UTM UNIVERSITI TEKNOLOGI MALAYSIA
INTERNATIONAL JOURNAL OF INNOVATIVE COMPUTING
ISSN 2180-4370
Journal Homepage : https://ijic.utm.my/

Group Project Monitoring System (TeamWatch)

Anas Ehab Nafei Mohamed[1], Marina Md-Arshad[2], Adlina Abdul-Samad[3*]
Faculty of Computing,
Universiti Teknologi Malaysia,
81310 UTM Johor Bahru, Malaysia
Email: anasnafei@gmail.com[1]; marinama@utm.my [2]; adlina6@graduate.utm.my [3]

Submitted: 22/2/2025. Revised edition: 6/5/2025. Accepted: 15/5/2025. Published online: 27/5/2025
DOI: https://doi.org/10.11113/ijic.v15n1.525

***Abstract*—Group projects in courses offer a valuable method of teaching students how to work in a collaborative setting, mirroring real-world scenarios. However, several challenges decrease the effectiveness of these projects, including the presence of free riders, the absence of a robust grading method, a lack of validation to ensure active participation, and the prevalence of lone wolves. These issues, if left unaddressed, degrade the educational benefits of group projects for students. Manual intervention by lecturers to tackle these challenges isn't practical as lecturers have many courses to follow and tracking each group manually will be difficult to do. Therefore, TeamWatch proposed web-based system with a desktop agent, designed to assist students and lecturers in the process of programming group projects. The development of TeamWatch employed the waterfall model, providing a structured approach to ensure efficient and systematic progress. The front-end is built using Django Templates, while Django served as the back-end framework. Postgres was employed as the database system, and Python was utilized for the development of the desktop agent. TeamWatch offers a range of features aimed at enhancing the group working experience. Students will be able to track their working hours in real-time, access the working hours of other group members, and even log offline working hours. Lecturers will have comprehensive visibility into each student's working hours and can generate reports providing valuable insights into individual performance and participation.**

***Keywords*—Teamwork, Time Tracking, Group Project, Team, Monitoring**

I. INTRODUCTION

Group projects are crucial for students, as they mirror the collaborative nature of real-world working environment [1]. Universities, including Universiti Teknologi Malaysia (UTM), have integrated group projects into their curricula to prepare students for team-based work environments. However, implementing group projects presents several challenges, such as fair assessment of individual contributions and equitable distribution of tasks.

Two significant problems identified are the difficulty in assessing individual contributions fairly and managing the uneven distribution of work within groups. Typically, all group members receive the same grade regardless of their input, leading to issues with "free riders" who contribute little work leaving other members with additional work and stress [2]. Additionally, there is the problem of "lone wolves" who dominate tasks, preventing others from learning effectively and they also put pressure on the other group members by neglecting their thoughts and ideas [3]. UTM currently lacks a standardized approach to address these issues, making it challenging to ensure fair participation and accurate assessment.

The proposed solution is a system that monitors individual contributions to group projects, aiming to identify free riders, lone wolves, and ensure fair evaluation. This system would track each student's work, provide data for accurate assessment, and help lecturers intervene when necessary. The project is significant as it promotes fairness, accountability, and enhanced learning outcomes in group settings. The primary aim is to develop an application that validates and tracks student contributions, ensuring that all members are actively engaged and assessed fairly in group projects.

This paper is structured as follows: Section II represents the literature review followed by the methodology in Section III. The proposed system was illustrated in Section IV. Besides, the result analysis was made in Section V. Finally, the conclusion was made in the last section, Section VI.

II. LITERATURE REVIEW

This section includes the case study that's carried out to understand and have a clearer view of the issue in hand, challenges, and other currently available systems.

A. UTM Faculty of Computing Case Study

The Faculty of Computing was chosen because most of the group projects in the Faculty of Computing are done on a computer, whether it's preparing documents or programming a system.

Group projects at UTM are initiated by lecturers who ask students to form their own groups or form the groups randomly for the students. Once groups are formed, students work on the projects with minimal lecturer supervision. If issues arise regarding workload distribution, students must manually report these to lecturers. All group members receive the same grade, with no differentiation for individual contributions.

To validate the existence of these issues a survey was conducted within the faculty of computing in UTM. The results of the survey showed that 70% of the students felt that the distribution of work among group members was unequal. 85% mentioned enocunting issues related to lack of commitment and unequal participations of other members. 70% mentioned over-reliance on a single member.

B. Current System Analysis

Group projects usually start with students forming their own groups, working with little oversight from the lecturer. Issues in work distribution are handled manually by reporting to the lecturer, and all members receive the same grade regardless of individual contributions. Some courses include peer evaluations to provide insights into each member's participation and some marks are based on that, but these evaluations vary in detail and effectiveness. For example, Programming Technique II (SECJ1023) uses a basic form, whereas Object-Oriented Programming (SECJ2154) uses a more detailed evaluation form, though both still suffer from subjective biases and lack a concrete evaluation metric.

In contrast, some courses use time-tracking software like Clockify and GitHub to monitor individual contributions more objectively. This method, was used in the application development course (SECR3104), attempts to quantify work by tracking hours spent on tasks. However, the effectiveness is limited by the lack of validation in the software, as students can log hours without necessarily working, making the data unreliable.

C. Existing Systems Analysis

To define the essential features for a proposed system and identify improvement opportunities, comparing it with existing systems is crucial. Table I shows the analysis of 3 systems that has a similar idea to the propsed system is provided.

1) Clockify

Clockify is a web-based time tracking tool that offers two primary methods for tracking time: live time tracking and offline time addition. Users can record time as they work or manually input hours after completing tasks. Features include the ability to assign work hours to specific projects and add descriptions to clarify the nature of the work. However, Clockify lacks validation for the online time tracked, any user can start tracking and go watch YouTube videos or just leave the computer for few hours and the hours will be logged as working hours which can lead to unreliable data.

2) Toggl

Toggl, similar to Clockify, provides both live and offline time tracking options. Users can log time as they work or add it afterward, with the capability to assign hours to projects and include detailed descriptions. Despite these functionalities, Toggl shares Clockify's limitations: it lacks validation for online time entries.

3) Screenshot Monitor

Screenshot Monitor stands out from Clockify and Toggl by incorporating validation into its time tracking process. It operates through a desktop agent that captures screenshots and measures mouse and keyboard activity, providing visual evidence and activity metrics for the time tracked. Users can review work on the web interface, where activities and associated screenshots are listed. However, Screenshot Monitor lacks functionality for organizing users into teams or classes, limiting its utility in the current case. Despite its robust validation features, it does not support the grouping of students, which is critical for educational or team-based projects.

4) Systems Comparsion

Each of the systems analyzed in the pervious section has a set of strengths and a set of weaknesses, the proposed system will be developed such that it incorporates the strengths of each of the systems while mitigating the weaknesses.

TABLE I. COMPARISON OF EXISTING SYSTEMS

Features/ Software	Clockify	Toggl	Screenshot Monitor	TeamWatch
Create projects	✓	✓	✓	✓
Create groups	✓	✓		✓
Create courses				✓
Different sections for each course				✓
Set deadline for project				✓
Tracking working time	✓	✓	✓	✓
Add offline time	✓	✓	✓	✓

Features/ Software	Clockify	Toggl	Screenshot Monitor	TeamWatch
Create projects	✓	✓	✓	✓
Take screenshot			✓	✓
Track activity			✓	✓
Produce report	✓	✓	✓	✓

III. METHODOLGY

This chapter outlines the methodology employed in the development of the system, covering various stages such as requirements gathering, system design, coding, testing, implementation, and maintenance. While many methodologies like the waterfall model, agile, Rapid Application Development (RAD), and the spiral model exist. The chosen methodology for this project is the iterative waterfall model, an adaptation of the traditional waterfall model that permits stage repetition. Unlike the sequential waterfall model, where each phase must be completed before moving on to the next, the iterative version allows revisiting previous stages for improvements. This flexibility is advantageous for small projects with clear objectives, where the need for substantial changes is minimal. The waterfall model's simplicity, clear structure, and extensive industry use make it suitable for the project's scope, providing a solid framework for organizing and controlling the development process. Its limitations, such as rigidity and lack of feedback, are mitigated by its iterative adaptation, which addresses these concerns by allowing for ongoing refinement. Each phase's purpose and execution within the iterative waterfall model are discussed in detail. Fig. 1 shows the waterfall model [5].

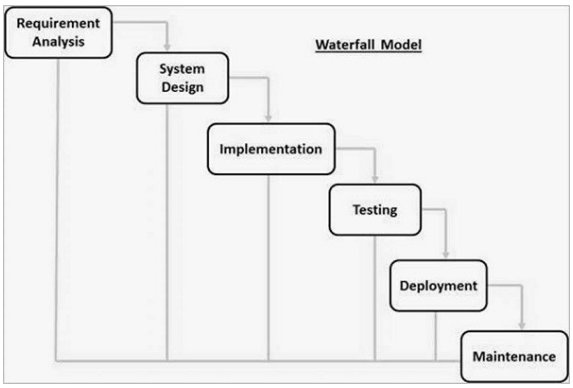

Fig. 1. Waterfall model [5]

A. Requirements Gathering and Analysis

In the initial phase, the focus is on planning and gathering user requirements through stakeholder interviews, surveys, and analysis of existing systems. This process identifies current system weaknesses and user needs, shaping the system's features and project milestones. This phase also involves extensive research into similar workflows and the development of a project plan, including Gantt charts to outline timelines and deliverables. Key outcomes include clear, well-defined requirements and insights into essential features like live time tracking and a simple dashboard, as highlighted by user feedback.

B. System Design

The system design phase transforms gathered requirements into a detailed design framework, including key system functions and interfaces. Various diagrams—use case, activity, class, and sequence—are created to document the system's structure and operations. A data dictionary, entity relationship diagram (ERD), and interface prototypes are also developed, resulting in a High-Level Design Document that comprehensively outlines the system's architecture, components, and data structures.

C. Implementation

In this phase, the system is coded and developed based on the design specifications. The project is divided into modules, each developed and tested individually before integration. The system comprises a desktop agent and a website, which are developed separately and then integrated. This phase focuses on implementing the system's features and functionalities, aiming to produce a working system that aligns with the design and requirements.

D. Testing

Testing involves verifying that the integrated system functions correctly and meets user requirements. After individual module testing in the implementation phase, the focus shifts to ensuring the system operates as a cohesive whole. This phase includes creating and executing test cases, documenting errors, and addressing issues. Test reports summarize the testing outcomes, detailing the severity of issues and resolutions. User Acceptance Tests (UAT) with students and lecturers ensure the system's reliability and performance.

E. Deployment

Once testing confirms the system's stability and error-free operation, deployment begins. This phase involves hosting the application and deploying the system to end-user devices. In the traditional waterfall model, deployment includes user training and ongoing follow-up. The primary objective is to ensure the system is properly installed and functional, enabling users to access and utilize its features effectively.

F. Maintenance

The final phase focuses on ongoing maintenance, including bug tracking, issue resolution, and integrating new features based on user feedback. It involves setting up feedback channels and regularly updating the system to address reported issues and enhance functionality. Although this phase is crucial for the system's long-term success, it is not within the scope of this project.

IV. REQUIRMENTS AND DESIGN

This section outlines the design of the system based on the requirements collected earlier. It covers various design diagrams such as use case, object-oriented class diagram, and database description. These diagrams provide a visual representation of the system's functionality, user interactions, data structure, and overall architecture. This structured approach aids in system development, ensuring that each component and interaction is clearly defined.

Finally, complete content and organizational editing before formatting. Please take note of the following items when proofreading spelling and grammar.

A. Use Case Diagram

The use case in Fig. 2 illustrates the interactions between the system and its primary users: coordinators, lecturers, and students. It showcases the different functionalities available to each user group. Coordinators manage courses, add students, and assign courses to lecturers. Lecturers view and manage their courses, track student activities, and generate reports. Students can register, log in, view courses, track and report their working hours, manage screenshots, and add offline time. The diagram captures the main functionalities and interactions within the system, providing an overview of the user-system interactions and the system's capabilities.

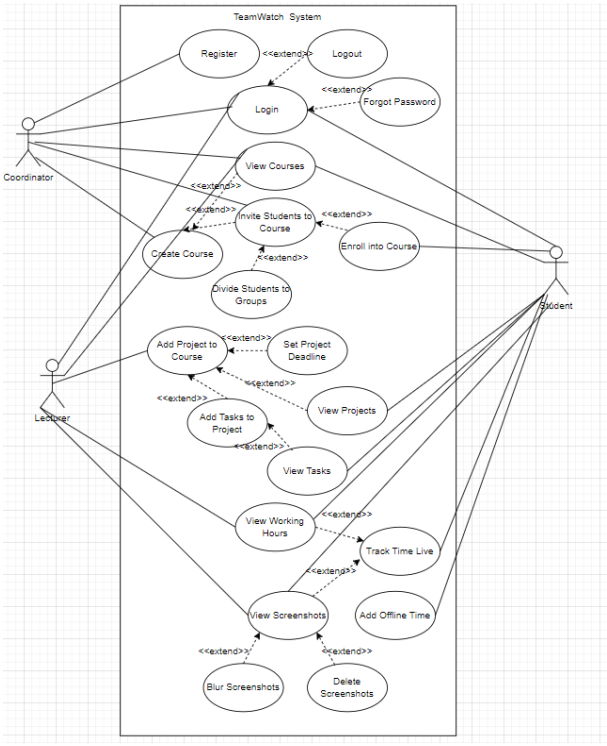

Fig. 2. Use Case Diagram

B. Object Oriernted Class Diagram

The Object-Oriented Programming (OOP) class diagramas shown in Fig. 3 depicts the structure of the system, including the main classes and their relationships. The diagram includes 11 classes: User, Coordinator, Lecturer, Student, Group, Course, Project, Working Hour, Task, Screenshot, and Report. Each class has attributes and methods relevant to its role in the system, and the relationships between the classes are clearly defined, showing how they interact to form the complete system.

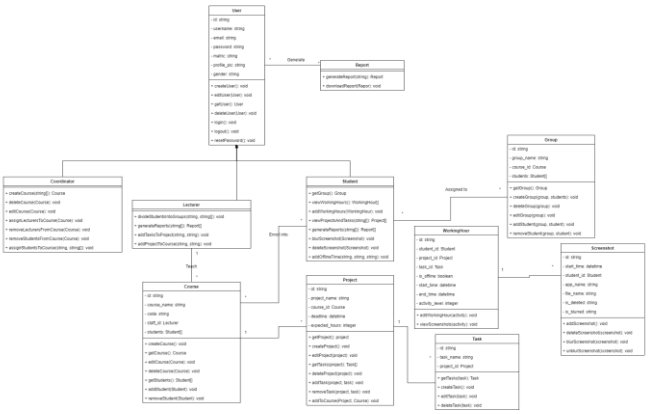

Fig. 3. Class Diagram (OOP)

C. Security Elements

Security is a crucial aspect of the system, focusing on protecting user credentials, student data, and ensuring secure communication.

In this system, security was a priority. Encryption and hashing are used to secure passwords. Passwords are stored as hashed values, making it difficult for attackers to retrieve them in case of a data breach. Authentication and access control mechanisms ensure that only authorized users can access and manipulate data. Middleware is used to handle requests and enforce access control. HyperText Transfer Protocol Secure (HTTPS) is used for secure data transmission between system components and user devices, protecting against eavesdropping and data interception. Django Object-Relational Mapping (ORM) is used to prevent Structured Query Language (SQL) injection by ensuring proper input validation and parameterized queries.

D. Network Elements

Network elements are essential for efficient system operation, focusing on data management and communication. A cloud database is used for scalability and high availability, ensuring efficient data storage and retrieval. Multiple Application Programming Interface (APIs) facilitate seamless communication between system components (server, database, client applications), enhancing functionality and providing real-time data access.

V. RESULT

This section will discuss the achievements of the project.

A. Web Application

The web application was successfully developed, the system maintained a simple modern design that's efficient and good looking. All the functions were successful. The web application is used by the 3 different user types, the coordinator can use the web application to create and manage courses, lectures, and students. The lecturers can use the system to view their courses, create and manage projects and groups, and view the activity of the students (working hours) with all of their details including the screenshots and activity levels. Finally, for the students, they use the system to view their courses, projects, their other group members and their own activity (working hours).

The system is full of features that make using the system easy for the different users, for example, for adding the students, lecturers or courses, there are options to import from Excel files. For the reporting module, there are options to export to Portable Document Format (PDF) or Excel. In the groups there are buttons to collapse/expand all. Even though such options aren't compulsory for the working of the system, it makes the user experience much better. Fig. 4 displays the student's view of group activity, showing working hours and contributions. Fig. 5 provides the student view of individual activity, detailing tasks and screenshots; Fig. 6 shows the student view of screenshots taken during work sessions.

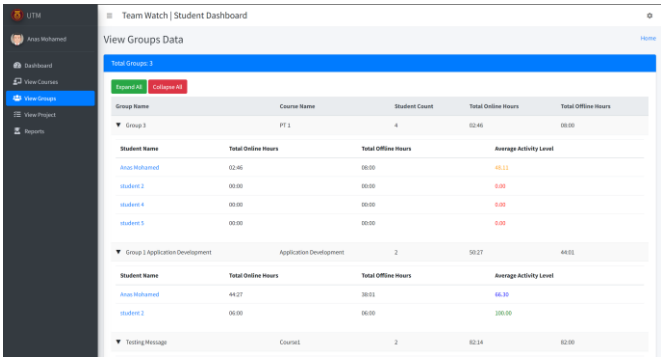

Fig. 4. Student View Group Activity Screen

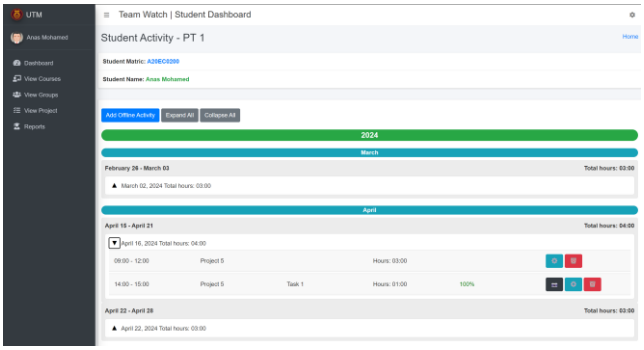

Fig. 5. Student View Student Activity Page

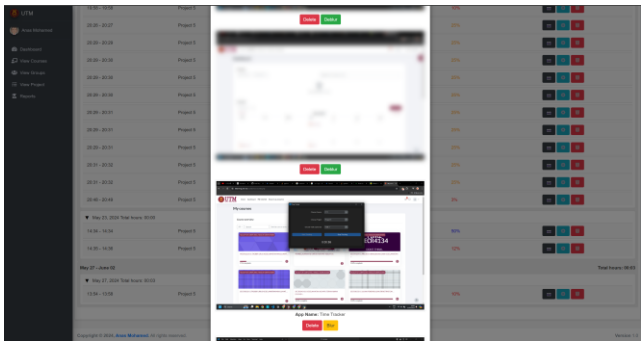

Fig. 6. Student View Screenshots Page

B. Desktop Agent

The desktop agent is used by the students, it's a simple application that allows the student to specify the course, project, and optionally the task they working on and then start or stop tracking.

While tracking is active, the system takes screenshots and identify the running application, and also calculate the activity of the keyboard and mouse. These data are then sent and stored at the server. The system also has the feature of stopping the time tracking if no activity (mouse and keyboard) is detected for a while. Students are notified of the various events, like when tracking starts, when inactivity is detected, when a screenshot is taken, and when tracking is stopped. Fig. 7 depicts the desktop agent interface for real-time time tracking and activity monitoring.

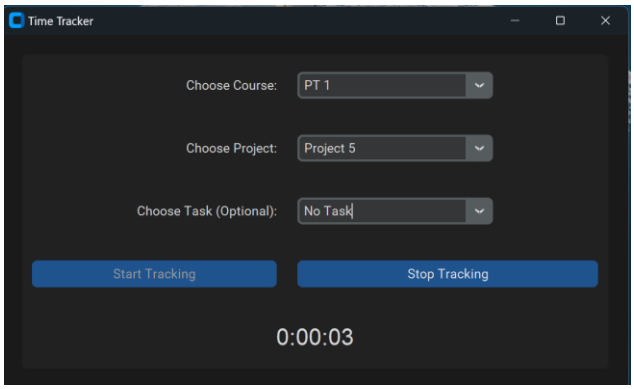

Fig. 7. Desktop Time Tracking Screen

C. User Acceptance Tests

Both the web application and the desktop application went through the user acceptance tests and received generally good feedback as shown in Table II. Some requested improvements were setting the frequency of screenshots manually and being able to set configuration settings by mouse and keyboard use of activity levels – all these will be taken into account if as suggestions for future improvements. One mentioned feature that was added in this version of the system was validation on project deadlines: a student cannot add working hours on a project whose due date has already passed. For the desktop

application, the feedback was also quite good, the students had a few suggestions. For instance, auto-start and background operation are some of the features mentioned by the testers. These are considered complementary, and will be considered as recommendations for future versions.

TABLE II. USER ACCEPTANCE TEST BASED ON ACTOR

Actor: Lecturer			
Related Module	***Requirements***	***Comments***	***Result***
Manage Project	• Be able to view projects for each course. • Be able to edit projects. • Be able to delete a project.	Students should be able to track for projects that the deadline of has passed already.	***Success***
View Activity Per Course	• Be able to view detailed activity for a course . • Be able to edit an activity. • Be able to view the screenshots associated with an activity. • Be able to blur a screenshot. • Be able to delete a screenshot.	Can add an option for lecturers to change the frequency of the taken screenshots and adjust the activity levels calculations (Keyboard and Mouse required clicks to be considered 100 percent active).	***Success***
View Activity Using Reports	• Be able apply filters. • Be able to group by the results to either students or groups. • Be able to export to PDF. • Be able to export to Excel.	Enhancements on the design can be made.	***Success***
Actor: Student			
Related Module	***Requirements***	***Comments***	***Result***
Tracking Working Time Using Desktop Agent	• Be able to select the course • Be able to select a project • Be able to select a task. • Be able to start tracking. • Be able to stop tracking.	• Can add an option to set the application to start automaticall y • Can add an option to let the program run in the background	***Success***

VI. CONCLUSION

The TeamWatch system has been successfully developed to meet its objectives and within the appropriate scope. TeamWatch effectively mitigates common issues in group projects such as free riding, unequal work distribution, lone wolves, and lack of participation tracking. The system combines a web-based platform and a desktop agent, enabling seamless tracking of individual contributions and working hours. The information is stored in a cloud database, accessible to lecturers and students through a Django-based application.

To evaluate the system's accuracy and functionality, user testing was conducted with students and lecturers. Feedback confirmed that TeamWatch functions as intended, with suggestions for future enhancements. Future improvements could include integration with other systems like Github or Jira or e-learning and expanding reporting capabilities for better insights.

ACKNOWLEDGMENT

The authors wish to thank everyone who provided their insights and advice for this system whether students or lecturers.

CONFLICTS OF INTEREST

The author(s) declare(s) that there is no conflict of interest regarding the publication of this paper.

REFERENCES

[1] Jun, H. (2010). Improving Undergraduates' Teamwork Skills by Adapting Project-based Learning Methodology. *2010 5th International Conference on Computer Science & Education.* Doi:10.1109/iccse.2010.5593527

[2] Gitinabard, N., Okoilu, R., Xu, Y., Heckman, S., Barnes, T., & Lynch, C. F. (2020). Student Teamwork on Programming Projects. What Can Github Logs Show Us. *Educational Data Mining*. http://files.eric.ed.gov/fulltext/ED608053.pdf.

[3] Barr, T. F., Dixon, A. L., & Gassenheimer, J. B. (2005). Exploring the "Lone Wolf" Phenomenon in Student Teams. *Journal of Marketing Education*, *27*(1), 81–90. Doi:10.1177/0273475304273459

[4] Feichtner, S. B., & Davis, E. A. (1984). Why Some Groups Fail: A Survey of Students' Experiences with Learning Groups. *Journal of Management Education*, *9*(4), 58–73. Doi:10.1177/105256298400900409

[5] Tutorialspoint. (2019). *SDLC Waterfall Model*. Www.tutorialspoint.com. https://www.tutorialspoint.com/sdlc/sdlc_waterfall_model.htm.