SPRINT 2 RETROSPECTIVE MEETING MINUTES

DATE:	START TIME:	END TIME:	MEETING PLATFORM:
November 15, 2024	10:30 PM	11:15 PM	Google Meet

ATTENDEES: Sadia(SH), Sumaiya(UJ), Jannnati Tajrimin(TM), Trisha Sarkar(TS), Rubayed All Islam(RAI)

MEETING CHAIR AND NOTE_TAKER: Akila Nipo (AN)

SPRINT 2 EVALUATION IN RETROSPECT

- **What went well?**
- **What went wrong?**
- **How can we improve?**
- **Improvements from Sprint 1**
- **What have we learnt?**

WHAT WENT WELL	WHAT WENT WRONG	HOW TO IMPROVE	IMPROVEMENTS FROM SPRINT 1
Schedule Class (Cancel,Confirm, Reschedule) :: The implementation of the Schedule Class feature was successfully completed according to the test cases defined for Test-Driven Development (TDD). The models, controllers, and views were implemented with thorough documentation, adhering to coding standards. Additionally, the Continuous Integration (CI) pipeline was successfully set up using a ci.yml file.	Test cases encountered timeout errors when run on GitHub Actions. The issue stemmed from GitHub Actions attempting to access MySQL during continuous integration, which caused the tests to fail.	The test files need to be revised and optimized to prevent issues during continuous integration (CI), ensuring that MySQL access does not cause timeouts or failures.	✓**Early Bug Detection:** ✓ With TDD, writing tests before implementation helps in catching bugs early in the development cycle, which reduces the cost of fixing defects. ✓ In **BDD**, since tests are focused on user stories, **some bugs related to implementation** details might only surface during later stages of testing or deployment.
Approve Rescheduling Request: Implementation completed Successfully with Proper UI,	Test cases showed error while doing CI/CD on GitHub Actions.	Update test cases and reviewing logs, fixing the issues, and rerunning the process may	✓TDD (Test-Driven Development) ensured that **code is developed**

Documentation, Coding Standard. The CI pipeline was successfully established with ci.yml file.				help.		**with a focus on passing tests before functionality**, leading to **more reliable and bug-free implementations**.

WHAT WENT WELL		WHAT WENT WRONG		HOW TO IMPROVE		IMPROVEMENTS FROM SPRINT 1
Filter Syllabus: **Feature Completion**: Successfully implemented the **Filter Syllabus** feature, including a well-designed **UI,** an additional **syllabus download option**. **Backend & Database**: Handled database queries effectively, and smooth backend logic. **Successful TDD**: Followed Test-Driven Development and all test cases passed successfully, ensuring robust feature functionality. **CI/CD Success**: Set up a CI pipeline using GitHub Actions, which ran smoothly and successfully deployed the feature. **Coding Standards & Documentation:** Followed coding standards and documented the code.		A few minor UI adjustments were needed after the initial implementation.		**UI Finalization**: Ensure UI designs are fully finalized before development to minimize post-development adjustments. **Expand Test Coverage**: Add more test cases to the CI pipeline for handling large datasets.		✓TDD approach ensured **logic met requirements before implementation.** ✓**Refactored logic into smaller, reusable components**, ensuring all test cases passed successfully. ✓Increased accuracy in Code **Logic by Refactoring with TDD** ✓Achieved **higher Test Coverage (93%)**

WHAT WENT WELL		**WHAT WENT WRONG**		**HOW TO IMPROVE**		**IMPROVEMENTS FROM SPRINT 1**
View Academic Calendar: View Academic Calendar feature implementation completed Successfully with Proper Documentation, maintaining Coding Standard. The CI pipeline was successfully established with ci.yml file.		Test cases showed Time out error when run on Github Actions.		Updating Test File to ensure that necessary code statement exists to resolve the Timeout Error.		✓By using **TDD the source** code generated in a way that ensures the **correctness of each individual function** while in BDD that wasn't completely possible and in that case **some of the test cases didn't pass** .
Generate Makeup Class Routine: Documentation, Coding Standard was done properly. Routine was generated according to prefferd time, day, room and class needef ro particular course.		UI could not generate routine format as expected.✓		Update the UI code. Add more test cases to the CI pipeline for handling large datasets.		✓By using **TDD** , ✓Code **quality and maintainability** is improved. ✓Enabled faster iterations and improved logic. ✓Involves writing tests before implementation, allowing quickly **verify correctness after**

The CI pipeline was successfully established with ci.yml file. Added more test cases and passed sucessfully.						**each change** which are difficult **in BDD** (implementing coding then testing) and also **some test cases are failed** in sprint 1

WHAT WENT WELL		WHAT WENT WRONG		HOW TO IMPROVE		IMPROVEMENTS FROM SPRINT 1
Update Class Representative Information: The implementation of the Update Class Representative feature was successfully completed following the test cases defined using Test-Driven Development (TDD). The models, controllers, and views were implemented with comprehensive documentation, adhering to coding standards. Additionally, the Continuous Integration (CI) pipeline was set up successfully using a ci.yml file.		The test cases failed when executed with Mocha, resulting in errors during the Continuous Integration (CI) process on GitHub Actions.		The test files, along with the model and controller functions, need to be reviewed and optimized to ensure the test cases pass successfully. This will help avoid issues during CI and improve overall reliability.		**✓Facilitates Continuous Integration (CI)** **✓TDD are usually faster to execute,** making it easier to integrate them into a CI pipeline. **✓BDD scenarios,** being broader and more complex**, may take longer to run, potentially slowing down the CI process,** especially if a **large number of scenarios are involved.**

What have we learnt?

In Sprint 2, we gained valuable insights and practical skills that enhanced our project management and collaboration. Here are the key takeaways:

Development Practices

- ### Test-Driven Development (TDD)
  - Followed TDD practices by writing tests before implementing functionality.
  - Improved code quality and ensured modular design, reducing bugs and enhancing maintainability.
- ### Continuous Integration (CI) with GitHub Actions
  - Implemented **GitHub Actions** for **CI/CD** workflows, automating tests and builds on each code push.
- ### Documentation Generation
  - Used **JSDoc** for automated documentation, making the codebase more accessible.
- ### Followed Coding Standards
  - The team followed coding standards set by the team strictly, ensuring consistent formatting, naming conventions, and code structure across the entire codebase.
  - **Improved Readability and Maintainability:** Adhering to these standards improved the readability of the code, making it easier for new team members to onboard and for existing members to maintain and extend the code.

- **Collaborative Coding**
  - Used **GitHub** for version control, branching, and team-based code review.
  - Used **Git Bash** for seamless CLI interactions with GitHub, streamlining code pushes, merges, and pulls.

Project Tools

- **Trello**
  - Managed tasks, prioritized work, and tracked progress visually.
  - Improved team organization, offering a shared view of task statuses and deadlines.

- **Discord**
  - Enabled real-time communication, making it easy to discuss and resolve issues instantly.
  - Facilitated quick updates and announcements, keeping the team in sync.

- **Toggl**
  - Used for time-tracking to log work hours and manage productivity.
  - Allowed for monitoring time spent on tasks, promoting efficient time management.

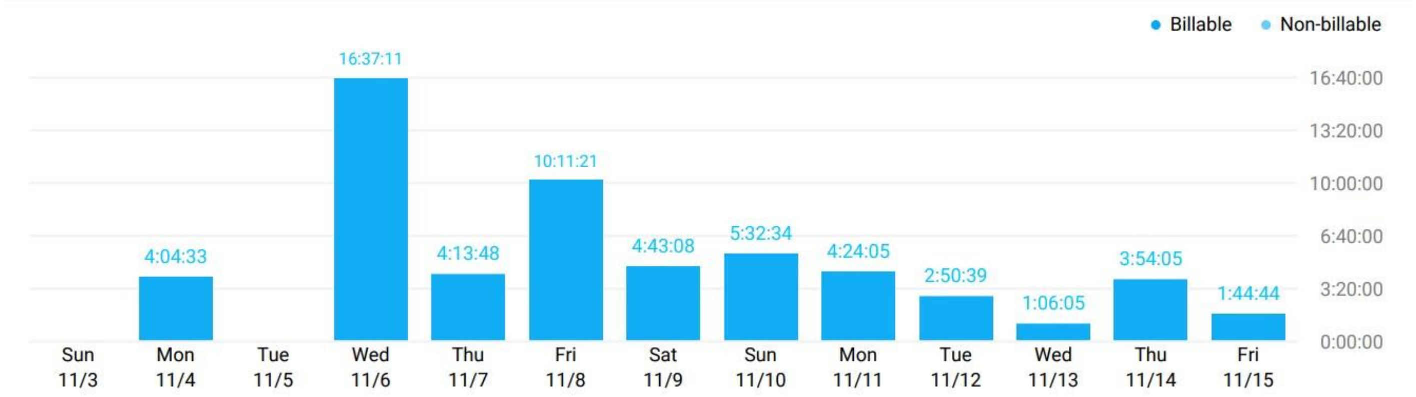
- **GitHub**
  - Served as the primary platform for version control, code collaboration, and code review.
  - Centralized all coding activities, ensuring every team member had access to the latest code.

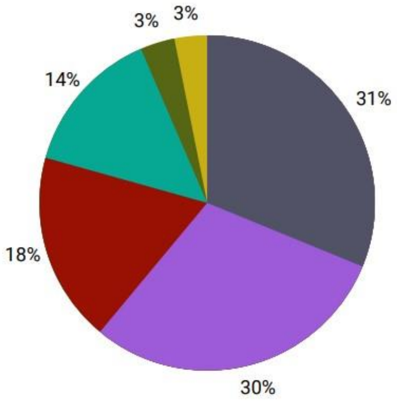
- **Git Bash**
  - Supported smooth command-line operations, streamlining code
  - management tasks.
  - Simplified branching, merging, and other Git processes.

Timely Conduct of Meetings

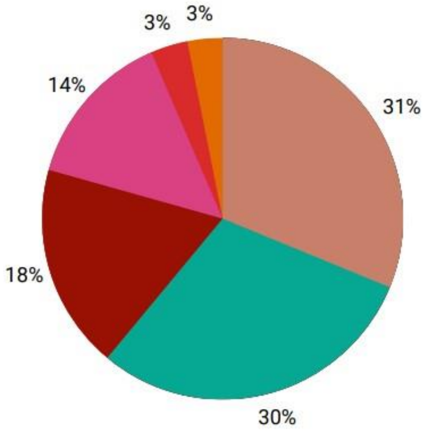
- **Regular Meetings**
  - Held consistent team meetings to review progress, discuss challenges, and set priorities.
  - Meetings provided a platform for feedback, updates, and realignment of project goals.
- **Coordination and Accountability**
  - Promoted responsibility and awareness, with each member understanding their role and contributions.

Time Tracked Using Toggl:





USER	DURATION
HO Hossainsadia2000	18:30:54
J3 Jucse29 368	17:43:29
J3 Jucse29 359	10:53:18
TM Tajrimin Mitu01	8:22:15
J3 Jucse29 370	2:00:59
UM Ummasumaiya207	1:51:18



TASK	DURATION
● Generate MakeUp Routine: Sprint-2 (Smart Class Routine Management System)	18:30:58
● Filter Syllabus: Sprint-2 (Smart Class Routine Management System)	17:43:25
● View Academic Calendar: Sprint-2 (Smart Class Routine Management System)	10:53:18
● Approve Rescheduling Class: Sprint-2 (Smart Class Routine Management System)	8:22:15
● Schedule Class: Sprint-2 (Smart Class Routine Management System)	2:01:00
● Update Class Representative Information: Sprint-2 (Smart Class Routine Management System)	1:51:17

https://github.com/JUCSE49-Mavericks/Smart-Class-Routine-Management-System/wiki/TDD-%E2%80%90-Test-Driven-Development#-report-analysis