# **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**: B

**Members Present**:

|  |  |
| --- | --- |
| 1. Abdullah Al Mahfuz | 4. Devarsh Shaileshbhai Patel |
| 2. Isabela Jorge Bulla | 5. |
| 3. Ketia Teta | 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** | **Tasks Completed** | **Tasks**  **Delayed/**  **Blocked** |
| **Abdullah Al Mahfuz** | **Create Github account and repository (initialize it). Add group members and the professor. Create jira account and Kanban board. Send invitations to group members and professor. Submit ms1 via mail.** | **Yes** |  |
| **Isabela Jorge Bulla** | **Create github and jira account. Accept the invitation to jira.** | **Yes** |  |
| **Ketia Teta** | **Create github and jira account. Accept the invitation to jira.** | **Yes** |  |
| **Devarsh Shaileshbhai Patel** | **Create github and jira account. Accept the invitation to jira.** | **Yes** |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

For every task delayed or blocked, describe the reason for the delay or block, how it impacts the project and the proposed solution or workaround**.**

|  |  |
| --- | --- |
| **Delayed or Blocked Task** |  |
| **Reason for delay or block** |  |
| **Impact on Project** |  |
| **Solution or work-around** |  |
|  |  |
| **Delayed or Blocked Task** |  |
| **Reason for delay or block** |  |
| **Impact on Project** |  |
| **Solution or work-around** |  |

**Summary of Meeting:**

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

|  |  |  |
| --- | --- | --- |
| Topic | Discussion Summary | Outcome |
| Inquiry about the accounts and invitations | **Everyone has created their account and has accepted the invitation to Jira. Also everyone can see the repository for the team work project.** | **Positive.** |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |

**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 |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |

**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? |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

**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 |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |

**Major Outcomes of Meeting:**

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

|  |  |
| --- | --- |
| Outcome | Impact on Project |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |

**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 |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |

**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 |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |

**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.

**Answer**: Version control systems like Git are essential tools in modern software development. 3 benefits of using a version control system-

1. **Collaboration and Coordination** – Version control systems (VCS) facilitate seamless collaboration among multiple developers. They provide a centralized repository where all changes are tracked, making it easy for team members to work on different parts of a project simultaneously without overwriting each other’s work. Developers can create branches to work on new features, bug fixes, or experiments without affecting the main codebase. Once the changes are stable and tested, branches can be merged back into the main branch. There is also a good way for conflict resolution. Git provides tools to identify and resolve conflicts. Every change made to the code is recorded with a timestamp, author information, and a commit message describing the change. This allows the teams to understand the evolution of the code, pinpoint who introduced specific changes, and trace the origin of bugs.
2. **Code Integrity and Backup** – VCSs ensure the integrity and reliability of the codebase by maintaining a complete history of all changes. This comprehensive tracking provides several layers of security and backup. Every modification is logged, allowing developers to revert to previous versions if a problem is detected. This is crucial for recovering from mistakes or unforeseen issues introduced by recent changes. Systems like Git are distributed, meaning that each developer’s local repository is a complete copy of the project history. This decentralization ensures that the project is not dependent on a single point of failure and can be restored from any developers local copy if needed.
3. **Enhanced Productivity and efficiency** – Using a VCS significantly enhances productivity and efficiency by automating and simplifying many aspects of the development workflow. They integrate seamlessly with automated build systems and continuous integration pipelines. This allows for automatic testing and deployment of changes, ensuring that new code validated and deployed quickly and reliably. They also support code review workflow through pull requests or merge requests. These features enable peer reviews before changes are merged into the main branch, improving code quality and fostering knowledge sharing within the team.
4. What is a version control system? Why does GitHub qualify as a version control system?

**Answer**: A Version Control System (VCS) is a software tool that helps manage changes to source code or other collections of files over time. It keeps track of every modification made to the files in a project, enabling multiple developers to collaborate efficiently.

GitHub qualifies as a version control system because – it provides all the necessary tools and infrastructure for managing and collaborating on software projects. GitHub is built on top of Git, a powerful distributed version control system. Git itself provides all the core functionalities of a VCS including: Tracking changes, Branching and merging, Distributed version control. GitHub hosts Git repositories in the cloud, providing a central place for storing and managing project code. It offers features like – creating and managing repositories (which are collections of files and their files and their history), access control (who can view and modify the repository). GitHub also enhances collaboration among developers with tools that extend the basic functionalities of Git with Pull requests, code reviews and issues and project management. GitHub integrates with various CI/CD tools, enabling automated testing, building, and deployment of code changes. So, GitHub qualifies as a version control system because it is built on Git, a robust distributed VCS, and extends its capabilities with cloud-based repository hosting, collaboration tools, CI/CD integrations, and community features. These attributes make GitHub a comprehensive platform for managing and collaborating on software development projects.

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

**Answer**: Jira is a powerful project management and issue-tracking tool developed by Atlassian. It is widely used in software development and other industries to track tasks, bugs, and project progress. Jira provides a comprehensive platform for planning, tracking, and managing software development projects. It supports various methodologies, including scrum kanban, and other agile frameworks. For our project- we will be using **Kanban**. Here is how we will be using Jira for our project –

1. **Creating Issues -** When there is a need to write a document, a function, test code, or fix a bug, an issue will be created in Jira. Each issue will include:

**Title**: A short description summarizing the task or problem. This serves as a quick reference.

**Description**: A detailed explanation of what needs to be done. **This should include**:

The specific tasks involved. Any files that need to be created or modified. Any specific details or requirements for the task.

**Tags:** Each issue will be tagged with relevant milestones or other identifiers to categorize and prioritize them.

**Relationships**: Indicate if the issue is related to other issues, such as being blocked by another issue or blocking progress on other issues.

2. **Using the Kanban Board** - Our team will manage the workflow using a Kanban board in Jira. The board will have columns representing different stages of work:

**To Do**: This column will contain all newly created issues that are yet to be started. When an issue is created, it is added to this column.

**In Progress**: When a team member starts working on an issue, they will assign the issue to themselves and move it to the "In Progress" column. This indicates that the task is currently being worked on.

**In Review**: Once the work on an issue is completed, it will be moved to the "In Review" column for peer review or testing.

**Done**: After passing the review, the issue will be moved to the "Done" column, indicating that the task is completed.

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

**Answer**: A Kanban board is a visual tool used to manage and optimize workflow in software development. It helps teams visualize their work, identify bottlenecks, and continuously improve processes. Kanban, which means "signboard" or "billboard" in Japanese, originated from manufacturing practices but has been widely adopted in software development for its effectiveness in managing work.

**Advantages of using Kanban Board**-

1. **Visualizing work** – One of the primary advantages of a Kanban board is its ability to visualize the entire workflow. The visual representation helps teams understand the current status of tasks and identify where attention is needed. All team members can see what tasks are in progress, whats completed, and whats pending. A glance at the board provides immediate insight into the state of the project, making it easy to see the progress and identify any tasks that may be stuck.
2. **Improving Workflow Efficiency** - Kanban boards help improve workflow efficiency by identifying bottlenecks and optimizing processes. By setting WIP limits, teams can control the amount of work in progress at any given time. This prevents overloading team members and ensures a smooth flow of tasks. Visualizing work allows teams to spot bottlenecks quickly. For example, if tasks are piling up in one column, it indicates a problem in that phase of the workflow that needs to be addressed. Kanban promotes a continuous flow of work, reducing the wait time between different stages of development and minimizing idle time for team members.
3. **Enhancing Team Collaboration and Communication** - Kanban boards enhance team collaboration and communication by providing a centralized platform for tracking progress and discussing tasks. Teams can use the Kanban board during daily stand-up meetings to discuss the status of tasks, address issues, and plan the day’s work. Team members can collectively plan and prioritize tasks, ensuring that everyone is aligned on the project goals and responsibilities. The board provides visual feedback on the team’s performance, allowing members to celebrate successes and address challenges collectively.
4. **Flexibility and Adaptability** - Kanban boards offer flexibility and adaptability, making them suitable for various projects and team sizes. Teams can customize the Kanban board to fit their specific workflow. Columns can be added, removed, or renamed to reflect different stages of the project. Kanban boards can scale with the team and the project. Whether managing a small team or a large, complex project, Kanban can adapt to the needs of the team. Kanban can be integrated with other agile practices, such as Scrum, to create a hybrid approach that leverages the strengths of both methodologies.
5. **Promoting Continuous Improvement** - Kanban emphasizes continuous improvement, helping teams refine their processes and enhance productivity over time. Kanban encourages teams to continuously evaluate and improve their processes. Regular retrospectives can be held to discuss what’s working well and what can be improved. Kanban boards often come with built-in metrics and analytics, such as cycle time and lead time, which help teams measure their performance and identify areas for improvement. Short feedback loops, facilitated by the visual nature of Kanban, enable teams to quickly respond to changes and iterate on their processes.

Kanban boards are highly beneficial in software development for their ability to visualize work, improve workflow efficiency, enhance collaboration, and promote continuous improvement.