needs to continue even with communication. Try to get others involved in conversations to make sure the developer is on task and not side tracking.

23. User misunderstands the purpose of the website. A user's original thought might be to purchase a PC on a website; however, we assist in building and have no purchasing whatsoever.
    - **Mitigation**: The home page will inform our users as clearly as possible in order to tell our user that we solely assist in building PCs and not selling.

24. Developers are not included in the information circuit, leaving them out of the loop and making the group subject to roadblocks. Without the developer present it is possible there is an issue they will not have the chance to raise any flags before any actual work is completed.
    - **Mitigation**: We need to ensure that all team members and stakeholders are included in any relevant project planning and implementation of operations communications. We do this by double checking any emails or phone numbers making sure that nobody who needs to be communicated to is not left out by mistake.

25. New skills are often complicated and time consuming for new users or those with limited experience in practical application. With the time it takes to learn these skills it can overflow into the time you need for implementation.
    - **Mitigation**: The most important mitigation technique is to acknowledge when you don't know something and ask for help as early as possible. Start early with learning new skills and ask for help from other project members or anyone with experience in the required skill. The last thing we want to happen is doing too little too late and not meeting our project's deadlines.

26. As new developers most of our skill sets are not attuned to those of real world development, with product releases. Most of our skills reside in the area of console output, and generic data structures, with some level of database

manipulation. This creates an environment of fairly weak developers trying to keep each other afloat.
    - **Mitigation**: Get outside help, research, use all available resources to make sure that you grow as a developer. Growth will help the team as a whole and benefit our team members when they move on from the project.

27. Team members are not able to perform the tasks assigned, or take on certain tasks they cannot complete.
    - **Mitigation**: Be honest, if you see a task you do not know how to complete, ask for assistance or do not take it. Request work you are comfortable with rather than work that is outside your abilities. Also avoid over committing, when you commit to work it is expected you finish it within the allotted timeline, if you do not complete it the other team members suffer.

28. Team members can become unmotivated and not want to put in their full effort due to constant struggle and stressful deadlines. If team members do not pull their weight it only makes it harder on the others which will increase the likelihood that they develop a negative attitude towards the project.
    - **Mitigation**: Set realistic and short term goals that can be accomplished rather than tackling large problems all at once. Break it up into small steps and have multiple people work together to accomplish harder implementations.

29. The module plans are not flexible. As in, when we create a logging module it should be able to interact with all sections of the website, if not we have specific, exclusive, and redundant code that becomes limited in utility. This results in more work and wasted resources.
    - **Mitigation**: We want to make as much code as modular as possible, allowing for reuse between different portions of the project. This should be standard practice going forward to reduce overall workload.

30. A design plan for a class or feature does not fulfill the expected or necessary purposes and leads to extra work to rework the component(s).
    - **Mitigation**: Ensure plans for classes are overviewed by multiple team members for correctness and validity. We want to ensure that this passes several sets of eyes for confirmation before beginning work on a non-relatable, or simply unusable class.

31. Our testing environment is not yet made, and we could run into problems while creating it, since we have not created one before. We will also run into a scenario of pushing a production ready build out, which we are unprepared for.
    - **Mitigation**: Learn what we can ahead of time. Research into test environments and determine the process in which we would like to release our product, from pre-production to production.

32. The services we will use to launch our service, and the servers we will use to test our service have the ability to go down. Personal internet can go down. Hardware can fail. These are all considerations in regards to technology problems.
    - **Mitigation**: There is not much mitigation can be done when the servers at amazon go down, or if the internet goes down within our homes. We will have to try to make up for lost time in production, and hope that the downtime is negligible on amazon's end.

33. Problems with tools we use to communicate, and tools we use to assign work, could occur preventing developer participation in sprint planning, or velocity calculations, or hours recording. Resulting in unusable metrics for calculating work loads, and the inability to assign appropriate work or tasks to individuals.
    - **Mitigation**: Check back often with team members to ensure that nothing was intended to be assigned and not done, that a task was assigned in your absence, or work is not adequately covered. Use alternative communications to make sure that you are up to date.

34. There is a chance that we hit some legal complications with our web crawler, and data collection techniques. We do not plan to gather user information, but do intend to collect data and benchmarks regarding computer components.
    - **Mitigation**: Research and make necessary requests to appropriate sources to ensure the lawful gathering and redistribution of data from prospective websites. Making sure that we do not receive the ire of the companies we wish to work with.

35. If what is expected and required for our project can be interpreted ambiguously, then there is going to be a disconnect in interpretation between team members and lead to non cohesive work.
    - **Mitigation**: The way to remove ambiguity from the project is to have regular team meetings to discuss the work that needs to be done to make sure that everyone understands it in the same way.

36. Our requirements documentation had many holes and we have been slowly filling them with ideas, and plans, for expected outcomes regarding the end product. This is a mixture of things we believe that would be beneficial and things that would be expected from our users. This leads to scope creep, and moving goal posts, resulting in unobtainable completion criteria.
    - **Mitigation**: Be firm in our decisions to complete what we have currently decided to work on. Do not add additional work to the workload that pushes the goal further out of reach.

37. Some of our decisions are vague, to maintain civility, or a level of "plausible deniability", however this creates ambiguous results to important decisions. We have had several times where we have each heard the same answers from

someone and the same questions remain because the answer did not satisfy the question.
    - **Mitigation**:Be clear, be decisive, do not leave a question unanswered if you feel that your understanding is different from another's.

38. Some issues that are addressed are not completely, or correctly, resolved. Meaning that one resolution is proposed, no agreement is made, and the issue is forgotten, or the resolution is not acknowledged or adhered to.
    - **Mitigation**: Ensure that decisions are clear during all team meetings. All relevant team members should be updated on the status of the resolution, and the resolution must be enforced. If an issue goes unresolved it must be addressed at the next possible meeting, or through discord as soon as possible.

39. Since our website will be used throughout the United States of America, and laws regarding the internet are continuously changing, we could potentially face legal ramifications as if any of our functionality infringes a law in another state, as well as California and federal law.
    - **Mitigation**: Be mindful of new laws, as well as the resolutions and ways to work around them, while maintaining as much functionality as possible and reducing future work. Do not break the law, work within it.

40. With the upcoming release of .NET 5 we are likely to want to update our framework to the new system, provided it is stable and suits our needs. This could provide an opportunity as much as a chance to fail with the chance to learn how to upgrade to a new framework while our project is small. However, this will result in added workload, and lost time.
    - **Mitigation**: Research the benefits, the time cost, and the potential gains. Avoid updating to .NET 5 if the work outweighs the benefits.

41. Our production schedule will likely not be as rock solid as we plan for the future, creating potential slow down in production, and leading to piling up work.
    - **Mitigation**: Follow the production schedule as closely as we can, and be ahead of as much work as possible. Completing tasks early will allow for more time on additional work, and potentially out of scope features we want to include.

42. Our UI is hard to follow along, slow, or not visually appealing.
    - **Mitigation**: Ensure all features' implementations are efficient while also maintaining a user-friendly UI.

43. Users won't like our product or our website.
    - **Mitigation**: Ensure that our product solves a problem that will benefit enough people while also maintaining a user-friendly UI to ensure a good user experience.

44. Our information gathering relies on a web crawler, this could end up being a legal problem if the information gathered is not freely given and we do not adhere to certain regulations and rules.
    - **Mitigation**: Ensure that none of our features or our data gathering techniques violate any legal agreements.
45. The product displays false or incorrect information or has quality issues.
    - **Mitigation**: Ensure that all features are properly tested and make sure that all third party information is regularly updated to prevent any misleading information.

5 References

Risk Assessment Inspiration:

- https://management.simplicable.com/management/new/130-project-risks
- https://www.guru99.com/what-everybody-ought-to-know-about-test-planing.html