SWE20001: Managing Software Projects

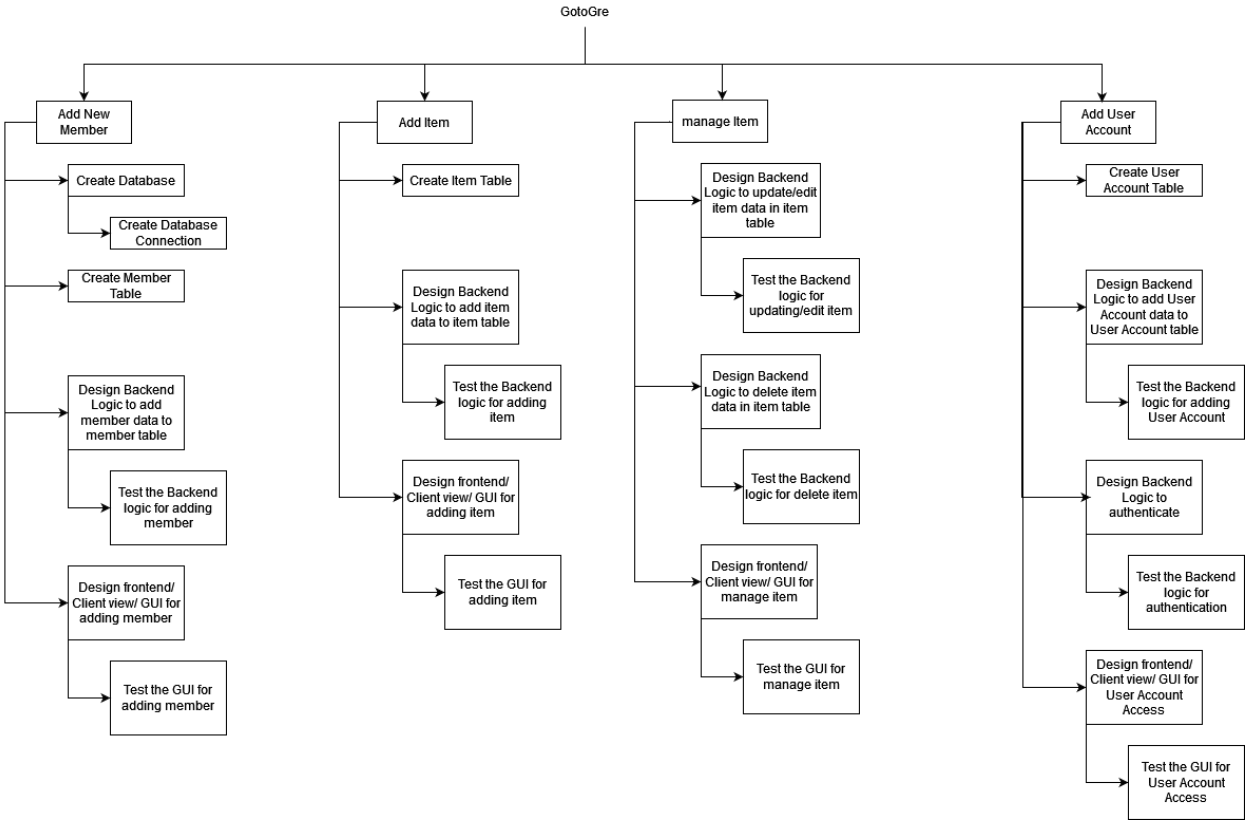
Name: S M Ragib Rezwan

ID: 103172423

Tutor: Naveed Ali | 12.30pm Tuesday | EN31

Project Proposal1 : GotoGro :

In the group work task (08P), we had decided upon the following WBS where we have decided to use product based approach to ensure that no product planned to be developed in sprint 1 is missing from the task breakdown structure/chart.



Here, you can see that we have the following leaf nodes that we must create by the end of Sprint 1:

1. Creating Database
2. Creating the connection of the Database
3. Creating Member Table
4. Designing the backend functionality and logic to add Member to the member table
5. Testing the backend functionality and logic to add Member to the member table
6. Designing the GUI feature for adding the Members

7. Testing the GUI feature for adding the Members
8. Creating Item Table
9. Designing the backend functionality and logic to add Item to the item table
10. Testing the backend functionality and logic to add Item to the item table
11. Designing the GUI feature for adding the Items
12. Testing the GUI feature for adding the Items
13. Designing the backend functionality and logic to update/edit Item to the item table
14. Testing the backend functionality and logic to update/edit Item to the item table
15. Designing the backend functionality and logic to deleting Item to the item table
16. Testing the backend functionality and logic to deleting Item to the item table
17. Designing the GUI feature for managing the Items
18. Testing the GUI feature for managing the Items
19. Creating User Account Table
20. Designing the backend functionality and logic to add User Account data to the User Account table
21. Testing the backend functionality and logic to add User Account data to the User Account table
22. Designing the backend functionality and logic to authentication
23. Testing the backend functionality and logic for authentication
24. Designing the GUI feature for User Account Access
25. Testing the GUI feature for User Account Access

Now here, considering Ideal Time, Ideal Effort and Effective effort, we can notice the following table:

[Note: here I am using the fact the each person will only be able to productively work for 70% of their time and using that in my calculation of real effort for each of the leaf nodes.]

[Note: the leaf node numbers refer to their task names here and I have not written their names as it would be too long and their brief abbreviations would be confusing]

Task no (following the list created above)	Ideal Time	Ideal Effort	Real Effort	Justification
1	60 mins	60 mins	120mins	Here we are creating the database for the entire system in C# and then verify that it works by using a test case table (this will be removed later on after troubleshooting is over). Although this should ideally take 60 mins; considering the fact that we are working in C# and the different types of errors that usually pop up, we have decided to keep some

				leeway time for troubleshooting and research. Furthermore, there is also the fact that each person will only be effectively working 70% productively.
2	60 mins	60 mins	120 Mins	Here we are creating the connection for the database for the entire system in C# and then verify that it works by using a test case table and querying for viewing data (this will be removed later on after troubleshooting is over). Although this should ideally take 60 mins; considering the fact that we are working in C# and the different types of errors that usually pop up, we have decided to keep some leeway time for troubleshooting and research. Furthermore, there is also the fact that each person will only be effectively working 70% productively.
3	60 mins	60 mins	120 Mins	Here we are creating the member table in the database in C# and then verify that it works by using a test case data (this will be removed later on after troubleshooting is over). Although this should ideally take 60 mins; considering the fact that we are working in C# and the different types of errors that usually pop up, we have decided to keep some leeway time for troubleshooting and research. Furthermore, there is also the fact that each person will only be effectively working 70% productively.
4	60 mins	60 mins	120 Mins	Here we are creating the functionality and logic to add members to the member table in C# and then verify that it works by using a test case data (this will be removed later on after troubleshooting is over). Although this should ideally take 60 mins; considering the fact that we are working in C# and the different types of errors that usually pop up, we have decided to keep some leeway time for troubleshooting and research. Furthermore, there is also the fact that each person will only be effectively working 70% productively.
5	60 mins	60 mins	120 Mins	Here we are vigorously testing the functionality and logic to add members to the member table in C# following our Definition of done (noted before in previous tasks) Although this should ideally take 60 mins; considering the fact that we are working in C# and the different types of errors that usually pop up, we have decided to keep some leeway time for troubleshooting and research. Furthermore,

				there is also the fact that each person will only be effectively working 70% productively.
6	60 mins	60 mins	120mins	Here we are creating the GUI feature for adding the members in C# and then verify that it works by using a test case table (this will be removed later on after troubleshooting is over). Although this should ideally take 60 mins; considering the fact that we are working in C# and the different types of errors that usually pop up, we have decided to keep some leeway time for troubleshooting and research. Furthermore, there is also the fact that each person will only be effectively working 70% productively.
7	60 mins	60 mins	120 Mins	Here we are vigorously testing the GUI feature for adding the members in C# following our Definition of done (noted before in previous tasks) Although this should ideally take 60 mins; considering the fact that we are working in C# and the different types of errors that usually pop up, we have decided to keep some leeway time for troubleshooting and research. Furthermore, there is also the fact that each person will only be effectively working 70% productively.
8	60 mins	60 mins	120 Mins	Here we are creating the Item table in the database in C# and then verify that it works by using a test case data (this will be removed later on after troubleshooting is over). Although this should ideally take 60 mins; considering the fact that we are working in C# and the different types of errors that usually pop up, we have decided to keep some leeway time for troubleshooting and research. Furthermore, there is also the fact that each person will only be effectively working 70% productively. **[Note: But since we have done similar task for task no 3, it might end up taking half time in reality. But that is something that must be verified first during this sprint]**
9	60 mins	60 mins	120 Mins	Here we are creating the functionality and logic to add Items to the Item table in C# and then verify that it works by using a test case data (this will be removed later on after troubleshooting is over).

				Although this should ideally take 60 mins; considering the fact that we are working in C# and the different types of errors that usually pop up, we have decided to keep some leeway time for troubleshooting and research. Furthermore, there is also the fact that each person will only be effectively working 70% productively. **[Note: But since we have done similar task for task no 4, it might end up taking half time in reality. But that is something that must be verified first during this sprint]**
10	60 mins	60 mins	120 Mins	Here we are vigorously testing the functionality and logic to add Items to the Item table in C# following our Definition of done (noted before in previous tasks) Although this should ideally take 60 mins; considering the fact that we are working in C# and the different types of errors that usually pop up, we have decided to keep some leeway time for troubleshooting and research. Furthermore, there is also the fact that each person will only be effectively working 70% productively. **[Note: But since we have done similar task for task no 5, it might end up taking half time in reality. But that is something that must be verified first during this sprint]**
11	60 mins	60 mins	120mins	Here we are creating the GUI feature for adding the Items in C# and then verify that it works by using a test case table (this will be removed later on after troubleshooting is over). Although this should ideally take 60 mins; considering the fact that we are working in C# and the different types of errors that usually pop up, we have decided to keep some leeway time for troubleshooting and research. Furthermore, there is also the fact that each person will only be effectively working 70% productively. **[Note: But since we have done similar task for task no 6, it might end up taking half time in reality. But that is something that must be verified first during this sprint]**
12	60 mins	60 mins	120 Mins	Here we are vigorously testing the GUI feature for adding the Items in C# following our Definition of done (noted before in previous tasks)

				Although this should ideally take 60 mins; considering the fact that we are working in C# and the different types of errors that usually pop up, we have decided to keep some leeway time for troubleshooting and research. Furthermore, there is also the fact that each person will only be effectively working 70% productively. **[Note: But since we have done similar task for task no 7, it might end up taking half time in reality. But that is something that must be verified first during this sprint]**
13	60 mins	60 mins	120 Mins	Here we are creating the functionality and logic to update/edit Item to the item table in C# and then verify that it works by using test case data (this will be removed later on after troubleshooting is over). Although this should ideally take 60 mins; considering the fact that we are working in C# and the different types of errors that usually pop up, we have decided to keep some leeway time for troubleshooting and research. Furthermore, there is also the fact that each person will only be effectively working 70% productively.
14	60 mins	60 mins	120 Mins	Here we are vigorously testing the functionality and logic to update/edit Item to the item table in C# Although this should ideally take 60 mins; considering the fact that we are working in C# and the different types of errors that usually pop up, we have decided to keep some leeway time for troubleshooting and research. Furthermore, there is also the fact that each person will only be effectively working 70% productively.
15	60 mins	60 mins	120 Mins	Here we are creating the functionality and logic to delete Item from the item table in C# and then verify that it works by using test case data (this will be removed later on after troubleshooting is over). Although this should ideally take 60 mins; considering the fact that we are working in C# and the different types of errors that usually pop up, we have decided to keep some leeway time for troubleshooting and research. Furthermore, there is also the fact that each person will only be effectively working 70% productively.
16	60	60	120	Here we are vigorously testing the functionality and logic to

	mins	mins	Mins	delete Item from the item table in C# Although this should ideally take 60 mins; considering the fact that we are working in C# and the different types of errors that usually pop up, we have decided to keep some leeway time for troubleshooting and research. Furthermore, there is also the fact that each person will only be effectively working 70% productively.
17	60 mins	60 mins	120 Mins	Here we are creating the GUI feature for managing the Items in C# and then verify that it works by using test case data (this will be removed later on after troubleshooting is over). Although this should ideally take 60 mins; considering the fact that we are working in C# and the different types of errors that usually pop up, we have decided to keep some leeway time for troubleshooting and research. Furthermore, there is also the fact that each person will only be effectively working 70% productively. **[Note: But since we have done similar task for task no 6, it might end up taking half time in reality. But that is something that must be verified first during this sprint]**
18	60 mins	60 mins	120 Mins	Here we are vigorously testing the GUI feature for managing the Items in C# Although this should ideally take 60 mins; considering the fact that we are working in C# and the different types of errors that usually pop up, we have decided to keep some leeway time for troubleshooting and research. Furthermore, there is also the fact that each person will only be effectively working 70% productively. **[Note: But since we have done similar task for task no 7, it might end up taking half time in reality. But that is something that must be verified first during this sprint]**
19	60 mins	60 mins	120 Mins	Here we are creating the User Account Table in C# and then verify that it works by using test case data (this will be removed later on after troubleshooting is over). Although this should ideally take 60 mins; considering the fact that we are working in C# and the different types of errors that usually pop up, we have decided to keep some

				leeway time for troubleshooting and research. Furthermore, there is also the fact that each person will only be effectively working 70% productively. **[Note: But since we have done similar task for task no 3, it might end up taking half time in reality. But that is something that must be verified first during this sprint]**
20	60 mins	60 mins	120 Mins	Here we are creating the functionality and logic to add User Accounts to the User Account table in C# and then verify that it works by using test case data (this will be removed later on after troubleshooting is over). Although this should ideally take 60 mins; considering the fact that we are working in C# and the different types of errors that usually pop up, we have decided to keep some leeway time for troubleshooting and research. Furthermore, there is also the fact that each person will only be effectively working 70% productively. **[Note: But since we have done similar task for task no 4, it might end up taking half time in reality. But that is something that must be verified first during this sprint]**
21	60 mins	60 mins	120 Mins	Here we are vigorously testing the functionality and logic to add User Accounts to the User Account table in C# Although this should ideally take 60 mins; considering the fact that we are working in C# and the different types of errors that usually pop up, we have decided to keep some leeway time for troubleshooting and research. Furthermore, there is also the fact that each person will only be effectively working 70% productively. **[Note: But since we have done similar task for task no 5, it might end up taking half time in reality. But that is something that must be verified first during this sprint]**
22	60 mins	60 mins	120 Mins	Here we are creating the functionality and logic to authenticate the User accounts in C# and then verify that it works by using test case data (this will be removed later on after troubleshooting is over). **[Note: This is a completely new thing that none of the**

				group members have ever done before in any of their units. Thus we would need to research a lot on this topic and might even take "double" or "quadruple" amount of time on it. Need to note the actual time taken for this after the sprint to have a good idea on how long our team takes to create something new and complex]
23	60 mins	60 mins	120 Mins	Here we are vigorously testing the functionality and logic to authenticate the User accounts in C# **[Note: This is a completely new thing that none of the group members have ever done before in any of their units. Thus we would need to research a lot on this topic and might even take "double" or "quadruple" amount of time on it. Need to note the actual time taken for this after the sprint to have a good idea on how long our team takes to create something new and complex]**
24	60 mins	60 mins	120 Mins	Here we are creating the GUI feature for User Account Access in C# and then verify that it works by using test case data (this will be removed later on after troubleshooting is over). Although this should ideally take 60 mins; considering the fact that we are working in C# and the different types of errors that usually pop up, we have decided to keep some leeway time for troubleshooting and research. Furthermore, there is also the fact that each person will only be effectively working 70% productively. **[Note: But since we have done similar task for task no 6, it might end up taking half time in reality. But that is something that must be verified first during this sprint]**
25	60 mins	60 mins	120 Mins	Here we are vigorously testing the functionality and logic to delete Item from the item table in C# Although this should ideally take 60 mins; considering the fact that we are working in C# and the different types of errors that usually pop up, we have decided to keep some leeway time for troubleshooting and research. Furthermore, there is also the fact that each person will only be effectively working 70% productively. **[Note: But since we have done similar task for task no 7, it might end up taking half time in reality. But that is something that must be verified first during this sprint]**

Total time taken (with respect of the WBS diagram): **50 hrs** (=3000 mins)

Now, in our project we are planning on using pair programming method where two people will be tasked to code for a single functionality. This is done for three reasons:

a) Following the information in the lecture about "trunk factor" in risk and by also having firsthand experience of it (as a member of our team had been in a car crash in the early weeks and thus had missed for several later weeks due to recuperation), we have decided to proceed with the pair programming method in order to ensure that even if one person is unable to deliver, the other one can and thus the functionalities present in sprint 1 can be delivered in time (whilst fulfilling the definition of done related to them)

b) Different people have different way of approaching a single problem and thus can come up with different unique backend logic to provide the same outcome. Thus by using this system we can ensure that we can have a variety of method logics and thus can choose the best one in a short period of time

c) Last but not the least, our team has issue in the skills required for the task as almost half of our members have never done such a project before from end to end, let alone in C#. So by following this technique, we can provide them with the time and guidance needed to hone their skills, whilst not delaying the progress of the project itself.

Thus, we end up with the time of **100hrs** (= 2 * Total time taken) which is just **4 hrs exceeding** our intended team time of **96hrs**. Since it is just **4.17% extra time** from the initial intended time, we believe it to be alright and thus believe that the backlog items chosen for sprint 1 *(Adding members, Adding Item, Managing Item and Adding User Accounts)* are adequate.