# **Milestone 1 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 1 Tasks**

In this phase of the project you will:

* Setup teams of about 3-5 developers (6 is too large)
* Write and sign a team contract
* Create a GIT account
* Create a Jira account
* Add your professor to the GIT and Jira accounts
* Update Jira with the work performed and planned

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

* Completed team contract.
* Fully initialized Git repository. **Be sure to send your professor the link to your GitHub repository and a screenshot of the GitHub users.**
* Fully setup Jira project. **Be sure to send your professor the link to your Jira Project.**
* Completed scrum report including reflection questions answered.

**Rubric**

|  |  |  |
| --- | --- | --- |
| **Individual** | Group participation | 80% |
| Teamwork | 20% |
| **Group** | Contract | 25% |
| Git repository | 25% |
| Jira project | 25% |
| Scrum report & reflections | 25% |
| **Deadline** | 20% deduction for each day you are late |  |
| **NOTE** | Both the individual and group marks are calculated separately. Each member of the group will have their mark calculated based on their contribution to the group work and their contributions to the team. The group participation is a percentage that your professor feels you contributed to the group work. This is multiplied by the weight of the group participation component to determine your grade. |  |

**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 | Set up GitHub repository and Jira project |  |
| Md Arafat Koyes | Wrote scrum report |  |
| Md Asif Karim | Wrote group contract |  |
| Peter Bryson | Meeting invitation |  |

**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 |
| Adjust GitHub permissions | To ensure all team members can access the repository |
|  |  |
|  |  |

**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 cannot be completed, the student should indicate why this was not possible.

|  |  |  |  |
| --- | --- | --- | --- |
| Member | Task Attempted | Time Spent | Complete? |
| Chia-Ming Cheng | Set up GitHub repository and Jira project | 1 hour | Yes |
| Md Arafat Koyes | Writing scrum report | 1.5 hour | Yes |
| Md Asif Karim | Writing group contract | 1 hour | Yes |
| Peter Bryson | Meeting invitation and organize | 30 mins | 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 | Writing scrum report |
| Md Arafat Koyes | Update Jira with new tasks |
| Md Asif Karim | Begin initial coding for project |
| Peter Bryson | Review and update GitHub |
|  |  |
|  |  |

**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 (to be answered by the group)**:

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

1. GIT is an example of a version control system. List and explain 3 benefits of using a version control system.

A:

1. Tracking changes and version history:

One of the benefits of version control is ability to track changes made to files over time. Every change made by team members is recorded along with metadata such as who made the change and when. This history allows developers to understand how a project has evolved, review previous versions, and revert to earlier states if needed. This capability is important for collaboration, troubleshooting bugs introduced by recent changes, and maintaining a stable codebase.

2. Good for team collaboration:

Version control systems enable seamless collaboration among team members, even if they are in different places. Multiple developers can work on the same files concurrently without conflicts, thanks to features like branching and merging. Branches allow developers to work on new features or fixes independently, keeping the main codebase stable. When features are complete, branches can be merged back into the main branch, integrating everyone’s work. This process prevents overwriting each other’s changes and ensures that the team’s efforts are efficiently combined.

3. Backup and restore capabilities:

Version control systems serve as a backup mechanism for code. The entire project history is stored on the version control server, which typically includes offsite backups for redundancy. This feature is invaluable in case of hardware failures, accidental deletions, or other emergencies. Developers can restore the project to any previous state recorded in the version history, ensuring data integrity and project continuity.

1. What is a version control system? Why does GitHub qualify as a version control system?

A:

A version control system is a software tool that helps track and manage changes to files, particularly source code, over time. It allows developers to collaborate on projects, maintain a history of the changes in their code, and revert to previous versions if needed. GitHub qualifies as a version control system for the following reasons:

1. Change tracking: GitHub uses Git, a distributed version control system, to track and maintain a complete history of changes, including who made them and when to files in a repository.

2. Collaboration support: GitHub enables multiple developers to work on the same project simultaneously by providing features like branching, merging, and pull requests, enabling teams to collaborate efficiently without interfering with each other's work.

3. Version management: GitHub stores multiple versions of files and allows users to access and restore previous versions easily.

4. Distributed architecture: As a Git-based platform, GitHub follows a distributed version control model, where each developer has a complete copy of the repository on their local machine. This allows for offline work and provides redundancy.

5. Code review and issue tracking: GitHub provides built-in tools for code review, issue tracking, and project management, which are essential components of modern version control systems.

6. Centralized repository: While Git itself is distributed, GitHub acts as a centralized hub for storing and sharing code repositories, making it easier for teams to collaborate and maintain a single source of truth.

7. Branching and merging: GitHub supports Git's powerful branching and merging capabilities, allowing developers to create separate lines of development and integrate changes seamlessly.

8. Access control and security: GitHub provides features for managing access to repositories, protecting sensitive code, and ensuring the integrity of the codebase

1. What is Jira? How are we going to use Jira for this project?

A:

Jira is a project management and issue-tracking application designed by Atlassian for teams of software developers. Jira supports extensive functionality, from bug tracking and agile project management to task tracking, with enhanced productivity tools in the hands of a project manager.

For our project, we will use Jira for tasks and progress tracking and to ensure effective collaboration. Each task or issue will be logged in Jira, assigned to team members, and tracked from its creation to its lifecycle. We will use boards in Jira that will show us our workflow in a way that maintains transparency and responsibility. Custom workflows will help shape the project management process according to our needs. Furthermore, the use of reporting tools in Jira should enable us to track project metrics, deadlines, and progress for us to keep on track and be able to catch up quickly with any lost time.

1. Why is a Kanban board useful in software development. What are the advantages of using Kanban board?

A:

A Kanban board is a tool that we use in software development. It helps the teams manage their work on the project. On the Kanban board, team members can see their task and the progress of their work, which makes it easier for each member to understand their task for the project. If anyone adds or changes the tasks, everyone can see them inside the tool, and members can focus on which one they have to give priority to. By seeing this, team members can save a delay in work and do the tasks more smoothly. The Kanban board has an update system, which can help members fix issues and deliver new features quickly. Also, it has a task manager that shows the tasks one by one so that one member does not overload their work. In Kanban Board, everyone can see what their team members are doing or the progress of everyone’s tasks, and it boosts teamwork and communication for all members. Overall, Kanban Board is so useful for managing projects that the group members work on. It helps the team work together and finish the project with no errors.

GitHub:

<https://github.senecapolytechnic.ca/ccheng73/summer24-sft221-zcc-c>

Jira:

<https://summer24-sft221-zcc-c.atlassian.net/jira/software/projects/KAN/boards/1>