SORT IT! GROUP MANAGEMENT SYSTEM

NUR ALEEYA DEENA BINTI MOHAMMAD ISHAK

UNIVERSITI TEKNIKAL MALAYSIA MELAKA

SORT IT! GROUP MANAGEMENT SYSTEM

NUR ALEEYA DEENA BINTI MOHAMMAD ISHAK

This report is submitted in partial fulfilment of the requirements for the

Bachelor of Computer Science (Database Management) with Honours.

FAKULTI TEKNOLOGI MAKLUMAT DAN KOMUNIKASI

UNIVERSITI TEKNIKAL MALAYSIA MELAKA

2024

DECLARATION

I hereby declare that this project report entitled

SORT IT! GROUP MANAGEMENT SYSTEM

is written by me and is my own effort and that no part has been plagiarized without citations.

STUDENT:  Date: 6 September 2024

(NUR ALEEYA DEENA BINTI MOHAMMAD ISHAK)

I hereby declare that I have read this project report and found this project report is sufficient in term of the scope and quality for the award of Bachelor of Computer Science (Database Management) with Honours.

SUPERVISOR:  Date: 6 September 2024

(DR NUR ATIKAH BINTI ARBAIN)

DEDICATION

To all those people who have been my backbone when writing this thesis, this thesis is dedicated to you.

To my family especially ibu, you are my source of strength and always believed in me when I didn't believe in myself. This encouragement and sacrifices you gave me gave me the right start I needed to be able to push through.

For my friend (Fiqa, Wawa, Zaza and others), thanking for the company and support, which always helps me to struggle and carry on. This simply means that occupying your seats has been a great source of help, particularly when my psychological integrity was in turmoil.

To my supervisor: without your advice, understanding nature and support as well as sharing of knowledge on how to go about this final year project, it could not have been done. You have encouraged me and stood behind me when the way appeared steep.

Nonetheless, these are all the feats I have achieved today despite the obstacles that I had to struggle with and the daily fluctuations of mental health. Such a great achievement is attributed to the strength that comes along with such a stupendous support base.

You all are amazing; thank you for being with me and for supporting me.

ACKNOWLEDGEMENTS

First and foremost, my appreciation extends to my family, and this is because. Love, hope and encouragement you give me have been my pillars throughout this period. I want to start by saying that I wouldn't have been able to accomplish this journey if it wasn't for your confidence in me.

I would also like to express my deep gratitude to one close person, without whom I could never complete such a project as the present work. Nur Melissa Maisarah Binti Shahrulnizam, this is to express my appreciation to you for availing your support and company during the process through which my mind has been considerably unstable. I am so grateful that you have been of great support throughout the time.

To my supervisor Madam Nur Atikah Binti Arbain, I am particularly grateful for his advice, knowledge, and tolerance all through my writing of this thesis. It is my pleasure to acknowledge that you have been very much helpful in this project through your valuable comments and consistencies. I am very much thankful for your training that you have given to me and for challenging me to the fullest.

I would also like to thank my professors and fellow students who helped me with ideas and motivation along the way.

Finally, I would like to say that one has to be strong and persistent as well as never give up on herself. Thus, I have been able to undertake this final year project despite the hurdles that come with being mentally ill. It is therefore for this reason that this accomplishment mirrors and acknowledges effort from all the names listed above and my ability to work beyond barriers.

To all of you for your unrelenting support and for having faith in me as a writer, this thesis is dedicated to you and we are all in this together.

ABSTRACT

Group work is an essential component of modern education, promoting collaboration and shared learning experiences. However, organizing students into balanced and fair groups can be a challenge for educators, often leading to issues of inequity and inefficiency. This project introduces *Sort It!* a web-based application designed to streamline group formation by categorizing students based on their GPA into three distinct classes: first class, second class, and third class. This approach ensures that each group is composed of students with a diverse range of abilities, fostering fairness and minimizing bias. By applying a systematic sorting method, *Sort It!* eliminates the need for manual group assignments, saving educators time and effort. The goal of this system is to create balanced, effective group dynamics that enhance the quality of collaborative work and improve student outcomes. The application promotes equity by ensuring that all students, regardless of their academic performance, are distributed fairly, supporting a more inclusive learning environment. The expected outcomes include more efficient group formation, fairer distribution of group members, and enhanced collaborative project success, ultimately improving both the learning process and project results in educational settings.

ABSTRAK

Kerja berkumpulan adalah komponen penting dalam pendidikan moden yang menggalakkan kerjasama dan pengalaman pembelajaran bersama. Walau bagaimanapun, mengatur pelajar ke dalam kumpulan yang seimbang dan adil boleh menjadi cabaran bagi pendidik, yang sering membawa kepada isu ketidaksaksamaan dan ketidakcekapan. Projek ini memperkenalkan *Sort It!* sebuah aplikasi berasaskan web yang direka untuk memperkemaskan pembentukan kumpulan dengan mengkategorikan pelajar berdasarkan GPA mereka kepada tiga kelas yang berbeza: kelas pertama, kelas kedua, dan kelas ketiga. Pendekatan ini memastikan setiap kumpulan terdiri daripada pelajar dengan pelbagai kebolehan, mewujudkan keadilan dan mengurangkan prasangka. Dengan menggunakan kaedah pengisihan yang sistematik, *Sort It!* menghapuskan keperluan untuk pembahagian kumpulan secara manual, menjimatkan masa dan tenaga pendidik. Matlamat sistem ini adalah untuk mencipta dinamik kumpulan yang seimbang dan berkesan yang dapat meningkatkan kualiti kerja kolaboratif serta memperbaiki hasil pembelajaran pelajar. Aplikasi ini menggalakkan kesaksamaan dengan memastikan semua pelajar, tanpa mengira prestasi akademik, diagihkan secara adil, menyokong persekitaran pembelajaran yang lebih inklusif. Hasil yang dijangka termasuk pembentukan kumpulan yang lebih cekap, pengagihan ahli kumpulan yang lebih adil, dan kejayaan projek kolaboratif yang lebih baik, yang akhirnya memperbaiki proses pembelajaran dan hasil projek dalam suasana pendidikan.

Table of Contents

Page

List of Figures

Page

List of Tables

Page

CHAPTER 1: INTRODUCTION

1.1 Introduction

Collaborative learning has been emphasized as a cornerstone of modern education, with teamwork and collective achievement among students being highlighted. The acquisition of knowledge is not merely prioritized; instead, crucial interpersonal skills such as critical thinking, problem-solving, and reasoning are also fostered. However, logistical hurdles are often posed for teachers when collaborative activities are orchestrated. Valuable teaching time is consumed by the manual formation of student groups, and ensuring fairness while accommodating diverse student backgrounds has remained a persistent challenge.

To tackle these obstacles, the creation of a web-based application has been proposed to streamline the group formation process. The sorting of students into groups would be automated according to user-defined parameters and rules. Through the use of technology, the burden of manual group formation can be alleviated for teachers, allowing more focus to be placed on facilitating meaningful learning experiences. Moreover, equity and inclusivity could be promoted, as group assignments would be made fair and considerate of each student's unique strengths and needs.

In essence, this proposed solution aims to harness the power of technology so that the effectiveness and efficiency of collaborative learning can be enhanced in educational settings. By providing teachers with a user-friendly tool for group formation, the creation of environments conducive to student collaboration and holistic skill development can be empowered.

1.2 Problem Statement

In various educational settings such as classrooms, workshops, project and more, students need to collaborate in groups. Traditionally, teachers manually form these groups based on the predefined criteria made by the teachers. Unfortunately, the traditional way of manually do

grouping often fall short in addressing a few challenges such as unfairness, time-consuming and fixed and inflexible groups.

- **Unfairness**: The traditional way of creating group can unintentionally favour certain students or exclude others. Potential biases related to gender, ethnicity, social dynamics, and skill levels, including the presence of highly capable students within a single group, may inadvertently influence outcomes.
- **Time-consuming**: The teachers will spend most of their time assigning the group for the student. The manual process of creating balanced groups for large classes is highly inefficient. Valuable teaching or learning time is often consumed by administrative tasks, which can be a significant waste. Implementing an automated system would expedite group formation, thereby freeing up time for more meaningful interactions.
- **Fixed and Inflexible Groups**: Once group are formed manually, they remain static throughout the whole project. Fixed groups do not adapt to changing circumstances. As student skills may improve over time or maybe the availability might change due to external factors such as illness or scheduling conflicts, dynamic group that adjust based on evolving needs are more effective for collaborative learning

1.3 Objective

This project embarks on the following objectives:

- **Minimize Unfairness**: This objective directly addresses the problem of unfairness by creating groups without favouring specific participants based on personal characteristics. By randomly assign participants using the mix of Monte Carlo, Randomized Quick Sort algorithm and customised random sampling to groups and ensuring diversity such as skills within each group, we can promote equal opportunities and a positive learning environment.

- **Increase Efficiency and Time Optimization**: Addressing these objectives resolve the issue of time-consuming process by swiftly forming balanced group without manual efforts. By importing data that the teachers have such as Excel into Sort It! it can reduce the time consume by the teachers to sort the student into groups. Reducing the administrative overhead allows more time for teaching and learning will maximize the valuable classroom time with an efficient group formation.

- **Foster a Dynamic and Adaptable Groups**: This objective directly tackles the challenges of fixed and inflexible group by creating groups that adapt to changing circumstances. By periodically re-randomize group for example at the beginning of a new project phase, it will allow participants to express their preferences such as what skills they seek. We can use machine learning models to predict optimal compositions. Since we do not have any labelled data, we can use unsupervised learning to help make the groups adaptable. This will enhance collaboration and accommodate evolving needs.

1.4 Scope

- Login Module

i. In this module, the user will use their email and password for login to ensure the system cannot be accessed or changed by any unauthorized user.
ii. Allow new users to register themselves by providing necessary information such as email, name, phone number, password, and creating login credentials.

- Modules

- **Data collection and Preprocessing Module**: Collect relevant data about students from the school or by using data collection module, and preprocess the data by cleaning, transforming, and organizing data for further analysis. Use machine learning, unsupervised learning to train the data.

- **Group Formation Algorithm**: Design and implement the core algorithm for group formation. Randomly assign participants to groups while adhering to user-defined rules such as no duplicates, different skills or skills diversity. Make sure to consider factors like availability, preferences and constraints.
- **User-Interface (UI) Module**: Create an intuitive web interface for users (teachers, organizers, etc.) that include input user-defined rules (constraint, preferences), display randomized groups clearly and allow user feedback or adjustments.
- **Dynamic Adaptation Module**: Implementing mechanisms for dynamic adjustments by re-randomizing groups periodically at the start of a new project and allowing student to express preferences like preferred teammates or skill they seek.
- **Feedback and Evaluation Module**: Collecting feedback from users (teachers, student, etc.) on group satisfaction and evaluating the effectiveness of the system based on user experience and learning outcomes.
- **Integrations and deployment Module**: Integrate all modules into a system and deploy the system in educational settings such as classrooms.

1.5 Project Significant

- Enhanced collaboration
    - The Sort It! Group Randomizer System helps in Improving the project-related cooperation of the teams because it establishes well-balanced and heterogeneous groups. As to skill, prefer, and availability factors the system qualifies that more suitable personnel are brought together in the teams, thus more productivity and innovation is likely to be achieved.
- Efficient Resource Utilization:
    - The system also follows the aspect of resource management where proper participants are grouped together with due regard to their abilities. This makes sure

that each group that is formed has the necessary competencies of handling each task in a specific project in the most efficient way that would eliminate squandering of resources.

- Improved Project Outcomes:
    - In this respect, the ability to foster collaboration and design teams with the help of the system enhance the results of the projects. Fewer project setbacks are likely to go unnoticed or unaddressed, while more opportunities are likely to be seized that leads to the production of higher quality consumables and ultimately, project success.
- Enhanced User Experience:
    - The system also improves the experience of project participants where the intention of the system is to ease formation of a group. It makes it very convenient for the participants to express their preferences, see the compositions of the groups, and communicate with the other members of the team, which generally makes the user experience more several and positive.

- Promotion of Diversity and Inclusion:
    - The system also enriches the teams involved in projects with diversity and inclusion since among members of teams selected; some are bound to come from different backgrounds and have different opinions on various issues. This gives diversity in teamwork a positive aspect since it caters for team creativity and enhances the achievement of good results because of the multiple perspectives in tackling the issues.
- Facilitation of Project Management:

- The system enables project management by offering project managers relevant information on group compositions and interactions. In this case, the project leaders

can monitor the group performance and other issues of concern and be able to make follow-ups hence improving the management of the project.

- Support for Research and Innovation:

  - Universality of the system fosters research and innovation since it facilitates the formation of cross-disciplinary research teams. Merging people of diverse backgrounds enriches the work of the system with different points of view and contributes to the introduction of interdisciplinarity into solving multifaceted problems during research.

1.6 **Expected Output**

- Web-Based Application
  - In this case, a fully functional, web-based application that allows the users to set the parameters and rules for group formation and selects the best setting itself.
  - Interface design to be friendly and easily manageable from the teachers' and administrators' end.
- Efficient Group Formation:
  - Program that directly ensures that people are grouped thus eliminating the time and effort that would otherwise be used when grouping himself.
  - Opportunity to input the student information into the system from other sources, for instance, Excel files, in order to configure the system rapidly.
- Fair and Balanced Groups:
  - Predefined groups in that the distinguished rate and selection of the groups makes them fairly randomized.
  - Using algorithms or probabilities in assignments (e.g., Monte Carlo, Randomized Quick Sort) so that this strategy can produce equality; students' skills and characteristics are balanced.
- Dynamic and Adaptable Groups:
  - Procedures of re-organizing groups to fit the changing dynamics like the different phases of a project or changes in the students' requirements.

- Ability of students to provide expectations and needs while increasing group's flexibility and cooperation.

- Enhanced Collaboration and Learning:
  - Enhanced students' interactivity because students were placed in well-mixed groups in terms of their strengths and weaknesses.
  - In the specific area of group projects 'fair' and 'efficient' distribution of students has a favourable effect on students' learning engagement and achievements.
- Time and Resource Optimization:
  - There is the opportunity to reduce overhead administrative duties so that teachers can spend more time on actual teaching and on creating good learning environments.
  - Class time management, optimization of the learning time available in a classroom.
- Inclusivity and Equity:
  - Equality since the group assignments have to be made with respect to each student's ability and/or disability.
  - Formation of an academic climate in the classroom that would enable all students in the class to contribute in one way or the other as per their potential.
- Feedback and Evaluation:
  - Usage of feedback from the users (teachers and students) and integration of feedback into the system towards improvement.
  - Evaluation of the supportive role of the developed system in improving collaboration in learning and accomplishing the project goals.
- Documentation and Maintenance:
  - Detailed records of the processes of installation, implementation, and management of the system.
  - Structure of components as well as clear commenting to make changes and debugs as easy as possible.

1.7 Conclusion

In conclusion, Sort It! Groups Formation System aims to solve major issues of the processes of collaboration learning and project management in educational institutions. This web-based application will therefore be a tool to coordinate the grouping of students and fairly, efficiently and with flexibility do so. The system is free of bias, efficient, and develops intelligent groups depending on the contextual factors and students' requirements through complex algorithms and machine learning. This approach does not only improve the process of collaboration with co-workers but also helps in achieving diversity and inclusion, both of which will benefit the projects that we will work on and produce better results that can help in the learning process. The features of the system's architecture and the presence of such modules as login, data acquisition module, formation of groups, user interface, dynamic adaption, feedback, integration, etc make the system one of the most robust and user friendly. In this way, the indicated system contributes to rational use of resources and offers useful information related to the practical management of projects to both educators and students. Also, the encouragement of interdisciplinary cooperation enriches both innovation and research, thus benefiting the overall education agenda. Finally, the Sort It! Group Randomizer System lets the teachers concentrate on the process of delivering the lesson and allows students to be innovative when sharing and achieving group formation. This project shows that the idea of using a technology in education to improve the organizational dynamics of grouping can be an efficient approach, which is fair and flexible to meet modern demands of the educational system.

CHAPTER 2: PROJECT METHODOLOGY AND PLANNING

2.1 Introduction

Sort It! will be developed using the Agile approach, a methodology that provides the flexibility and responsiveness required for projects that evolve over time based on continuous feedback and real-world application. Agile is particularly well-suited for this project because it allows for small, incremental improvements through iterative cycles, which is essential when dealing with a system that aims to streamline group formation in educational settings. The development process will be broken into several stages, each involving active input from stakeholders to ensure that the system meets the needs of its users—both teachers and students.

The project will begin with the planning phase, where the overall goals, objectives, and deliverables are clearly defined. This phase will include identifying the key participants, their roles, and the specific tasks they will undertake. User stories and use cases will be gathered from educators and other stakeholders, providing a detailed understanding of the system's requirements. These stories will guide the system's design, ensuring that the final product addresses the real challenges faced by teachers in organizing group work. This phase will also include risk assessment and the establishment of contingency plans, ensuring that potential obstacles are anticipated and managed effectively.

Next is the design phase, where the conceptual framework of Sort It! will be created. This includes developing wireframes and prototypes for the user interface to ensure the system is intuitive and user-friendly. The database schema will also be designed at this stage, laying the groundwork for how student data, such as GPAs, will be processed and used for group formation. The design will cover all major components, including the login module, the group formation module, dynamic adaptation for changing student data, and feedback mechanisms. The goal during this phase is to create a scalable and flexible system architecture that can adapt to future updates and requirements.

The development phase will take place over several 2- 4 weeks sprints, following Agile's iterative structure. In each sprint, high-priority features will be developed, tested, and reviewed. This approach allows for continuous progress while ensuring that any issues are addressed

promptly. The Agile framework encourages collaboration between developers and users, meaning that educators will have the opportunity to provide feedback during each sprint. As a result, the system can be fine-tuned during development, preventing larger issues from arising later in the project. By the end of each sprint, functional components of Sort It! will be ready for testing and refinement.

Testing will be an integral part of each development sprint, ensuring that the system functions as expected at every stage. This phase will include unit tests, integration tests, system tests, and user acceptance tests (UAT). The purpose of this rigorous testing process is to catch and address any potential bugs, usability issues, or functional gaps before deployment. The continuous feedback loop during testing will allow stakeholders to assess the system's performance, ensuring that it meets the needs of its users and works seamlessly within educational settings.

The deployment phase will follow, where Sort It! will be implemented in a selected educational institution as a pilot. During this phase, real-world testing will be conducted, and the system's performance will be carefully monitored to identify any areas that need further adjustment. The pilot implementation will provide valuable insights into how the system functions in an actual classroom environment, allowing for additional refinement before a broader rollout.

Finally, the post-deployment and maintenance phase will ensure that the system continues to operate effectively and that updates are made as necessary. This phase includes regular monitoring, feedback collection, and implementing updates or patches based on new requirements or challenges that arise from its use. Continuous updates will keep Sort It! aligned with the needs of its users, ensuring that the system remains stable, secure, and relevant over time. Agile's flexibility allows for these iterative improvements without disrupting the system's overall functionality.

The entire project is expected to span approximately 7 months, including planning, design, development, testing, and deployment. Throughout this period, Agile's iterative and flexible nature will ensure that the system evolves in line with user feedback and real-world needs. By applying the Agile methodology, Sort It! will be developed as a highly functional, user-friendly application that enhances collaborative learning in educational institutions. The project aims not only to simplify the group formation process but also to ensure fairness, inclusivity, and

efficiency in group assignments, ultimately improving the quality of group work and student outcome.

2.2 Project Methodology

Several phases that make up the Database Life Cycle (DBLC) are as follows, to enhance systematic development, implementation, and maintenance of a database system. For the "Sort It!" project, which aims to streamline the process of forming student groups based on various criteria, the DBLC phases can be described, and the tasks planned as follows:

2.2.1 Database Initial Study

At the planning stage of the Sort It! project, the main focus is on defining the system's need, scope, and goals. This involves gathering detailed requirements from stakeholders such as teachers, administrators, and students using questionnaires, interviews, and group discussions. The information collected is essential for understanding the current challenges in group formation and identifying key features that the system should address. A feasibility study is then conducted to assess the technical, operational, and financial viability of the project. This includes evaluating the available technology infrastructure, ensuring that the system aligns with the workflow of educators, and estimating the project's costs and benefits.

Once the feasibility is confirmed, the scope of the project is clearly defined. This includes specifying the data requirements, such as student GPAs, group preferences, and any other relevant information necessary for creating balanced and fair groups. The goals of the system are also outlined, focusing on improving efficiency, fairness, and flexibility in group formation while reducing the workload for teachers and promoting inclusivity in the classroom.

Next, resources are allocated, including the assignment of project personnel, hardware, and software tools needed for development. A detailed timeline is created, breaking the project into phases such as design, development, testing, and deployment, with specific milestones and deliverables for each stage. This structured plan ensures that the project proceeds smoothly, with built-in contingencies to handle potential risks or delays. With these elements in place, the project is positioned for successful execution, aiming to meet the needs of its users and improve collaborative learning in educational settings.

2.2.2 Design

This pertains to the conceptualization of the document plan for the database in relation with the given project requirements. Conceptual design is initially done, with creation of an Entity-Relationship Diagram (ERD) for modelling the Students, Groups, Teachers, Assessment, and their relations. This is followed quickly by the logical design in which tables are defined, primary keys, and foreign keys are identified, and relationships ensured while attempting to normalize the table to eliminate redundancy. The organization of the database design aspect deals with defining the external organisation of the database that include; layouts, indices, and partitions that help in nature of the database and storage functionality. The activities involved in this phase include the coordination with the database designers and the reviewing sessions**.**

2.2.3 Implementation

This phase pertains to the conceptualization and design of the database plan, closely aligned with the Agile approach to ensure flexibility and responsiveness to evolving project requirements. In Agile, the database design process is iterative, allowing for continuous feedback and adjustments. The initial step is the conceptual design, where an Entity-Relationship Diagram (ERD) is created to model the key entities—Students, Groups, Teachers, and Assessments—along with their relationships. This ERD provides a clear, visual representation of how data will be organized, laying the groundwork for future development.

Following this, the logical design stage is carried out iteratively. In each sprint, tables are defined, and primary and foreign keys are identified. This ensures that relationships between tables are properly established while adhering to the project's requirement for a normalized database to eliminate redundancy. Since Agile encourages flexibility, feedback gathered during early development phases can be used to modify or improve the database structure as new needs arise or as the system scales.

Agile's focus on collaboration and stakeholder involvement is evident in the organization of the database design activities. The external aspects of the database, such as layouts, indices, and partitions, are defined to enhance performance and storage efficiency. Regular coordination with database designers, combined with frequent review sessions, ensures that the database structure evolves alongside the system's functionality. This iterative process allows the

database to remain flexible and scalable as user requirements change, reflecting the adaptability and responsiveness at the core of the Agile methodology.

2.2.4 Testing

The testing and evaluation phase is a critical component in the Agile approach, where continuous validation and feedback loops ensure that the database functions as expected and meets performance benchmarks. In this phase, the functionality of the entire application is rigorously tested, specifically focusing on CRUD (Create, Read, Update, and Delete) operations to verify that the database can handle these actions efficiently. Agile's iterative nature allows for testing in sprints, meaning that potential issues can be identified and resolved early in development, ensuring smooth database performance as the system evolves.

Performance testing is also conducted to assess the database's ability to manage the expected traffic load. This is particularly important in educational settings where multiple users—students, teachers, and administrators—may be accessing the system simultaneously. The database is evaluated under various loads to ensure it maintains optimal performance without bottlenecks. In Agile, this type of testing is repeated at various stages to continuously assess performance as new features are added.

Additionally, data integrity testing ensures that all constraints and relationships between tables are properly enforced, and that the accuracy of the data is maintained. This phase validates the correctness of primary and foreign keys, as well as any rules governing data entry, ensuring the reliability of the database. Agile's emphasis on collaboration plays a role here, as feedback from stakeholders, including teachers and administrators, is gathered continuously. This feedback is used to fine-tune the system, addressing any issues or inconveniences users might encounter, thus ensuring that the database is not only functional but also user-friendly and aligned with real-world needs.

This iterative testing process ensures the database remains reliable, efficient, and scalable, supporting the long-term goals of the Sort It! application in providing a seamless, fair, and effective group formation process. By incorporating regular feedback and validation, the Agile approach ensures that the system is constantly improving and adapting to meet the evolving requirements of its users

2.2.5 Operation and Maintenance

In the operation phase, the database is deployed to the production environment, marking a significant milestone in the Agile approach. During this phase, the database is transferred to the production server, where its performance, usage, and overall health are carefully monitored. The Agile methodology's iterative nature is again crucial here, as any issues that arise can be promptly addressed through continuous feedback and adjustments, ensuring that the database functions smoothly in a live setting.

Technical assistance plays a key role during this phase, as any problems or errors encountered in the production environment are quickly resolved. Agile encourages ongoing collaboration between developers, database administrators, and users, allowing for efficient troubleshooting and quick deployment of solutions. This real-time support helps minimize downtime and ensures that the system remains fully operational for end users. Additionally, if new challenges emerge during operation, Agile's flexible approach allows for quick patches and updates without disrupting overall functionality.

To safeguard against data loss, robust data redundancy and backup procedures are implemented during this phase. This ensures that in the event of a system failure or error, the data can be restored with minimal disruption. Agile's focus on adaptability means these backup and restoration procedures can evolve over time, ensuring the system remains secure and reliable as the user base grows or new features are introduced.

Continual monitoring is established as part of the Agile workflow, using automated tools and methods to track the database's performance over time. This monitoring ensures that key metrics, such as query speed, server load, and data accuracy, are constantly checked, allowing the team to detect and resolve potential issues before they escalate. Regular health checks and audits are also performed to ensure that the database remains efficient, scalable, and aligned with the needs of its users. This proactive approach, combined with Agile's iterative feedback loops, ensures that the database remains stable, secure, and optimized for long-term use in the Sort It! system..

2.2.6 Maintenance and Evolution

In the maintenance and evolution phase, the focus shifts to ongoing improvements and adjustments to ensure the database remains efficient and scalable. Agile's iterative approach is particularly beneficial during this phase, as it allows for constant evaluation and optimization of the database's performance. Regular monitoring ensures that the database operates at peak efficiency, with any emerging issues quickly identified and resolved. Feedback loops from end-users and technical teams play a key role in ensuring that the system continues to meet evolving requirements.

One critical aspect of this phase is the regular updating of the Database Management System (DBMS). Patches provided by DBMS vendors are applied to address vulnerabilities, enhance security, and ensure the system stays current with the latest technologies and industry standards. Agile's flexibility ensures that these updates can be implemented with minimal disruption to the overall functionality, allowing the system to evolve smoothly over time.

As the application grows and user needs change, schema modifications may be necessary. This includes adapting the database to accommodate new features or requirements, ensuring that the database structure supports the evolving functionality of the application. Agile's iterative nature allows for these changes to be incorporated seamlessly, with continuous testing and feedback ensuring that any updates enhance the overall performance of the system.

Additionally, records management policies are crucial during this phase to ensure proper data archiving and compliance with data lifecycle management standards. Data retention policies are reviewed and updated to comply with industry regulations, ensuring the integrity and security of stored data. Agile's proactive approach allows for constant adjustments to ensure that both current and future needs are met efficiently.

The ongoing checks and improvements ensure that the database remains scalable and adaptable to future requirements. With Agile's focus on evolution and continuous improvement, this phase ensures that the Sort It! database will continue to support the application's goals, maintaining its reliability and efficiency over the long term.

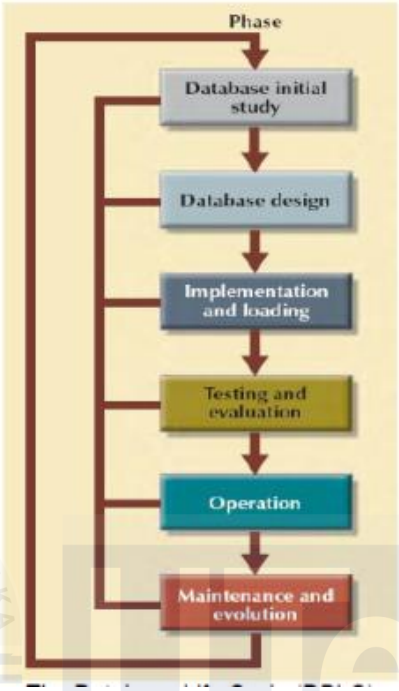

Figure 2.1 Database Life Cycle

Figure 2.2 Agile's Approach

2.3 Project Milestone

Task	Month						
	1	2	3	4	5	6	7
Planning and Analysis							
Database Design							
Implementation							
Testing and Evaluation							
Operation							
Maintenance							

Table 2.1 Gantt Chart

2.4 Summary

When dealing with the "Sort It!" project that is targeted at the automation of the student group formation, the Database Life Cycle (DBLC) implies a set of logical steps that will promote proper, and thorough development and manipulation of the database. The first process of the acquisition plan is the study phase that comprises requirement gathering and apprehension of feasibility and scope of the project through consultations with stakeholders. Components in the design phase include concept models, logical design, and physical design, while internal building blocks are developed into a full design during this phase. Implementation means the organization and installation of the DBMS, designing of the actual database and managing data consistency. During the testing and evaluation phase, functional and performance tests are done as well as the gathering of user feedback. The operation phase that is established ensures the right implementation and ongoing inspection of the solution within the production setting. Moreover, the maintenance and evolution phase entail regular performance enhancement, upgrading, altering the schema, and archiving data to maintain the database stability. This systematic approach guarantees that the development of the "Sort It!" project's database is properly implemented, accurate and optimized over its life cycle.

CHAPTER 3: ANALYSIS

3.1 Introduction

Many standard online group randomizers often rely on pure chance to assign students to groups, which can lead to imbalanced group compositions and inefficiencies in collaborative learning. These systems typically do not account for critical factors such as academic performance, which can result in groups with significant disparities in ability levels or uneven distribution of students' strengths and weaknesses.

In contrast, the analysis phase of the Sort It! Group Randomizer Management System project addresses these shortcomings by focusing on a more nuanced approach to group formation. This phase involves a thorough evaluation of existing processes and a detailed assessment of stakeholder needs, including the importance of fairness and balance in group assignments. By creating a comprehensive requirements document, the Sort It! system is designed to sort students based on GPA, ensuring that each group is composed of a diverse range of academic performances. This method promotes fairness and equity in group formation, avoiding the pitfalls of random assignment and providing a more structured approach that aligns with educational goals and operational requirements.

This approach ensures that the system not only meets but exceeds user expectations, creating groups that are more balanced and effective for collaborative learning.

3.2 Problem Analysis

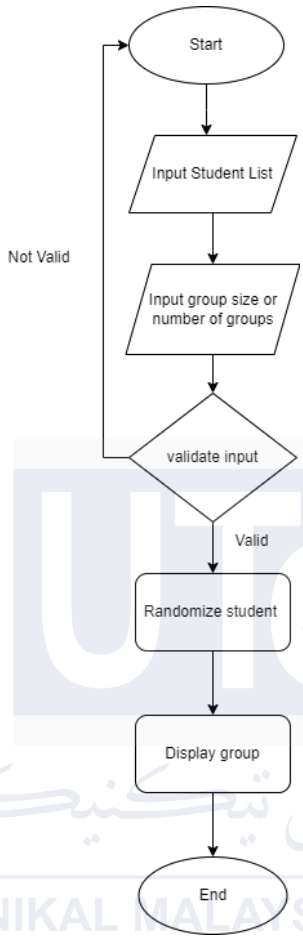

Figure 3.1 Current System Group Randomizer Flowchart

The flowchart shows the functioning of a student group randomizer system. The first action is that the user enters a list of students After that, the user designates either the number of students in a group or the number of groups. The system then cross checks the inputs with the student list by checking whether the list is empty, the group size or the number of groups is realistic. After validation the system goes ahead and randomly assigns students into groups as per the specified input parameters. Lastly, the group labels are randomized and after the execution of all the above process the program stops which helps in creating groups of students randomly.

3.3 The proposed improvements/solutions

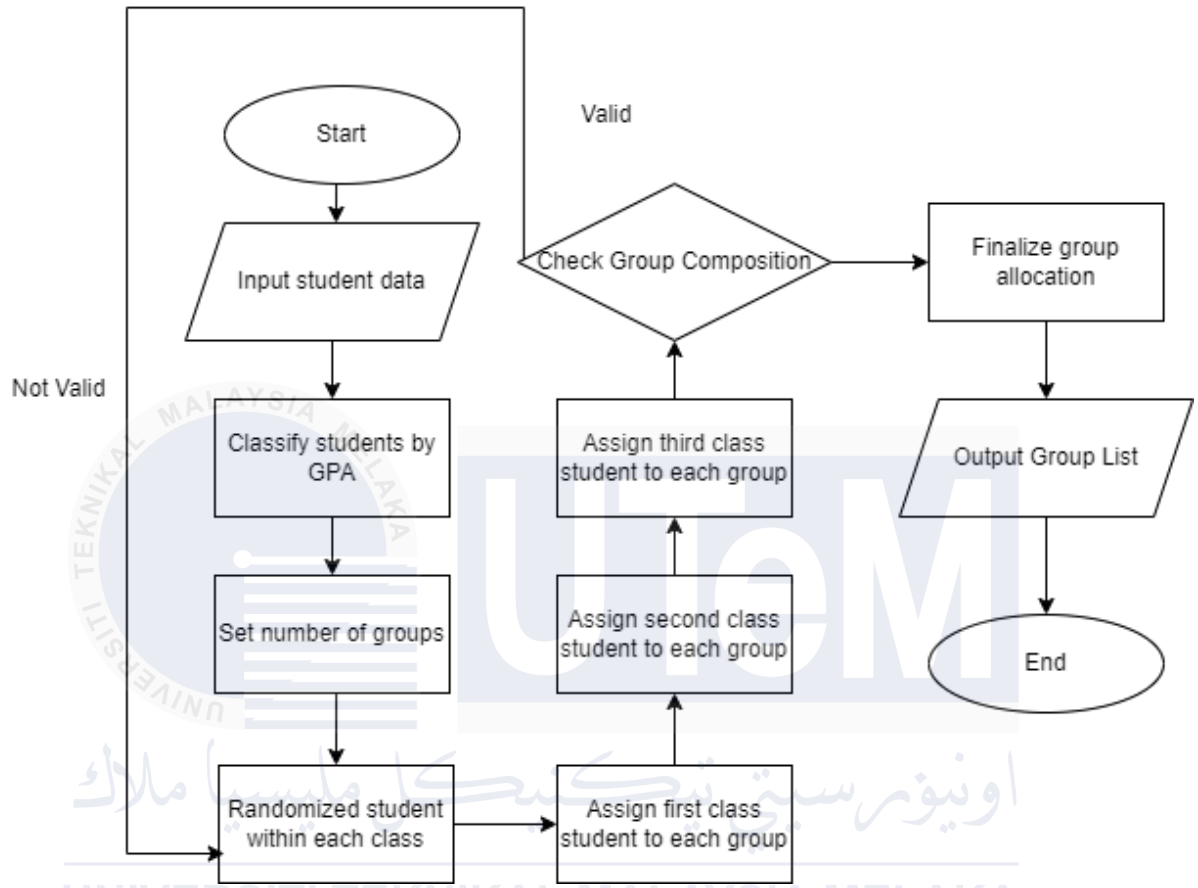

Figure 3.2 Proposed System Flowchart

The following flowchart depicts the proposed system of grouping the students based on their respective GPAs. The process begins with inputting student data, followed by classifying the students into three categories based on their GPA: First-class, second-class and third-class. It also determines the number of groupings and goes ahead to allocate one learner from the first class, second class and third class to form the groupings uniformly. Before assignment (student in each class is taken randomly). Once the organizational grouping has been checked to make sure that it is correct, the system then seals of grouping and displays the list of the groups as well as bring the process to an end. This flow will help to organize the groups in such a way that each group will have a right proportion of students according to their performances.

3.4 Requirement analysis of the to-be system

3.4.1 Functional Requirement (Process Model)

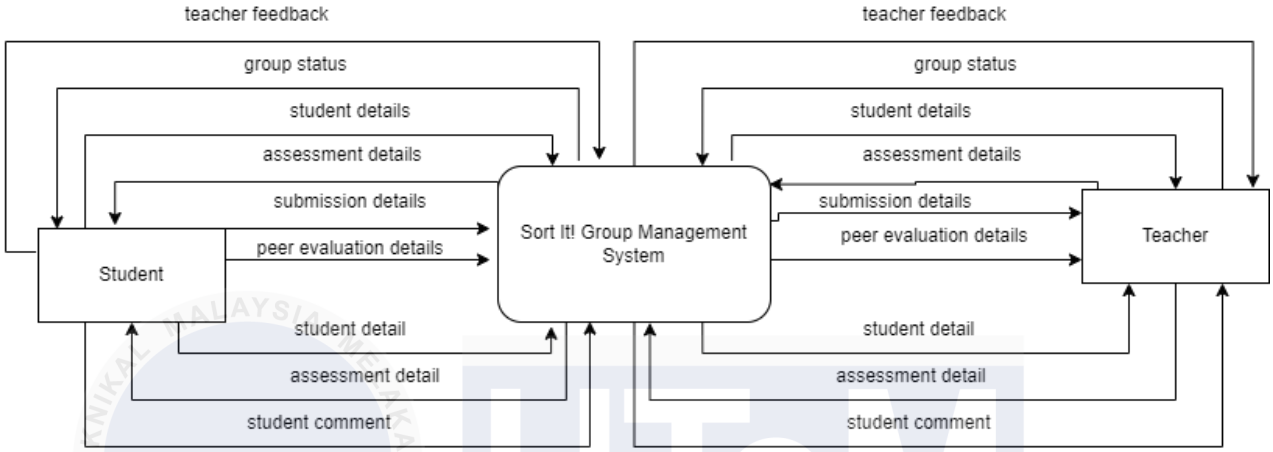

Figure 3.3 Context Diagram

This context diagram illustrates the interaction between the **Sort It! Group Management System**, students, and teachers. The system acts as the central point where both students and teachers input and retrieve data. Students interact with the system to submit their details, possibly manage submissions, or review group allocations, while teachers use the system to manage assessments, provide feedback, and oversee group formations. The diagram indicates a high-level view of data flow, where the system facilitates group management by receiving input from both students and teachers, processing it, and then outputting results back to the respective users.

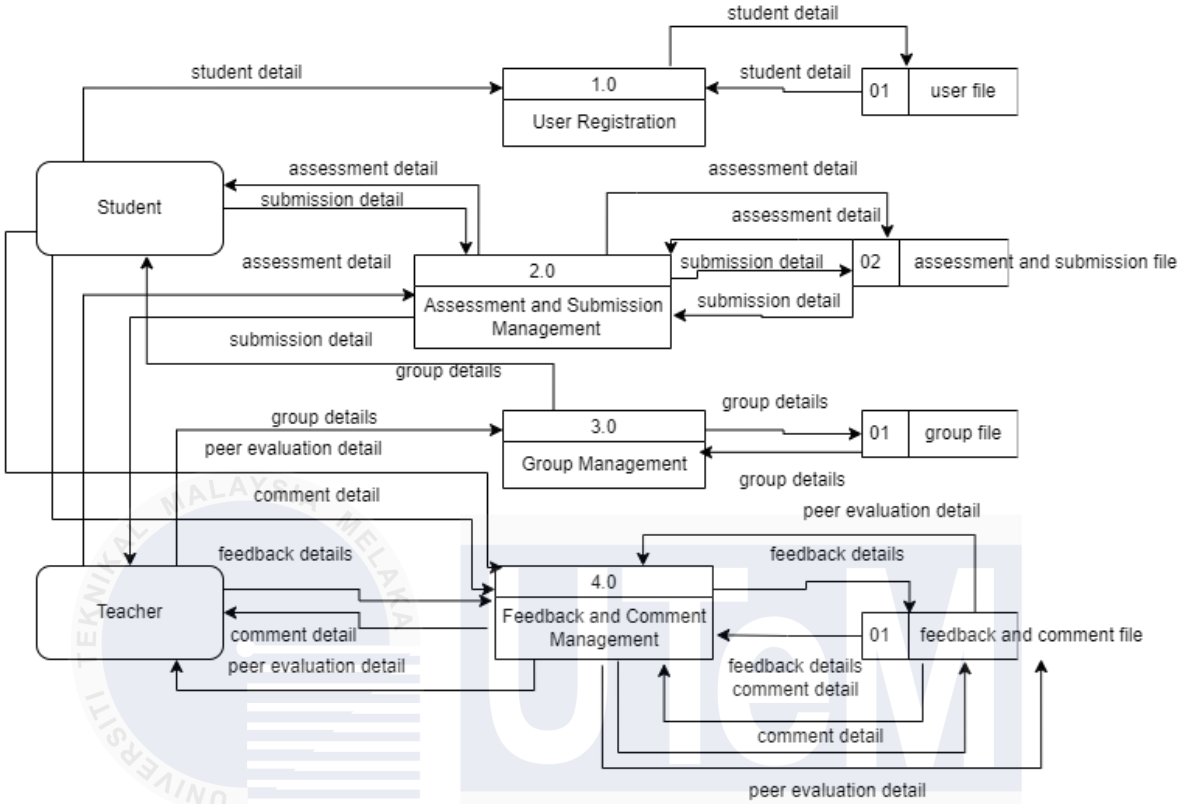

Figure 3.4 Data Flow Diagram Level 1

This Data Flow Diagram (DFD) provides a detailed breakdown of how data flows between students, teachers, and the system's core processes. There are four main processes depicted: **User Registration (1.0)**, where student data is stored in the user file; **Assessment and Submission Management (2.0)**, where assessments are handled and stored in an assessment file; **Group Management (3.0)**, where students are assigned to groups and the group file is updated; and **Feedback and Comment Management (4.0)**, where both students and teachers provide and receive feedback stored in the feedback archive. The diagram shows interactions between students and teachers with the system and how data is stored, processed, and retrieved within each module.

3.4.2 Non-functional Requirements

3.4.2.1 Quality Standards

- **Accuracy**

The system ensures accuracy by categorizing students into three distinct classes—First Class, Second Class, and Third Class—based on their GPAs. The sorting algorithm is built to evaluate student GPA data and assign them to the appropriate class without errors. This method eliminates any randomness and focuses solely on GPA as the determining factor for group formation. The clear and deterministic nature of this sorting method ensures that all students are grouped fairly, based on their academic performance, maintaining consistency across the system.

Before assigning groups, the system validates all inputs (such as GPA data and group sizes) to ensure no errors occur during processing. By strictly adhering to the set GPA thresholds for each class, the system guarantees that students are placed in the correct class, eliminating any bias or unfair advantage. Additionally, feedback and evaluation processes are tracked and maintained accurately to ensure that all interactions within the system are correctly logged and evaluated.

- **Reliability:**

Reliability is a key focus, with the system designed to maintain 99.9% availability. This ensures that users, especially during peak usage times like at the start of a semester or during group assignment periods, can rely on the system without experiencing downtime. The GPA-based sorting algorithm is efficient, meaning it can handle large volumes of student data quickly without overwhelming the system.

The system is hosted on a reliable server infrastructure that includes failover mechanisms and load balancing to ensure that performance is not affected by high traffic. Regular system updates and performance monitoring will be conducted to ensure that any potential issues are addressed before they cause interruptions. Moreover, the system is scalable, allowing it to accommodate an increasing number of students and institutions without performance degradation.

- **Usability:**

The system's interface is designed to be highly intuitive, making it easy for teachers and administrators to upload student GPA data and generate groupings without specialized

training. From the teacher dashboard, users can upload a list of students, view their sorted GPA classifications, and assign them into groups with ease. The interface guides users through each step, providing clear instructions and feedback to minimize errors or confusion.

The design ensures that users with varying levels of technical expertise can navigate the system efficiently. Detailed tooltips, instructional pop-ups, and visual cues will assist users in completing their tasks quickly and confidently. Teachers can also view the final group compositions, ensuring that every group has a balanced mix of students from different GPA categories, which promotes diversity and fairness in collaborative learning environments.

- **Maintainability:**

The system is designed with modularity in mind, allowing different components (such as the GPA sorting mechanism, user interface, and reporting tools) to be updated independently. This modular design enables easy maintenance and updates, ensuring the system can be adapted to changing educational needs. For example, if GPA thresholds change or additional categories need to be introduced, the system can be easily modified to reflect these adjustments without overhauling the entire platform.

Comprehensive documentation will accompany the system, explaining the structure of each module, the GPA sorting logic, and how to debug or extend the system as needed. Developers can use version control tools to track changes and updates, ensuring that any adjustments can be reverted or reviewed. Scheduled maintenance routines, such as database backups and performance optimizations, will be automated, allowing the system to maintain high performance with minimal manual intervention.

3.4.2.2 Security Requirements:

- **Data Encryption:**

In Sort It! Group Management System, student passwords are encrypted to protect sensitive login information. This encryption ensures that even if unauthorized access

occurs, student passwords cannot be easily deciphered or used. A strong hashing algorithm, such as bcrypt, is used to store passwords securely, ensuring that they cannot be reversed back to plain text.

While the student passwords are the only data being encrypted, this still adds a crucial layer of security to protect student accounts. Passwords are never stored or transmitted in plain text. During login, the system compares the hashed version of the entered password with the stored hash to authenticate the user.

- **Role-Based Access Control**:

  In Sort It! there are two primary user roles: teachers and students, and Role-Based Access Control (RBAC) is used to distinguish their access permissions. Teachers are granted access to key functionalities such as uploading student data, sorting students into groups based on their GPAs, and managing group assignments. This allows them to oversee and manage the data of all students within their respective classes. However, teachers do not have access to system-level configurations or settings, ensuring that administrative and core system functions remain restricted.

  On the other hand, students can log into the system to view their personal information, including which group they have been assigned to. Their access is limited to their own data, and they do not have visibility or control over other students' information. Additionally, students cannot perform administrative tasks such as group management or GPA sorting.

  By implementing RBAC, Sort It! ensures that users can only access the data and functions relevant to their roles. This structure helps to maintain the security and privacy of sensitive student information, ensuring that only authorized personnel can manage and view it.

- **Data Integrity:**

  Sort It! ensures the integrity of data through several mechanisms. First, it incorporates validation methods to verify the accuracy and format of any data entered or updated

within the system. For example, GPA values are validated to confirm they fall within appropriate ranges, while student passwords must adhere to predefined complexity standards to ensure robust security.

In addition, Sort It! uses audit trails to track all key actions. These logs record events such as when a teacher uploads or updates student information or when a student accesses their data. The logs detail who made the changes, what was altered, and when the changes occurred. This level of tracking ensures full accountability and traceability, making it easy to audit the system if needed.

Finally, Sort It! performs regular data checks to maintain consistency across records. It periodically verifies that no data has been inappropriately altered and that all group assignments remain accurate. If any inconsistencies or errors are detected, they are flagged for further review to ensure the integrity of the system is maintained.

3.4.3 Others Requirement

3.4.3.1 Software Requirement

Software	Description
Microsoft Visual Code	Microsoft Visual Code is well-suited to my project based on the powerful features and usage flexibility. There is a lot of settings and available extensions and the option to set it up exactly how I want is very appealing for web development using PHP, Laravel and other technologies. Microsoft Visual Code also has an in-built system of Git integration for checking and administration of types of change to codes, which is crucial in this aspect. The code editors, debugging tools and collaboration facilities have made the job smarter and efficient while cutting down development time from the original version. Also, Microsoft Visual Code is relatively lightweight, and this makes it

	compatible with any system, thus, Microsoft Visual Code is an efficient tool to be used in accomplishing development projects.
XAMPP	XAMPP is also suitable for my project due to the package for local web server development. It has necessary modules that are required for development and assessment of Web applications in the PHP environment like Apache, MySQL, PHP, and Perl etc which are all Open-source and free to use. XAMPP helps me not to face difficulties while organizing my local server it helps arrangement of local serves easy way. The easy-to-use control panel that it has for managing the server components helps me to test several components of my application with ease during developing and identifying bugs. Further, XAMPP supports cross-platform development where I know that my AP will be built on Windows, Mac OS, or LINUX operating system.

Table 3.1 Database Development for Software Requirement

3.4.3.2 Hardware Requirement

Hardware	Specification	Reason Of Using
Laptop Dell Inspiron 15 3530	• Windows 11 (64 bit) • 13th Gen Intel(R) Core (TM) i7-1355U • 14" HD (1366 x 768) • 16GB RAM • 512GB SSD	Speedy Connectivity

Table 3.2 Database Development for Hardware Requirement

3.5 Summary

In Chapter 3, the functional and non-functional requirements for the Sort It! group formation system are outlined as well as the system's key features these include: user authentication, data collection and preprocessing, identification of prospective group formation algorithm, dynamic adaptation, and feedback and evaluation components. The functional requirements revolve around total quality, assurance and reliability, speed, dependability, and data protection. These are the general processes of system development such as architecture and database design, user interface development and implementation where actual programming of the system components begins following the design layouts.

CHAPTER 4: DESIGN

4.1 Design

In this chapter, the authors describe the main aspects of the design of the "Sort It!" group formation system. Design, as mentioned in previous chapters, is a very crucial stage in the process, that physical conceptualized ideas, which were explained before, turn into physical designs and schemas. The areas to be addressed include the system graphic model, data model, human computer interface model among other areas that are crucial in designing a reliable and user-friendly system. In this chapter, the foundation for specification of requirements under consideration into actual parts of the subsequent system's architecture is laid down in a proper manner so that the eventual implemented system reliably meets the needs of project stakeholders**.**

4.2 System Architecture

The architecture view of the Sort It! group management system is designed to ensure scalability, maintainability, and efficiency. The system adopts a three-tier architecture comprising the presentation layer, business logic layer, and data layer.

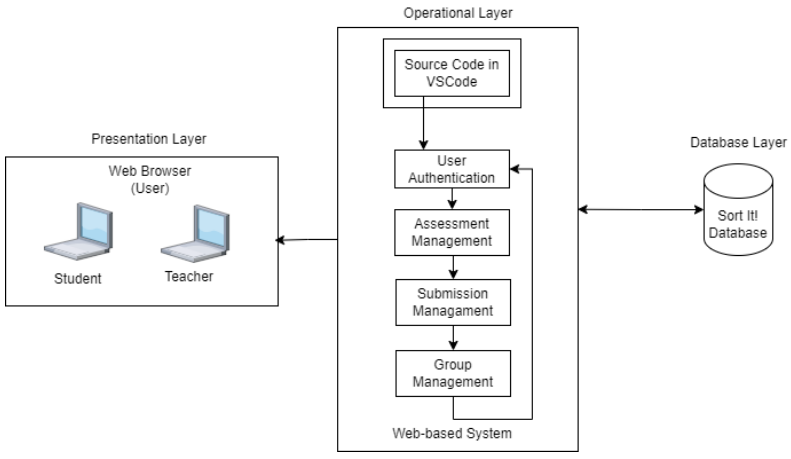

Figure 4.1 Sort It! System Architecture

- Presentation Layer: This layer includes the user interfaces for different users

(student and teacher). It is developed using HTML,

CSS, and JavaScript, providing a responsive and intuitive user experience.

- Operational Layer: This layer contains the core functionalities of the system,

including user authentication, assessment management, submission management and group management. It is implemented using PHP and integrated with PL/SQL for database operations.

- Data Layer: This layer is responsible for data storage and management. MySQL

with phpMyAdmin is used for database management, ensuring data integrity,

security, and efficient data retrieval

4.3 Database Design

The database design process involves three main phases: conceptual design, logical design, and physical design. Each phase aims to ensure the database structure supports the system's operational requirements and objectives.

4.3.1 Conceptual Design

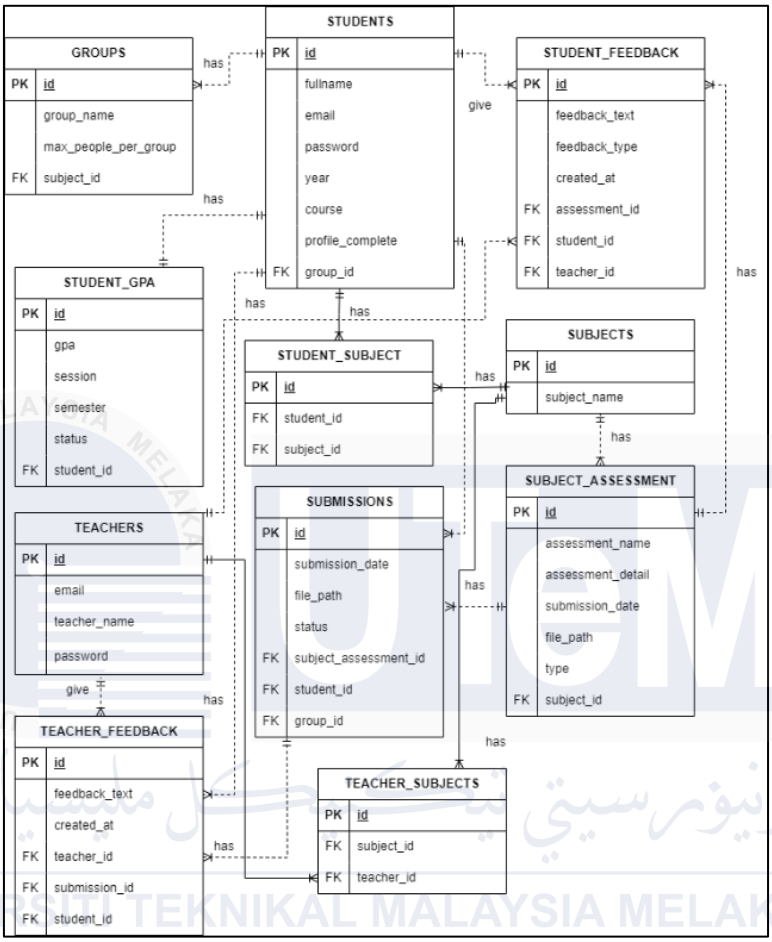

Table 4.1 Conceptual Design, Entity Relationship Diagram of the system

Business Rules

- A student belongs to one and only one group (in one subject), but a group can have one or many students.
- A student has one and only one GPA, but a GPA belongs to one and only one student.
- A student can enroll in one or many subjects, but a subject can have one or many students enrolled.
- A group belongs to one and only one subject, but a subject can have one or many groups.
- A submission belongs to one and only one student, but a student can submit one or many submissions.

- A submission belongs to one and only one group, but a group can have one or many submissions.
- A submission belongs to one and only one assessment, but an assessment can have one or many submissions.
- An assessment belongs to one and only one subject, but a subject can have one or many assessments.
- A student can receive one or many feedback entries, but a feedback entry belongs to one and only one student.
- A teacher can give one or many feedback entries, but a feedback entry belongs to one and only one teacher.
- A teacher can teach one or many subjects, but a subject can have one or many teachers.

4.3.2 Logical Design

4.3.2.1 Normalization

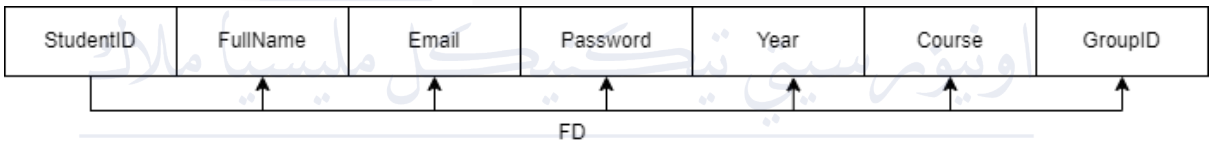

Figure 4.2 The Normalization of Student

The figure illustrates the structure of a student table, where `StudentID` serves as the unique identifier (primary key) for each student. This table consists of several attributes, including `FullName`, `Email`, `Password`, `Year`, `Course`, and `GroupID`. Each of these attributes is functionally dependent on `StudentID`, as indicated by the arrows pointing from `StudentID` to the other columns. This means that for every unique `StudentID`, there is a corresponding unique set of values for `FullName`, `Email`, `Password`, `Year`, `Course`, and `GroupID`.

In terms of functional dependency, the relationship shown indicates that `StudentID` fully determines the values of all other attributes. This is crucial for maintaining data integrity, ensuring that each student's information is uniquely identifiable and consistent across the table. The structure adheres to the principles of normalization, specifically 1st Normal Form (1NF), where each attribute contains atomic values, and each row is uniquely identified by the

`StudentID`. Furthermore, since all non-key attributes depend entirely on the primary key (`StudentID`), the table also satisfies the conditions of 2nd Normal Form (2NF).

In summary, the figure represents a table where `StudentID` is the primary key, functionally determining all other attributes. This functional dependency ensures that the database structure is both efficient and free from redundancy, which is a fundamental aspect of database normalization.

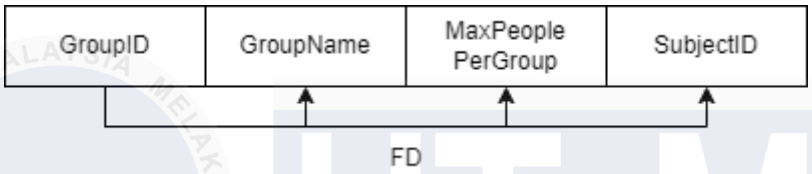

Figure 4.3 The Normalization of Group

The figure represents a table for student groups, where `GroupID` acts as the primary key (unique identifier) for each group. The table contains several attributes, including `GroupName`, `MaxPeoplePerGroup`, and `SubjectID`. Each of these attributes is functionally dependent on `GroupID`, as indicated by the arrows pointing from `GroupID` to the other columns. This signifies that for every unique `GroupID`, there is a corresponding unique set of values for `GroupName`, `MaxPeoplePerGroup`, and `SubjectID`.

The functional dependency depicted in the diagram demonstrates that `GroupID` fully determines the values of the other attributes. This means that for every unique `GroupID`, there will be a unique group name, a specific maximum number of people allowed in that group, and an associated subject. The functional dependency ensures that each group is uniquely identifiable based on its `GroupID`, and all related information is consistent and non-redundant.

In terms of normalization, this structure follows the principles of 1st Normal Form (1NF), where each attribute holds atomic values, and each row is uniquely identified by the `GroupID`. Additionally, since all non-key attributes (`GroupName`, `MaxPeoplePerGroup`, `SubjectID`) are fully dependent on the primary key (`GroupID`), the table is also in 2nd Normal Form (2NF).

In summary, the figure shows a table where `GroupID` serves as the primary key and functionally determines `GroupName`, `MaxPeoplePerGroup`, and `SubjectID`. This functional dependency helps to maintain the integrity of the table, ensuring that the data is well-organized and free from redundancy, which is a key principle of database normalization.

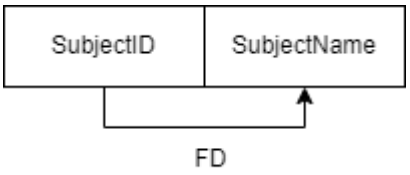

Figure 4.4 The Normalization of Subject

The figure illustrates a subject table where `SubjectID` serves as the primary key (unique identifier) for each subject. This table consists of two attributes: `SubjectID` and `SubjectName`. The arrow in the diagram, pointing from `SubjectID` to `SubjectName`, signifies a functional dependency (FD). This means that `SubjectID` uniquely determines `SubjectName`.

In other words, for each unique `SubjectID`, there is a corresponding unique `SubjectName`. This relationship ensures that each subject is uniquely identified by its `SubjectID`, and there is a direct mapping to the subject's name, ensuring consistency and uniqueness in the data.

From a normalization perspective, this table satisfies the requirements of 1st Normal Form (1NF) since each attribute contains atomic values, and every row is uniquely identifiable by the `SubjectID`. Moreover, the table adheres to the principles of 2nd Normal Form (2NF), as there are no partial dependencies—`SubjectName` is fully dependent on the primary key `SubjectID`.

In conclusion, the figure represents a table where `SubjectID` is the primary key, and it functionally determines `SubjectName`. This functional dependency ensures that the data is structured efficiently, avoiding redundancy and maintaining the integrity of the subject information.

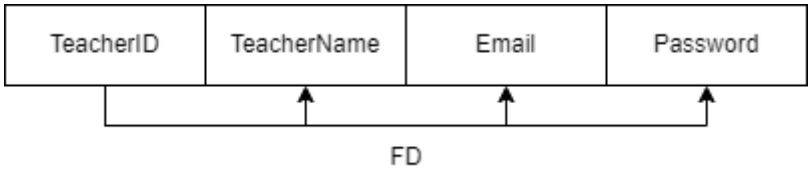

Figure 4.5 The Normalization of Teacher

The figure represents a teacher table, where `TeacherID` is the primary key (unique identifier) for each teacher. The table consists of several attributes, including `TeacherName`, `Email`, and `Password`. Each of these attributes is functionally dependent on `TeacherID`, as indicated by the arrows pointing from `TeacherID` to the other columns. This shows that for each unique `TeacherID`, there is a corresponding unique set of values for `TeacherName`, `Email`, and `Password`.

In terms of functional dependency (FD), the relationship depicted in the diagram indicates that `TeacherID` fully determines all the other attributes in the table. For every unique `TeacherID`, there is a unique teacher name, email, and password associated with that ID. This functional dependency ensures that each teacher's information is uniquely identifiable and organized, helping to maintain consistency and integrity in the data.

From a normalization perspective, the table satisfies the requirements of 1st Normal Form (1NF) because each attribute holds atomic values, and each row is uniquely identified by the `TeacherID`. Additionally, since all non-key attributes (`TeacherName`, `Email`, `Password`) are fully dependent on the primary key (`TeacherID`), the table also complies with the principles of 2nd Normal Form (2NF).

In conclusion, the figure shows a table where `TeacherID` is the primary key that functionally determines `TeacherName`, `Email`, and `Password`. This ensures data consistency and eliminates redundancy by maintaining the unique relationship between a teacher's identifier and their corresponding attributes.

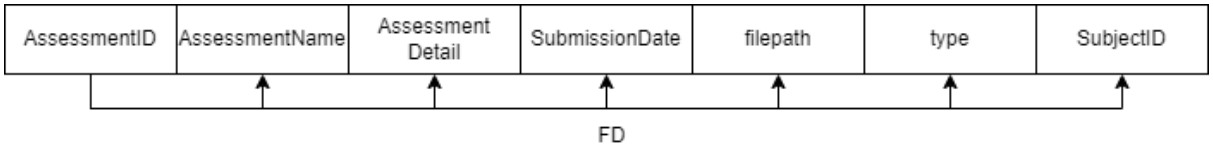

Figure 4.6 The Normalization of Assessment

The figure represents an assessment table, where `AssessmentID` is the primary key (unique identifier) for each assessment. The table contains several attributes, including `AssessmentName`, `AssessmentDetail`, `SubmissionDate`, `filepath`, `type`, and `SubjectID`. Each of these attributes is functionally dependent on `AssessmentID`, as shown by the arrows pointing from `AssessmentID` to the other columns. This indicates that for each unique `AssessmentID`, there is a corresponding unique set of values for `AssessmentName`, `AssessmentDetail`, `SubmissionDate`, `filepath`, `type`, and `SubjectID`.

In terms of functional dependency (FD), the diagram illustrates that `AssessmentID` fully determines all the other attributes. For every unique `AssessmentID`, there is a unique name for the assessment, specific details, a submission date, a file path for submission, the type of assessment, and the associated `SubjectID`. This functional dependency ensures that each assessment's data is uniquely identifiable and organized, maintaining data consistency and integrity.

From a normalization perspective, the table satisfies the requirements of 1st Normal Form (1NF) as each attribute contains atomic values, and each row is uniquely identified by `AssessmentID`. Additionally, since all non-key attributes are fully dependent on the primary key (`AssessmentID`), the table complies with the principles of 2nd Normal Form (2NF).

In conclusion, the figure represents a table where `AssessmentID` is the primary key that functionally determines `AssessmentName`, `AssessmentDetail`, `SubmissionDate`, `filepath`, `type`, and `SubjectID`. This structure ensures efficient data organization and eliminates redundancy by maintaining a unique relationship between the assessment identifier and its corresponding attributes.

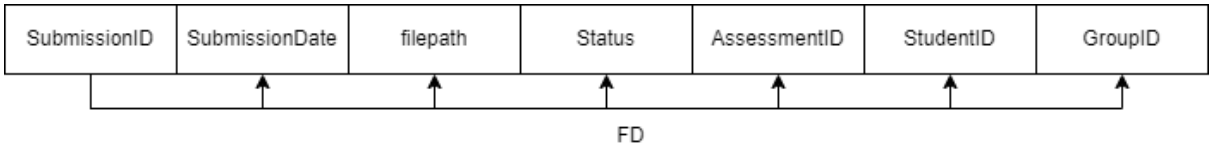

Figure 4.7 The Normalization of Submission

The figure represents a submission table, where `SubmissionID` serves as the primary key (unique identifier) for each submission. The table consists of several attributes: `SubmissionDate`, `filepath`, `Status`, `AssessmentID`, `StudentID`, and `GroupID`. Each of these attributes is functionally dependent on `SubmissionID`, as indicated by the arrows pointing from `SubmissionID` to the other columns. This implies that for each unique `SubmissionID`, there is a unique set of values for `SubmissionDate`, `filepath`, `Status`, `AssessmentID`, `StudentID`, and `GroupID`.

In terms of functional dependency (FD), the diagram shows that `SubmissionID` fully determines the values of all the other attributes in the table. For each submission identified by a unique `SubmissionID`, there is a specific submission date, file path, status, associated assessment (`AssessmentID`), student (`StudentID`), and group (`GroupID`). This functional dependency ensures that each submission's data is organized consistently and uniquely identified.

From a normalization perspective, the table adheres to the requirements of 1st Normal Form (1NF), where each attribute holds atomic values, and each row is uniquely identifiable by the `SubmissionID`. Additionally, since all non-key attributes are fully dependent on the primary key (`SubmissionID`), the table also complies with the principles of 2nd Normal Form (2NF).

In conclusion, the figure represents a table where `SubmissionID` is the primary key that functionally determines `SubmissionDate`, `filepath`, `Status`, `AssessmentID`, `StudentID`, and `GroupID`. This ensures efficient organization of data, reducing redundancy, and maintaining consistency by uniquely identifying each submission and its related information.

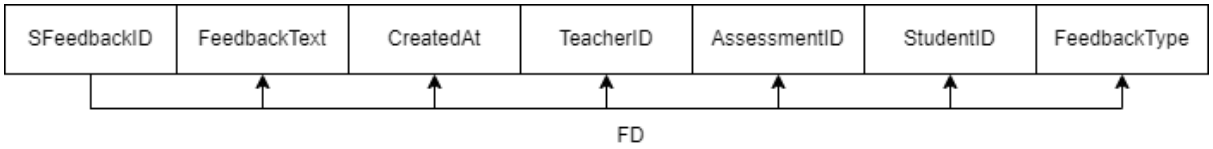

Figure 4.8 The Normalization of Student_Feedback

The figure represents a `student_feedback` table, where `SFeedbackID` is the primary key (unique identifier) for each piece of student feedback. The table consists of several attributes: `FeedbackText`, `CreatedAt`, `TeacherID`, `AssessmentID`, `StudentID`, and `FeedbackType`. Each of these attributes is functionally dependent on `SFeedbackID`, as indicated by the arrows pointing from `SFeedbackID` to the other columns. This implies that for each unique `SFeedbackID`, there is a unique set of values for `FeedbackText`, `CreatedAt`, `TeacherID`, `AssessmentID`, `StudentID`, and `FeedbackType`.

In terms of functional dependency (FD), the diagram shows that `SFeedbackID` fully determines the values of all the other attributes in the table. For each feedback entry identified by a unique `SFeedbackID`, there is specific feedback text, a timestamp for when the feedback was created, the teacher who provided the feedback (`TeacherID`), the associated assessment (`AssessmentID`), the student receiving the feedback (`StudentID`), and the type of feedback (`FeedbackType`). This functional dependency ensures that each feedback entry is uniquely identifiable and organized with all its associated details.

From a normalization perspective, the table adheres to the requirements of 1st Normal Form (1NF), where each attribute contains atomic values, and each row is uniquely identifiable by the `SFeedbackID`. Furthermore, as all non-key attributes are fully dependent on the primary key (`SFeedbackID`), the table also complies with the principles of 2nd Normal Form (2NF).

In conclusion, the figure represents a table where `SFeedbackID` is the primary key that functionally determines `FeedbackText`, `CreatedAt`, `TeacherID`, `AssessmentID`, `StudentID`, and `FeedbackType`. This structure helps in organizing feedback data efficiently, ensuring that each feedback entry is uniquely identified and linked to the relevant teacher, student, and assessment while maintaining consistency and eliminating redundancy.

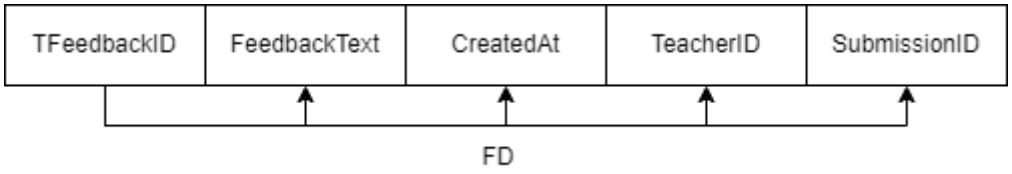

Figure 4.9 The Normalization of Teacher_Feedback

The figure represents a `teacher_feedback` table, where `TFeedbackID` serves as the primary key (unique identifier) for each feedback entry given by a teacher. The table contains several attributes, including `FeedbackText`, `CreatedAt`, `TeacherID`, and `SubmissionID`. Each of these attributes is functionally dependent on `TFeedbackID`, as indicated by the arrows pointing from `TFeedbackID` to the other columns. This implies that for each unique `TFeedbackID`, there is a corresponding unique set of values for `FeedbackText`, `CreatedAt`, `TeacherID`, and `SubmissionID`.

In terms of functional dependency (FD), the diagram shows that `TFeedbackID` fully determines the values of all the other attributes. For every unique feedback entry, identified by its `TFeedbackID`, there is a specific piece of feedback (`FeedbackText`), a timestamp indicating when the feedback was created (`CreatedAt`), the teacher who provided the feedback (`TeacherID`), and the submission it relates to (`SubmissionID`). This functional dependency ensures that each feedback entry is uniquely identifiable and all related information is consistent.

From a normalization perspective, the table satisfies the requirements of 1st Normal Form (1NF), where each attribute holds atomic values, and each row is uniquely identified by the `TFeedbackID`. Additionally, as all non-key attributes are fully dependent on the primary key (`TFeedbackID`), the table also complies with the principles of 2nd Normal Form (2NF).

In conclusion, the figure represents a table where `TFeedbackID` is the primary key that functionally determines `FeedbackText`, `CreatedAt`, `TeacherID`, and `SubmissionID`. This structure helps in organizing feedback data efficiently, ensuring each entry is uniquely identified and linked to the relevant teacher and submission, maintaining consistency and reducing redundancy.

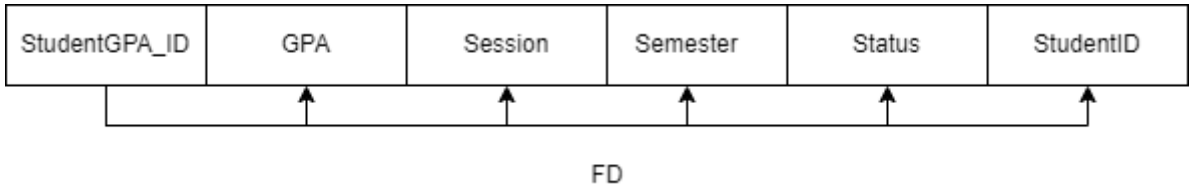

Figure 4.10 The Normalization of Student_GPA

The figure represents a `student_gpa` table, where `StudentGPA_ID` serves as the primary key (unique identifier) for each GPA record. The table consists of several attributes, including `GPA`, `Session`, `Semester`, `Status`, and `StudentID`. Each of these attributes is functionally dependent on `StudentGPA_ID`, as indicated by the arrows pointing from `StudentGPA_ID` to the other columns. This means that for each unique `StudentGPA_ID`, there is a corresponding unique set of values for `GPA`, `Session`, `Semester`, `Status`, and `StudentID`.

In terms of functional dependency (FD), the diagram illustrates that `StudentGPA_ID` fully determines the values of all the other attributes. For every unique GPA entry, identified by its `StudentGPA_ID`, there is a specific GPA score (`GPA`), the academic session (`Session`), the semester (`Semester`), the status of the GPA (`Status`), and the student to whom it belongs (`StudentID`). This functional dependency ensures that each GPA record is uniquely identifiable and the associated data is consistent.

From a normalization perspective, the table satisfies the requirements of 1st Normal Form (1NF), as each attribute contains atomic values, and each row is uniquely identified by the `StudentGPA_ID`. Moreover, since all non-key attributes are fully dependent on the primary key (`StudentGPA_ID`), the table also adheres to the principles of 2nd Normal Form (2NF).

In conclusion, the figure shows a table where `StudentGPA_ID` is the primary key that functionally determines `GPA`, `Session`, `Semester`, `Status`, and `StudentID`. This design ensures efficient organization of GPA records, linking them to the relevant student and maintaining consistency while eliminating redundancy.

4.3.2.2 Data Dictionary

STUDENT table

Attribute Name	Description	Data type and size	Null	Default Value	Unique	PK or FK	Reference table
Id	Student ID	int (11)	No	-	Yes	PK	-
Fullname	Student Full Name	varchar (255)	No	-	No	-	-
Email	Student Email	varchar (50)	No	-	Yes	-	-
Password	Password	varchar (50)	No	-	No	-	-
Group_id	Student Group ID	int (11)	Yes	-	No	FK	Group
year	Year of Studying	int (11)	Yes	-	No	-	-
Course	Course Taking	varchar (50)	Yes	-	No	-	-
Profile_complete	Role in Group	int (11)	Yes	-	No	FK	Role

Table 4.2 Data Dictionary Student

TEACHERS Table

Attribute Name	Description	Data type and size	Null	Default Value	Unique	PK or FK	Reference table
Id	Teacher ID	varchar (11)	No	-	yes	PK	-
Email	Teacher Email	varchar (50)	No	-	Yes	-	-
Password	Password	varchar (255)	No	-	No	-	-
Teacher_name	Teacher Name	varchar (100)	No	-	No	-	-

Table 4.3 Data Dictionary Teachers

STUDENT_GPA Table

Attribute Name	Description	Data type and size	Null	Default Value	Unique	PK or FK	Reference table
Id	Student GPA ID	Int(11)	No	-	No	PK	-
GPA	Student GPA	float	Yes	-	No	-	-
Session	The year/session of the GPA	int	Yes	2023/2024	No	-	-
Semester	The semester of Student Pursuing	int	Yes	-	No	-	-
Status	The status of GPA	int	Yes	1	No	-	-
Student_id	Student ID	int	No	-	No	FK	Student

Table 4.4 Data Dictionary Student_GPA

Groups Table

Attribute Name	Description	Data type and size	Null	Default Value	Unique	PK or FK	Reference table
id	Group ID	Int (11)	No	-	-	PK	-
Group_name	Group Name	varchar (50)	No	-	-	-	-
Max_people_per_group	Maximum People Per Group	Int (11)	Yes	-	-	-	-
Subject_id	Subject ID	Varchar(10)	No	-	Yes	FK	Subject

Table 4.5 Data Dictionary Group

Submission Table

Attribute Name	Description	Data type and size	Null	Default Value	Unique	PK or	Reference table

						FK	
Submission _id	Submision ID	Int (11)	No	-	No	PK	-
Subject_id	Subject ID	Varchar(10)	No	-	No	FK	subject
Assessment_name	Assessment Name	Varchar(255)	No	-	No	-	-
Assessment_details	Assessment detail	Varchar(255)	Yes	-	No	-	-
Submission_date	Submission Date	date	No	-	No	-	-
File_path	File Path of the submission file	Varchar (255)	No	-	No	-	-
type	Type of assessment	Varchar(255)	No	-	No	-	-

Table 4.6 Data Dictionary Submission

TEACHER_FEEDBACK Table

Attribute Name	Description	Data type and size	Null	Default Value	Unique	PK or FK	Reference table
Id	Student Role ID	Int (11)	No	-	No	PK	-
Teacher_id	Teacher ID	Varchar(255)	No	-	Yes	FK	Teachers
Student_id	Student ID	Int (11)	No	-	No	FK	Student
Submission_id	Submission ID	Int(11)	no	-	Yes	FK	Submission
Feedback_text	Feedback Text	Varchar(255)	No	-	No	-	-
Created_at	Time the feedback is created	time	no	-	No	-	-

Table 4.7 Data Dictionary Teacher_Feedback

STUDENT_FEEDBACK Table

Attribute Name	Description	Data type and size	Null	Default Value	Unique	PK or FK	Reference table
Id	Student Role ID	Int (11)	No	-	No	PK	-

Teacher_id	Teacher ID	Varchar(255)	No	-	Yes	FK	Teachers
Student_id	Student ID	Int (11)	No	-	No	FK	Student
Assessment_id	Assessment ID	Int(11)	no	-	Yes	FK	Submission
Feedback_text	Feedback Text	Varchar(255)	No	-	No	-	-
Feedback_type	Feedback Type	Varchar(255)	No	-	No	-	-
Created_at	Time the feedback is created	time	no	-	No	-	-

Table 4.8 Data Dictionary Student_Feedback

STUDENT_SUBJECTS Table

Attribute Name	Description	Data type and size	Null	Default Value	Unique	PK or FK	Reference table
Id	Student Assessment ID	Int (11)	No	-	No	PK	
Student_id	Student ID	Int (11)	No	-	No	FK	Student
Subject_id	Subject ID	Varchar(10)	No	-	No	FK	Subject

Table 4.9 Data Dictionary Student_Subjects

SUBJECTS Table

Attribute Name	Description	Data type and size	Null	Default Value	Unique	PK or FK	Reference table
Id	Subject ID	Varchar(10)	No	-	No	PK	-
Subject Name	Subject Name	Varchar(255)	No	-	No	-	-

Table 4.10 Data Dictionary Subjects

Assessment Table

Attribute Name	Description	Data type and size	Null	Default Value	Unique	PK or FK	Reference table
Id	Assessment ID	Int (11)	No	-	No	PK	-

Assessment_Name	Assessment Name	Varchar (11)	No	-	No	-	-
Assessment_Detail	Details of assessment	Varchar (255)	No	-	No	-	-
Submission _date	Submission Date	Date	No	-	No	-	-
File_path	File Path	Varchar (255)	No	-	No	-	-
Type	Type of Assessment	Varchar (255)	No	-	No	-	-
Subject_ID	Subject ID	Varchar (10)	No	-	No	FK	Subjects

Table 4.11 Data Dictionary Assessment

TEACHER_SUBJECTS Table

Attribute Name	Description	Data type and size	Null	Default Value	Unique	PK or FK	Reference table
Id	Teacher Subject ID	INT	No	-	No	PK	-
Subject_ID	Subject ID	Varchar(255)	No	-	No	FK	Subject
Teacher_ID	Teacher ID	Varchar(10)	No	-	Yes	FK	Teacher

Table 4.12 Data Dictionary Teacher_Subject

4.3.3 Physical Design

i. Simple Query

```
SELECT * FROM `groups;
```

ii. Join

```
SELECT students.id AS student_id, students.fullname,
groups.group_name
                FROM students
```

```
                    LEFT JOIN `groups` ON students.group_id =
groups.id;
```

iii. Trigger

```
BEGIN
INSERT INTO sequence_teacher VALUES (NULL);
SET NEW.id = CONCAT ('S',LPAD (LAST_INSERT_ID (),3,'000'));
END
```

iv. Aggregate

```
$member_count_result = $conn->query("SELECT COUNT(*)
AS member_count FROM students WHERE group_id =
'$group_id'");
```

4.4 Physical Design

4.4.1 Security Mechanism

```php
// Check if the user is a teacher
$sql_teacher = "SELECT * FROM teachers WHERE email='$email'";
$result_teacher = $conn->query($sql_teacher);
if ($result_teacher->num_rows > 0) {
    $teacher = $result_teacher->fetch_assoc();
    if ($password === $teacher['password']) {
        $_SESSION['teacher_id'] = $teacher['id'];
        $_SESSION['teacher_name'] = $teacher['teacher_name']; // Store teacher name in session
        $_SESSION['role'] = 'teacher'; // Set role as 'teacher'
        header("Location: dashboard_teacher.php");
        exit();
    }
}

// Check if the user is a student
$sql_student = "SELECT * FROM students WHERE email='$email'";
$result_student = $conn->query($sql_student);
if ($result_student->num_rows > 0) {
    $student = $result_student->fetch_assoc();
    if ($password === $student['password']) {
        $_SESSION['student_id'] = $student['id'];
        $_SESSION['fullname'] = $student['fullname'];
        $_SESSION['email'] = $student['email'];

        // Check if the student is a leader
        $sql_check_leader = "SELECT sr.role_id, r.role_name
                             FROM student_roles sr
                             JOIN roles r ON sr.role_id = r.id
                             WHERE sr.student_id = '{$student['id']}' AND r.role_name = 'leader'";
        $result_check_leader = $conn->query($sql_check_leader);

        if ($result_check_leader->num_rows > 0) {
            $_SESSION['role'] = 'leader'; // Set role as 'leader'
        } else {
            $_SESSION['role'] = 'student'; // Set role as 'student'
```


Figure 4.11 Security Mechanism

4.5 GUI Design

4.5.1 Register Interface Design

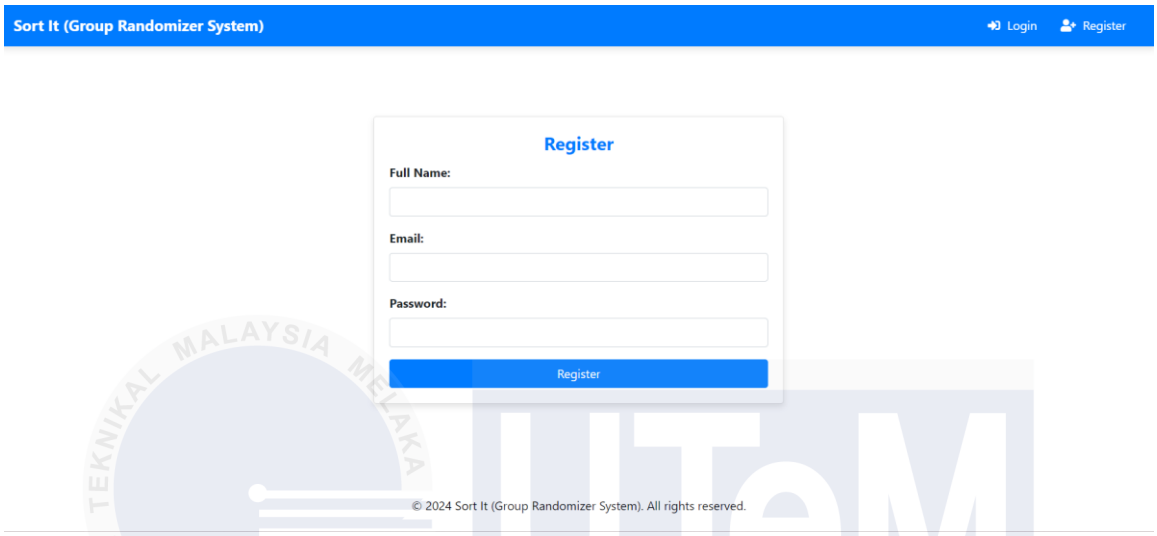

Figure 4.12 Page of Register

4.5.2 Log In Interface System

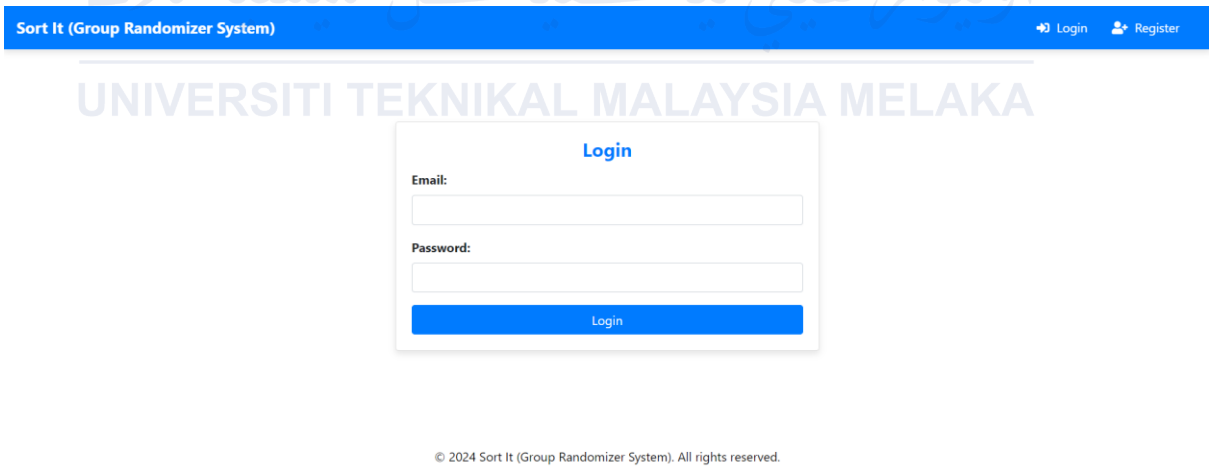

Figure 4.13 Login Page

4.5.3 Teacher Dashboard

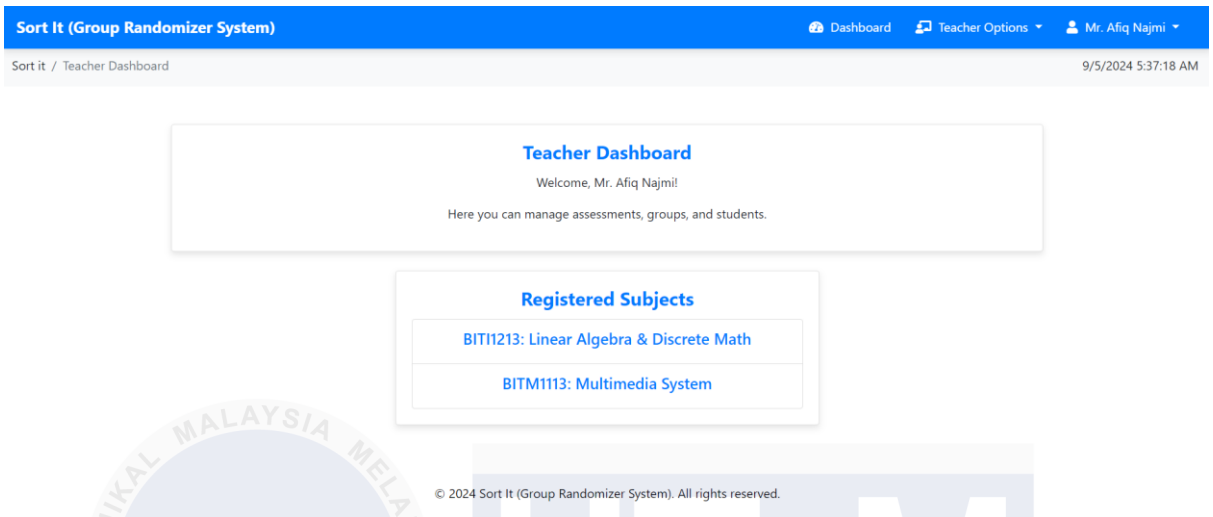

Figure 4.14 Teacher Dashboard

4.5.4 Subject Dashboard

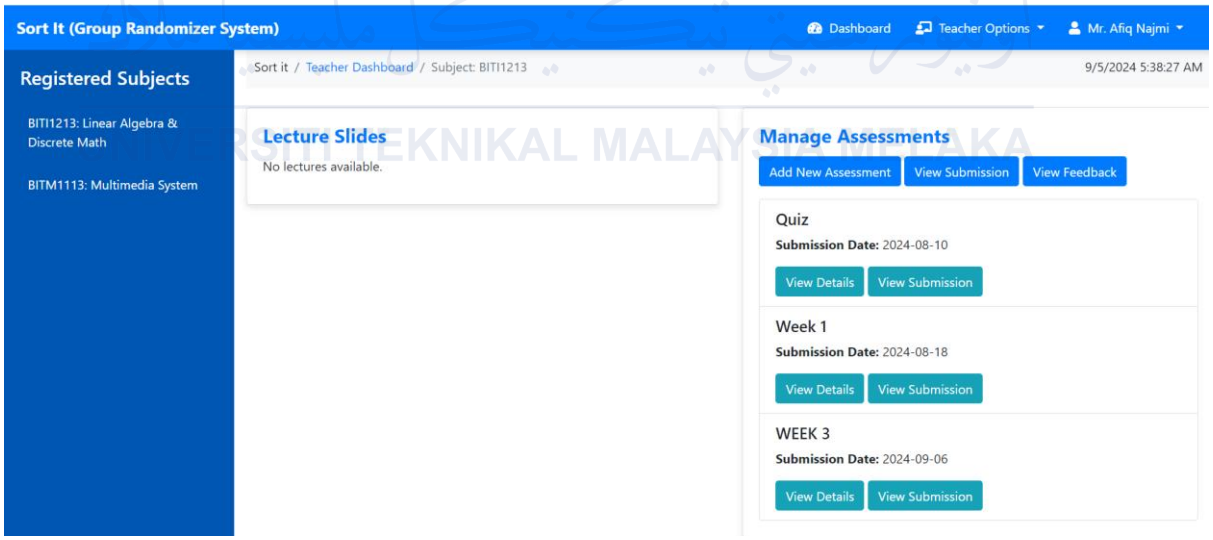

Figure 4.15 Subject Dashboard Page

4.5.5 Add Assessment

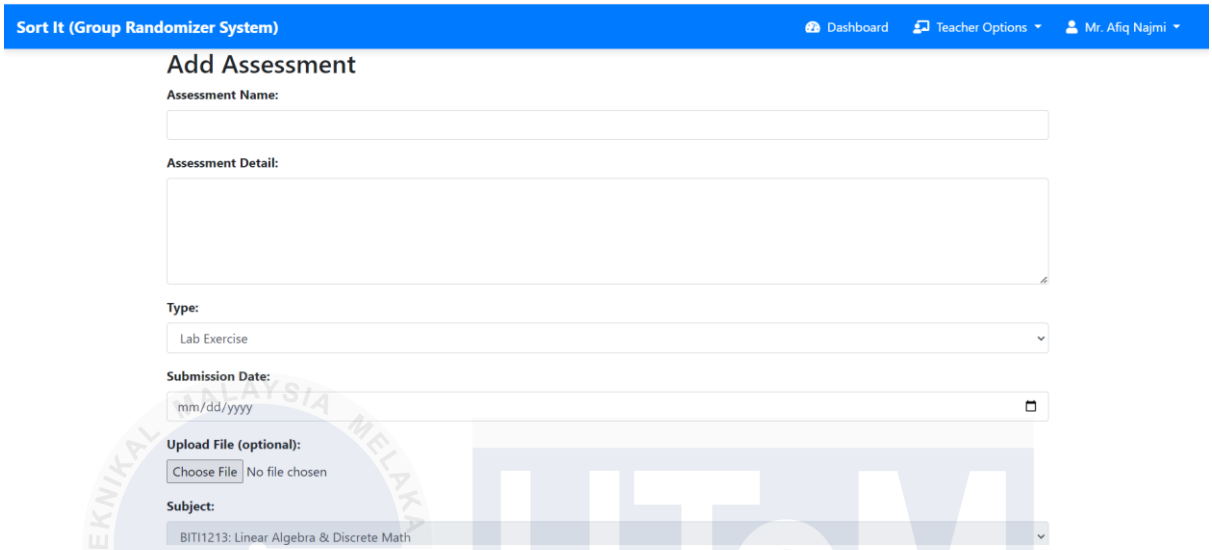

Figure 4.16 Add Assessment Page

4.5.6 View Submissions

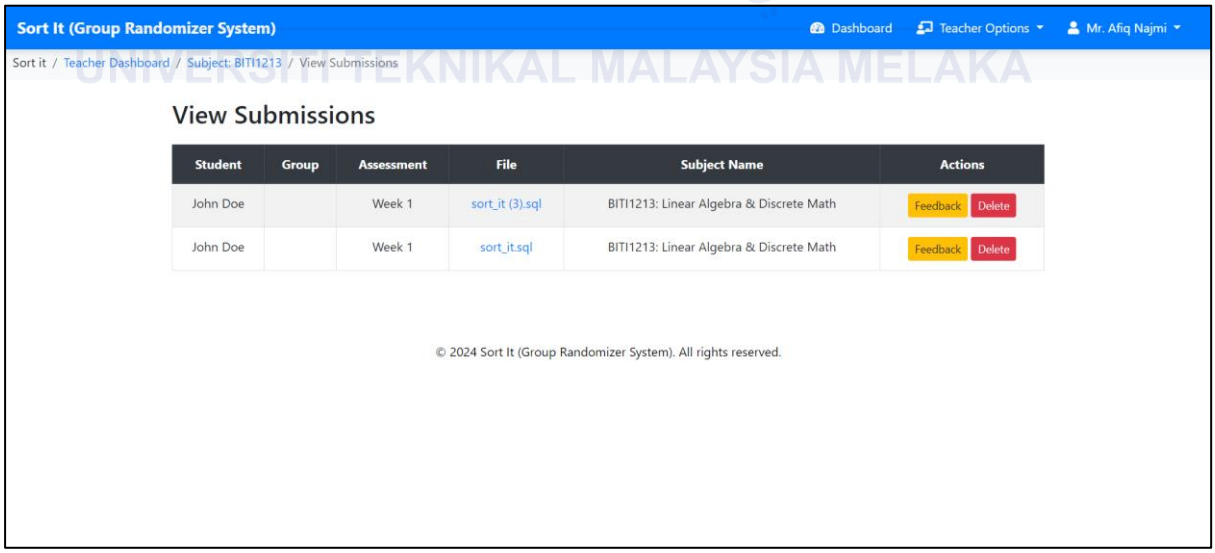

Figure 4.17 View Submissions

4.5.7 Provide Feedback

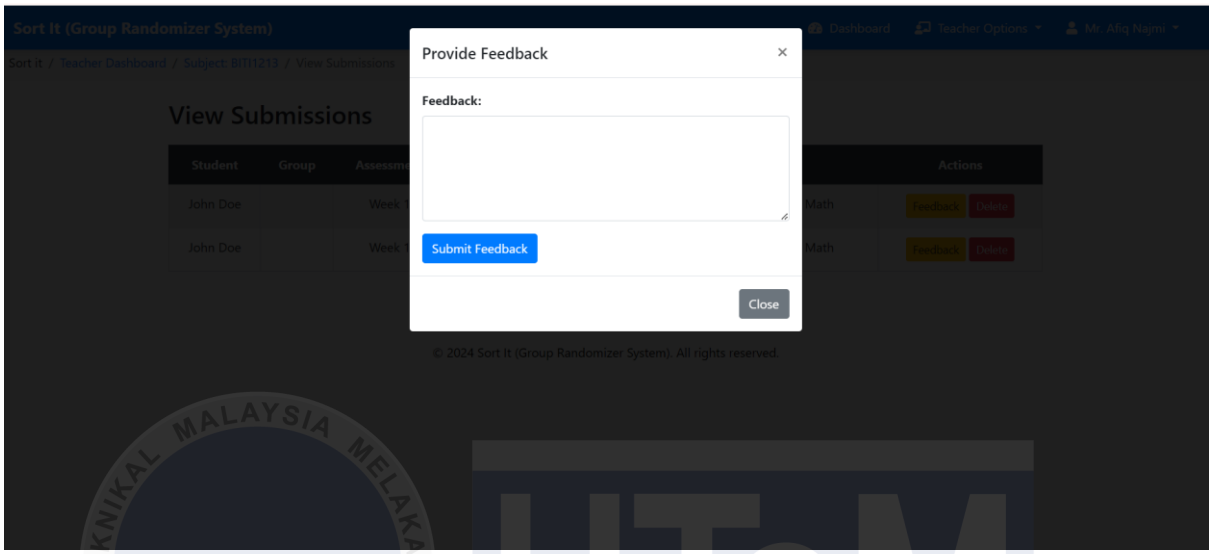

Figure 4.18 Provide Feedback Page

4.5.8 Assign Group

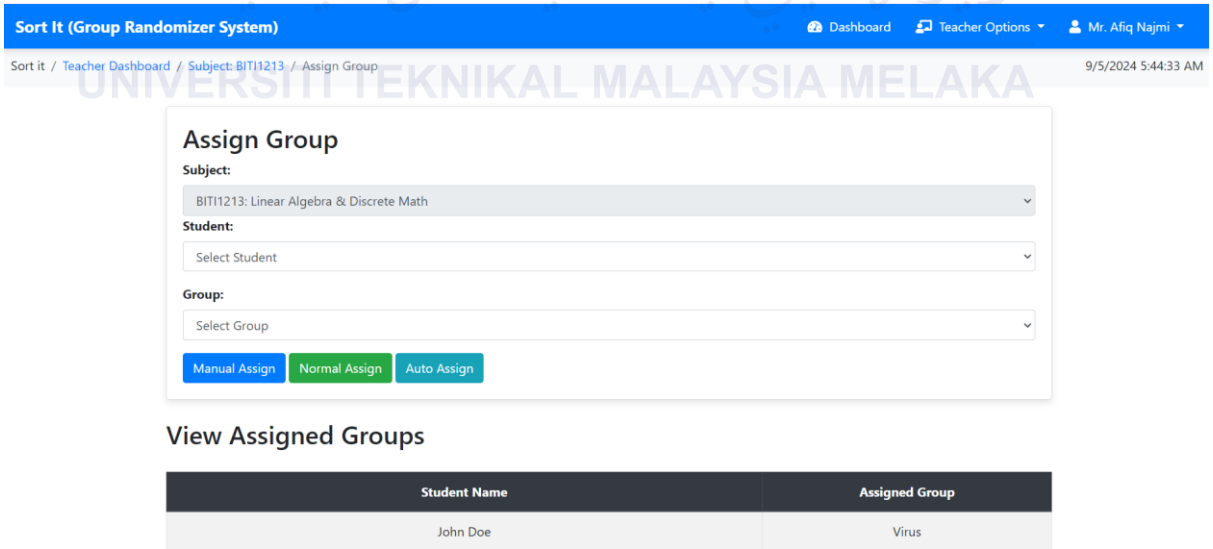

Figure 4.19 Assign Group Page

4.5.9 View Feedback

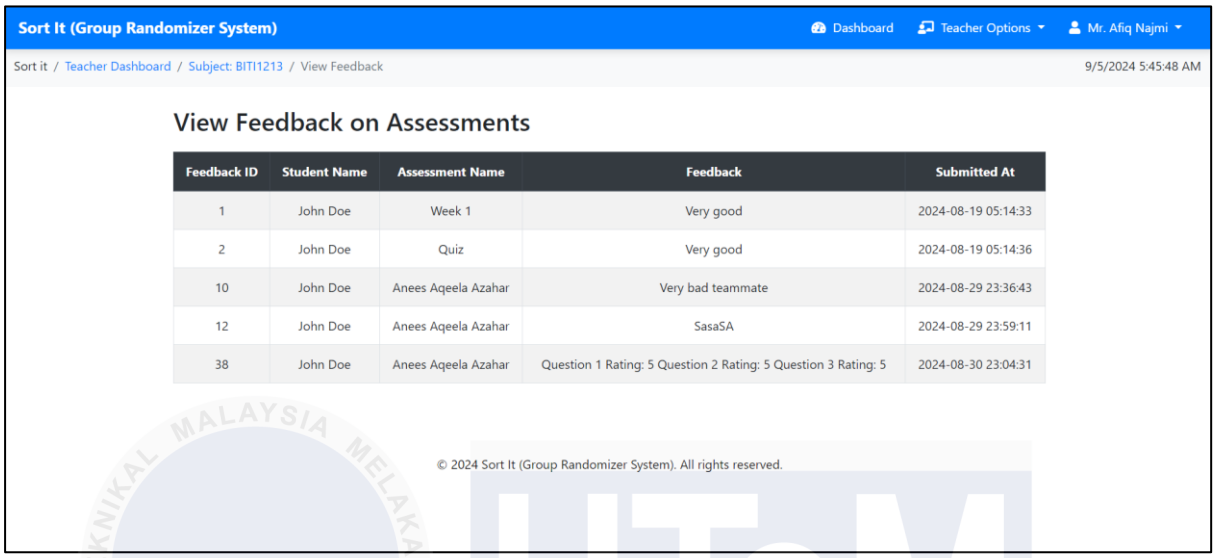

Figure 4.20 View Feedback Page

4.5.10 Student Dashboard

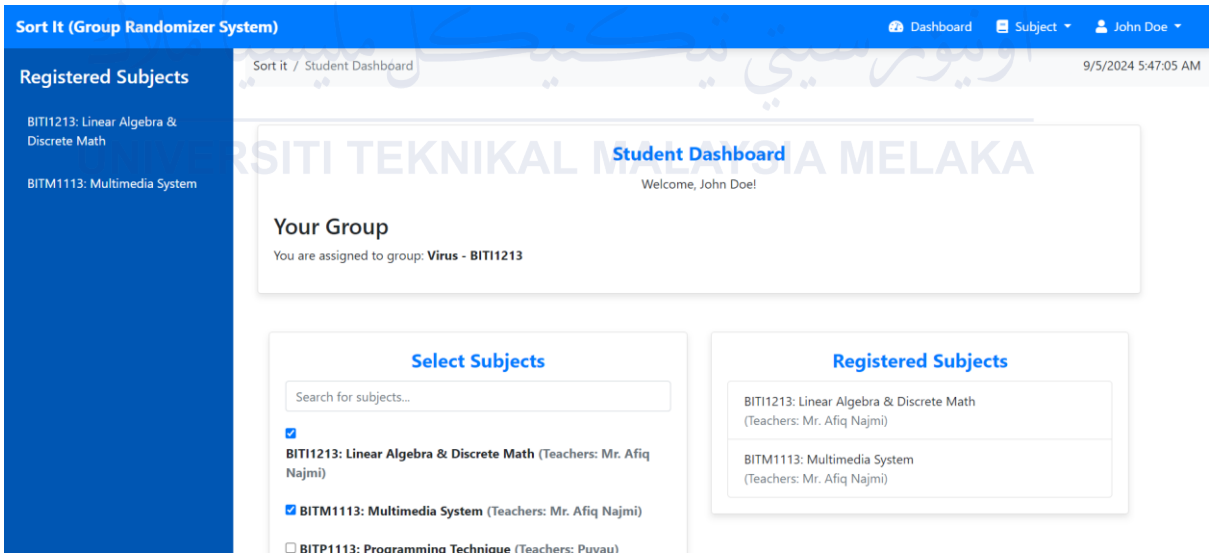

Figure 4.21 Student Dashboard Page

4.5.11 Enrol Subject

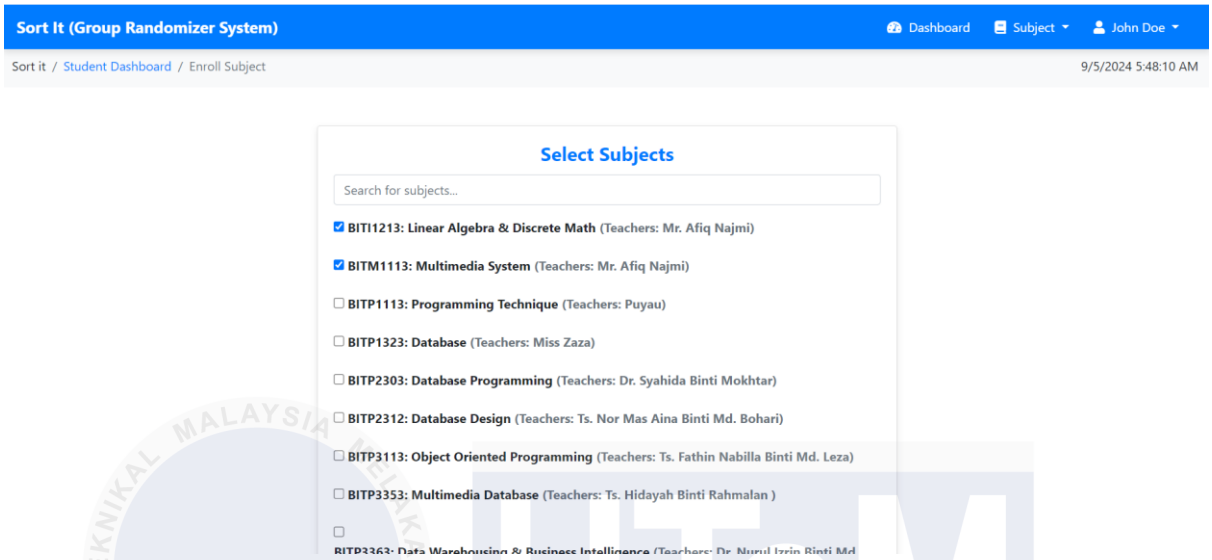

Figure 4.22 Enrol Subject Page

4.5.12 Submission

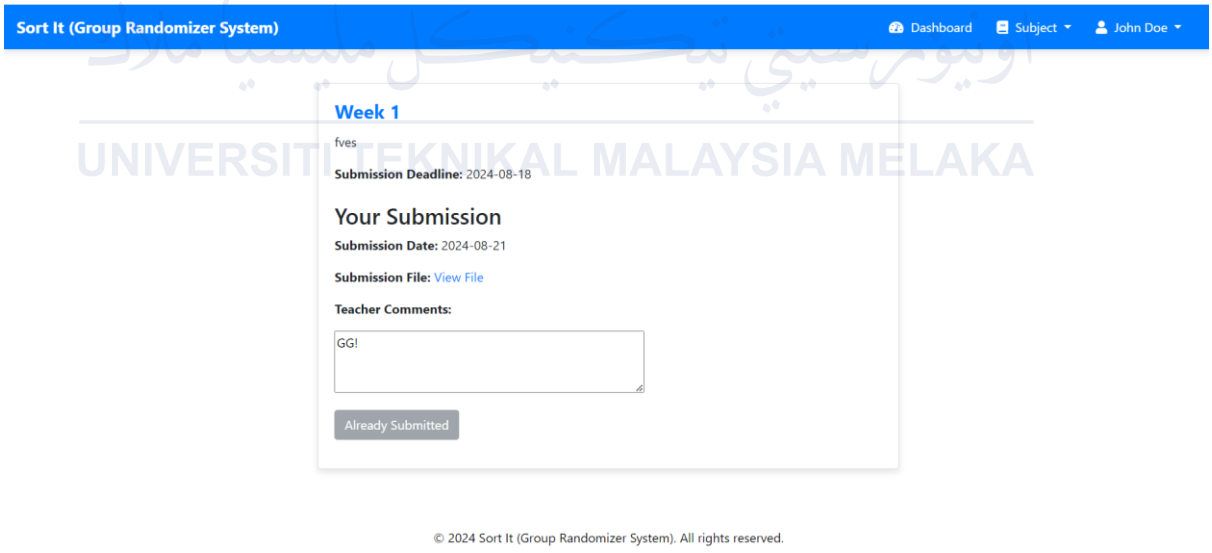

Figure 4.23 Submission Page

4.5.13 Peer Evaluation Page

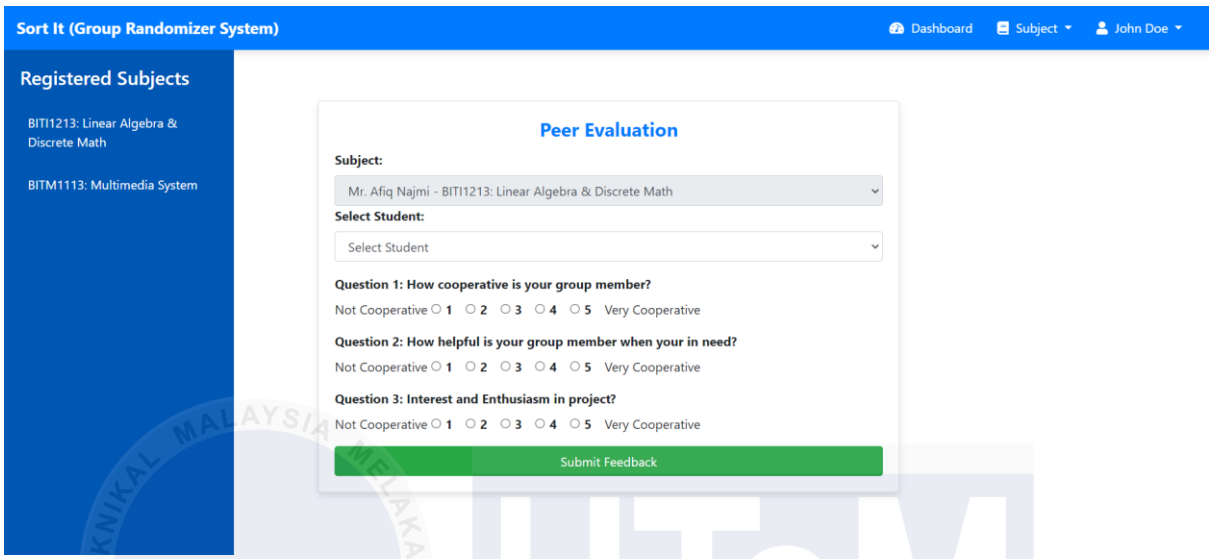

Figure 4.24 Peer Evaluation Page

4.6 Conclusion

The database architecture for the "Sort It!" group formation system is critical to ensuring smooth operations by meeting each user's specific needs, combining scalability and rigorous security measures, and facilitating efficient data management.

CHAPTER 5: IMPLEMENTATION

5.1 Introduction

During the installation phase, the system is installed and configured before advancing to the subsequent stage, the testing phase. When a system is finished being created, it moves into the implementation stage. This is where it is tested, and any bugs are fixed. This phase is crucial for transitioning the project from the development stage to the production stage. The chapter is divided into four sections: setting up the software development environment, implementing the database, managing software configuration, and discussing the implementation status. The software development environment setup will explore two subtopics of the system, which will be thoroughly researched.

5.2 Software Development Environment Setup

Environment configuration of the software development is an important factor that will enable efficient creation of the Sort It system. This phase focuses on setting up of the key tools and environments that are necessary for designing, developing as well as the managing of the application. For local server and database management, XAMPP is utilized since it forms a strong base of the backend of the system. Just like any other development environment, Visual Studio Code assists in the development and debugging of codes making it an efficient platform in which to code. Together they provide a very sound framework for the web development environment which in turn provides the necessary foundation for the successful development of the Sort It! system.

Hardware Requirement	• Laptop Dell • Windows 11 (64 bit) • Intel® Core™ i7-10210U • 15.6" HD (1366 x 768) • 512GB SSD

Software Requirement	• Microsoft Visual Studio • XAMPP
Database Requirement	• PHPMyAdmin(MySQL)

Table 5.1 Environment Setup

5.2.1 System and Database Installation Setup

i. XAMPP

1. Step 1: Download XAMPP Package Installer from the website.

Link - https://www.apachefriends.org/download.html

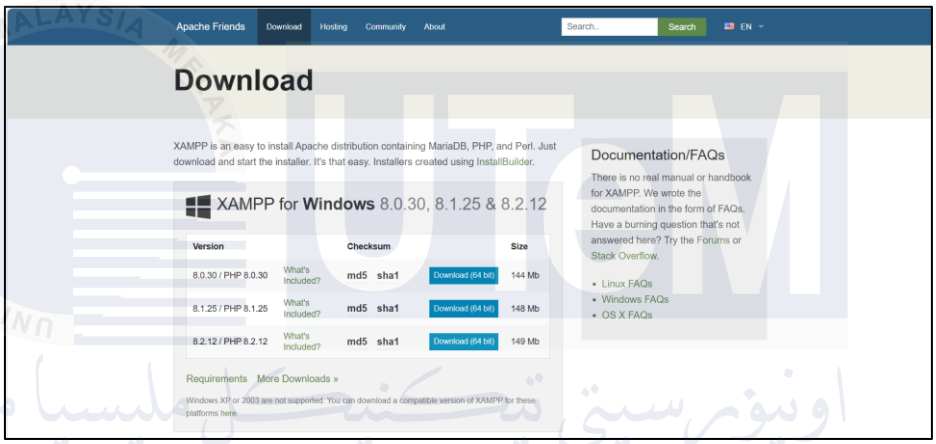

Figure 5.1 Download Page

2. Step 2: Locate the download destination folder and after that, make sure antivirus software that in laptop has been turn off before run the XAMPP.

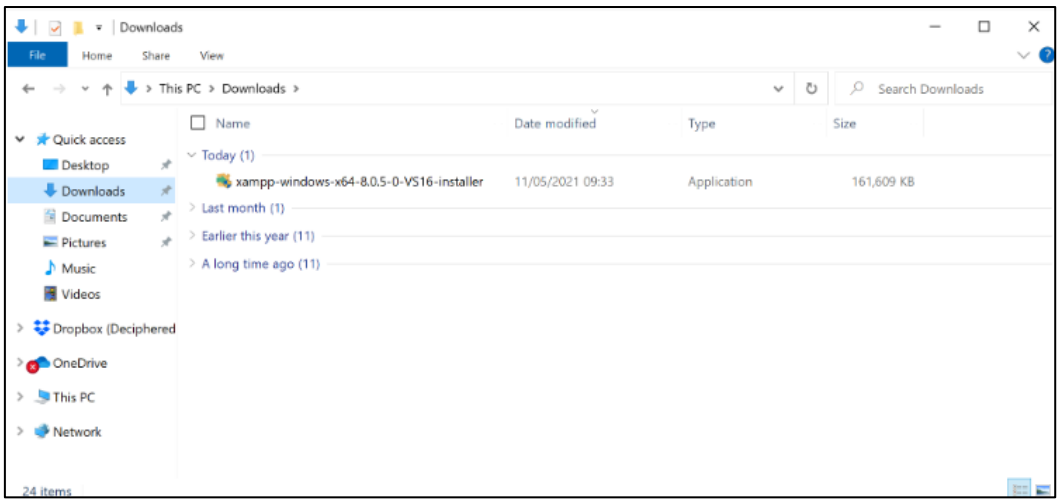

Figure 5.2 Location of XAMPP Application After download

3. Click the installation package until the window above appears and click next.

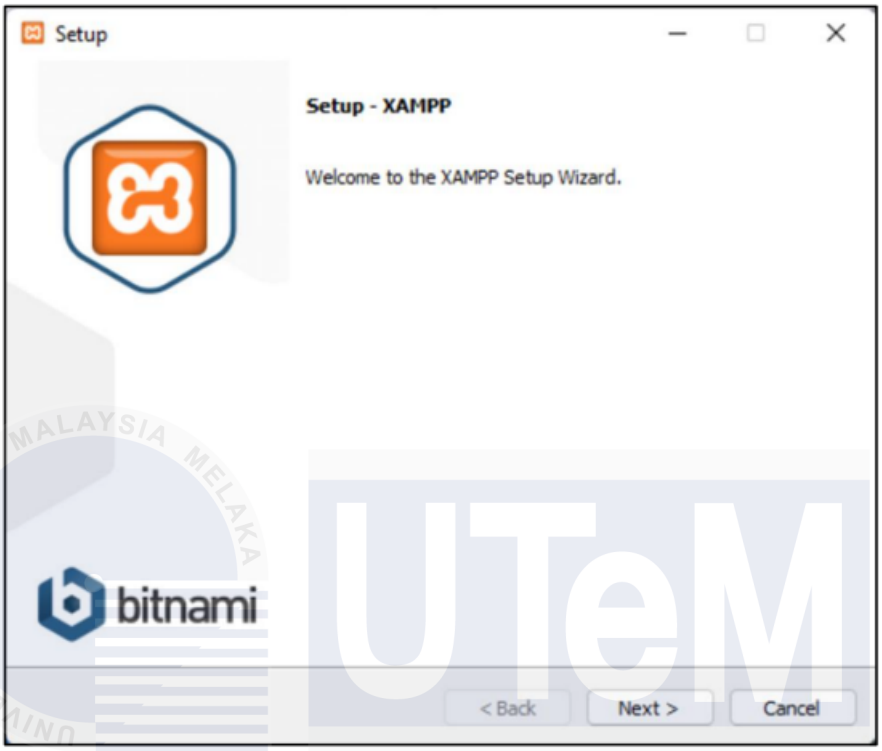

Figure 5.3 Setup Page

4. Make sure all the boxes are checked and click next.

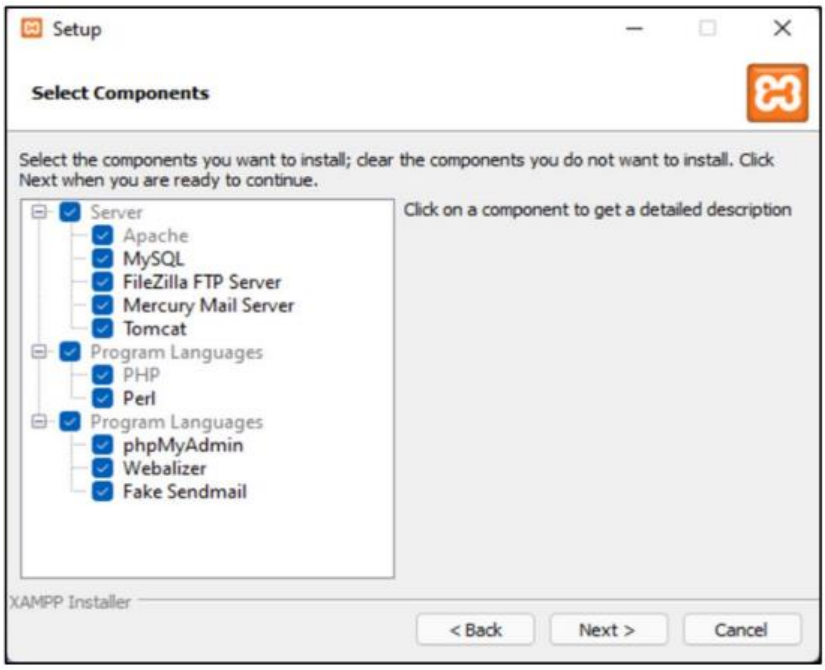

Figure 5.4 Select Component Page

5. Choose the file destination folder for the installation and click Next.

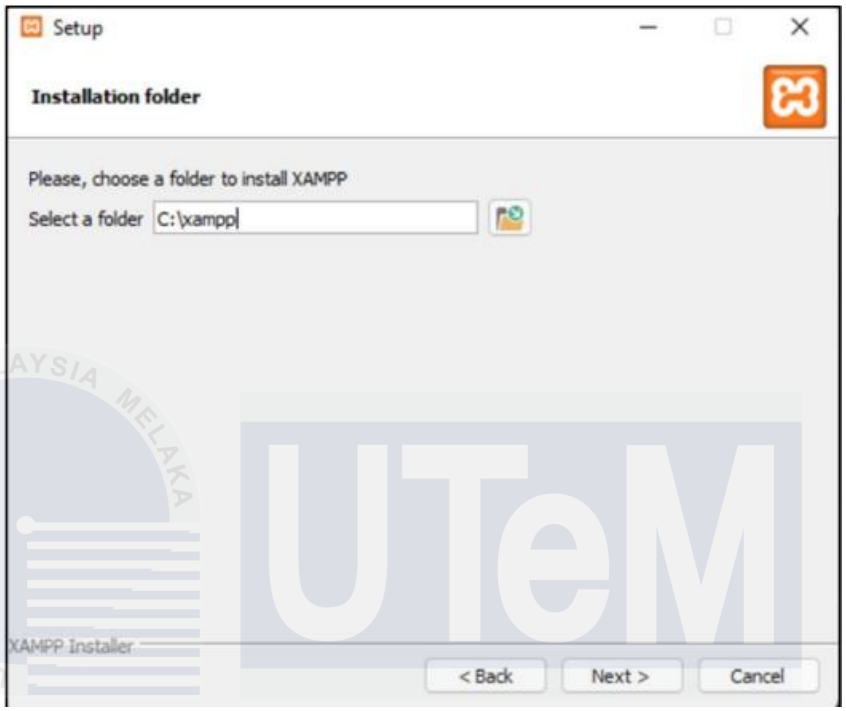

Figure 5.5 Installation Folder Page

6. Choose preferred language and click Next.

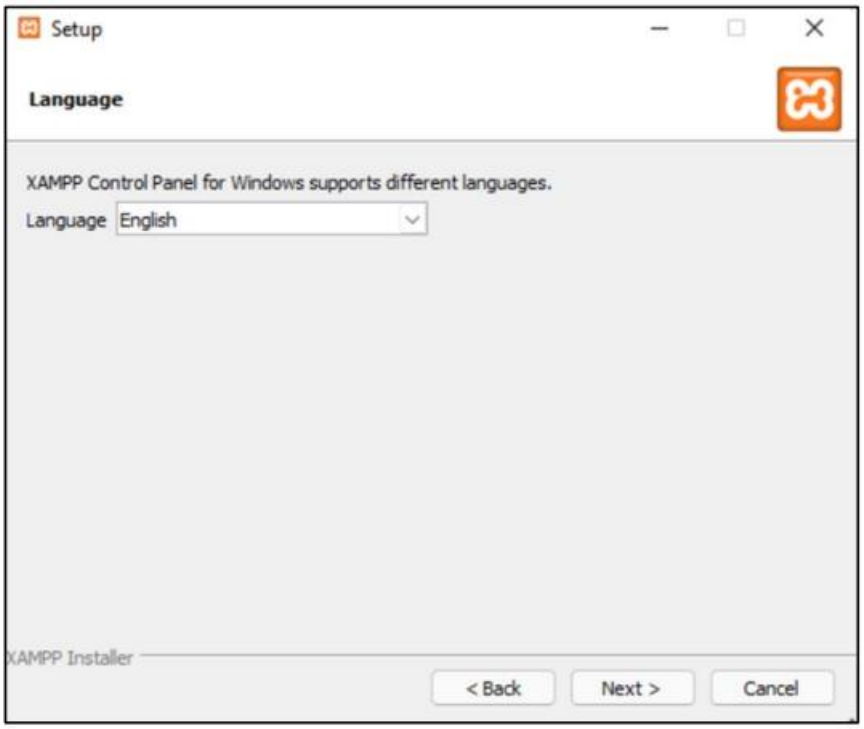

Figure 5.6 Language Page

7. The download progress bar will appear indicating the installation process has begun.

Figure 5.7 Installation in Progress

8. Click Finish.

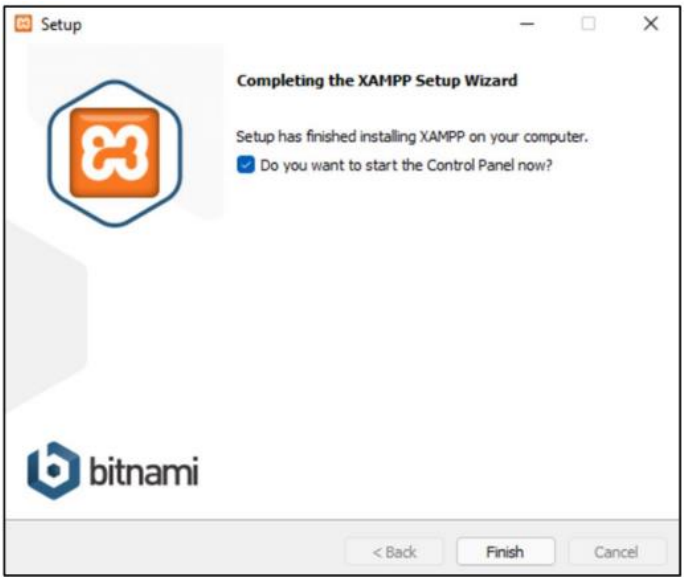

Figure 5.8 Completing the XAMPP Setup Wizard Page

9. After the installation is done, open XAMPP and the main window will show as below.

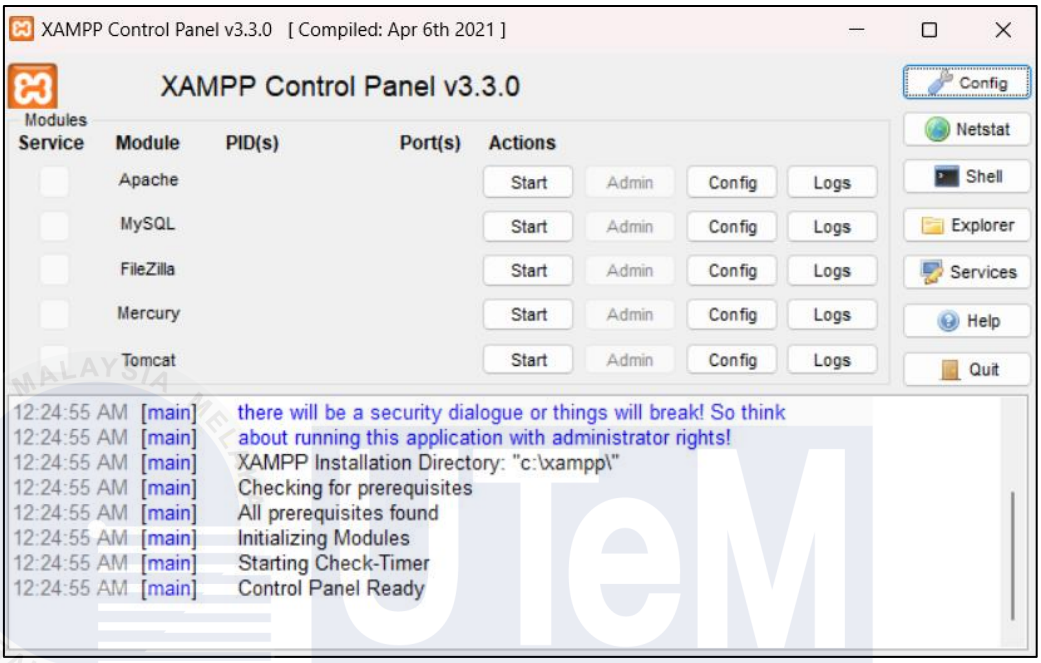

Figure 5.9 Control Panel

10. Click start button on Apache and MySQL to start. Click on MySQL Admin button to go to PhpMyAdmin page.

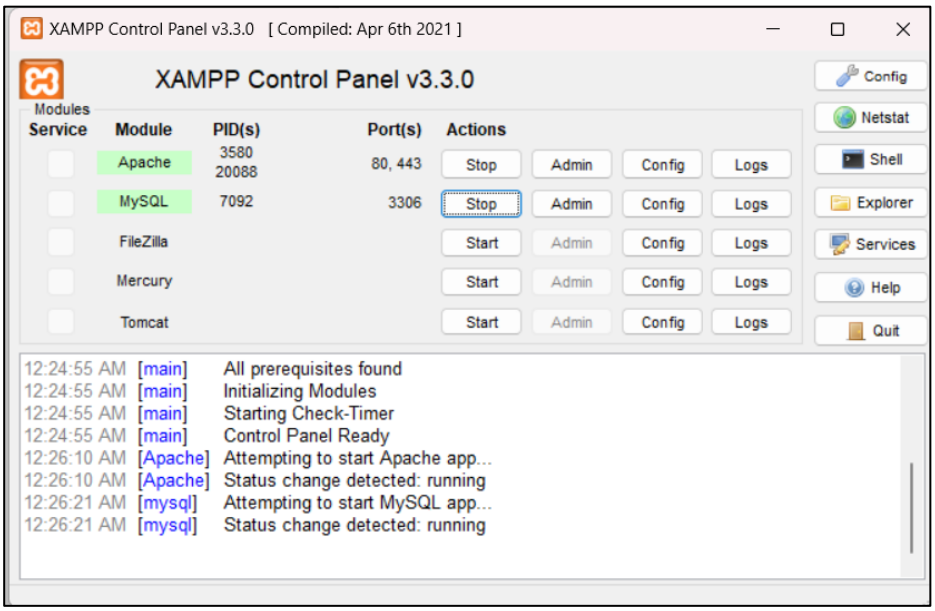

Figure 5.10 Click start at Apache and MySQL

11. The localhost will appear on browser as shown below.

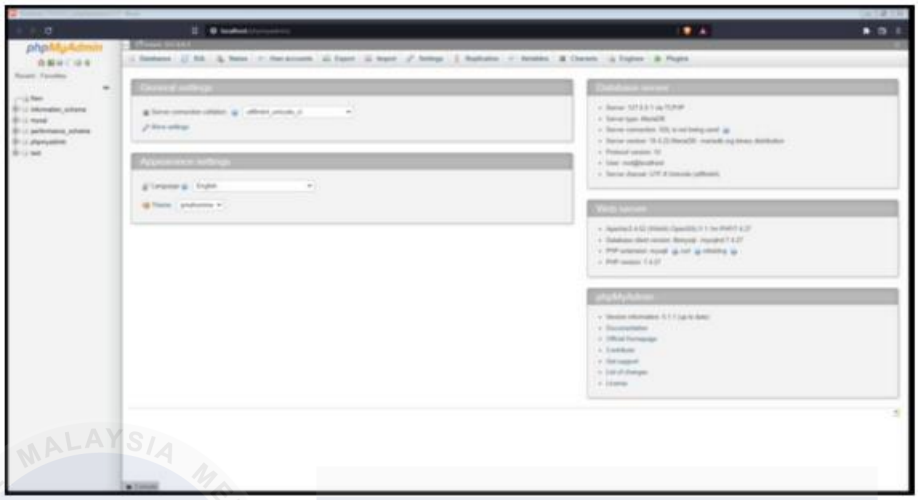

Figure 5.11 phpMyAdmin

5.3 Database Implementation

Database implementation is a critical phase in developing a comprehensive management system, ensuring that the database structure is effectively set up and integrated with the application. This process involves defining the database schema using Data Definition Language (DDL) and Data Control Language (DCL) statements to create and manage database objects such as tables, indexes, and permissions. Additionally, the implementation includes developing stored procedures and triggers to automate and streamline complex processes, enhancing data integrity and operational efficiency. The data loading process populates the database with initial test data to verify system functionality and performance. This phase is essential for ensuring that the database supports the application's requirements and operates seamlessly in real-world scenarios.

5.3.1 Data Definition Language (DDL)

1. Table Feedback

```
CREATE TABLE `feedback` (
  `id` int(11) NOT NULL,
  `student_id` int(11) NOT NULL,
  `group_id` int(11) NOT NULL,
  `preference_rating` int(11) NOT NULL,
  `adaptability_rating` int(11) NOT NULL,
  `feedback_text` text DEFAULT NULL,
  `submitted_at` timestamp NULL DEFAULT current_timestamp()
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_general_ci;
```

Figure 5.12 DDL Table Feedback

2. Table Groups

```
CREATE TABLE `groups` (
  `id` int(11) NOT NULL,
  `group_name` varchar(50) NOT NULL,
  `max_people_per_group` int(11) NOT NULL,
  `subject_id` varchar(50) NOT NULL
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_general_ci;
```

Figure 5.13 DDL Table Groups

3. Table Students

```
CREATE TABLE `students` (
  `id` int(11) NOT NULL,
  `fullname` varchar(255) NOT NULL,
  `email` varchar(50) NOT NULL,
  `password` varchar(255) NOT NULL,
  `group_id` int(11) DEFAULT NULL,
  `year` int(11) DEFAULT NULL,
  `course` varchar(255) DEFAULT NULL
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_general_ci;
```

Figure 5.14 DDL Table Students

4. Table student_feedbacks

```
CREATE TABLE `student_feedback` (
  `id` int(11) NOT NULL,
  `assessment_id` int(50) DEFAULT NULL,
  `student_id` int(11) DEFAULT NULL,
  `teacher_id` varchar(255) DEFAULT NULL,
  `feedback_text` text DEFAULT NULL,
  `feedback_type` varchar(50) NOT NULL,
  `created_at` timestamp NOT NULL DEFAULT current_timestamp()
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_general_ci;
```

Figure 5.15 DDL Table student_feedbacks

5. Table student_subjects

```
CREATE TABLE `student_subjects` (
  `id` int(11) NOT NULL,
  `student_id` int(11) NOT NULL,
  `subject_id` varchar(255) NOT NULL
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_general_ci;
```

Figure 5.16 DDL Table student_subjects

6. Table subjects

```
CREATE TABLE `subjects` (
  `id` varchar(255) NOT NULL,
  `subject_name` varchar(255) NOT NULL
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_general_ci;
```

Figure 5.17 DDL Table subjects

7. Table subject_assessment

```
CREATE TABLE `subject_assessment` (
  `id` int(11) NOT NULL,
  `subject_id` varchar(255) NOT NULL,
  `assessment_name` varchar(255) NOT NULL,
  `assessment_detail` varchar(200) DEFAULT NULL,
  `submission_date` date DEFAULT NULL,
  `file_path` varchar(255) DEFAULT NULL,
  `type` varchar(50) NOT NULL
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_general_ci;
```

Figure 5.18 DDL Table subject_assessment

8. Table submission

```
CREATE TABLE `submissions` (
  `id` int(11) NOT NULL,
  `subject_assessment_id` int(11) DEFAULT NULL,
  `student_id` int(11) NOT NULL,
  `group_id` int(50) DEFAULT NULL,
  `submission_date` date DEFAULT NULL,
  `file_path` varchar(255) DEFAULT NULL,
  `status` varchar(50) NOT NULL
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_general_ci;
```

Figure 5.19 DDL Table submission

9. Table teacher

```
CREATE TABLE `teachers` (
  `id` varchar(11) NOT NULL,
  `email` varchar(50) NOT NULL,
  `teacher_name` varchar(255) NOT NULL,
  `password` varchar(255) NOT NULL
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_general_ci;
```

Figure 5.20 DDL Table teacher

10. Table teacher_feedback

```
CREATE TABLE `teacher_feedback` (
  `id` int(11) NOT NULL,
  `teacher_id` varchar(50) DEFAULT NULL,
  `student_id` int(50) NOT NULL,
  `submission_id` int(50) DEFAULT NULL,
  `feedback_text` text DEFAULT NULL,
  `created_at` timestamp NOT NULL DEFAULT current_timestamp()
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_general_ci;
```

Figure 5.21 DDL Table teacher_feedbacks

11. Table teacher_subjects

```
CREATE TABLE `teacher_subjects` (
  `id` int(11) NOT NULL,
  `teacher_id` varchar(255) NOT NULL,
  `subject_id` varchar(255) NOT NULL
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_general_ci;
```

Figure 5.22 DDL Table teacher_subjects

5.3.2 Implementation of Main Process

5.3.2.1 Trigger

- Trigger for 'teachers'table

  The teacherIDincrement trigger automatically generates a unique teacher ID for each new booking inserted into the bookings table. It uses the sequence_booking table to maintain a sequence of unique IDs.

```
DELIMITER $$
CREATE TRIGGER `teacherIDincrement` BEFORE INSERT ON `teachers` FOR EACH ROW
BEGIN
INSERT INTO sequence_teacher VALUES (NULL);
SET NEW.id = CONCAT ('S',LPAD (LAST_INSERT_ID (),3,'000'));
END
$$
DELIMITER ;
```

Figure 5.23 SQL trigger teacherIDincrement

5.3.3 Data Loading Process

The data loading process for the database involves populating it with initial test data to ensure proper functionality and performance of the system. This is achieved through SQL INSERT statements, which create records in the database tables with predefined values. For example, the subject table was populated with sample data using the following SQL script:

```
INSERT INTO `subjects` (`id`, `subject_name`) VALUES
('BITI1213', 'BITI1213: Linear Algebra & Discrete Math'),
('BITM1113', 'BITM1113: Multimedia System'),
('BITP1113', 'BITP1113: Programming Technique'),
('BITP1323', 'BITP1323: Database'),
('BITS1123', 'BITS1123: Computer Organisation & Architecture');
```

Figure 5.24 SQL script INSERT statement table 'subject'

In this script, each INSERT statement adds a new record into the subject table. The subjectID serves as a unique identifier for each record, while the other fields such as subject_name are populated with sample data. This test data helps verify that the database schema is correctly defined, and that the application can interact with the database as expected. Additionally, it

allows for testing of database queries, 89 application logic, and user interface components before deploying the system with real data.

5.4 Conclusion

The implementation phase of the Sort It project is pivotal in transforming design concepts into a functional system, encompassing coding, configuration, and deployment. By establishing a robust software development environment with tools such as XAMPP, and Visual Studio Code, the project ensures a solid foundation for development and testing. The database implementation, involving DDL and DCL statements, and the creation of triggers, is crucial for maintaining data integrity and operational efficiency. The data loading process, utilizing SQL INSERT statements, populates the database with initial test data, validating the system's functionality and performance. This comprehensive approach ensures that the system meets its requirements and is prepared for effective real-world deployment.

CHAPTER 6: TESTING

6.1 Introduction

This chapter provides an overview of the testing phase for the Sort It! Group Management System. Testing ensures that the system functions as intended, meets all specified requirements, and is free of defects. The testing strategy includes white-box testing for internal code verification and black-box testing for functional verification. Additionally, testing for both system and database components is included.

6.2 Test Plan

A test plan is a document that outlines the strategy, scope, resources, and schedule for testing a system or product. It provides a structured approach to testing by detailing how testing will be performed to ensure that the system meets the required standards and functions correctly.

6.2.1 Test Organization

The test organization explains the personnel involved in the testing phases of the Sort It! Group Management System. In this part, a test group will be established and responsible for managing, executing, designing, reviewing, and completing the testing tasks. Within the Sort It! Group Management System context, the system developer serves as the primary tester. The developer is well-positioned to identify bugs and errors firsthand. The system developer will conduct tests on all modules to ensure the system's integrity during development. This approach aims to minimize bugs and reduce errors in the final product. Additionally, end users contribute to the testing process by verifying system functionalities. Figure 6.1 illustrates the test organization in a hierarchical structure.

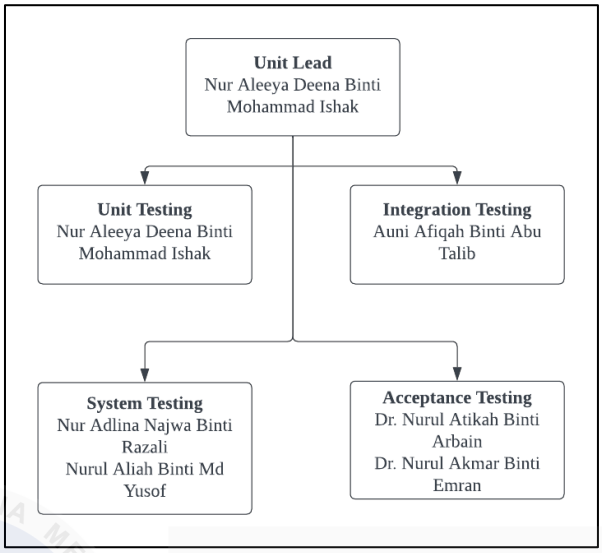

Figure 6.1 Hierarchy of Test Organization

6.2.2 Test Environment

Table 6.1 shows the details of test environment has been made.

System Configuration	Specification (Server)
Operating System	Windows 11(64 bit)
Database	phpMyAdmin
Random Access Memory (RAM)	16GB
Hard Disk	512GB
Processor	13th Gen Intel(R) Core (TM) i7-1355U 1.70 GHz
Software	Apache Tomcat

Table 6.1 Test Environment

6.2.3 Test Schedule

Testing for the Sort It! Group Management System is categorized into three types: unit testing, integration testing, and user acceptance testing. Each test output is documented to ensure that any necessary adjustments can be made after each testing phase. Table 6.2 provides a detailed overview of the tests conducted.

Modules	Type	Duration/ Cycles
Registration	Unit Test Integration Test User Acceptance Test	5 days/ 5 times
Log In System	Integration Test User Acceptance Test	3 days/ 5 times
Project Management	Integration Test User Acceptance Test	4 days/ 5 times
Score Recording	Integration Test User Acceptance Test	7 days/ 10 times
Milestone Tracking	Unit Test Integration Test User Acceptance Test	5 days/ 6 times

Table 6.2 Test Schedule

6.3 Test Strategy

- Black Box Testing Strategy

  Black-box testing is an important type of test which sits on the functional aspect of the Sort It! Group Management System with excluding internal code of this program. Testers create situations depending on the user needs and the system characteristics, as far as that all relevant features function as intended from users' point of view. This approach comprises of functional testing, usability testing, compatibility testing and regression testing used to check that the system executes necessary function correctly precisely and is not cumbersome despite the features incorporated, is compatible with multiple devices and browsers.

- White Box Testing Strategy

  White-box testing is when the code of the software is analyze with much focus being put on the inner structures of the Sort It! Group Management System. This approach requires knowledge of the system's code to make

certain that the different parts of the code are invoked functioning as expected. It is not limited to unit testing, which should be used to verify each piece of code as complete in isolation subsystems, system integration testing to confirm between subsystems and code reviews to assess the possibility of several problems and to prevent them when possible. The purpose is to build that its internal workings, or what goes on inside it that is so far not openly observable, is functioning correctly and effectively.

6.3.1 Classes of Tests

Functional Testing, Unit Testing, Integration Testing, and User Acceptance Testing are the four tests used to assess the Sort It! Group Management System's competencies and guarantee that it meets its needed outcomes. These exams are used for various modules, including participant registration, project management, evaluator assignment, score recording, and milestone tracking. Table 6.3 outlines the different types of tests in detail.

Class of Test	**Explanation**
Functional Testing	Evaluate the system's functionality to ensure that it meets the requirements, especially in group management and assessment management.
Unit Testing	Test separate system components, such as assign group logic, to ensure each portion works appropriately.
Integration Testing	Ensure several modules (such as Student dashboard and Subject Dashboard) work seamlessly.
User Acceptance Testing	The system meets user demands and performs well in real-world circumstances, emphasizing the entire user experience within the Sort It! Group Management System

Table 6.3 Description of classes of test

6.4 Test Design

Test design refers to the process of creating detailed test cases and scenarios based on the requirements and specifications of a software system. The goal is to ensure that the system functions correctly and meets its intended goals. Effective test design involves planning how to verify that the system performs as expected in different conditions.

6.4.1 Test Description

6.4.1.1 User Registration Module

The Registration module is essential for gaining access to the Sort It! Group Management System. Users are required to give their personal information to acquire a username. After completing the registration process successfully, individuals can access their accounts by using the username and password they just made. In the event of an unsuccessful registration, users will be required to retry the registration procedure

Test Case ID	Description	Action	Expected output
CS01	Full Name = blank Email = blank Password = blank	No input provided	ERROR
CS02	Full Name = Syed Mir Iqbal Email = blank Password = blank	Email and password left empty	ERROR
CS03	Full Name = Syed Mir Iqbal Email = smiball13@mail.com Password = blank	Password left empty	ERROR
CS04	Full Name = Syed Mir Iqbal Email = smiball13@mail.com Password = iqiemercury	All necessary input inserted	SUCCESS

Table 6.4 Description of User Registration Module

6.4.1.2 Login Module

Test Case ID	Description	Action	Expected output
CSL01	Email = blank Password = blank	No input provided	ERROR
CSL02	Email = smibal31 Password = blank	Email format is wrong, password left empty	ERROR
CSL03	Email = smiball13@mail.com Password = blank	Password left empty	ERROR
CSL04	Email = smiball13@mail.com Password = iqiemercury	All necessary input inserted	SUCCESS

Table 6.5 Description for Login Module

6.4.1.3 Assessment Module

Test Case ID	Description	Action	Expected output
CSA01	Assessment Name = blank Assessment Detail = blank Type = blank Submission date = blank Upload File = blank Subject = blank	No input provided	ERROR
CSA02	Assessment Name = Week 1 Assessment Detail = blank Type = blank Submission date = blank Upload File = blank Subject = blank	Assessment detail, type, submission date and subject are empty	ERROR

CSA03	Assessment Name = Week 1 Assessment Detail = Please do your work Type = blank Submission date = blank Upload File = blank Subject = blank	Type, submission date and subject are empty	ERROR
CSA04	Assessment Name = Week 1 Assessment Detail = Please do your work Type = Lab Exercise Submission date = blank Upload File = blank Subject = blank	Submission date and subject are empty	ERROR
CSA05	Assessment Name = Week 1 Assessment Detail = Please do your work Type = Lab Exercise Submission date = 26/09/2023 Upload File = blank Subject = blank	Submission date is past the day assessment was added, subject is empty	ERROR
CSA06	Assessment Name = Week 1 Assessment Detail = Please do your work Type = Lab Exercise Submission date = 26/09/2024 Upload File = blank Subject = blank	Subject is empty	ERROR
CSA07	Assessment Name = Week 1 Assessment Detail = Please do your work	All necessary input inserted	SUCCESS

	Type = Lab Exercise Submission date = 26/09/2024 Upload File = blank Subject = BITI1213		

Table 6.6 Description for Assessment Module

6.4.2 Test Data

In this part, the real data will be used to ensure the system for correctness and system effectiveness. Table 6.7 show the example of test data.

COMPONENT: LOGIN		
Test No	**Attribute**	**Data**
Test01	Teachers	
	Username	afiqnajmi@mail.com
	Password	*******
Test02	Student	
	Username	smiball13@mail.com
	Password	**********

Table 6.7 Description of Login test data

6.5 Test Results and Analysis

System: Sort It! Group Management System

Version: 1.0

Module: Registration Module

Test Number	Action	Result	Pass Initials (OK/Fail)
	Valid input: Based on each input type Condition: User	System will prompt with new username based on user priority	OK

	enters personal details		

Table 6.8 Test Result and Analysis for Registration Module

System: Sort It! Group Management System

Version: 1.0

Module: Registration Module

Test Number	Action	Result	Pass Initials (OK/Fail)
	Valid input: Condition: email and password are already in the database Input: Email: afiqnajmi@mail.com Password: *******	Able to access the system	OK
	Valid input: Condition: email and password are not in the database Input: Email: testing Password: testing	Display error message	OK

Table 6.9 Test Result and Analysis for Login Module

6.6 User Acceptance

The Sort It! Group Management System project utilized black-box testing techniques during the User Acceptance Testing (UAT) phase to ensure the system met the functional requirements and expectations of its end-users, such as teachers and students. The goal of UAT was to verify that the system operated correctly from the user's perspective, without focusing on the underlying code or logic. Black-box testing was essential in evaluating how the system behaved

when users interacted with it, ensuring it delivered accurate results based on inputs provided by the users.

The UAT process involved administering a Google Form survey to students and teachers, who represent the actual end-users of the system. They were asked to use the group management features, including registering, forming groups, submitting assignments, and viewing feedback. The survey gathered their feedback on various aspects of the system, such as user-friendliness, system performance, and overall satisfaction. The feedback was critical in identifying areas where the system could be improved, especially before the final deployment phase.

Survey results showed that most users found the system responsive, particularly when switching between modules like group creation and assessment submission. Users rated the system's responsiveness between 4 and 5 on a scale, indicating general satisfaction. However, a few users reported occasional delays or lags, suggesting that further optimization might improve the user experience.

Moreover, most users rated the system as intuitive, with core functionalities like uploading assignments and viewing group lists being straightforward. Ratings between 4 and 5 for ease of use were common, although some participants identified areas, such as the layout of buttons or navigation paths, that could be more intuitive.

Despite these minor issues, the feedback was predominantly positive, with many users agreeing that the system helped streamline group management and fostered a smoother learning experience. However, the testing phase also revealed that additional features, such as the ability to personalize the interface or clarify certain error messages, would further enhance the system’s usability.

In conclusion, the UAT feedback provided valuable insights, affirming that the Sort It! system was generally well-received, but with room for minor improvements in performance and user interface design before full-scale deployment.

6.7 Conclusion

The Sort It! Group Management System was tested to guarantee its dependability and correctness. Through unit, integration, and user acceptance testing, the system developer and end users engaged in a well-organised approach whereby several modules were tested. Black-box and white-box methods were used to investigate internal logic as well as outside capability. Actual test data was used to verify the performance of the system; the test environment, timetable, and techniques were extensively recorded. The findings verified that the system satisfied the necessary criteria; any found flaws were fixed to improve the quality of the finished good

CHAPTER 7: CONCLUSION

7.1 Introduction

The Sort It! Group Management System is designed to streamline the management and oversight of student group allocations, addressing the complexities involved in forming balanced and diverse student groups based on their GPA. The system includes a variety of functionalities to manage group creation, submissions, and feedback efficiently. During the development of this project, several challenges were encountered, such as limited familiarity with specific technologies, a lack of comprehensive reference materials and source code, and constraints related to time and other commitments. However, through perseverance, hard work, and collaboration with peers, we were able to successfully develop and complete the system.

7.2 Observation on Weakness and Strengths

Strength	Weaknesses
User-Friendly Interface: The system is designed to be intuitive and easy to navigate for both students and teachers, reducing the learning curve and making it accessible to a wide range of users.	Limited Personalization Options: While the system is functional, some users may find the lack of customization (e.g., interface themes or role-based access) limiting, which can affect user experience for those looking for more control over their interface.
Automated Group Allocation: By automating group creation based on GPA, the system ensures a balanced and fair group composition, saving time and effort for teachers and administrators.	Potential for Performance Issues: As reported during testing, there may be occasional delays when switching between system modules, indicating that further optimization is needed to enhance speed and performance.
Seamless Integration: The system integrates different functionalities such as submissions, group management, and	Dependency on Data Accuracy: The effectiveness of the group randomization feature heavily depends on the accuracy of GPA data. Any errors or

feedback collection, making it a one-stop solution for managing collaborative work.	inconsistencies in input data can lead to imbalanced group allocations.
Customizable Features: The ability to customize group sizes, assign different roles, and view feedback in real-time enhances the overall flexibility of the system.	Scalability Challenges: The current system might face challenges when scaling to a larger number of students or groups, as performance and efficiency could degrade without sufficient system resources or optimization.
Increased Efficiency: Automating the manual tasks of group assignment and tracking submissions greatly reduces human error and improves the overall efficiency of the group management process.	Limited Error Handling: While most error messages are clear, there are instances where they may lack clarity or fail to provide detailed guidance for users, which could cause frustration, particularly for less tech-savvy individuals.

Table 7.1 Strength and Weakness of the System

7.3 Propositions for Improvement

1. Enhanced Customization Options:

- User Interface Personalization: Allow users to customize the system interface by adjusting text sizes, color themes, and layout preferences to enhance the user experience and cater to individual needs.
- Role-Based Access Control: Implement role-based permissions, allowing different levels of access for teachers, students, and administrators. This could provide better control and security, especially for managing sensitive data like grades and submissions.

2. Performance Optimization:

- Optimize System Response Time: Address the occasional delays noted during testing by optimizing system performance, particularly when switching between modules or processing large datasets. Consider

implementing efficient data processing algorithms and reducing resource-heavy operations to ensure smooth performance.

- Scalability Enhancements: Improve the scalability of the system so that it can handle larger numbers of students and groups without sacrificing performance. This could involve improving server infrastructure, database indexing, and caching mechanisms.

3. Improve Error Handling:

- More Detailed Error Messages: Improve the clarity and detail of error messages, providing users with actionable steps for resolving issues. This can be especially helpful for less technical users who may struggle with vague or generic error warnings.
- Error Logging and Reporting: Implement a comprehensive error logging and reporting system to help administrators identify and resolve issues quickly.

4. Advanced Grouping Features:

- Dynamic Group Composition Rules: Offer more advanced group composition options, such as allowing teachers to define additional criteria for group randomization beyond GPA (e.g., skills, interests, or project preferences). This could help create more cohesive and well-rounded groups.
- Manual Overrides for Group Assignment: Provide teachers with the option to manually adjust group assignments after the system randomization, giving them more flexibility to fine-tune group composition based on their specific requirements.

5. Additional Functionalities:

- Real-Time Collaboration Tools: Integrate features like shared workspaces, document collaboration, and chat functionalities, allowing students to work on group projects within the system itself.

- Progress Tracking and Analytics: Add functionality for teachers and students to track group progress, analyze group performance over time, and identify areas for improvement. This could include visual dashboards with completion percentages, group contribution metrics, and more.
- Notifications and Alerts: Implement an automatic notification system to remind students of upcoming submission deadlines or notify them when a teacher provides feedback.

7.4 Project Contribution

The Sort It! Group Management System contributes significantly to the educational environment by streamlining the process of group formation and collaboration. By automating the creation of student groups based on GPA, the system ensures fair and balanced group compositions, saving teachers considerable time and effort. This approach fosters a collaborative environment where students with varying academic strengths can work together, enhancing peer-to-peer learning experiences. The system helps teachers reduce the manual work involved in managing groups and instead focus on instructional activities and providing meaningful feedback to students.

Additionally, the system promotes transparency and accountability within group projects. Teachers can track each student's contributions and monitor group progress, making the allocation and submission process more transparent and reducing potential conflicts within groups. The centralized submission system also simplifies the process for students, ensuring that they can easily track deadlines and submission statuses, while teachers can review and grade assignments efficiently. This streamlined approach benefits both students and teachers, enhancing overall productivity and workflow in the classroom.

The project also offers flexibility and scalability, making it adaptable to different educational settings. The system is customizable, allowing institutions to tailor its features based on specific needs, and it can scale to accommodate larger groups or more complex settings. Moreover, with the feedback mechanism integrated into the system, both students and teachers can

continuously provide evaluations, ensuring that the system remains responsive to user needs and helps improve the group learning experience over time.

Data-driven decision-making is another important contribution of the system. By tracking student performance, submission rates, and group progress, teachers and administrators gain access to valuable data that can inform decisions about group dynamics, individual performance, and curriculum planning. Combined with a user-friendly interface, the system ensures that even non-technical users can navigate and utilize it effectively, making it a valuable tool for any academic setting.

Overall, the Sort It! Group Management System contributes to creating a more efficient, transparent, and equitable learning environment by automating group assignments, facilitating collaboration, and supporting teachers in managing group-based learning. This helps improve the overall experience for both students and educators, promoting better learning outcomes and fostering a more collaborative classroom environment.

7.5 Conclusion

The development of the Sort It! Group Management System represents a significant advancement in addressing the operational challenges faced by teachers, students, and educational administrators. By automating key processes such as group formation, submission tracking, and feedback collection, the system not only enhances operational efficiency but also contributes to a better overall user experience for all parties involved. The system's ability to balance group compositions based on GPA fosters collaboration among students of varying academic strengths, which promotes a fairer and more dynamic learning environment.

While the system demonstrates considerable strengths in terms of efficiency, usability, and scalability, there are still limitations that need attention. Some of these include the limited customization options for users, occasional performance lags, reliance on accurate input data,

and potential challenges with scalability as the number of users increases. These areas for improvement present opportunities for refinement and further enhancement, particularly in optimizing system performance and expanding personalization options to meet user needs more effectively.

Proposed improvements, such as enhanced customization features, improved system performance, expanded scalability, and the integration of real-time collaboration tools, provide a clear path for future iterations of the system. By addressing these areas, the system can evolve into an even more comprehensive and user-friendly tool that better meets the diverse needs of students, teachers, and administrators alike.

In conclusion, the Sort It! Group Management System has made a significant contribution to group-based learning and educational management. By automating and streamlining complex tasks, it offers a modern, efficient solution to longstanding challenges in student group management. With ongoing enhancements and improvements, the system is well-positioned to become a leading solution in educational settings, delivering tangible benefits to all stakeholders involved.

REFERENCES

Burkart, N., & Huber, M. F. (2021). A survey on the explainability of supervised machine learning. *Journal of Artificial Intelligence Research*, *70*, 245–317. https://doi.org/10.1613/jair.1.12228

GfG. (2024, February 22). *Randomized algorithms*. GeeksforGeeks. https://www.geeksforgeeks.org/randomized-algorithms/

Haelermans, C. (2022). The effects of group differentiation by students' learning strategies. *Instructional Science*, *50*(2), 223–250. https://doi.org/10.1007/s11251-021-09575-0

Liang, C., Majumdar, R., & Ogata, H. (2021). Learning log-based Automatic Group Formation: System Design and classroom implementation study. *Research and Practice in Technology Enhanced Learning*, *16*(1). https://doi.org/10.1186/s41039-021-00156-w

Wang, X., Zhang, Y., & Zhu, R. (2022). A brief review on Algorithmic Fairness. *Management System Engineering*, *1*(1). https://doi.org/10.1007/s44176-022-00006-z

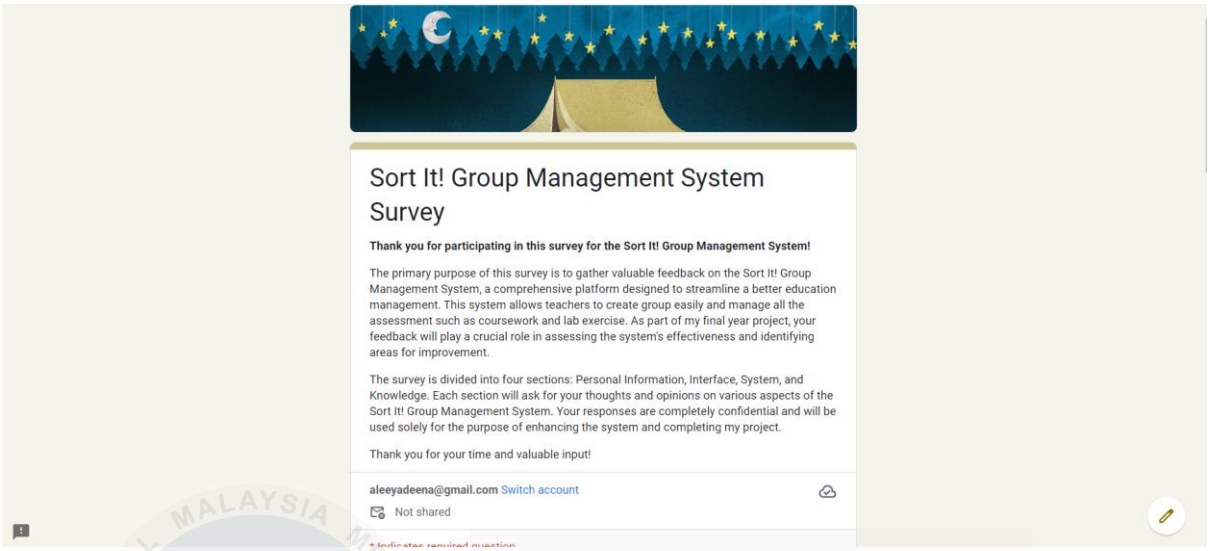

Figure 2 Permission Letter for Survey Purpose

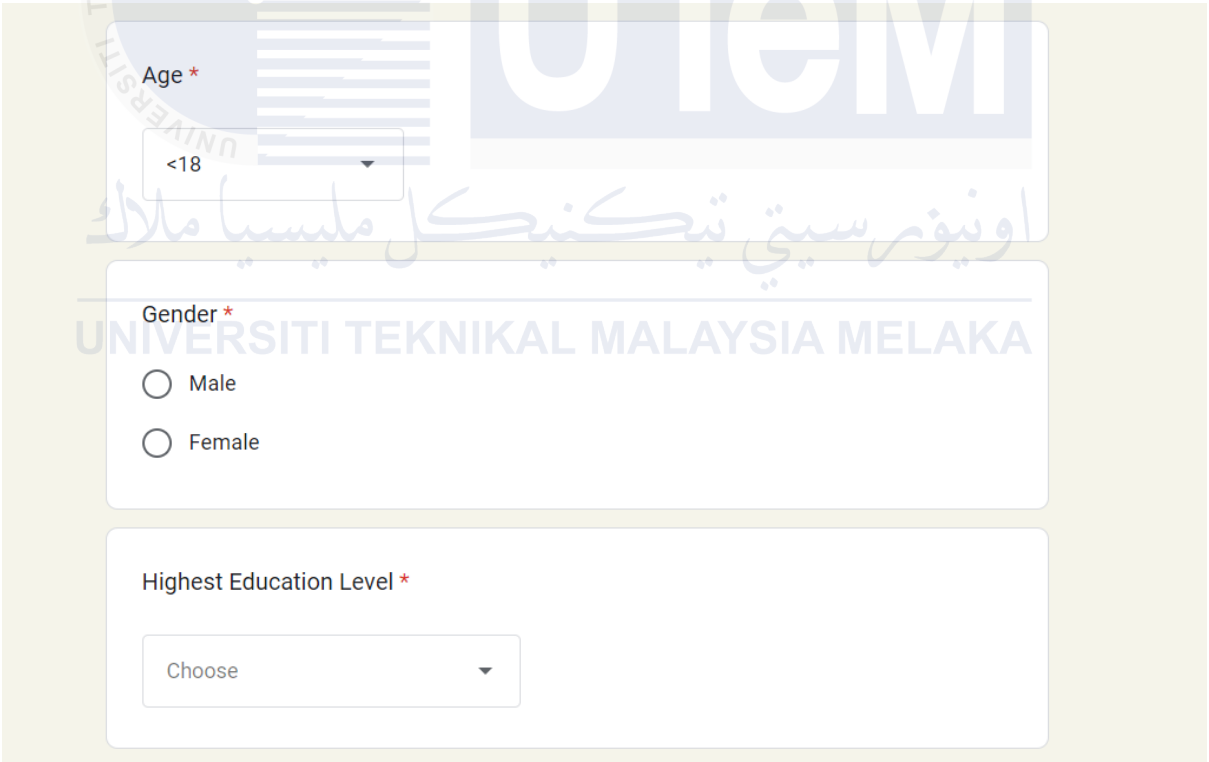

Figure 3 Survey form for testing purpose (b)

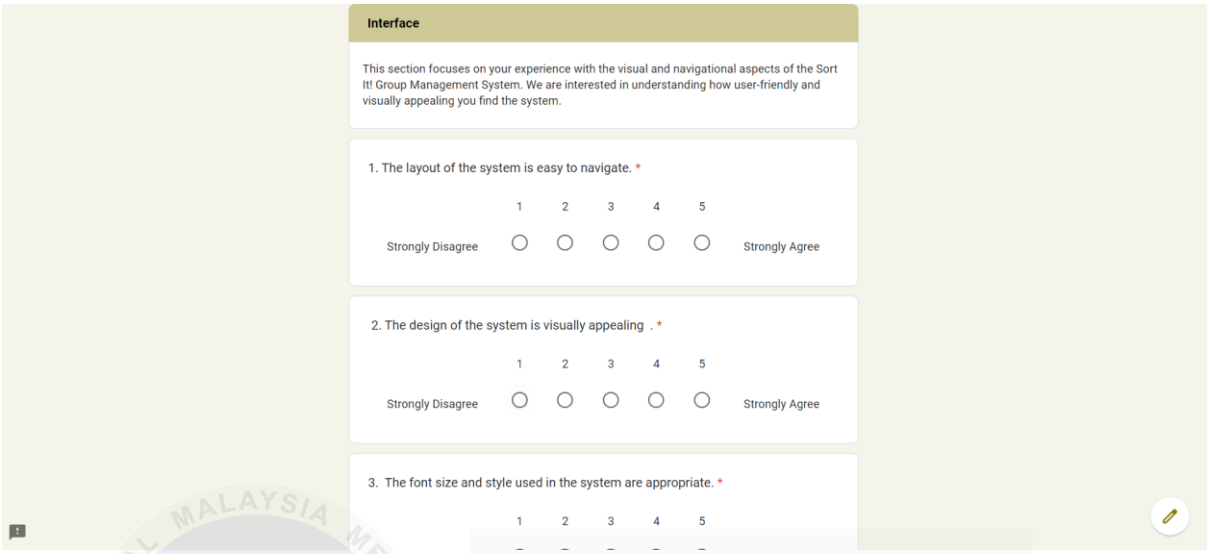

Figure 4 Survey form for testing purpose (c)

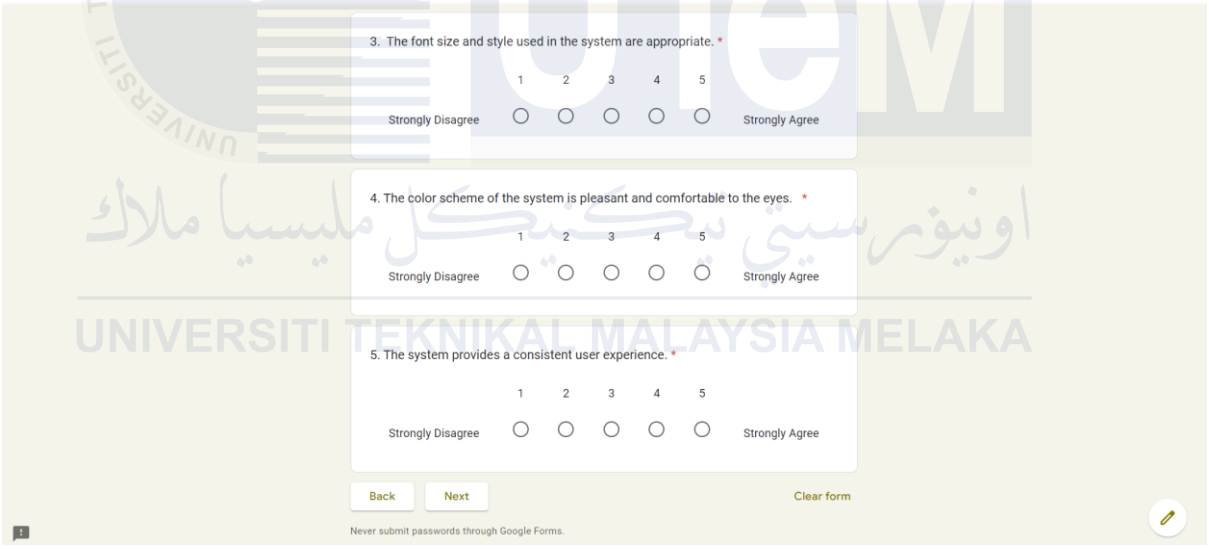

Figure 5 Survey form for testing purpose (d)

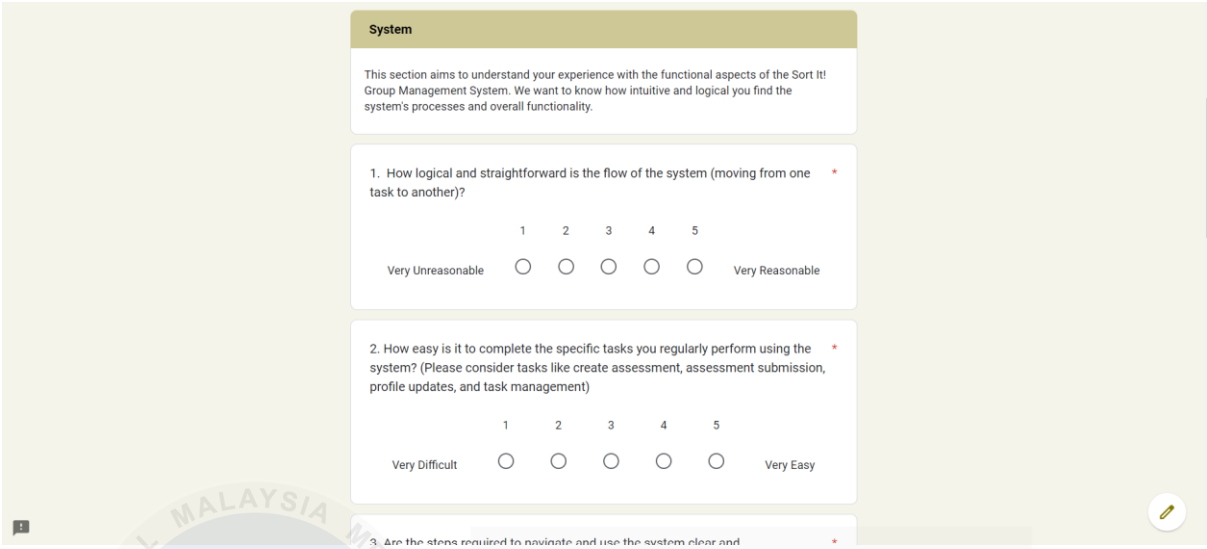

Figure 6 Survey form for testing purpose (e)

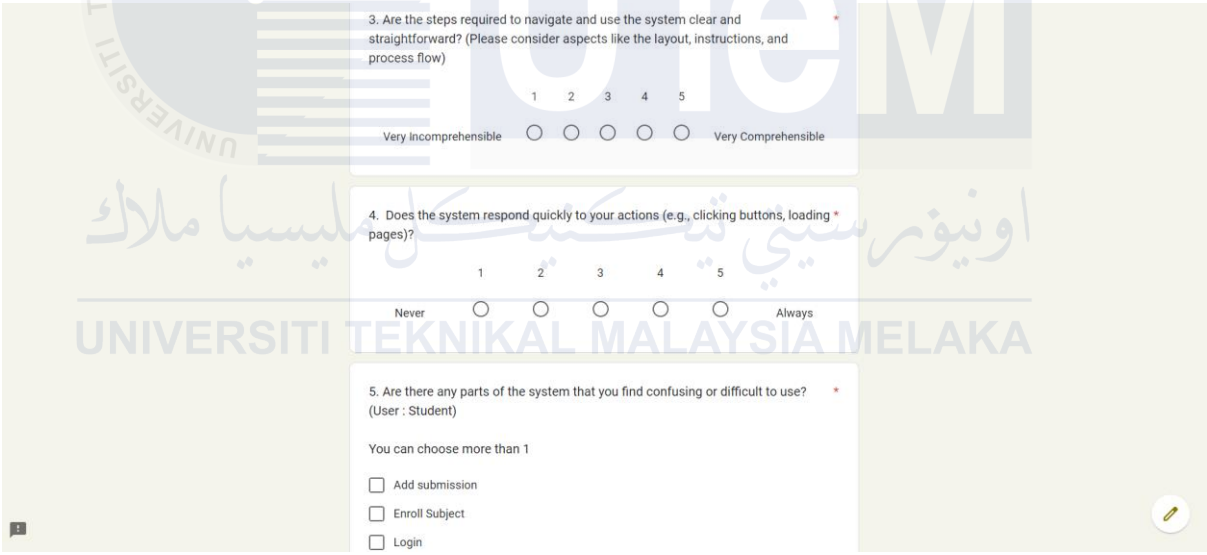

Figure 7 Survey form for testing purpose (f)

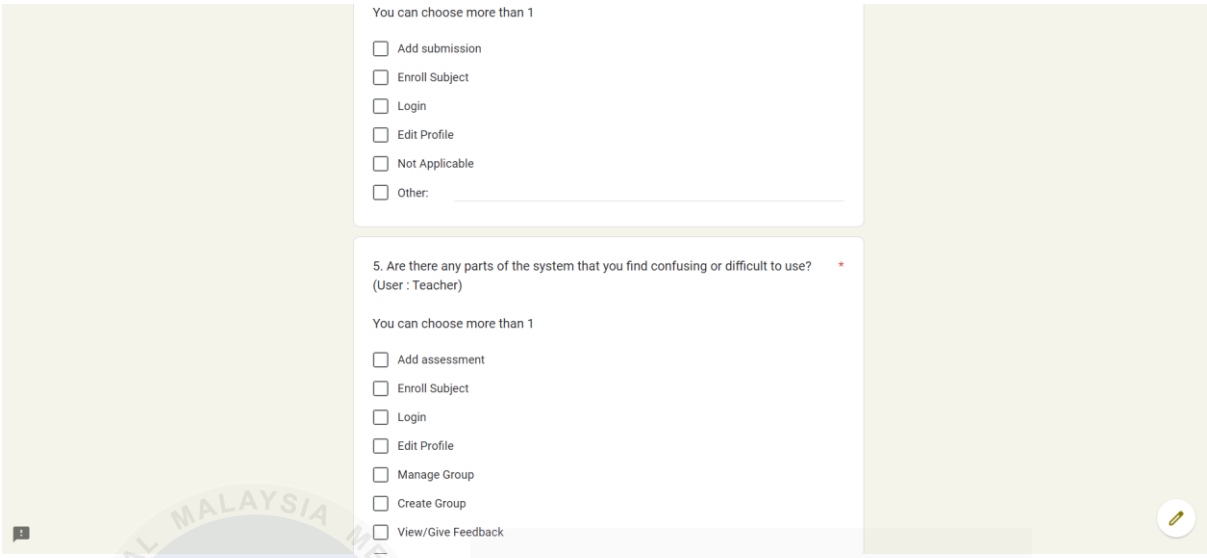

Figure 8 Survey form for testing purpose (g)

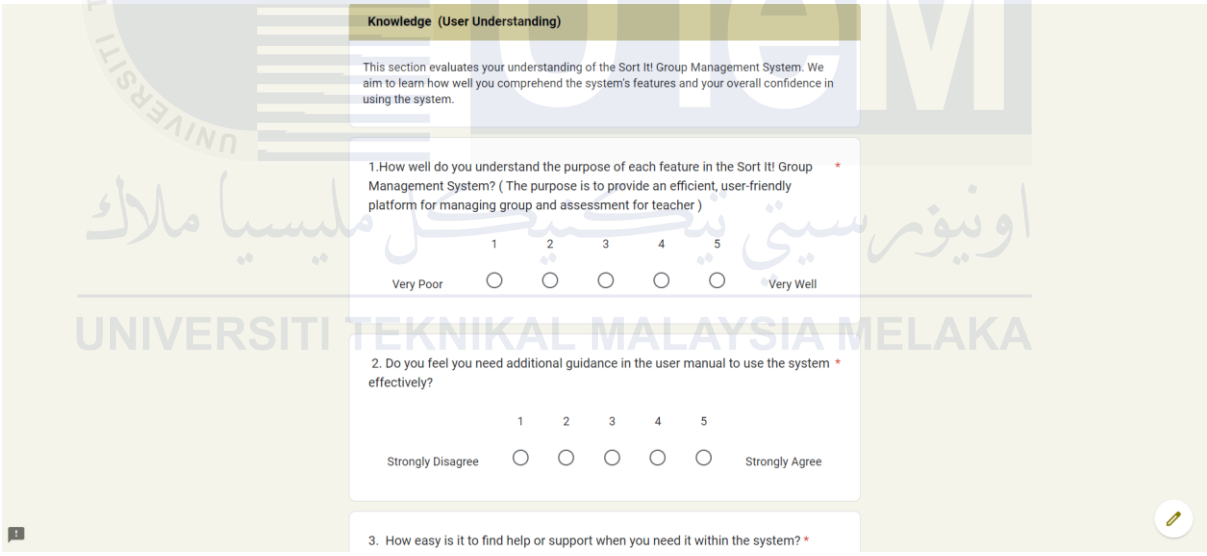

Figure 9 Survey form for testing purpose (h)

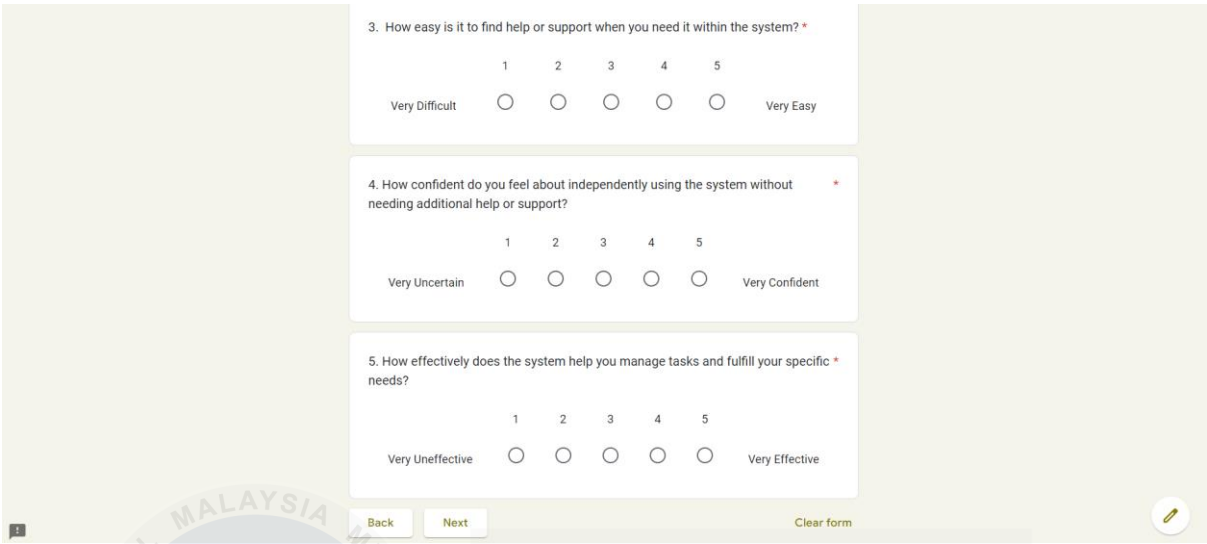

Figure 10 Survey form for testing purpose (i)

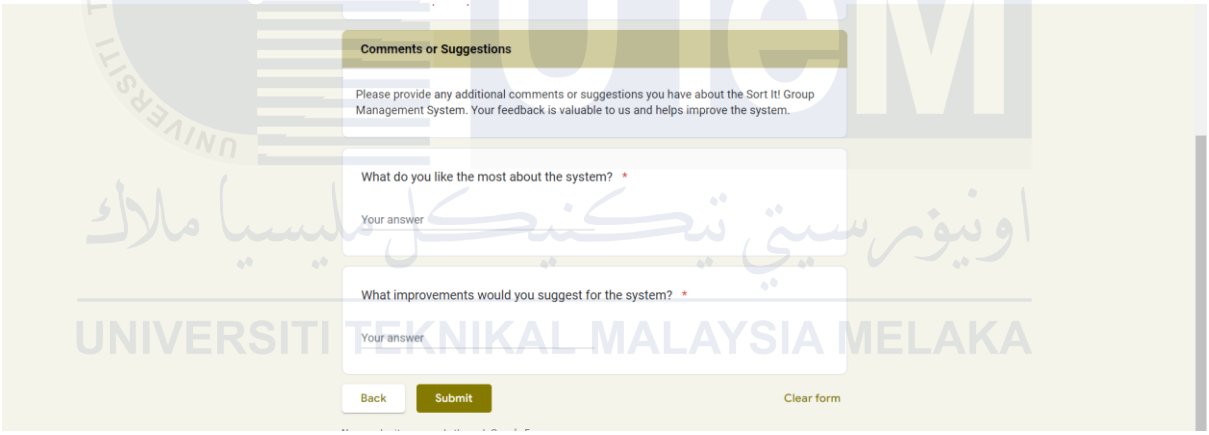

Figure 11 Survey form for testing purpose (j)

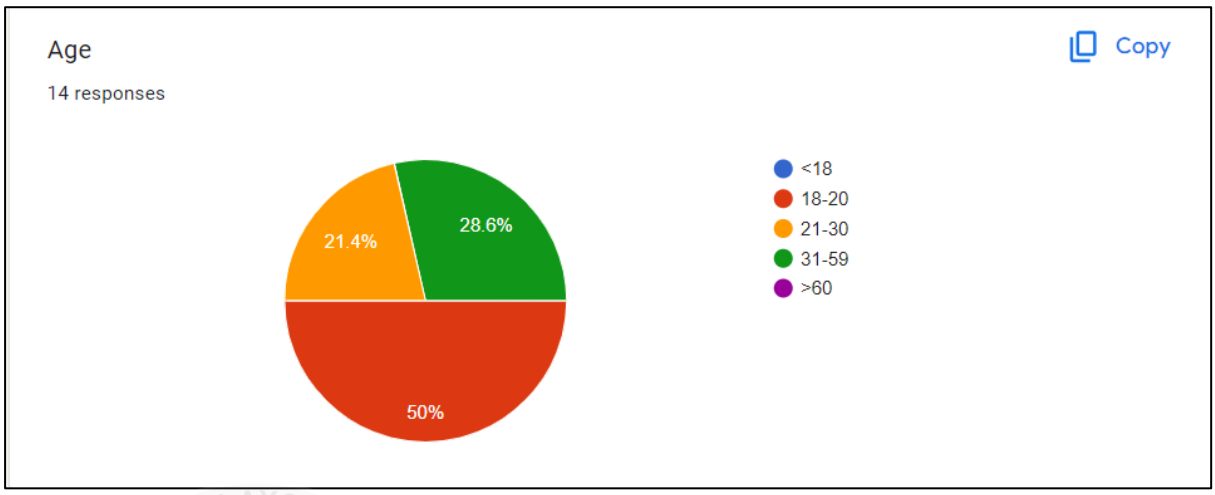

Figure 12 : Age Distribution of Survey Respondents: Majority (50%) are between 18-20 years old

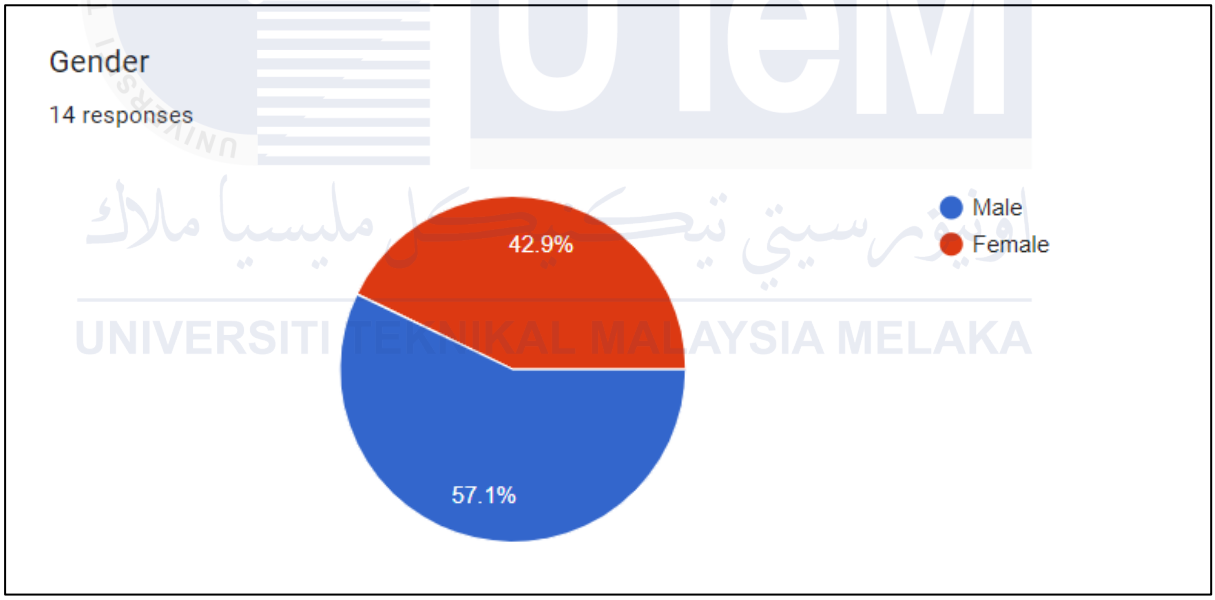

Figure 13 Gender Distribution of Survey Respondents: Majority (57.1%) are Male

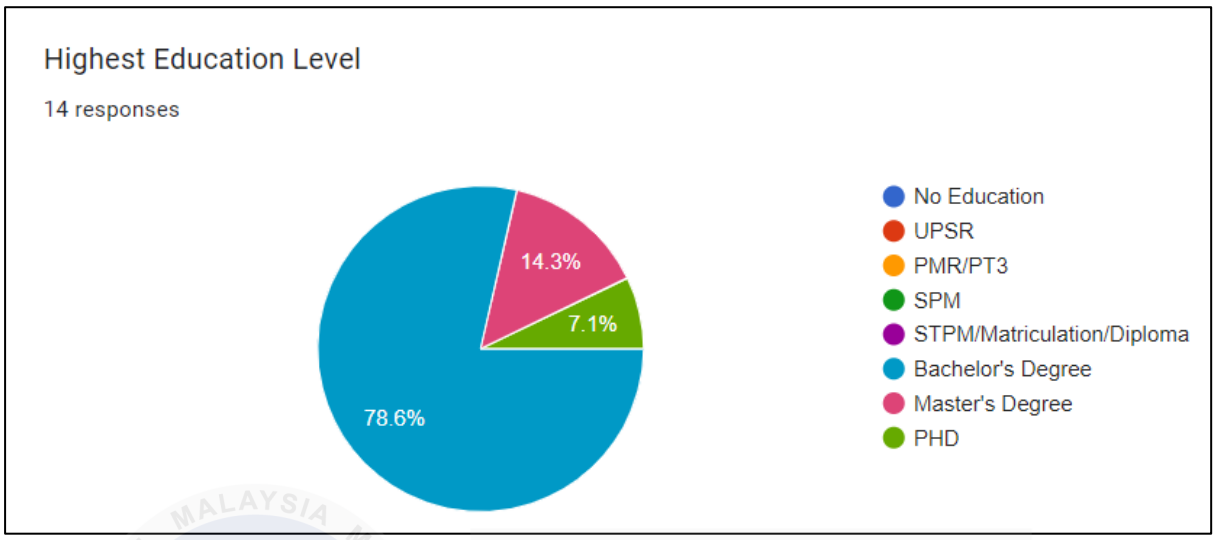

Figure 14 Education Level Distribution of Survey Respondents: Majority (78.6%) are Bachelor's Degree

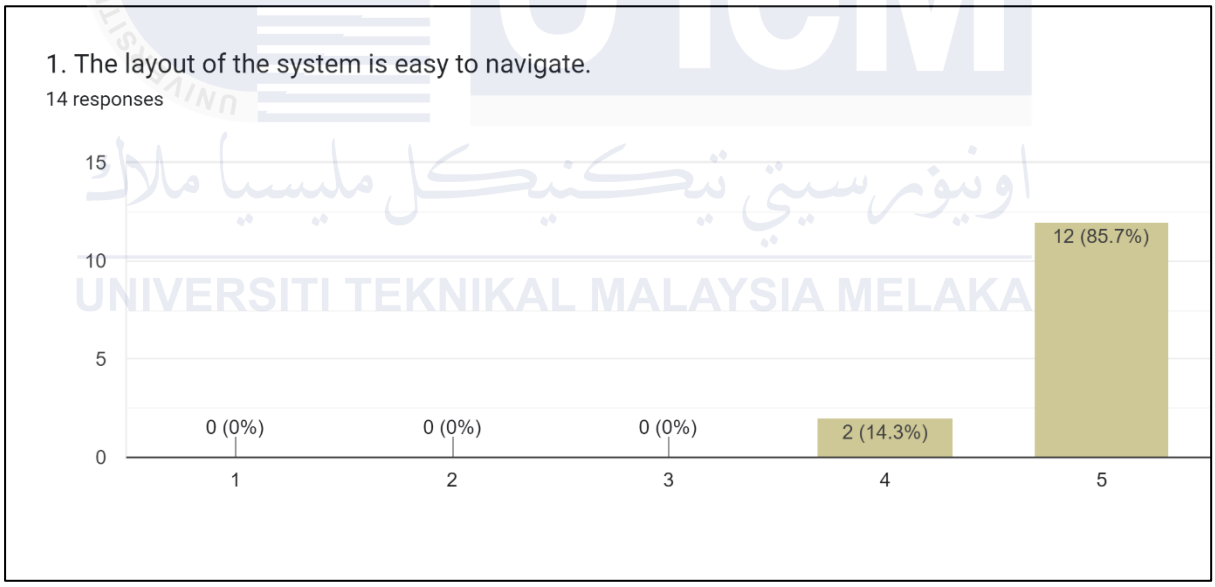

Figure 15 : Question 1 Interface Distribution of Survey Respondents: Majority

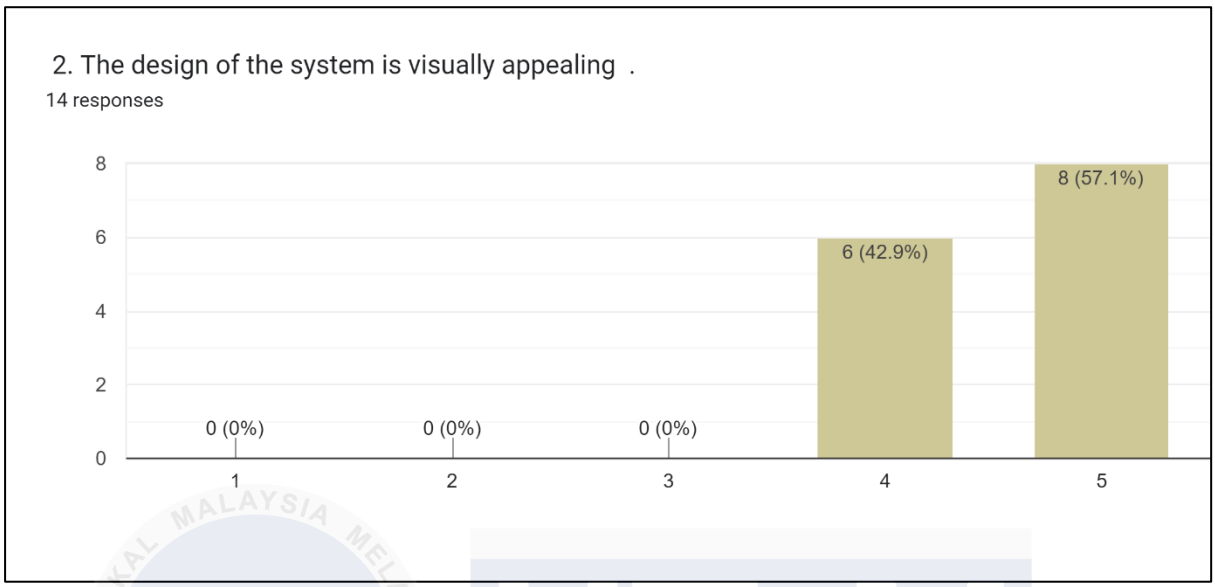

Figure 16 Question 2 Interface Distribution of Survey Respondents: Majority rate 5

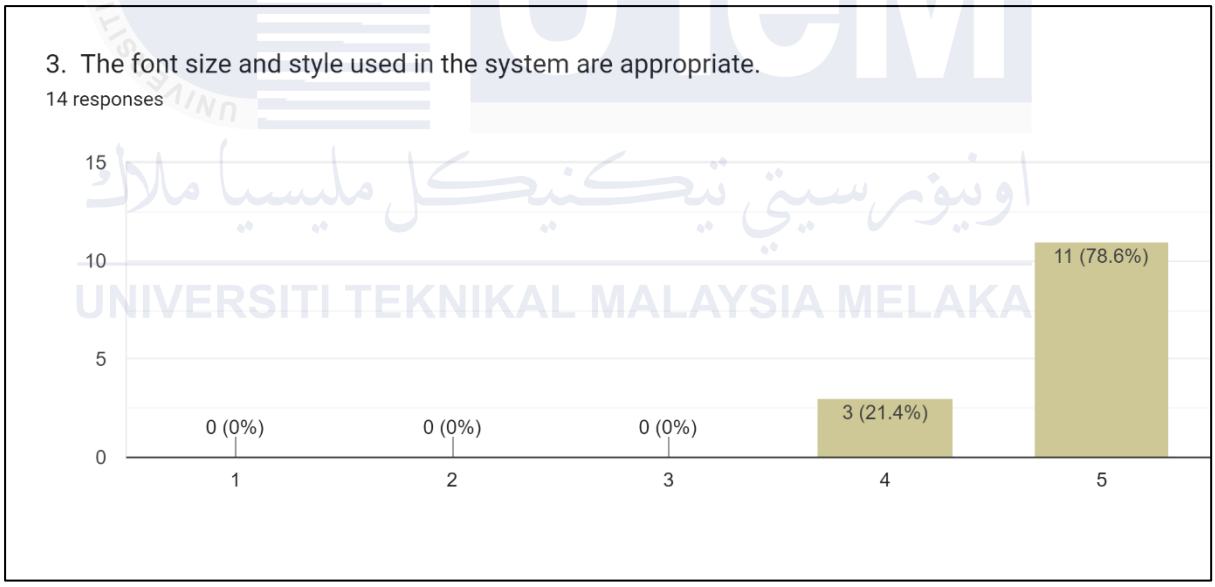

Figure 17 Question 3 Interface Distribution of Survey Respondents: Majority rate 5

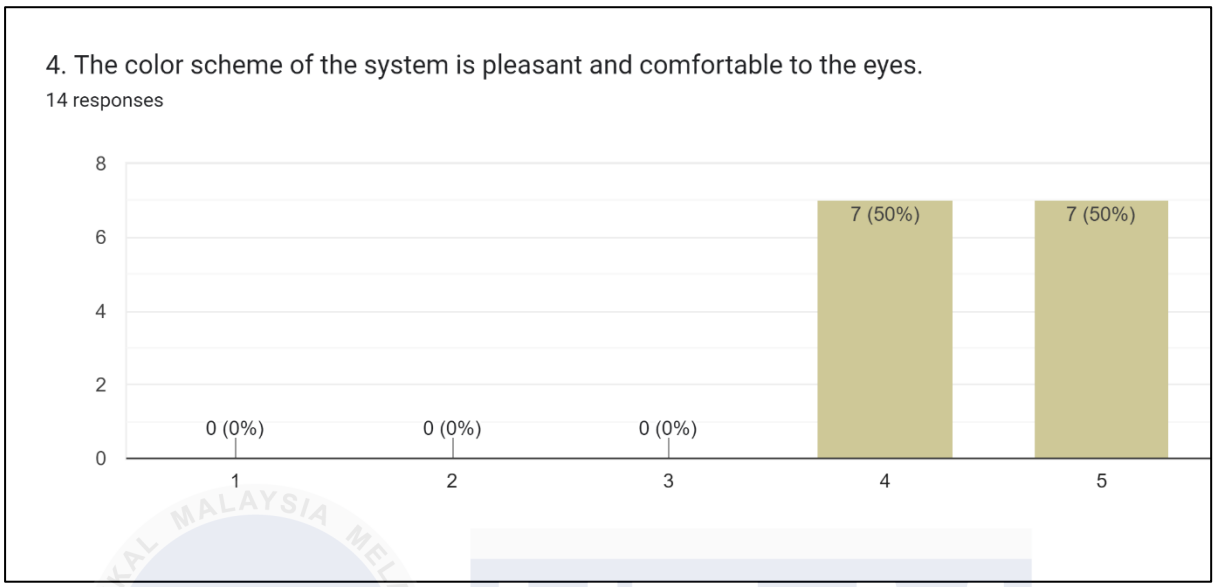

Figure 18 Question 4 Interface Distribution of Survey Respondents: Majority rate 4 and 5

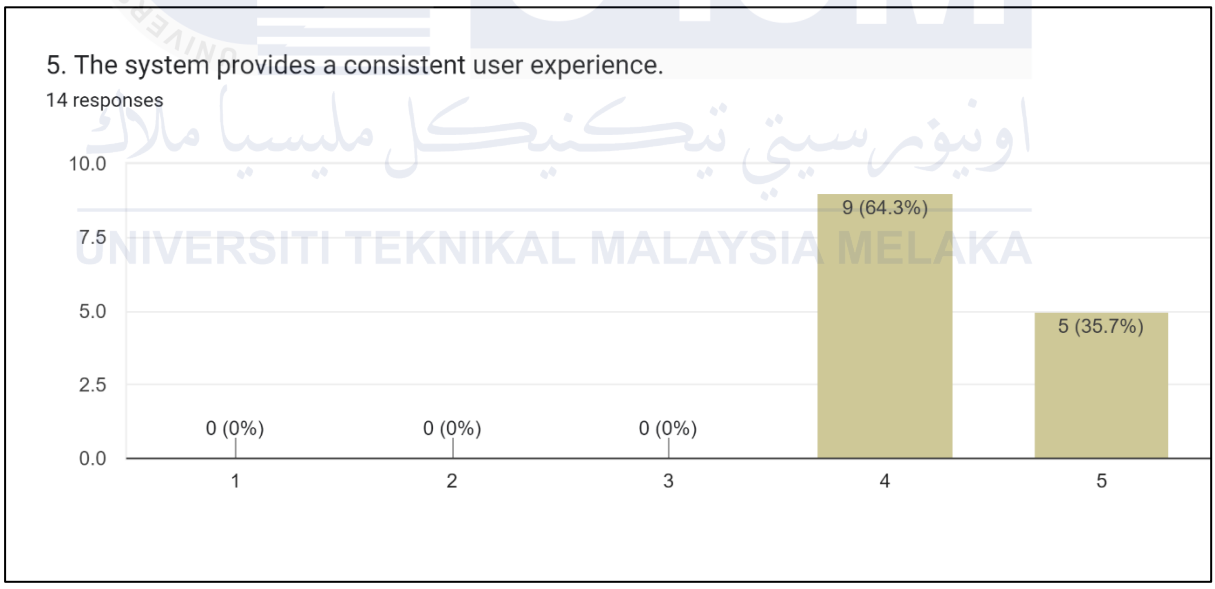

Figure 19 Question 5 Interface Distribution of Survey Respondents: Majority rate 5

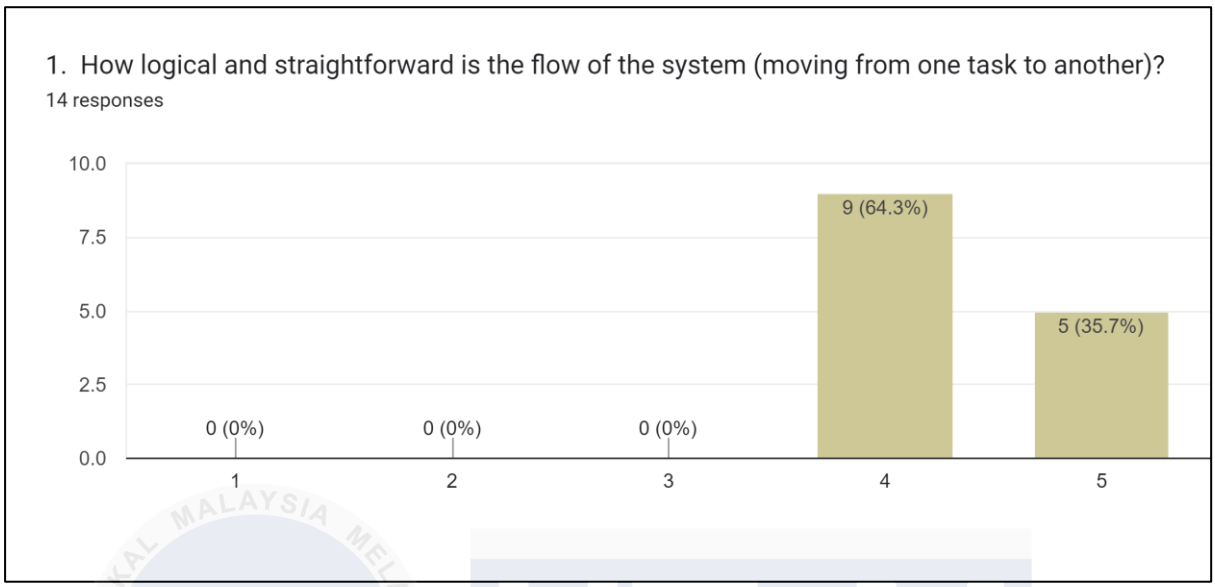

Figure 20 1 System Distribution of Survey Respondents

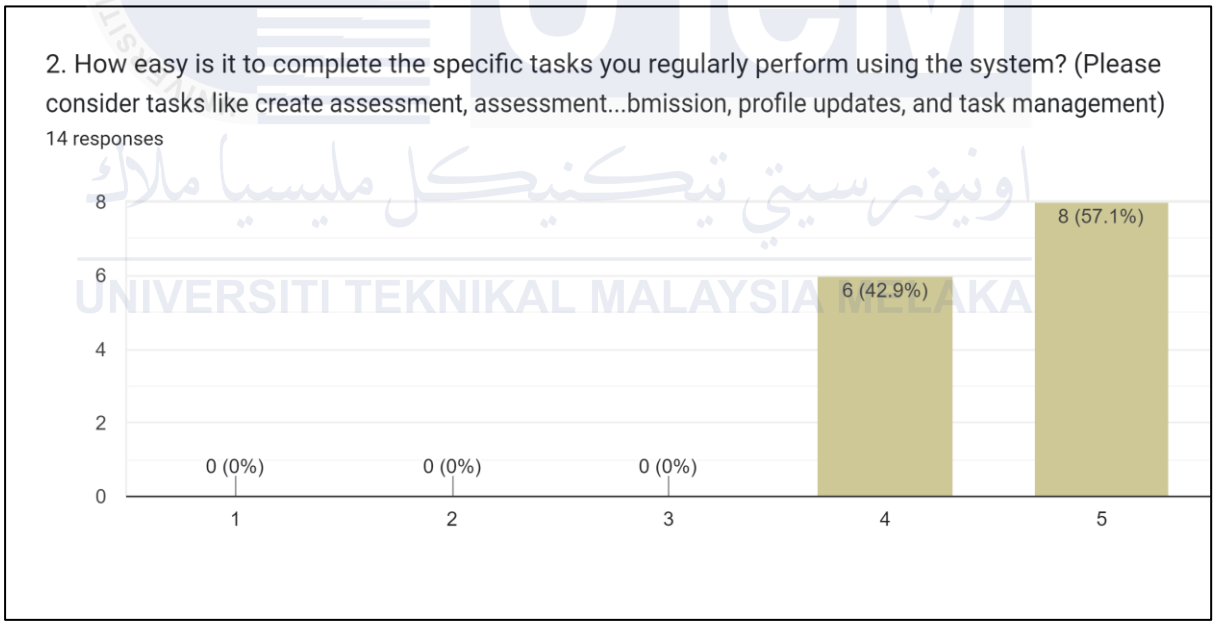

Figure 21 System Distribution of Survey Respondents

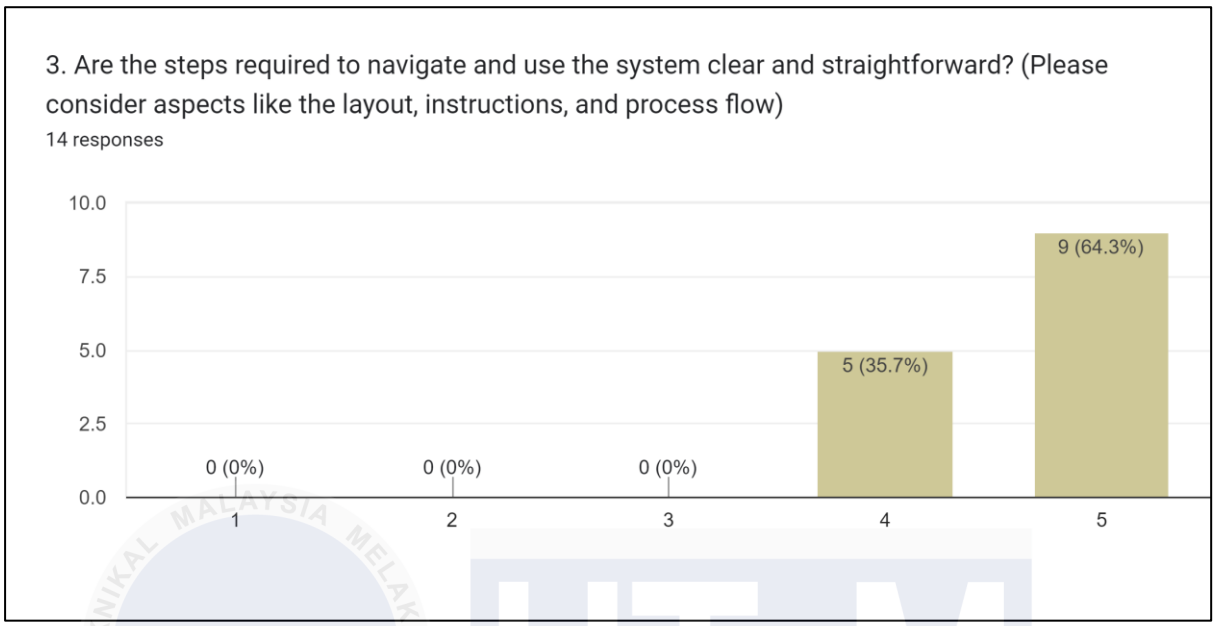

Figure 22 System Distribution of Survey Respondents

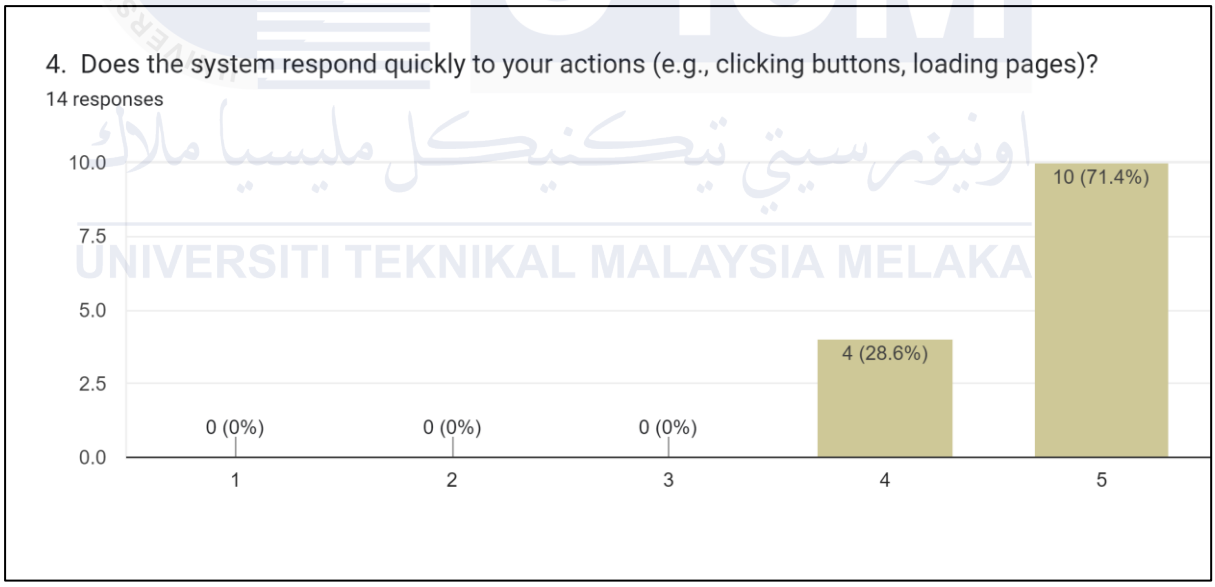

Figure 23 System Distribution of Survey Respondents

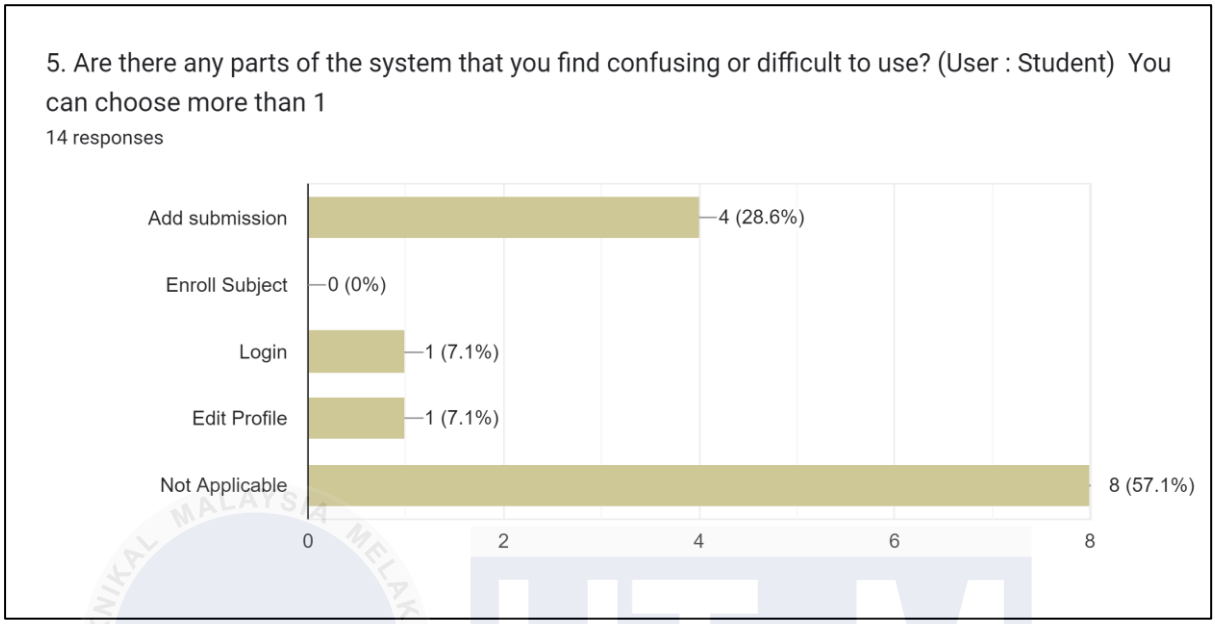

Figure 24 System Distribution of Survey Respondents

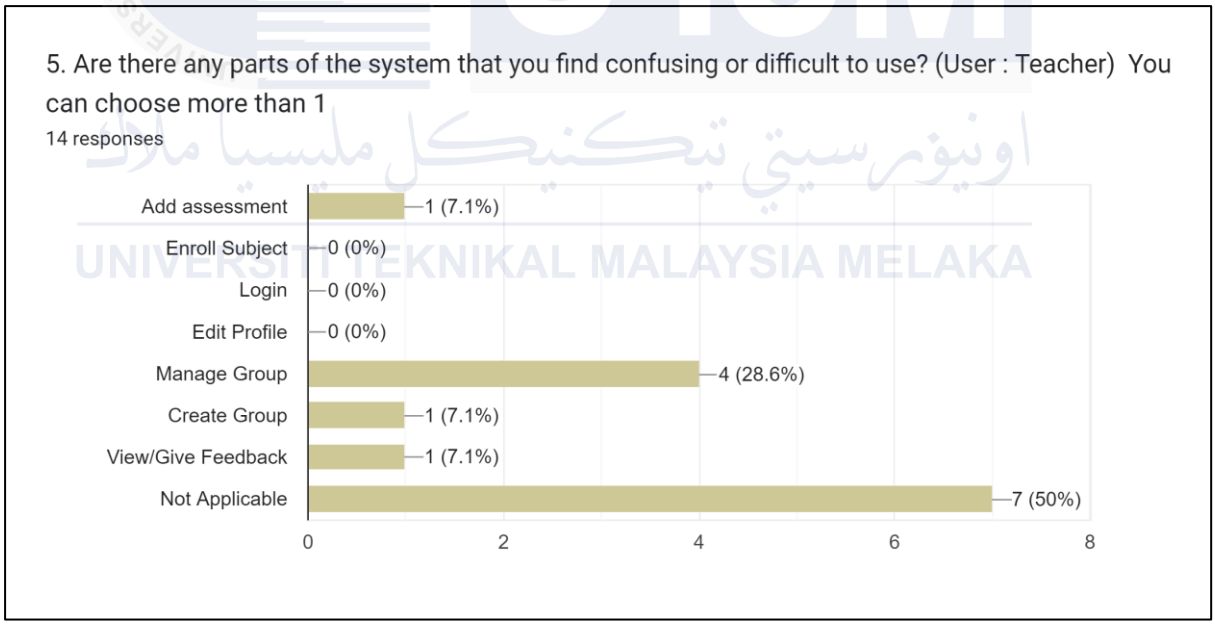

Figure 25 System Distribution of Survey Respondents