# **Milestone 3 Scrum Report**

All students are expected to attend the scrum meetings and to participate. Failure to do so will result in greatly reduced grades.

**GROUP**: **Group C**

**Members Present**:

|  |  |
| --- | --- |
| 1. Chia-Ming Cheng | 4. Peter Bryson |
| 2. Md Arafat Koyes | 5. |
| 3. Md Asif Karim | 6. |

## Milestone 3 Tasks

In this milestone you will create issues to design the functions, design all of the functions you need to complete the project and store the specifications in the repository. As soon as the specifications start to be produced, you can start to design the blackbox tests (what they test, how to perform them and test data). Once tests are written, they can be implemented and added to the repository and any team members not otherwise busy can start to implement the functions. You will also build a function-test matrix that shows the blackbox tests for each function. This will be maintained through the testing cycle as new tests are added.

**Deliverables due 4 days after your lab day:**

* A set of AT LEAST 4 function specifications added to a new header file and stored in the repository.
* A set of blackbox tests as test documents (in an Excel file) with test data for the functions you created. At least 4 sets of test data are required for each function. You must have test cases for at least 6 functions (including all your custom function). Stored in the repository.
* **Create and add a C++ testing project to your solution.**
* Start writing blackbox test code (for the functions above) and store in repository (at least 1 is required for this milestone).
* Start implementing the functions and store them in repository (optional).
* A requirements traceability matrix added to the repository and shows the mapping between the requirements and test cases.
* Updated Jira project to show activities and progress.
* Completed scrum report including reflection questions answered.

**Rubric:**

|  |  |  |
| --- | --- | --- |
| **Individual** | Group participation (includes GitHub commits and Jira usage) | 80% |
| Teamwork | 20% |
| **Group** | Function specifications (documented, complete, well-written, added to the project) | 10% |
| Blackbox test cases document (well-written, complete, good test data) | 10% |
| Blackbox test code (in the C++ project) well-designed and documented | 10% |
| Functions implementation (coded in the C project & well documented) | 15% |
| Requirements traceability matrix (complete and added to GitHub) | 15% |
| Git usage (used properly with good structure) | 10% |
| Jira usage (creates issues, tracks progress) | 15% |
| Scrum report & reflections | 15% |
| **Deadline** | 20% deduction for each day you are late |  |

**Scrum Report**

**Summary of Tasks Completed or Delayed in the last week:**

Here you can list all of the tasks completed in the last week along with any tasks which could not be completed with a reason why they could not be completed.

|  |  |  |
| --- | --- | --- |
| **Member** | **Tasks Completed** | **Tasks Delayed/Blocked** |
| Chia-Ming Cheng | Group meeting invitation, update GitHub and Jira, assign tasks, proofread, and submission |  |
| Md Arafat Koyes | Wrote Test Plan |  |
| Md Asif Karim | Wrote scrum report reflection |  |
| Peter Bryson | Created data structures |  |

**Summary of Meeting:**

A summary of the main points discusses in the meeting and the outcomes of the discussions.

|  |  |  |
| --- | --- | --- |
| Topic | Discussion Summary | Outcome |
| Task arrangement | Every member knows their job | On time |
|  |  |  |
|  |  |  |

**Summary of Decisions Made:**

This will include major architecture and design decisions, testing decisions, prioritization of tasks, dealing with problems encountered and other major outcomes from the meeting.

|  |  |
| --- | --- |
| Decision | Rationale |
| Assign appropriate tasks to each member | To ensure all team members can understand the project |
|  |  |
|  |  |

**Tasks Attempted During Meeting:**

Each member is assumed to participate in the scrum meeting and contribute to the completion of the scrum report and reflections. Since the scrum meeting will not take more than 20-30 minutes, there is lots of time left to undertake some of the actual work tasks. In the table below, each member should list what they did to complete the scrum report, the reflections, and 1-4 other tasks they completed during the class period. If a task could not be completed, the student should indicate why this was not possible.

|  |  |  |  |
| --- | --- | --- | --- |
| Member | Task Attempted | Time Spent | Complete? |
| Chia-Ming Cheng | MS 3 Group meeting invitation and assign tasks | 30 mins | Yes |
| Chia-Ming Cheng | MS 3 Jira task and GitHub update | 30 mins | Yes |
| Chia-Ming Cheng | MS 3 Proofread, review and submission | 1 hours | Yes |
| Md Arafat Koyes | Writing black box test cases | 1 hours | Yes |
| Md Asif Karim | Writing black box test code | 1 hours | Yes |
| Peter Bryson | Writing function specifications | 2 hours | Yes |
|  |  |  |  |
|  |  |  |  |

**Scrum Tasks Selected for Next Week**:

The tasks each member has selected to pursue for this class or the next week.

|  |  |
| --- | --- |
| Group Member | Task Description |
| Chia-Ming Cheng | Group meeting invitation and assign tasks, update Jira project and GitHub, update requirements traceability matrix, write white box test document and scrum report, proofread and review everything, submission |
| Md Arafat Koyes | Write white box test code, at least 1 |
| Md Asif Karim | Write black box test code |
| Peter Bryson | Write function code |
|  |  |
|  |  |

**Major Outcomes of Meeting:**

This is where you should highlight the major accomplishments of the class.

|  |  |
| --- | --- |
| Outcome | Impact on Project |
| Every member knows their job | The project is progressing perfectly |
|  |  |
|  |  |

**Things That Went Well in This Meeting:**

Here you can highlight things which worked well. This indicates that the way you worked on these items is working and should be continued.

|  |  |
| --- | --- |
| Topic/Work Item | Reason for Success |
| Task arrangement | Every member is good at communicate |
|  |  |
|  |  |

**Things That Did NOT go Well in This Meeting:**

This is where you can list things which did not go well in the class. You should analyze why this happened and suggest how you can improve it next time. This will lead to the goal of *continuous process improvement*.

|  |  |
| --- | --- |
| Topic/Work Item | Reason for Problem and How to do Better |
| N/A |  |
|  |  |
|  |  |

**Reflections**:

Answer the following questions using your own words. Make sure that each answer comprises a minimum of 100 words.

1. What is the difference between blackbox tests cases and blackbox test code? Explain how we use assertion in Visual Studio to execute tests.

A:

Blackbox test cases give us specific scenarios for testing the software system's functionality without knowing how it works. It focuses on the input that we provide to the system and the expected output to see that the software system is working as specified.

Blackbox test code is the actual code that we write to execute the test cases. The code gives input to the software system and checks if it gives the expected output. Because of the way the code is written, the system is a black box and does not require any knowledge of its internals.

In Visual Studio, assertions in test code are crucial for verifying the system's behavior under different conditions. Assertion checks only particular conditions; if the condition fails, the testing will also fail. It highlights between expected and actual behavior.

1. How can a traceability matrix help in the testing process?

A:

Traceability matrix is a very important tool for the testing process itself, as it provides assurance of detailed test coverage that meets the requirements. The traceability matrix maps the test cases against some specific requirements in order to ensure that all the functionalities are tested and not a single requirement is missed. This will bring out the holes in the test coverage and hence assure that every aspect of the system is evaluated. It also supports impact analysis, in which testers can see how requirements get affected by the changes within the code. It improves traceability and accountability by giving a clear link between the requirements, test cases, and defects. Such an organized approach helps ensure consistency, eventually improving the overall quality of software and that the final product meets its specified requirements.

1. Write down two of the function prototypes you submitted. Why did do you need each one of them and how will each one help you achieve the project needs?

A:

1. int findTruckForShipment(struct Map map, struct TruckCapacity trucks[], int numTrucks, struct Delivery shipment);

* Map (struct Map map): Provides information about the delivery area, including buildings that trucks need to navigate around.
* Trucks (struct TruckCapacity trucks[]): An array of trucks, each with its current route.
* Number of Trucks (int numTrucks): Specifies how many trucks are available for consideration.
* Shipment (struct Delivery shipment): Contains information about the size and weight of the shipment that needs to be assigned to a truck.

This function optimizes truck assignment by considering load capacity, route proximity to the destination, and whether a truck can physically accommodate the shipment. It returns the index of the best truck for the shipment or `-1` if no suitable truck is found.

2. struct DeliveryRoute determineTruckRoute(struct Map map, struct Point start, struct Point destination);

* Map (struct Map map): Provides the layout of the delivery area, including obstacles like buildings.
* Start (struct Point start): Specifies where the truck begins its journey.
* Destination (struct Point destination): Indicates where the truck needs to deliver the shipment.

By using pathfinding algorithms on the provided map, this function calculates the shortest route from the starting point to the destination, avoiding obstacles. It returns a structured route (struct DeliveryRoute) that the truck can follow to efficiently complete its delivery.