Arraid Journal of Science and Technology
Higher Institute of Science and Technology - AShmokh

مجلة الرائد للعلوم والتقنية
المعهد العالي للعلوم والتقنية - الشموخ

Streamlining University Examination Scheduling: Leveraging Kruskal's Algorithm and Z Specification for Efficiency

Salem Ekare[1], Ebtesam Taktek [2], Mostafa Brka[3], Osama Shtewi[4]

[1,2,3,4] Department of Information Technology
Higher Institute of Science & Technology – AlShomokh، Tripoli, Libya

* Corresponding: Ebtesam.tk@shomokh.edu.ly
Ebtesam.tk@gmail.com

ABSTRACT

Automating the examination scheduling process in universities is essential to streamline operations and reduce manual effort. This paper addresses the challenges of manual scheduling, such as time consumption, paper usage, and human effort. The goal is to efficiently schedule exams within a limited timeframe, ensuring no conflicts and adhering to constraints. Traditional methods are impractical for final examinations due to the growing number of students and courses. Various search algorithms can be employed for scheduling, including Kruskal's Algorithm, which this paper focuses on. The system's description and specification will use the Z specification language for formal analysis. Additionally, the Unified Modeling Language (UML) diagrams will visualize the system's architecture and design, using use case, class, sequence, and activity diagrams to represent functionality, structure, and behavior. This combination offers a robust framework for an effective and reliable examination scheduling system.

KEYWORDS: Examination Scheduling System, Kruskal's Algorithm, Z Specification.

1. INTRODUCTION

Examination scheduling is a significant challenge for teaching institutions, particularly universities, due to the involvement of a large number of students and courses. Each academic year requires the creation of a new examination schedule to accommodate the growth in students and changes in courses, leading to considerable effort. The traditional manual approach to creating examination schedules is being replaced by examination scheduling systems [1].

Manual scheduling poses several challenges [1], including: the difficulty in: Finding a feasible exam schedule with minimal duplicate exams, facilitating the creation of multiple exam sessions, ensuring that each student can take all required exams. The burden of examination scheduling in universities is alleviated by the existence of Examination Scheduling Systems (ESS), which reduce the required workforce through computer-supported scheduling mechanisms [2].

In this paper, the authors aimed to develop a comprehensive solution to the complex problem of exam scheduling in universities. The ever-increasing number of students and courses each year makes manual scheduling during midterms and finals arduous and prone to errors such as subject or student overlaps. Our system focuses on automating this process using Kruskal's algorithm for efficient searching. It accepts input data such as subject names, exam times, and durations from users, conducts room availability checks, and schedules exams accordingly. To ensure the accuracy of the search algorithm, the system relies on a predefined database with preset examination slots.

The remaining sections of this paper are organized as follows: first, a review of related work in this area is presented. A description of the UML specifications follows this. Next, the formal project specifications in Z notation are discussed. Finally, the paper concludes with recommendations for future research directions.

2. RELATED WORK

Several algorithms are available for solving scheduling problems, each with strengths and applications. The Graph coloring algorithms play a crucial role in efficiently solving the exam timetabling problem by representing each exam as a node in a graph. The Graph Coloring Algorithm represents exams as nodes with edges showing conflicts, ensuring no two adjacent nodes (exams) have the same color (time slot). Different colors represent different periods, allowing optimal scheduling without disputes [3].

Iterated Compound Multi-Agent (ICMA) method in workforce scheduling [4], specifically in nurse rostering problems. It highlights that ICMA stands out from other meta-heuristics by efficiently utilizing the hierarchical structure in the scheduling problems. The study emphasizes the potential benefit of incorporating hierarchical structure into other meta-heuristics to enhance performance. It suggests preserving the structure when converting real instances into benchmark problems for accurate evaluation. Furthermore, the paper suggests testing ICMA on domains with structure-preserving benchmarks, such as the PSTP and the ITC 2007 course timetabling track. It also mentions using integer programming methods to exploit grouping structures in problem-solving, indicating the overall advantage of structure exploitation in optimization. Moreover, the paper proposes exploring specific mechanisms for subset selection and termination criteria within the ICMA strategy. It discusses the gradual increase in the active set of events in ICMA, highlighting the importance of intermediate sizes of active sets in optimizing search efforts. The study also delves into the interleaving effect within ICMA as a means of diversification, aiming to maintain an informative yet flexible search space for efficient problem-solving [4].

The research paper in [5] investigates the efficient resolution of Examination Timetabling Problems by adapting heuristic orderings. The study compares different methods while considering both hard and soft constraints. The adaptive approach stands out, producing the best overall outcomes in six out of eleven test cases and the worst in only one case. The results of the adaptive method consistently fall within a narrow

range, with even its worst performance often surpassing the best results of other approaches. Notably, employing randomized adaptation with extended run times led to the best results in ten problems, emphasizing its robustness. The detailed findings and solutions from the study can be accessed online. Furthermore, the paper compares its results with [5, 6] offering an additional perspective. The problems analyzed in this study reflect real-life situations, incorporating constraints such as limits on seating capacity per period. The penalty function utilized follows a specific expression. When compared with the setup yielding the best results from [5]approach, the adaptive method with specific configurations still excelled. However, the decomposition approach from [5] work generally outperformed the others for the examined problems.

Kruskal's algorithm is a standout choice among search algorithms for efficiently creating examination timetables. Its particular strength lies in handling sparse graphs, where it excels at finding minimum spanning trees. The algorithm systematically constructs connected components of vertices, which eventually form the minimum spanning tree. Initially, each vertex is considered an independent component within the prospective tree, and the algorithm merges these components in a step-by-step manner to achieve the optimal solution [7]. As the algorithm progresses, it prioritizes the lightest remaining edges, ensuring that no cycles form within the developing tree. Specifically, it evaluates whether the endpoints of an edge belong to the same connected component. If they do, the edge is disregarded to prevent cycle formation. Conversely, if the endpoints belong to different components, the algorithm incorporates the edge, merging the respective components into a unified whole [7].

Kruskal's Algorithm offers a straightforward and efficient approach to finding the minimum spanning tree (MST) of a graph. By prioritizing lightweight edges and avoiding cycles, it provides a clear and optimal solution for connecting all vertices with minimal total weight. The algorithm's simplicity and effectiveness make it a popular choice in various applications requiring the generation of minimum spanning trees [8].

The research paper [9]discusses the implementation of Kruskal's Algorithm for finding the minimum spanning tree in a graph. Kruskal's Algorithm excels at identifying the least-weighted edges to create efficient paths. Unlike Dijkstra's Algorithm, Kruskal's Algorithm's strength lies in its ability to easily find the shortest path by sorting edge weights. However, a drawback is its slower speed when dealing with a large number of nodes, due to the necessity of sorting numerous edges before path formation [9]. The application of Kruskal's Algorithm in graph theory offers a practical approach to optimizing pathfinding and tree creation. It emphasizes efficiency and accuracy in determining the minimum spanning tree, making it a valuable tool for solving complex problems in this domain[9].

The research paper by [10] compares the performance of the Prim and Kruskal algorithms in constructing a minimum spanning tree within a super metric space. The study utilizes complexity analysis and experimental methods to assess the two algorithms. Analysis of daily sample data from the Shanghai and Shenzhen 300 indexes

between the second half of 2005 and 2007 showed that the Kruskal algorithm is more space-efficient when the number of shares is under 100, while the Prim algorithm outperforms in terms of time complexity when the number of shares exceeds 100. A minimum weight spanning tree is described as a connected graph with non-negative edge weights, with the challenge being to find a maximum weight spanning tree [10]. The study proposes greedy algorithms based on Prim and Kruskal for finding a minimum weight spanning tree, highlighting the importance of these algorithms in graph theory and minimal spanning trees[10].

The paper [11] discusses the implementation of the Kruskal algorithm to find the shortest path to a building store in Bogor city. The study focuses on improving navigation efficiency in urban settings. By utilizing the Kruskal algorithm, the research aims to optimize route planning to enhance accessibility to the designated store location. The implementation of this algorithm is crucial for streamlining navigation processes and ensuring effective paths are chosen within the city environment [11]

The research bulletin discusses the use of formal methods [12], specifically the Z language, to enhance the security of e-commerce systems. It addresses the importance of security threats in e-commerce systems and the need to gain users' trust by improving security measures. The paper [12] focuses on developing specifications for a secure e-commerce system using the Z language and highlights the significance of monitoring customer behavior to prevent security threats.

The research [2] discusses the formal specification for an inventory system using Z language to remove ambiguity in requirements specification and improve software reliability. The authors present the Unified Modeling Language (UML) specification through use case and class diagrams, and then translate it into Z schema. The consistency between UML and Z schema is shown, emphasizing the benefits of using Z language in improving system reliability and reducing defects in system development.

This research introduces the Z specification language, one of the most widely used formal methods developed by the Programming Research Group at Oxford University. Z employs mathematical notation and classical two-valued logic to ensure precision and identify inconsistencies in specifications. The use of theorem provers further verifies that the software implementation meets its specifications. [13]

3. UML SPECIFICATIONS

The Examination Scheduling System (ESS) encompasses five key functions: Login, Enter Details, Search Room, Schedule Exam, and View Timetable. Table 1 provides a detailed description of each function within the system.

Table 1: ESS Use Case with Description

No	Use Case Name	Description
1	Login	The user must enter their details, including name and password.
2	Enter details	This function allows the administrator to input exam details, including the subject name, exam time, and duration. (name _subject, time, duration_ exam)
3	Searching room	The administrator should enter the room ID to verify its availability (room_id).
4	Schedules exam	After verifying room availability, the user must schedule the exam by entering details such as the room ID, subject name, exam time, and duration. (room_id, name _subject, time, duration_exam)
5	View timetable	This function displays the examination schedule, including the subject name, exam time, and duration. (name_subject, time, duration_ exam).

3.1 USE CASE DIAGRAM OF EXAMINATION SCHEDULING SYSTEM

Figure 1 illustrates the use-case diagram for the Examination Scheduling System (ESS). The system involves two actors: the administrator and the user. There are five primary use cases in ESS: LoginProgram, EnterDetails, SearchRoom, ScheduleExam, and ViewTimetable. All these use cases interact with the system's database. Users can only log in to the system and view the schedules previously created by the administrator.

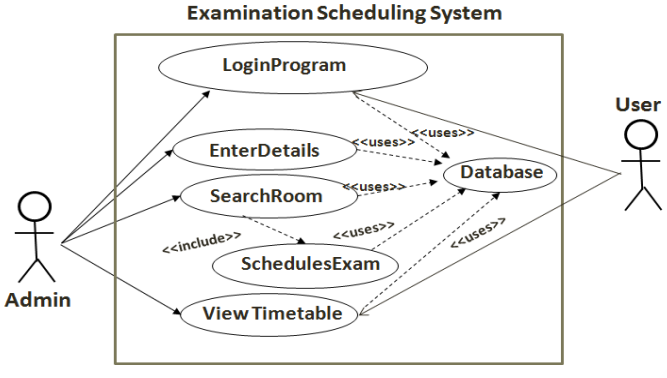

Fig. 1: Use-case Diagram for Examination Scheduling System

3.1.1 Class Diagram

Based on the use-case diagram for the ESS, three classes have been defined: DataScheduling, Activity, and Kruskal. The DataScheduling class includes the attributes: Name_subject, Time, and Duration_exam. Each of these attributes has

corresponding Set and Get methods. Figure 2 illustrates the DataScheduling class that has been created.

DataScheduling
- Name _subject : string - Time : int - Duration _exam : int
+ Set_name_subject (string : name) + Get_name_subject (string : name) + Set_time (int : time) + Get_time (int : time) + Set_duration_exam(int : dur_exam) + Get_duration_exam(int : dur_exam)

Fig. 2: DataScheduling Class

The Activity class inherits attributes and functions from the DataScheduling class, making it dependent on DataScheduling. The functions of the Activity class include Login, Add_Details, Search_Room, Schedule_Exam, ViewTimeTable, and Main_Menu. Since Activity inherits all attributes from DataScheduling, it has no additional attributes of its own. Figure 3 depicts the Activity class inheriting from the DataScheduling class.

The Kruskal class serves as the backbone for the searching technique essential in managing the examination schedule. Figure 4 illustrates the Kruskal class. It encompasses six attributes: Parent, Noofedges, Visited, Min, Mincost, and Cost. The class's methods are categorized into two: Read and Kruskals, each contributing to the effective execution of the searching algorithm.

The comprehensive class diagram for the Examination Scheduling System is depicted in Figure 5. In this diagram, the DataScheduling class acts as the superclass, with the Activity class serving as its subclass. Additionally, the Kruskal class is included as part of the system, providing crucial functionality for the searching technique.

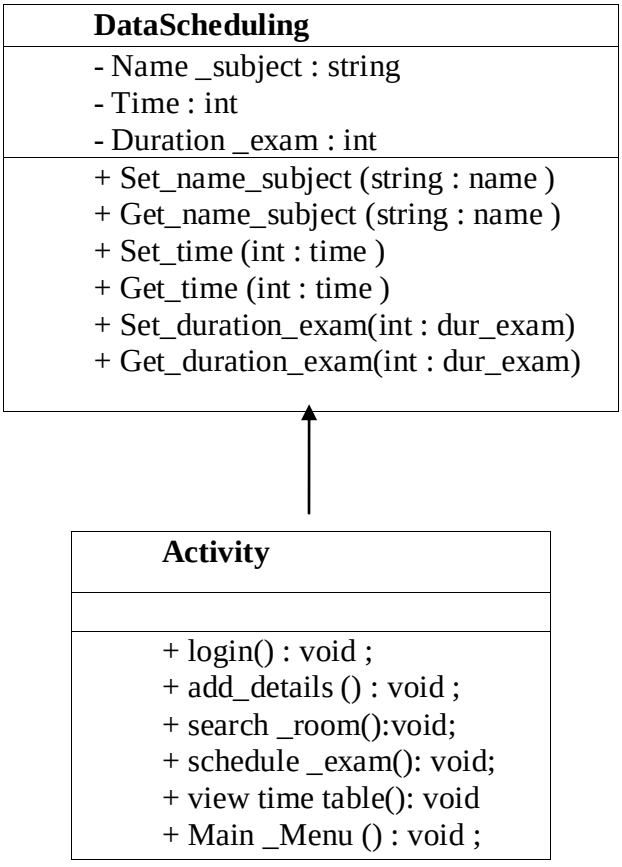

Fig. 3: Activity Class using Inheritance Concept

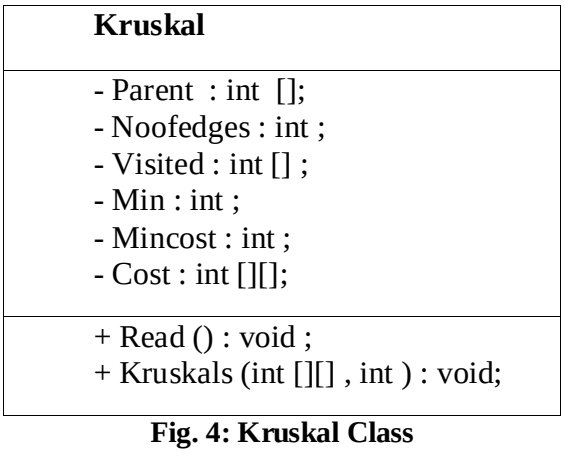

Fig. 4: Kruskal Class

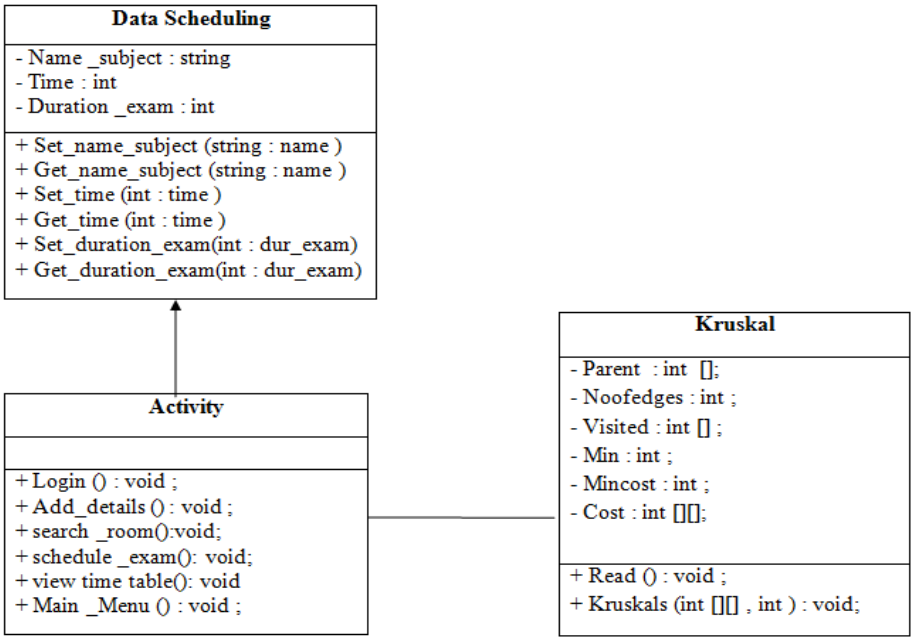

Fig. 5: Class Diagram of ESS

3.1.2 Sequence Diagram

The sequence diagram of the Examination Scheduling System (ESS) is divided into two parts: the sequence diagram for the User (Figure 6) and the sequence diagram for the Administrator (Figure 7) [14].

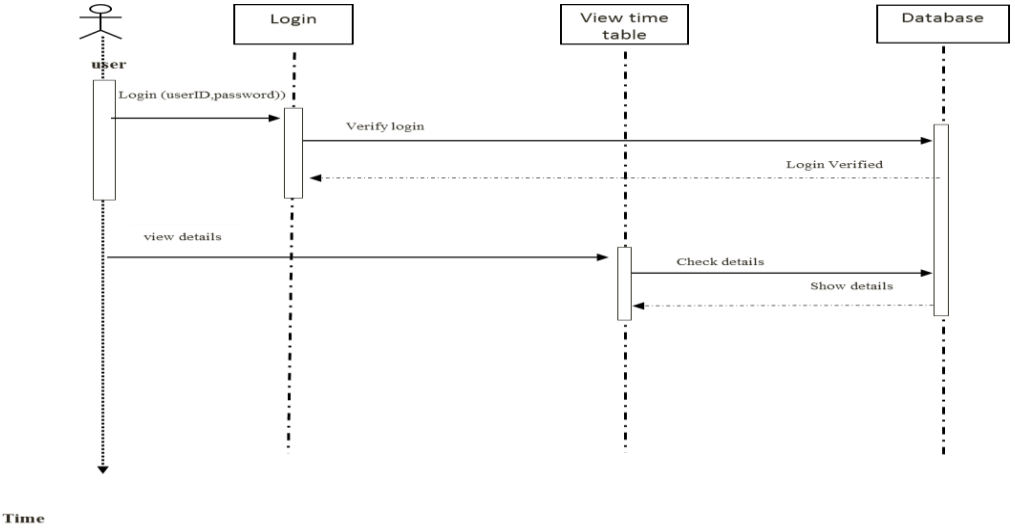

Fig.6: Sequence Diagram for User

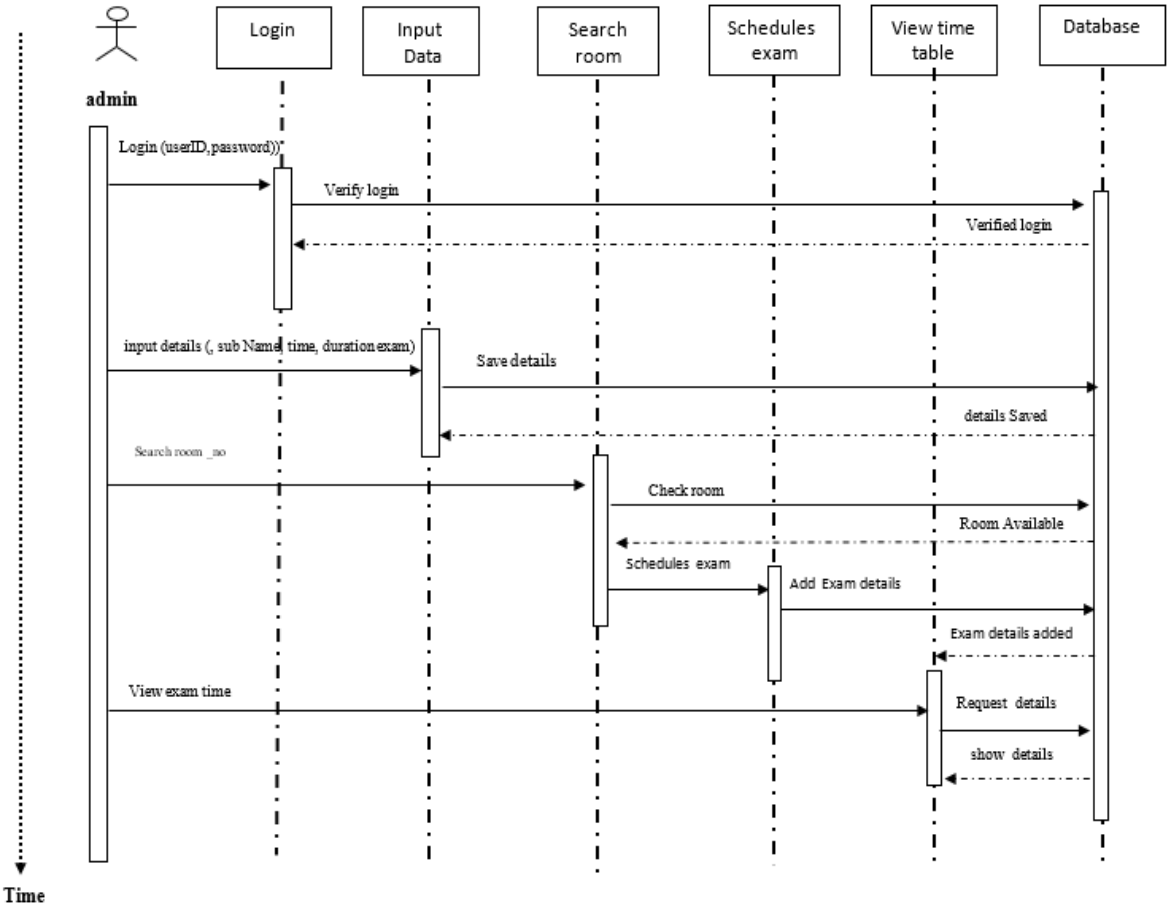

Fig. 7: Sequence Diagram for Administrator

3.1.3 Activity Diagram

Activity diagrams are graphical representations of workflows, detailing step-by-step activities and actions with support for choices, iterations, and concurrency. In the Unified Modeling Language (UML), activity diagrams are designed to model both computational and organizational processes [14]. Figure 8 illustrates the activity diagram for users in this system. Every user must log in before performing any functions. The system verifies the user ID and password; if the password is incorrect, an error message is displayed, prompting the user to re-login or exit the system. The only available menu option for users is to view the timetable.

Figure 9 illustrates the activity diagram for the system administrator. Like the user, the administrator must log in using a user ID and password. Upon successful login, the administrator can choose from three available functions: Input Data, Search Room, and View Timetable.

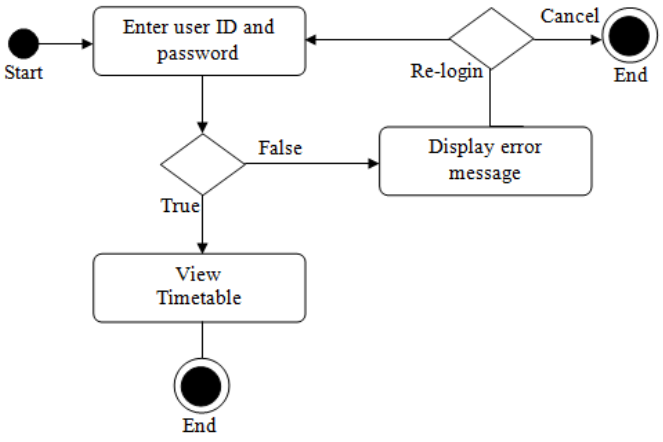

Fig. 8: Activity Diagram for User

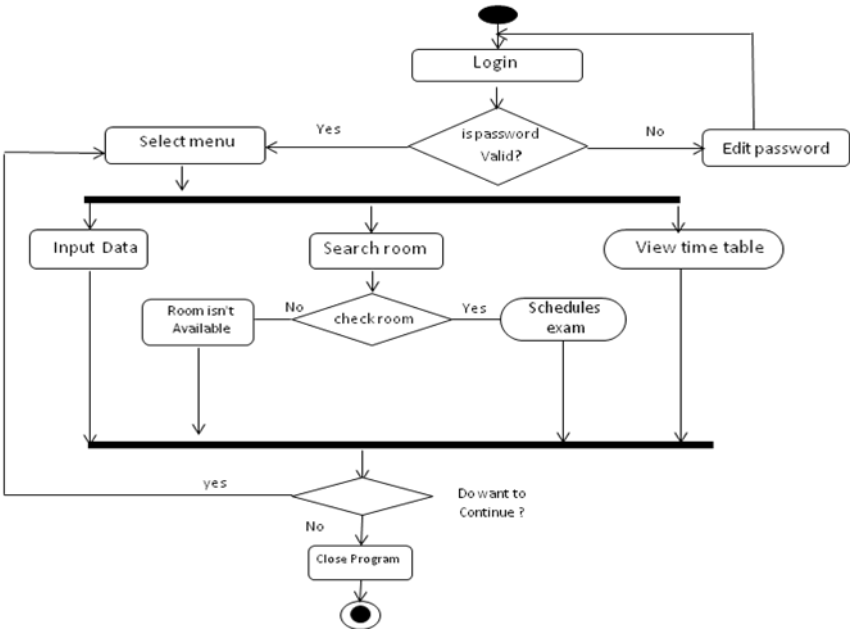

Fig. 9: Activity Diagram for Administrator

4. From Unified Modeling Language (UML) diagrams to Z Specification

Formal methods utilize mathematical notations to express requirements specifications precisely, eliminating the ambiguity inherent in natural language specifications and enhancing software reliability [2]. One such formal specification language is Z-notation. Z-notation is a mathematical notation, a specification language, and a model-based notation. It cannot be compiled into a running program but can be broken down into smaller components called schemas [15].

To transition from UML to Z-notation for the Examination Scheduling System (ESS), we first identify the database structure. This approach is inspired by the methodology presented in [2]. ESS comprises three table objects: DataScheduling, Activity, and Kruskal. Each operation executed on these table objects requires a system response,

represented by the variable response. There are two possible responses: Success or Failed. A Success response indicates that the user's operation has been successfully executed and recorded in the database. In contrast, a Failed response signifies that the system did not overwrite the data in the database. The following subsections will provide a detailed explanation of the Z schemas

4.1 Database Representation in Z-Notation

To represent the attributes of the classes in the Examination Scheduling System (ESS) using Z-notation, we define the database schema as follows:

Database

StockSys: P DataScheduling // represents attribute for the DataScheduling class
StockSys: P Activity // represents attribute for Activity class
StockSys: P Kruskal // represents attribute for Kruskal class

Response by system:
Response ::= Success | Failed

4.2 Z-Scheme for Class Activity

Figures 11 to 14 illustrate the schemas representing all methods for the Activity class. The Activity class includes five methods: Login, AddDetails, SearchRoom, ScheduleExam, and ViewTimetable.

4.2.1 Login Function

Figure 11 shows the Z schema for the Login method. The ESS provides login features for each user, requiring them to enter a username and password to gain authorized access. There are two possible responses: login success and login failure. If the username and password match, the system responds with login success, allowing the user to access the system. Conversely, if the username and password do not match, the system responds with login failed, preventing the user from entering the system. Username and Password are input fields for the user credentials. The response is the system's output, indicating whether the login attempt was successful or failed.

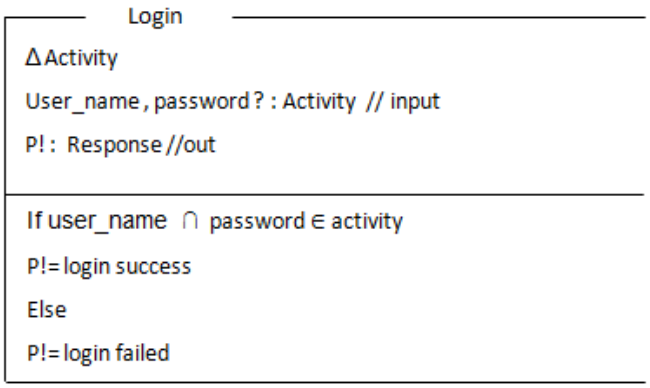

Fig. 11: Z Schema for Login Method in Class Activity

4.2.2 Add Details Function

In the AddDetails function, the administrator must enter the subject name, exam time, and exam duration. If all the details are entered correctly, the system will notify with a successful response; otherwise, it will respond with a failed message. The Z schema for the AddDetails method is shown in Figure 12. subjectName, examTime, and examDuration are the input fields for the exam details. The response is the system's output, indicating whether the details were added successfully or if there was an error.

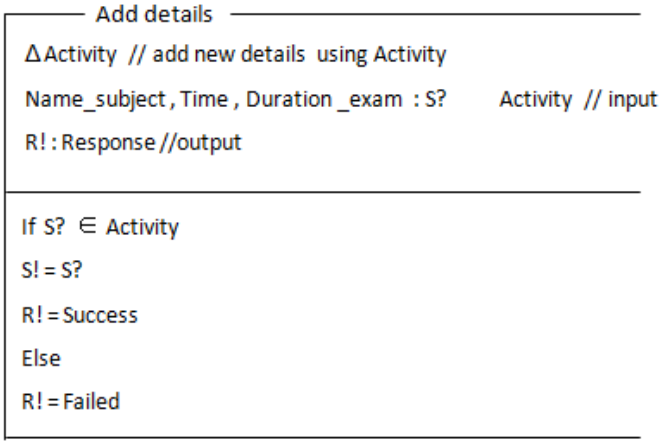

Fig. 12: Z Schema for Add Details Method in Class Activity

4.2.3 Search Room Function

Figure 13 shows the Z schema for the SearchRoom function. If the room has not been booked for any examination, the system will respond that the room is available. Otherwise, the system will respond that the room is not available. roomID is the input field for the room identification. Response indicates whether the room is available or not.

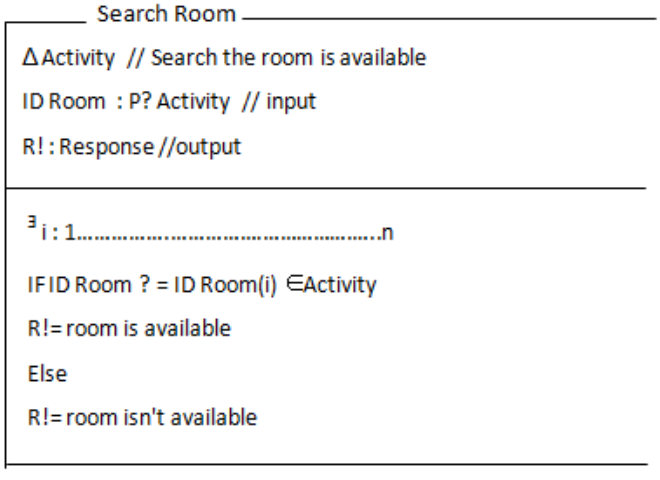

Fig. 13: Z Schema for Search Room Method in Class Activity.

4.2.4 Schedule Exam Method

In the ScheduleExam method, the administrator is required to input the subject name, exam time, and duration. The system then proceeds to schedule the exam in an available room. Figure 14 depicts the Z schema for the Schedule Exam method within the Activity class. subjectName, examTime, and examDuration represent the input fields for the exam details. The response indicates whether the scheduling of the exam was successful or failed.

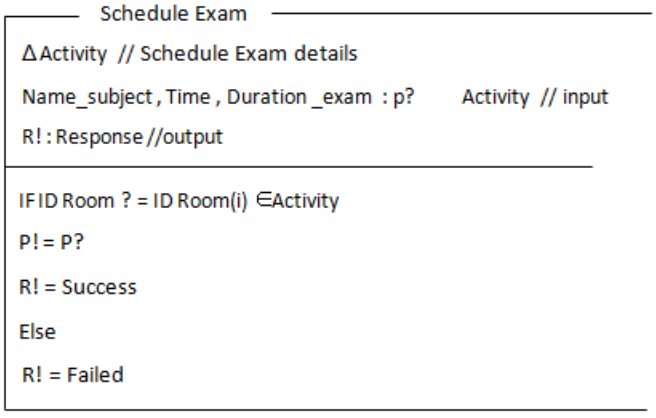

Fig. 14: Z Schema for Schedule Exam Method in Class Activity

4.2.5 View Timetable Method

Figure 15 illustrates the Z schema for the ViewTimetable method within the Activity class. This function facilitates the viewing of the overall timetable that has been created.

The response represents the system's output, indicating whether the operation to view the timetable was successful or failed.

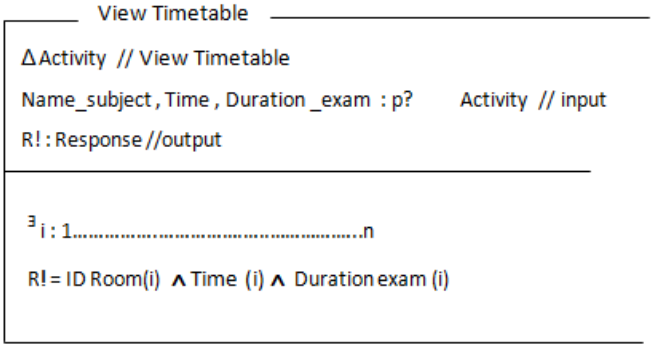

Fig. 15: Z Schema for View Timetable Method in Class Activity

4.3 Z-Schema for Class Kruskal's

Figure 16 presents the Z schema for Kruskal's algorithm. This algorithm operates by initially treating each vertex as its component. It then iteratively merges two components into one by selecting the lightest edge that connects them. Subsequently, it scans the set of edges in monotonically increasing order by weight. Finally, it utilizes a disjoint data structure to determine whether an edge connects vertices in different components. The graph represents the input graph on which Kruskal's algorithm is applied. The result is the resulting set of edges forming the minimum spanning tree.

Kruskal's algorithm

ΔKruskal's // search algorithm
A,G , V , E : N? Kruskal's // Input: A weighted connected graph G= (V, E)
E : Response //output Set of edges comprising a MST

sort the edges E by their weights
($E_T \leftarrow \emptyset$ while $|E_T| + 1 < |V|$ do) ∈ Kruskal's
e ← next edge in E
if $E_t \cup \{e\}$ does not have a cycle then
$E_T \leftarrow E_T \cup \{e\}$
Return ET

Fig. 16: Z Schema for Kruskal's Algorithm in Class Kruskal

5. Case Study: Validation of the Examination Scheduling System (ESS) Framework

Alshomokh Technical Higher Institute aims to streamline its examination scheduling process, which currently involves manual scheduling and is prone to errors and inefficiencies. To address these issues, we designed the Examination Scheduling System (ESS) framework. The following case study demonstrates the framework's validity through a simulation.

5.1 Objectives

1. Validate the ESS framework's functionality in a controlled, simulated environment.
2. Demonstrate the efficiency and accuracy of the system in scheduling exams and allocating rooms.
3. Collect and analyze feedback from users to evaluate satisfaction and ease of use.

5.2 Simulation Steps

Step 1: Administrator Login
Simulated administrators log in using the LoginProgram function.
Step 2: Entering Exam Details

Administrators use the EnterDetails function to input exam details, including subject names, exam times, and durations.

Step 3: Searching for Rooms

Administrators use the SearchRoom function to check room availability and ensure there are no conflicts.

Step 4: Scheduling Exams

Administrators schedule the exams using the ScheduleExam function.

Step 5: Viewing Timetables

Simulated students log in and use the ViewTimetable function to view their exam schedules.

5.3 Evaluation of the Framework

The simulation of the Examination Scheduling System (ESS) at Alshomokh Technical Higher Institute successfully demonstrated the functionality, efficiency, and accuracy of the proposed framework. The SearchRoom function effectively prevented room conflicts, demonstrating efficient room allocation. In addition, No double bookings or timing conflicts were detected, demonstrating the system's accuracy. The feedback from simulated users indicated high levels of satisfaction with the system's performance.

The simulation demonstrates the ESS framework's validity by highlighting its key features and functionalities: firstly, the use case and class diagrams were accurately followed, ensuring comprehensive coverage of system functions through well-defined interactions and processes. Secondly, the sequence diagrams detailed the operational flow, facilitating the smooth execution of the simulated scheduling and viewing processes. Thirdly, the workflows for both administrators and students, as depicted in the activity diagrams, were clear and easy to follow. Moreover, Z Notation provided formal specifications that ensured precise definitions of the system’s operations, reducing ambiguity and enhancing reliability.

This case study validates the framework without the need for full-scale implementation on the Institute's server, showing its potential for effective application in real-world scenarios.

6. CONCLUSION

In conclusion, this paper delves into the critical task of automating examination scheduling in university settings, aiming to enhance efficiency and reduce the burden of manual efforts. By exploring the limitations of traditional methods and the advantages of Examination Scheduling Systems (ESS) powered by algorithms like Kruskal's Algorithm, the study underscores the significance of technological solutions in tackling the complexities of scheduling exams amidst growing student numbers and evolving courses.

Through the utilization of the Z specification language for formal analysis and Unified Modeling Language (UML) diagrams for system visualization, the paper presents a robust framework for developing an effective examination scheduling system. It elucidates the various functionalities of the system, including user interactions, login processes, exam scheduling, and timetable viewing, ensuring a comprehensive understanding of the system architecture and operations.

Moreover, by reviewing related works on scheduling algorithms and highlighting the efficiency of Kruskal's Algorithm in handling sparse graphs, the paper emphasizes the importance of selecting appropriate algorithms to address specific scheduling challenges effectively.

In essence, this paper contributes to the ongoing discourse on optimizing university examination scheduling processes, offering insights into leveraging advanced technologies and formal methods to streamline operations, improve accuracy, and ultimately enhance the academic experience for both students and administrators. As universities continue to face the complexities of managing large-scale examinations, the adoption of automated scheduling systems supported by rigorous formal specifications and algorithmic techniques will undoubtedly play a pivotal role in driving efficiency and excellence in academic administration.

ACKNOWLEGEMENT

We extend our sincere appreciation to Prof. Dr. Rosziati Ibrahim for her invaluable guidance, understanding, and unwavering support throughout the completion of this paper. Her expertise and mentorship have been instrumental in shaping our work.

REFERENCES

[1] S. Wang, M. Bussieck, M. Guignard, A. Meeraus, and F. O'Brien, "Term-end exam scheduling at United States military academy/west point," *Journal of Scheduling,* vol. 13, no. 4, pp. 375-391, 2010.

[2] S. H. Bakri, H. Harun, A. Alzoubi, and R. Ibrahim, "The formal specification for the inventory system using Z language," 2013.

[3] A. Akbulut and G. Yilmaz, "University exam scheduling system using graphcoloring algorithm and rfid technology," *International Journal of Innovation, Management and Technology,* vol. 4, no. 1, p. 66, 2013.

[4] E. Özcan, A. J. Parkes, and A. Alkan, "The interleaved constructive memetic algorithm and its application to timetabling," *Computers & Operations Research,* vol. 39, no. 10, pp. 2310-2322, 2012.

[5] E. K. Burke and J. P. Newall, "Solving examination timetabling problems through adaption of heuristic orderings," *Annals of operations Research,* vol. 129, pp. 107-134, 2004.

[6] L. Di Gaspero and A. Schaerf, "Tabu search techniques for examination timetabling," in *Practice and Theory of Automated Timetabling III: Third International Conference,*

PATAT 2000 Konstanz, Germany, August 16–18, 2000 Selected Papers 3, 2001: Springer, pp. 104-117.

[7] S. S. Skiena, *The algorithm design manual*. Springer, 1998.

[8] K. D. Lee and S. Hubbard, "Graphs," in *Data Structures and Algorithms with Python: With an Introduction to Multiprocessing*: Springer, 2024, pp. 187-206.

[9] K. Salahddine, "The implementation of kruskal's algorithm for minimum spanning tree in a graph," in *E3S Web of Conferences*, 2021, vol. 297: EDP Sciences, p. 01062.

[10] R. Maurya and R. Sharma, "Comparison of Prim and Kruskal's Algorithm," *Global Journal of Computer Science and Technology,* vol. 23, no. C1, pp. 27-33, 2023.

[11] P. Citra, "The implementation of the Kruskal algorithm for the search for the shortest path to the location of a building store in the city of Bogor," in *IOP conference series: Materials Science and Engineering*, 2019, vol. 621, no. 1: IOP Publishing, p. 012010.

[12] M. M. Noaman, I. M. Alsmadi, and A. S. Jaradat, "The specifications of E-Commerce Secure System using Z."

[13] G. O'Regan, "Z formal specification language," in *Mathematical Foundations of Software Engineering: A Practical Guide to Essentials*: Springer, 2023, pp. 277-293.

[14] R. Ibrahim *et al.*, "Development of staff management system using UML-based object-oriented approach," in *Proceedings of the International Conference on Computing Technology and Information Management*, 2014, pp. 119-124.

[15] B. Mondal, B. Das, and P. Banerjee, "Formal specification of UML use case diagram–A CASL based approach," *International Journal of Computer Science and Information Technologies,* vol. 5, no. 3, pp. 2713-2717, 2014.