Timetable Wiz: Advanced Timetable Generation Solution for Educational Institutes

Kariyawasam Don Easara Ivanjaya Weerasinghe (IT21259852)
Wijayawardhana Nawoda Dhananjanee (IT21172182)
Dasanayaka Mudiyanselage Sasitha Udayantha (IT21266164)
Kaluthota Hewage Pasindu Nimsara De Silva (IT21208980)

BSc (Hons) Degree in Information Technology Specialization in Information Technology

Department of Information Technology

Sri Lanka Institute of Information Technology
Sri Lanka

April 2025

Timetable Wiz: Advanced Timetable Generation Solution for Educational Institutes

Kariyawasam Don Easara Ivanjaya Weerasinghe (IT21259852)

Wijayawardhana Nawoda Dhananjanee (IT21172182)

Dasanayaka Mudiyanselage Sasitha Udayantha (IT21266164)

Kaluthota Hewage Pasindu Nimsara De Silva (IT21208980)

Dissertation submitted in partial fulfilment of the requirements for the Bachelor of Science Special Honors Degree in Information Technology

Department of Information Technology

Sri Lanka Institute of Information Technology
Sri Lanka

April 2025

DECLARATION

We declare that this our own work & this proposal does not incorporate without acknowledgement any material previously submitted for a degree or diploma in any other university or Institute of higher learning & to the best of our knowledge & belief it does not contain any material previously published or written by another person except where the acknowledgement is made in the text.

Name	Student ID	Signature
Weerasinghe K.D.E.I	IT21259852	[illegible]
Wijayawardhana G.L.C.N.D	IT21172182	[illegible]
Udayantha D.M S	IT 2126614	Sasitha
De Silva K.H.P.N	IT21208980	[illegible]

The above candidate is carrying out research for the undergraduate Dissertation under my supervision.

---   ---------------------------------

Signature of the Supervisor   Date

---   ---------------------------------

Signature of the Co-Supervisor   Date

ACKNOWLEDGEMENT

We would like to thank our supervisor Mr. Jeewaka Perera for his immense support and guidance for this research project and our co-supervisor Mrs Sasini Hathurisnghe for her direction and support from the initial stage of this research project.

Also, a special thanks go to the students and parents for their contribution. In the end, it is with true pleasure that works with group members with their amazing contribution from the beginning of this research to the end.

ABSTRACT

Educational timetable scheduling presents a significant combinatorial optimization challenge, characterized by its NP-hard nature, complex constraints (both hard and soft), multiple conflicting objectives, and the need to satisfy diverse stakeholder requirements within dynamic environments. Traditional manual or simplistic rule-based methods often prove inadequate, leading to suboptimal resource utilization, scheduling conflicts, and stakeholder dissatisfaction. This research addresses these challenges by proposing an **Advanced Timetable Generation Solution**, integrating multiple sophisticated optimization paradigms and a comprehensive evaluation framework. The study explores and compares the efficacy of Evolutionary Algorithms (EAs) such as NSGA-II, MOEA/D, and SPEA2; Reinforcement Learning (RL) techniques including Q-Learning, DQN, and SARSA combined with Multi-Criteria Decision Making (MCDM); and bio-inspired Swarm Intelligence methods like Ant Colony Optimization (ACO), Bee Colony Optimization (BCO), and a novel hybrid Ant-Bee Swarm Optimization (ABSO). Central to this work is a robust Timetable Evaluation System designed to objectively assess schedule quality based on constraint satisfaction, resource utilization, workload balance, and institutional priorities, enabling meaningful comparison between algorithm outputs. The developed system incorporates practical features essential for real-world deployment, including Role-Based Access Control (RBAC) for personalized access, an Extract-Transform-Load (ETL) pipeline for data handling, interactive manual editing capabilities with real-time conflict validation, and API integration potential. Through comparative analysis and simulation on representative datasets, this research demonstrates the strengths and weaknesses of different optimization approaches, identifies effective strategies for handling dynamic constraints and multi-objective trade-offs, and validates a holistic system architecture. The findings contribute to the development of more efficient, adaptable, user-centric, and high-quality timetabling solutions for educational institutions, ultimately improving operational efficiency and stakeholder satisfaction.

Keywords—Timetable Scheduling, Optimization, Evolutionary Algorithms (EAs), Reinforcement Learning (RL), Swarm Intelligence, Ant Colony Optimization (ACO), Bee Colony Optimization (BCO), Multi-Objective Optimization, Constraint Management, Dynamic Scheduling, Timetable Evaluation, Role-Based Access Control (RBAC), Educational Institutions, NP-Hard Problems.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF ABBREVIATIONS

GA - Genetic Algorithm

CO - Colony Optimization

RL - Reinforcement Learning

EA- Evolutionary Algorithm

API - Application programming interface

MCDM - Multi-Criteria Decision Making

MOEA - Multi-Objective Evolutionary Algorithm

NSGA-II - Non-dominated Sorting Genetic Algorithm II

RBAC - Role-Based Access Control

VPS - Variable Population Size

PSO - Particle Swarm Optimization

ABSO - Ant-Bee Swarm Optimization

LIST OF APPENDICES

1 INTRODUCTION

1.1 Background

The scheduling of academic activities within educational institutions, commonly known as university timetabling, represents a fundamental operational necessity and a significant logistical challenge [1]. At its core, timetabling involves the allocation of limited resources—including teaching staff, student groups, classrooms, laboratories, and specific time slots—to a predefined set of events, such as lectures, tutorials, and practical sessions [2], [4]. The primary goal is to create a functional and conflict-free schedule that enables the smooth delivery of the academic curriculum.

However, the complexity of this task extends far beyond simple resource allocation. Educational institutions operate under a multitude of constraints that must be meticulously satisfied [3], [6]. These constraints are typically divided into two categories:

- **Hard Constraints:** These are rigid rules that *must* be met for a timetable to be considered feasible or valid. Violating a hard constraint renders the schedule unusable. Common examples include:
    - A lecturer cannot be assigned to two different classes simultaneously.
    - A student group cannot attend two different sessions at the same time.
    - A room cannot host more than one activity concurrently.
    - The number of students assigned to a room must not exceed its capacity.
    - All required academic activities must be scheduled.

- **Soft Constraints:** These represent preferences, desirable qualities, or optimization goals that enhance the quality and usability of the timetable, although violating them does not necessarily invalidate it. Examples include:
    - Minimizing idle time (gaps) between classes for students.
    - Balancing the teaching workload evenly among instructors.
    - Assigning classes to preferred time slots or rooms for specific lecturers or courses.
    - Ensuring compactness of schedules (e.g., avoiding single classes late in the day).
    - Respecting specific sequencing or proximity requirements between related courses.

The sheer number of activities, resources, and constraints, coupled with their intricate interdependencies, makes timetable a computationally difficult problem. It is formally recognized as an NP-hard combinatorial optimization problem [1], [8], meaning that finding a guaranteed optimal solution becomes computationally infeasible as the size of the institution (number of courses, students, rooms) increases.

Historically, many institutions relied on manual timetabling processes or basic, often spreadsheet-based, rule systems [1]. While manageable for very small entities, these traditional methods are fraught with limitations in modern, complex academic environments. They are notoriously time-consuming, often requiring weeks of effort by administrative staff. They are highly prone to errors, leading to conflicts that necessitate frequent and disruptive revisions. Furthermore, manual methods struggle to effectively balance the numerous soft constraints and stakeholder preferences, often resulting in schedules that are feasible but far from optimal in terms of resource efficiency or user satisfaction [3].

The dynamic nature of academic institutions further exacerbates these challenges. Unexpected events like instructor illness, room unavailability due to maintenance, changes in student enrollment numbers, or last-minute curriculum adjustments require rapid rescheduling capabilities that manual systems cannot easily provide. The inability to adapt quickly often leads to cascading disruptions and dissatisfaction among students and faculty.

Consequently, there is a clear and growing need for more sophisticated, automated, and optimized approaches to timetable generation and management. The limitations of traditional methods underscore the necessity for computational solutions that can navigate complexity, handle diverse constraints effectively, adapt to dynamic changes, and produce high-quality schedules that enhance operational efficiency and stakeholder satisfaction within educational institutions.

1.2 Literature Survey

In response to the complexities and limitations inherent in manual timetabling, a significant body of research has explored the application of various computational intelligence and optimization techniques. These approaches aim to automate the scheduling process, handle constraints more effectively, and optimize timetables according to multiple objectives. Key areas of investigation include:

- **Evolutionary Algorithms (EAs):** EAs have been widely applied due to their ability to explore large, complex search spaces.
    - **Genetic Algorithms (GAs):** Foundational work utilized GAs, inspired by natural selection, employing operators like crossover and mutation to evolve populations of candidate timetables [1], [8]. Studies explored variations, such as using dynamic chromosome sizes to better adapt to varying course loads across departments, demonstrating improved effectiveness over static approaches and manual scheduling [3]. Other preliminary studies focused on multi-parent crossover techniques, suggesting potential for more robust solutions compared to traditional two-parent approaches, although optimality was not always guaranteed [2].

    - **Multi-Objective Evolutionary Algorithms (MOEAs):** Recognizing the multi-objective nature of timetable, research has increasingly focused on advanced MOEAs such as **NSGA-II** [12], **SPEA2** [14], and **MOEA/D** [13]. These algorithms are designed to find a set of Pareto-optimal solutions, representing different trade-offs between conflicting objectives. NSGA-II uses fast non-dominated sorting and crowding distance, SPEA2 employs an external archive and fine-grained fitness assignment, while MOEA/D decomposes the problem into scalar subproblems. Comparative studies within this family aim to identify strengths in handling specific constraints or achieving better solution diversity. Research has also explored enhancing MOEAs like NSGA-II with variable population sizes (VPS) and lifetime concepts to improve performance, particularly in reducing soft constraint violations [4].

- Reinforcement Learning (RL): RL offers a different paradigm, framing timetable as a sequential decision-making process where an agent learns to make optimal assignments by interacting with the environment and receiving rewards or penalties.
    - Algorithms such as **Q-Learning, Deep Q-Learning (DQN), and SARSA** have been explored for their potential adaptability to dynamic environments and their ability to learn complex scheduling policies over time. Integrating RL with Multi-Criteria Decision Making (MCDM) techniques has been proposed to handle conflicting objectives and prioritize decisions based on user-defined criteria, aiming for both adaptability and optimality. However, challenges remain in defining effective reward structures, ensuring scalability, and guaranteeing convergence, particularly in highly constrained, large-scale problems.

- **Swarm Intelligence Algorithms:** These algorithms draw inspiration from the collective behavior of social organisms.
    - Ant Colony Optimization (ACO): Based on ants' pheromone-laying behavior, ACO has been applied to construct timetables where artificial ants probabilistically choose assignments guided by pheromone trails and heuristic information [11]. Research highlights its effectiveness in finding high-quality solutions by reinforcing good assignment patterns but notes its sensitivity to parameter tuning (e.g., pheromone evaporation rate) [6].
    - Bee Colony Optimization (BCO) / Artificial Bee Colony (ABC): Mimicking bee foraging, BCO/ABC uses employed, onlooker, and scout bees to perform local search, exploitation, and diversification, respectively [5], [14]. It is considered robust and adaptable, often requiring fewer parameters than ACO. Its effectiveness in generating feasible timetables has been demonstrated [5].
    - Particle Swarm Optimization (PSO): Inspired by flocking/schooling behavior, PSO uses particles representing timetables that adjust their "positions" based on personal best (pBest) and global best (gBest) solutions [4], [13]. PSO is often recognized for its rapid convergence, particularly towards feasible solutions, but can risk premature convergence to local optima [4].
    - Hybrid Swarm Methods: To leverage the strengths of different approaches, hybrid algorithms have been explored, including combinations like Ant-Bee Swarm Optimization (ABSO) which aims to merge ACO's guided construction with BCO's diversification mechanisms. Other hybrids combining swarm methods with GAs or local search have also shown potential [15].

Sequential hybrids (e.g., PSO followed by ACO) have also been considered.

Timetable Evaluation Methodologies: Beyond generation, literature emphasizes the need for robust evaluation. Early methods focused primarily on hard constraint satisfaction. More recent work stresses the importance of multi-criteria evaluation, incorporating soft constraints, resource utilization metrics (like room usage or teacher load balance), schedule compactness, and stakeholder preferences [4], [6]. Kingston [4] proposed unified evaluation metrics, while MirHassani [6] highlighted the need for multi-dimensional scoring systems with appropriate weighting. However, a standardized, comprehensive evaluation framework directly comparing outputs from diverse algorithmic paradigms remains an area needing further development.

Practical System Features: While algorithmic research is abundant, studies integrating these algorithms into practical, usable systems are less common. Some research touches upon necessary components like Role-Based Access Control (RBAC), interactive manual editing with conflict validation, API integration, and data handling pipelines (ETL). The need for user-friendly interfaces and features that support real-world administrative workflows is often highlighted as crucial for adoption.

This survey reveals a rich landscape of computational techniques applied to timetable. However, it also highlights the fragmentation of research efforts, often focusing on specific algorithms or limited aspects of the problem, thereby motivating the need for comparative studies, integrated evaluation frameworks, and holistic system designs as addressed in this research.

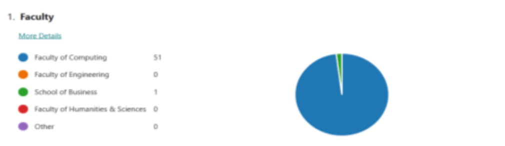
1.3 Background Survey

To gain a clearer understanding of the practical challenges and user dissatisfaction associated with existing timetabling systems, a background survey was conducted involving key stakeholders—students, academic staff, and administrative staff—at the Sri Lanka Institute of Information Technology (SLIIT). This survey aimed to identify specific pain points and unmet needs within a real-world university environment. The key findings and results extracted in below:

- **Overall Dissatisfaction:** A significant number of respondents across user groups expressed dissatisfaction with their current timetables, indicating that the schedules often only partially met their needs or preferences.
- **Frequent Timetable Issues:** Students reported frequently encountering problems such as overlapping classes (clashes) and inconveniently long gaps between scheduled lectures or labs. These issues directly impact the student learning experience and time management.
- **Classroom Availability and Suitability:** Instances were noted where allocated classrooms or labs were insufficient to accommodate the enrolled student numbers, hindering effective participation and teaching. Furthermore, the suitability of assigned spaces (e.g., appropriate facilities, size) was sometimes inadequate.
- **Misalignment with Preferred Schedules:** The survey highlighted a disconnect between the generated timetables and the preferred study or working patterns of students and staff. Many expressed preferences for avoiding very early morning or late evening classes, yet the existing timetables often failed to accommodate these common requests.
- **Communication and Notification Deficiencies:** A major point of frustration was the lack of timely and systematic communication regarding timetable changes. Notifications, when provided, were often delayed or managed through ad-hoc administrative channels, leading to confusion and

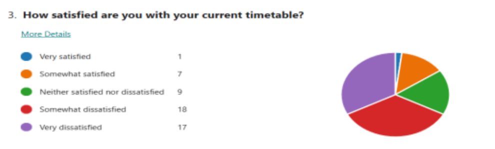
dissatisfaction among users who rely on accurate, up-to-date schedule information.

- **Desire for Improvements:** Stakeholders provided constructive suggestions for improvement. Common themes included the need for better logical grouping of lectures and labs (e.g., minimizing travel time between sessions), more predictable scheduling patterns, and the implementation of a reliable, centralized notification system for any changes.



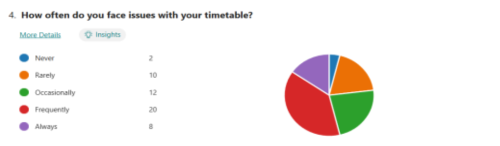
1. Faculty

● Faculty of Computing	51
● Faculty of Engineering	0
● School of Business	1
● Faculty of Humanities & Sciences	0
● Other	0



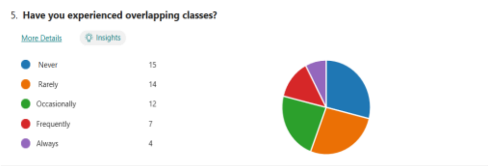
3. How satisfied are you with your current timetable?

● Very satisfied	1
● Somewhat satisfied	7
● Neither satisfied nor dissatisfied	9
● Somewhat dissatisfied	18
● Very dissatisfied	17



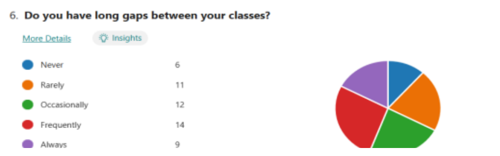
4. How often do you face issues with your timetable?

● Never	2
● Rarely	10
● Occasionally	12
● Frequently	20
● Always	8



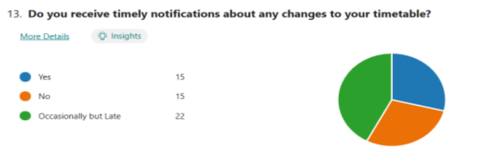
5. Have you experienced overlapping classes?

● Never	15
● Rarely	14
● Occasionally	12
● Frequently	7
● Always	4



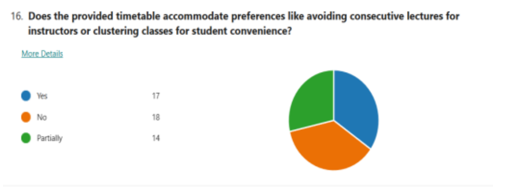
6. Do you have long gaps between your classes?

● Never	6
● Rarely	11
● Occasionally	12
● Frequently	14
● Always	9



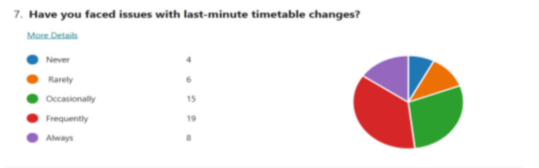
13. Do you receive timely notifications about any changes to your timetable?

● Yes	15
● No	15
● Occasionally but Late	22



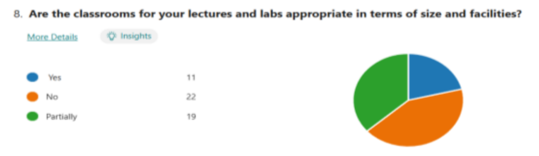
16. Does the provided timetable accommodate preferences like avoiding consecutive lectures for instructors or clustering classes for student convenience?

● Yes	17
● No	18
● Partially	14



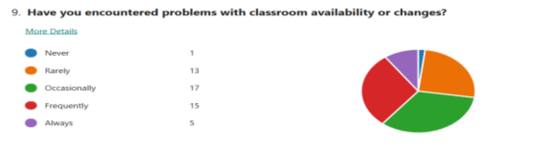
7. Have you faced issues with last-minute timetable changes?

● Never	4
● Rarely	6
● Occasionally	15
● Frequently	19
● Always	8



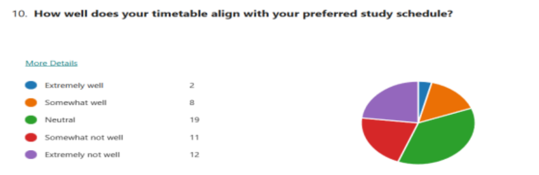
8. Are the classrooms for your lectures and labs appropriate in terms of size and facilities?

● Yes	11
● No	22
● Partially	19



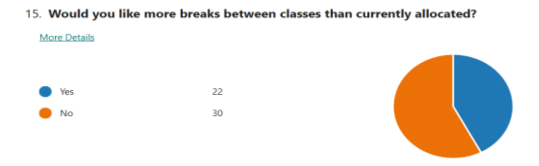
9. Have you encountered problems with classroom availability or changes?

● Never	1
● Rarely	13
● Occasionally	17
● Frequently	15
● Always	5



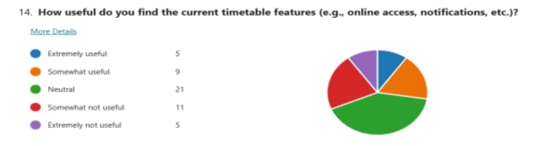
10. How well does your timetable align with your preferred study schedule?

● Extremely well	2
● Somewhat well	8
● Neutral	19
● Somewhat not well	11
● Extremely not well	12



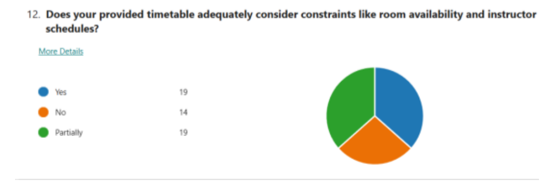
15. Would you like more breaks between classes than currently allocated?

● Yes	22
● No	30



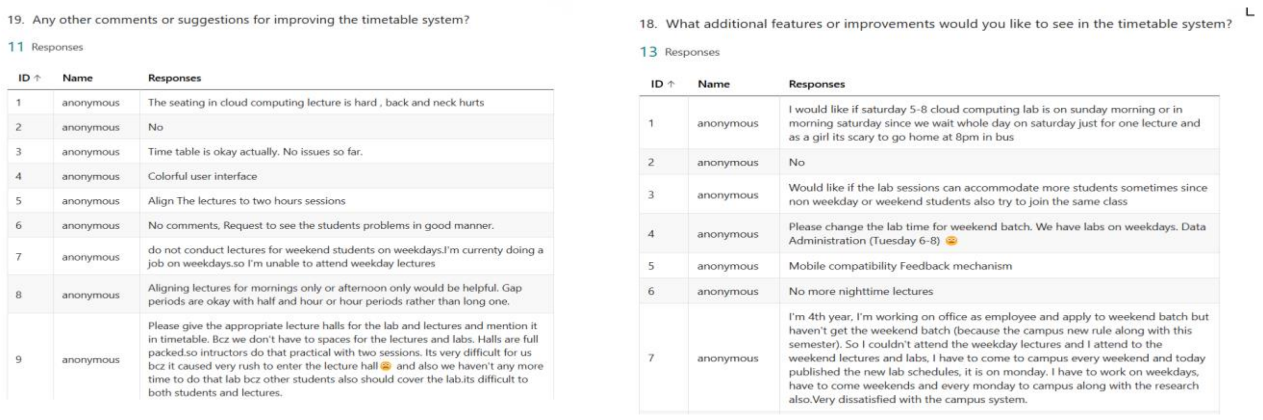
14. How useful do you find the current timetable features (e.g., online access, notifications, etc.)?

● Extremely useful	5
● Somewhat useful	9
● Neutral	21
● Somewhat not useful	11
● Extremely not useful	5



12. Does your provided timetable adequately consider constraints like room availability and instructor schedules?

● Yes	19
● No	14
● Partially	19



19. Any other comments or suggestions for improving the timetable system?

11 Responses

ID ↑	Name	Responses
1	anonymous	The seating in cloud computing lecture is hard , back and neck hurts
2	anonymous	No
3	anonymous	Time table is okay actually. No issues so far.
4	anonymous	Colorful user interface
5	anonymous	Align The lectures to two hours sessions
6	anonymous	No comments, Request to see the students problems in good manner.
7	anonymous	do not conduct lectures for weekend students on weekdays.I'm currenty doing a job on weekdays.so I'm unable to attend weekday lectures
8	anonymous	Aligning lectures for mornings only or afternoon only would be helpful. Gap periods are okay with half and hour or hour periods rather than long one.
9	anonymous	Please give the appropriate lecture halls for the lab and lectures and mention it in timetable. Bcz we don't have to spaces for the lectures and labs. Halls are full packed.so intructors do that practical with two sessions. Its very difficult for us bcz it caused very rush to enter the lecture hall 😩 and also we haven't any more time to do that lab bcz other students also should cover the lab.its difficult to both students and lectures.

18. What additional features or improvements would you like to see in the timetable system?

13 Responses

ID ↑	Name	Responses
1	anonymous	I would like if saturday 5-8 cloud computing lab is on sunday morning or in morning saturday since we wait whole day on saturday just for one lecture and as a girl its scary to go home at 8pm in bus
2	anonymous	No
3	anonymous	Would like if the lab sessions can accommodate more students sometimes since non weekday or weekend students also try to join the same class
4	anonymous	Please change the lab time for weekend batch. We have labs on weekdays. Data Administration (Tuesday 6-8) 😩
5	anonymous	Mobile compatibility Feedback mechanism
6	anonymous	No more nighttime lectures
7	anonymous	I'm 4th year, I'm working on office as employee and apply to weekend batch but haven't get the weekend batch (because the campus new rule along with this semester). So I couldn't attend the weekday lectures and I attend to the weekend lectures and labs, I have to come to campus every weekend and today published the new lab schedules, it is on monday. I have to work on weekdays, have to come weekends and every monday to campus along with the research also.Very dissatisfied with the campus system.

Figure 1 - Student Survey Data in SLIIT

1.4 Research Gap

The preceding literature survey and background analysis reveal substantial progress in applying computational techniques to educational timetabling. However, a closer examination exposes gaps between the capabilities of existing research or isolated systems and the requirements for a truly comprehensive, adaptable, and user-centric solution. Many studies focus on specific algorithms or constraints in idealized scenarios, often neglecting the integration needed for practical deployment or the dynamic complexities of real-world institutions. The following table consolidates the key research gaps identified across the reviewed literature and user surveys, highlighting the areas addressed by the proposed advanced timetable generation system:

Table 1 - Comparison of existing systems

Feature / Capability	Herath (2017) (GA Focus)	Abdullah et al. (2010) (NSGA-II+VPS)	Ai et al. (2022) [P1, Ref 1] (RL-Bus)	Ojha et al. (2019) [P4, Ref 5] (BCO)	Mazlan et al. (2019) [P4, Ref 6] (ACO)	Deb et al. (2002) (NSGA-II Algo)	Proposed System (Timetable Wiz)
Optimization Approach	GA	MOEA (NSGA-II)	Deep RL	BCO	ACO	MOEA (NSGA-II)	Multi-Paradigm (EA, RL, Swarm, Hybrid) ✓
Hard Constraint Handling	Basic (Clashes)	Handled	Dynamic Opt. Focus	Handled	Handled	Algorithm Focus	Robust & Explicit ✓
Soft Constraint Optimization	Limited	Addresses Soft Constraints	Not Primary Focus	Addresses Soft Constraints	Addresses Soft Constraints	Algorithm Focus	Multi-Dimensional & Weighted ✓
Dynamic Adaptation / Real-Time	Static	Static (VPS helps)	Dynamic Opt.	Static	Static	Static	Addresses Gap (RL+MCDM, Validation) ✓
Multi-Objective Handling	Not Explicit	MOEA	Not Explicit	Not Explicit	Not Explicit	MOEA	Core Feature (MOEAs, MCDM) ✓
Comprehensive Evaluation System	Not Addressed	Not Addressed	Not Addressed	Not Addressed	Not Addressed	Not Addressed	Key Contribution ✓
Integrated Software System	Algorithm Only	Algorithm Only	Algorithm Only	Algorithm Only	Algorithm Only	Algorithm Only	Holistic System ✓
Role-Based Access Control (RBAC)	Not Addressed	Not Addressed	Not Addressed	Not Addressed	Not Addressed	Not Addressed	Implemented ✓
Manual Editing & Validation	Not Addressed	Not Addressed	Not Addressed	Not Addressed	Not Addressed	Not Addressed	Implemented (Real-Time) ✓
Chatbot Integration	Not Addressed	Not Addressed	Not Addressed	Not Addressed	Not Addressed	Not Addressed	Proposed Feature ✓
Data Handling (ETL)	Not Addressed	Not Addressed	Not Addressed	Not Addressed	Not Addressed	Not Addressed	Implemented ✓
Scalability Testing	Limited Scope	Tested on Benchmarks	Domain Specific	Limited Scope	Limited Scope	Algorithm Focus	Addressed (SLIIT/Muni) ✓

Summary: This table illustrates that while individual studies in the literature have explored specific algorithms (GA, MOEA, RL, BCO, ACO) or features (like VPS or dynamic optimization in other domains), a system that **integrates multiple paradigms**, provides a **comprehensive evaluation framework**, handles **dynamic aspects**, and incorporates **practical software features** (RBAC, validated editing, ETL, potential chatbot) represents a significant advancement addressing multiple identified gaps.

1.5 Research Problem

Educational timetabling is a well-established NP-hard problem, critical to the functioning of academic institutions. While numerous computational approaches (including Evolutionary Algorithms, Reinforcement Learning, and Swarm Intelligence) have been developed to automate schedule generation, significant challenges and limitations persist, hindering the widespread adoption of truly effective and comprehensive solutions. Based on the identified gaps in the literature and practical shortcomings revealed by user surveys, the core research problem can be defined as:

The lack of an integrated, adaptive, and comprehensively evaluated timetable system capable of objectively comparing diverse advanced optimization algorithms (EA, RL, Swarm) while addressing real-world complexities, dynamic constraints, multi-objective trade-offs, and essential user-centric features required for practical deployment in educational institutions.

This overarching problem manifests in several specific issues:

1. **Difficulty in Algorithm Selection:** Institutions lack objective means to compare and select the most suitable optimization algorithm (e.g., NSGA-II vs. ACO vs. RL) for their specific needs, as cross-paradigm benchmarks using unified metrics are scarce.

2. **Inadequate Handling of Real-World Dynamics:** Existing systems often fail to efficiently adapt to real-time changes (e.g., staff absence, room changes), requiring cumbersome manual intervention or complete rescheduling.
3. **Suboptimal Solutions due to Poor Evaluation:** Without robust multi-criteria evaluation, generated timetables may be feasible but fail to satisfy crucial quality aspects like workload balance, student convenience, or resource utilization efficiency, leading to stakeholder dissatisfaction.
4. **Gap Between Algorithmic Potential and Practical Usability:** Advanced algorithms are often presented in isolation, lacking integration into functional software with essential features like role-based access, intuitive interfaces, validated manual editing, and seamless data handling, limiting their practical value.
5. **Uncertainty in Scalability and Hybrid Effectiveness:** The performance and scalability of various algorithms, especially novel hybrid approaches, on large, complex institutional datasets are not well-established, making it difficult to predict their effectiveness in demanding scenarios.

Therefore, the central challenge is to design, implement, and evaluate a system that bridges these gaps, moving beyond isolated algorithmic improvements towards a holistic, practical, and intelligent solution for advanced timetable generation and evaluation.

1.6 Research Objectives

1.6.1 Main Objective

To design, develop, and evaluate an Advanced Timetable Generation Solution that integrates multiple optimization paradigms (Evolutionary Algorithms, Reinforcement Learning, Swarm Intelligence), incorporates a comprehensive multi-criteria evaluation framework, and includes practical system features to effectively address the complex, dynamic, and user-centric requirements of educational timetable scheduling.

1.6.2 Specific Objectives

1. **Implement and Compare Diverse Optimization Algorithms:**
    - Implement representative algorithms from three major paradigms:
        - **Evolutionary Algorithms:** NSGA-II, MOEA/D, SPEA2.
        - **Reinforcement Learning:** Q-Learning, DQN, SARSA (potentially integrated with MCDM).
        - **Swarm Intelligence:** ACO, BCO/ABC, PSO, and a hybrid ABSO.
    - Adapt these algorithms for the specific constraints and structures of educational timetabling problems (using SLIIT and Muni datasets as test cases).
    - Conduct comparative performance analysis of these algorithms based on constraint satisfaction, solution quality, and convergence characteristics using standardized datasets and metrics.
2. **Develop a Comprehensive Timetable Evaluation System:**
    - Design and implement a multi-criteria evaluation module capable of assessing quality based on:
        - Hard constraint violations (feasibility).
        - Soft constraint satisfaction (e.g., student/lecturer fatigue, idle time, schedule spread, workload balance).
        - Resource utilization efficiency.

- Develop a robust, weighted scoring mechanism to provide a single, objective quality score for any given timetable, facilitating comparison.

3. **Design and Implement Integrated Software Architecture:**
   - Develop a modular software system incorporating:
     - Data input and preprocessing capabilities (ETL).
     - Multiple optimization algorithm modules.
     - The comprehensive evaluation and scoring engine.
     - A persistent database (e.g., MongoDB/PostgreSQL) for storing schedules, results, and user data.
     - A functional backend API (e.g., Python/Flask/FastAPI).
     - An interactive frontend user interface (e.g., React).

4. **Incorporate Practical User-Centric Features:**
   - Implement Role-Based Access Control (RBAC) to provide differentiated views and permissions for administrators, lecturers, and students.
   - Develop an interactive manual editing interface with **real-time conflict validation** (single and cross-timetable) to allow for safe and effective schedule refinement.
   - Explore and potentially integrate an interactive chatbot interface for user queries regarding schedules and constraints.
5. **Validate System Performance and Scalability:**
   - Conduct thorough testing (unit, integration, system, usability, performance) to ensure system correctness, robustness, and usability.
   - Evaluate the system's performance and the scalability of the implemented algorithms using datasets of varying complexity (e.g., SLIIT, Muni-fsps-spr17).
   - Analyze the trade-offs between different algorithms in terms of solution quality, computational cost, and suitability for different institutional contexts.

2 METHODOLOGY

This chapter outlines the systematic methodology adopted to design, implement, and evaluate the proposed Advanced Timetable Generation Solution. The approach integrates requirements engineering, advanced optimization techniques (Evolutionary Algorithms, Reinforcement Learning, Swarm Intelligence), the development of a comprehensive evaluation framework, and the implementation of a practical, user-centric software system.

2.1 Overall Approach

The research followed a multi-faceted approach:

1. **Requirement Analysis:** Gathering functional and non-functional requirements based on literature, existing system limitations, and user feedback (via background surveys).
2. **Algorithm Selection and Implementation:** Selecting, adapting, and implementing representative algorithms from diverse optimization paradigms (EA, RL, Swarm) suitable for educational timetabling.
3. **Evaluation Framework Development:** Designing and implementing a robust, multi-criteria evaluation system to objectively assess and score timetable quality.
4. **System Design and Integration:** Developing a modular software architecture integrating data handling, optimization algorithms, the evaluation engine, user management, and user interface components.
5. **Testing and Validation:** Employing a rigorous testing strategy (functional, non-functional, performance, usability) to validate the system's correctness, robustness, and effectiveness using real-world and benchmark datasets.
6. **Comparative Analysis:** Evaluating and comparing the performance of different algorithms within the integrated system to understand their trade-offs and suitability.

2.2 System Methodology

The proposed system employs a multi-layered and modular framework designed to tackle the complex and dynamic nature of academic timetable scheduling using advanced optimization techniques. The architecture, illustrated in Figures 2.1, is structured around three main pillars: the Frontend, the FastAPI Backend, and the Scheduler Engine.

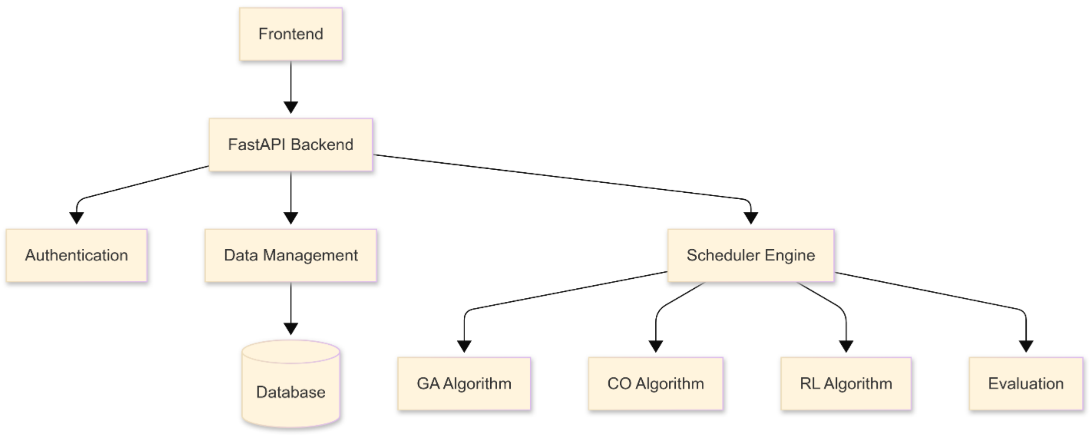


Figure 2 - System Architecture Diagram

- **Frontend Layer (React Application):** This layer serves as the Graphical User Interface (GUI) for all user roles (Admin, Lecturers, Students). Developed using **React.js**, it allows users to input preferences, constraints, and configurations, trigger timetable generation, and view schedules and evaluation feedback visually. It is structured internally with:
    - **React UI Layer:** Handles user interactions and data rendering.
    - **Redux State Layer:** Manages the application's global state centrally, ensuring data consistency across modules (using Redux Thunk for asynchronous actions, see Figure 2.2.2).
    - **API Integration Layer:** Connects the frontend to the backend via secure RESTful HTTP requests.

- **Logical Modules:** Includes an Auth Module (login/authentication), Admin Module (settings, permissions), Scheduling Module (interacting with backend for generation/display), and Data Module (handling constraint/metadata input/display). *(See Figure 2.2.3)*

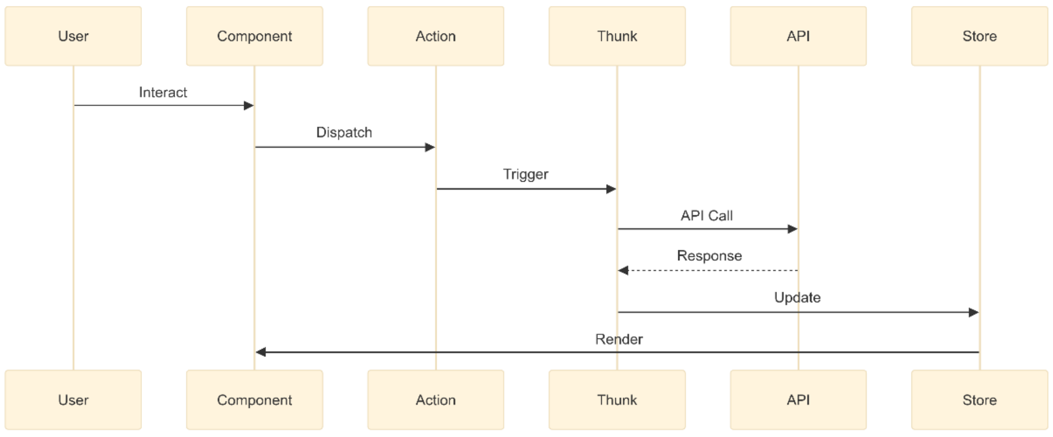


Figure 3 - using Redux Thunk for asynchronous actions

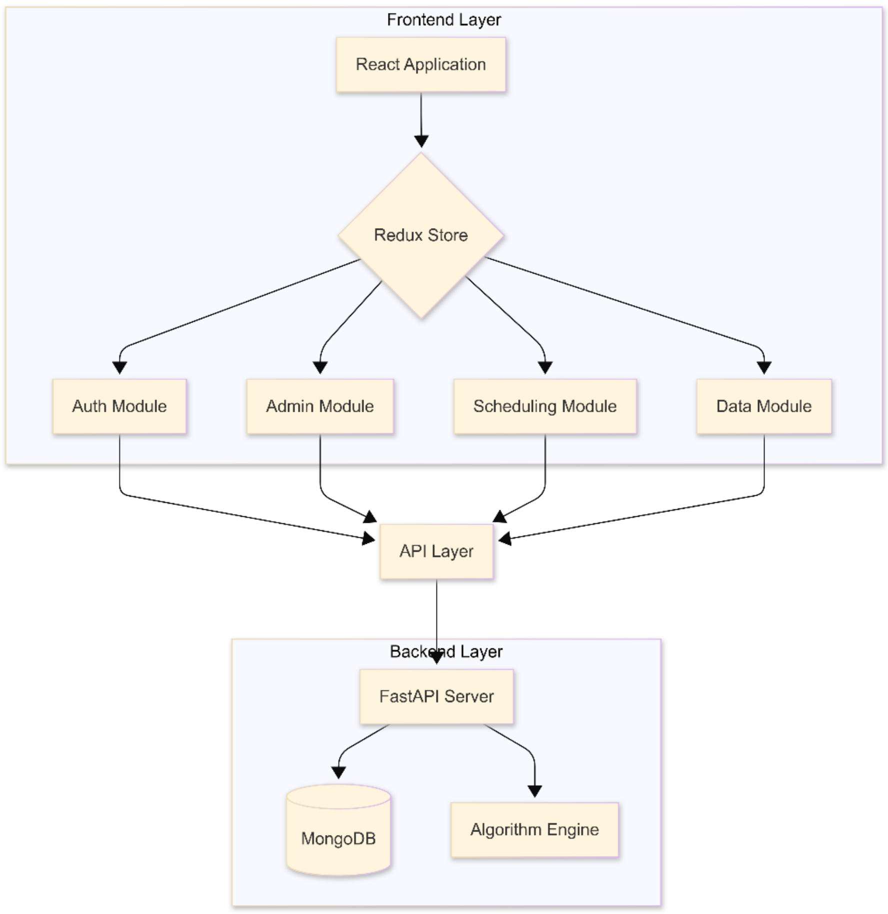
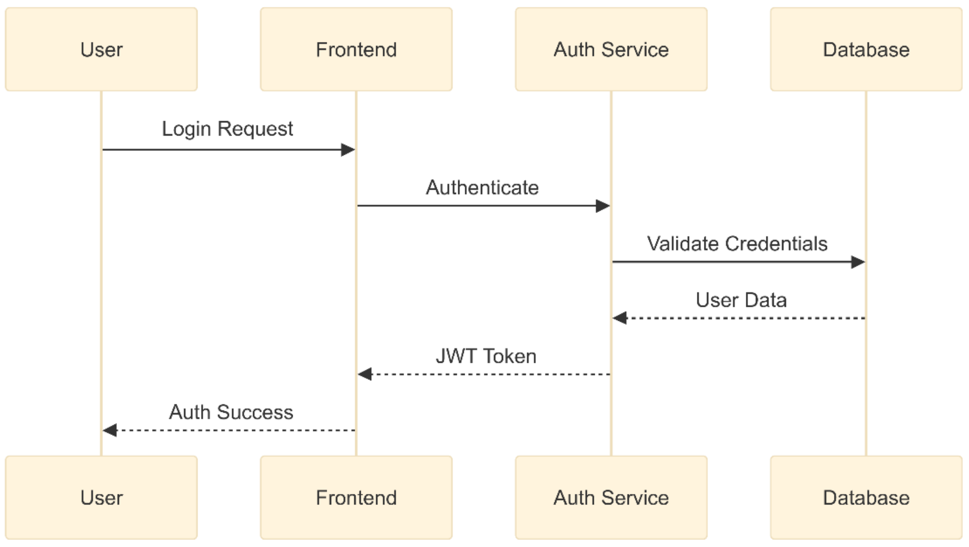


Figure 4 - handling constraint/metadata input/display

- **Backend Layer (FastAPI Application):** This is the core processing unit, built using **FastAPI** for high performance and asynchronous capabilities. It follows a modular structure with:
    - **Routers:** Manage incoming API requests from the frontend.
    - **Models (Pydantic):** Define data validation schemas for requests and responses.
    - **Services:** Contain business logic, process data, interact with external modules (Database, Scheduler Engine), and coordinate scheduling requests.
    - **Authentication Module:** Handles user validation and access control using **JSON Web Tokens (JWT)** for secure, stateless sessions. *(See Figure 2.2.4)*

- **Data Management Module:** Executes Create, Read, Update, Delete (CRUD) operations on the database for course schedules, room availability, instructor preferences, constraints, etc.
- **Logic Layer (L):** Promotes reusability via shared utility functions used across Routers and Models. *(See Figure 2.2.5)*



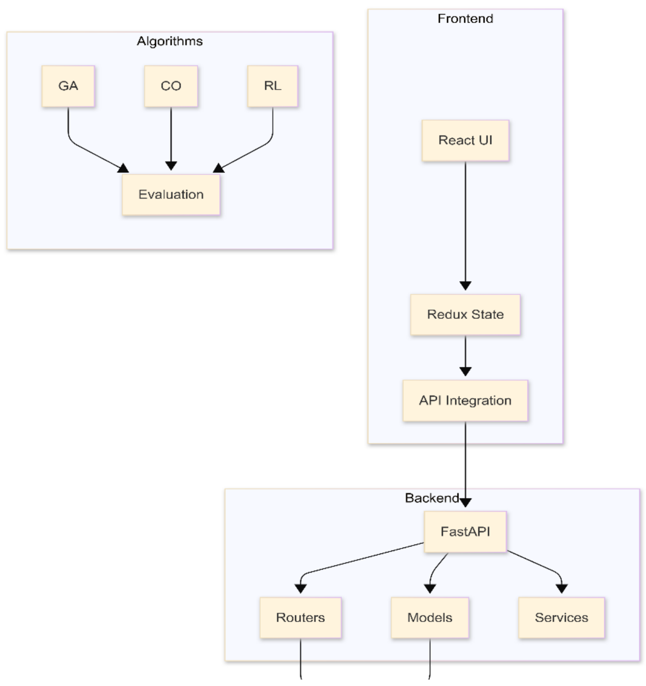


Figure 5 - Handles user validation and access control using JSON Web Tokens

- **Scheduler Engine (Standalone Component):** This engine houses the core optimization logic and is designed to be modular and extensible. It integrates multiple metaheuristic and machine learning algorithms:
  - **GA Algorithm Module:** Implements Genetic Algorithms .
  - **RL Algorithm Module:** Implements Reinforcement Learning agents (details in Section 2.4.1).
  - **Colony Optimization (CO) Module:** Implements swarm intelligence algorithms **ACO, BCO, and PSO** .
  - Outputs from these algorithms are routed to the Evaluation Module.
- **Evaluation Engine** Assesses the feasibility and quality of generated schedules based on hard and soft constraints, returning evaluation results (scores, violation details) to the backend/frontend. .

2.3 Data Handling and Representation

Two primary datasets were used for development, testing, and evaluation, representing different levels of complexity:

1. **SLIIT Computing Dataset:** A real-world dataset representing a faculty timetable scenario (e.g., Faculty of Computing at SLIIT), characterized by specific module structures, activity types (Lectures, Tutorials, Labs), room types, lecturer assignments, and student subgroupings based on specializations and years. This dataset features around 60 courses, 55 rooms, and 50 student groups, presenting significant capacity constraints and scheduling interdependencies.
2. **Muni-fsps-spr17 Dataset (ITC 2019):** A complex, large-scale benchmark dataset from the International Timetabling Competition 2019. It includes 226 courses, 561 classes, 44 rooms, 865 students, and a mix of hard and soft constraints with defined weights, representing a highly realistic and challenging scheduling environment.

2.4 **Core Scheduling Logic Implementation**

Before optimization, the system preprocesses institutional data:

- **Module Definition:** Modules (courses) are defined with attributes like code, name, credits, and associated specializations/student groups.
- **Activity Generation:** Based on module definitions, the system auto-generates specific activities (Lecture, Tutorial, Lab) that need scheduling. Shared modules generate activities for each relevant specialization.
- **Teacher Assignment:** Teachers are assigned to activities, either based on pre-defined module data or using rule-based/randomized assignment for tutorials/labs, ensuring fair workload distribution where possible.
- **Lab Subgrouping:** For lab activities exceeding room capacity, the system dynamically calculates the required number of subgroups ($g = \lceil N/C \rceil$, where N=students, C=capacity) and creates corresponding lab session activities. This calculation adapts to available room capacities and enrollment numbers. Fair teacher assignments (round-robin or tracked random selection) is applied to these subgroups.

2.4.1 Algorithm Implementation

- **Common Components:**
    - **Initialization:** Generating an initial population of candidate timetables using heuristics (e.g., greedy filler, constraint-aware sampling) appropriate for the chosen solution representation.
    - **Fitness Evaluation:** A shared module calculates the quality of a timetable based on hard constraint violations (penalty) and soft constraint satisfaction (score/penalty), as defined in the Evaluation System (Section 2.5).

    - **Variation Operators:** Standard genetic operators (e.g., uniform crossover, swap/random-reset mutation) adapted for the specific dictionary/assignment representations.
    - **Repair Mechanisms (Muni):** For the complex ITC dataset, optional repair functions attempt to fix hard constraint violations introduced by variation operators.
    - **Local Search (SLIIT):** Optional periodic local search (e.g., swapping nearby activities) applied to individuals to potentially accelerate convergence.
- **Specific Algorithm Details:**
    - **EAs (NSGA-II, MOEA/D, SPEA2):** Implemented using libraries like DEAP or custom code, incorporating mechanisms like fast non-dominated sorting, crowding distance (NSGA-II), Tchebycheff decomposition and neighborhood-based evolution (MOEA/D), and external archive management with fine-grained fitness and density estimation (SPEA2). Focus on finding Pareto-optimal fronts representing trade-offs.
    - **RL (Q-Learning, DQN, SARSA):** Implemented using TensorFlow/PyTorch and potentially OpenAI Gym environments. State representation encodes the current timetable state, actions correspond to scheduling decisions (assigning activity to slot/room). Reward functions were designed based on constraint satisfaction (large negative rewards for hard constraint violations, smaller positive/negative rewards for soft constraints). MCDM integration involves adjusting reward components based on prioritized criteria.
    - **Swarm (ACO, BCO, PSO):** Implemented using DEAP or custom code.
        - *ACO:* Graph representation (activities as steps, slot/room choices as edges), pheromone

        initialization/update/evaporation, heuristic calculation (e.g., inverse of conflicts), probabilistic state transition rule.

    - *BCO/ABC:* Food source representation (timetable), nectar calculation (fitness), employed bee phase (local search/neighbor generation), onlooker bee phase (probabilistic selection based on fitness), scout bee phase (abandonment/random reset based on 'limit' counter).
    - *PSO:* Particle representation (timetable), fitness calculation, pBest/gBest tracking, discrete velocity/position update rules adapted for timetable perturbations (e.g., guided swaps based on pBest/gBest differences).

A dedicated evaluation module was implemented in Python to provide objective scoring:

- **Hard Constraint Evaluation:**
    - Checks for lecturer conflicts, student group conflicts, room double-bookings, and room capacity violations using efficient methods (e.g., Python sets for overlap detection, direct comparison for capacity).
    - Tracks unassigned activities and potentially vacant rooms.
    - Calculates a **Total Hard Constraint Violation Score** (sum of all violations). A score of 0 indicates a feasible timetable.
- **Soft Constraint Evaluation:**
    - Calculates metrics related to user preferences and schedule quality:
        - ***Student Metrics:*** Fatigue (total lectures), Idle Time (gaps between classes), Lecture Spread (variance in lecture timing).
        - ***Lecturer Metrics:*** Fatigue, Idle Time, Lecture Spread, Workload Balance (variance vs. mean workload).

    - **Normalization:** Each metric is normalized (e.g., scaled to) to allow fair comparison.
    - **Weighted Scoring:** A **Final Soft Constraint Score (FSC)** is calculated as a weighted sum of the normalized metrics. Metrics representing undesirable qualities (e.g., fatigue, idle time, spread) are typically subtracted from 1 before weighting, so higher scores indicate better quality. Weights are configurable (e.g., student fatigue/idle time weighted higher).
- **Centralized evaluate() Function:** Provides a single interface that takes a timetable solution and returns both the hard violation count and the final soft constraint score, orchestrating calls to the respective evaluation functions.

2.5 System Features Implementation

- **RBAC:** Implemented using backend logic (e.g., decorators or middleware in Flask/FastAPI) that checks user roles associated with authenticated sessions against required permissions for accessing specific API endpoints or data views. Frontend components dynamically render options based on user role.
- **Manual Editing & Validation:**
    - The frontend displays timetables in an interactive grid. Drag-and-drop functionality triggers backend API calls.
    - **Real-Time Validation:** On attempting an edit (e.g., drop), the backend invokes:
        - ***Single Timetable Conflict Checker:*** Checks for overlaps/violations within the specific view being edited (e.g., student group's schedule).
        - ***Cross Timetable Conflict Checker:*** Checks for resource clashes (teacher, room) with activities in *other* related timetables.
    - Immediate feedback (allow/disallow, error message, highlighting) is sent to the frontend. Edits are only committed to the database if validation passes. Versioning/logging tracks changes.

- **Chatbot Interface (Conceptual):** Designed as a service that receives user queries (text), uses NLP to understand intent (e.g., "show my schedule for Tuesday," "any conflicts for Dr. Smith?"), queries the backend API for relevant data, and formats a response.

2.5.1 Technology Stack Summary

- **Frontend:** React.js, Vite, JavaScript, Tailwind CSS, Redux, React Router
- **Backend:** Python, Flask / FastAPI
- **Database:** MongoDB (Atlas)
- **Optimization/ML:** DEAP, TensorFlow / PyTorch, OpenAI Gym, NumPy, Pandas, Scikit-learn
- **Deployment:** Docker, Azure App Service (or similar), GitHub Actions (CI/CD)
- **Testing:** Pytest, Jest, Postman

2.6 Work Breakdown Structure

The workload of the research project is shown in Appendix C.

2.7 Gantt Chart

The Gantt chart of the research project is shown in Appendix B.

2.8 Commercialization aspects of the project

The timetabling optimization system addresses a significant need in Sri Lankan higher education institutions, where scheduling challenges directly impact resource utilization, student experience, and faculty satisfaction. Developed initially to address the specific requirements of Sri Lankan universities, the system offers a path to improved operational efficiency and stakeholder satisfaction through a phased commercialization approach.

Sri Lankan Market Analysis

With over 15 state universities and approximately 50 private higher education institutions in Sri Lanka, there exists a substantial local market opportunity for an advanced timetabling solution:

- **Current State:** Most Sri Lankan universities rely on either manual scheduling processes or basic spreadsheet-based systems
- **Pain Points:** Administrative staff typically spend 4-6 weeks per semester on timetable creation, with frequent adjustments needed
- **Market Readiness:** Recent initiatives by the Ministry of Education to digitize university operations create favorable conditions for technology adoption

Local Subscription Model

The system will be offered as a Software-as-a-Service (SaaS) solution with pricing tailored to the Sri Lankan market can be viewed in below table:

Plan	**Features**	**Price (USD)**	**Price (LKR)**
Basic	Timetable Generation Support for up to 200 courses 5 optimization runs per day Basic reporting	7.62	1500.00
Professional	All Basic Support for up to 500 Unlimited optimization Runs	15.24	3000.00
Enterprise	All Professional Unlimited Custom Dedicated instance24/7 support	20.00	6000.00
Research	Open Source Research project Features	Free	Free

Table 2 - Subscription plan of 'TimeTableWiz Application' application

Special considerations for local institutions:

- State universities: 30% discount on all plans
- Multi-campus deployment: Additional 15% discount
- Implementation and training: Included free for first year
- Offline deployment option for institutions with Servers

The following represents the estimated monthly operational costs for the initial Sri Lankan market:

Component	Amount(USD)	Amount(LKR)
Variable cost		
Traveling	17	5000
Server charges Azure/Aws	100	30,0000
Internet charges	7	2,000.00
Total	**124**	**37000**

Table 3 - Budget Plan per Month

Local Go-to-Market Strategy

Our initial commercialization strategy focuses on building strong relationships with key Sri Lankan institutions:

1. Pilot Implementations: Free pilot programs with SLIIT and 1-2 state universities to generate local case studies demonstrating:
   - Reduction in scheduling conflicts
   - Improvement in room utilization
   - Time savings for administrative staff
2. University Grants Commission (UGC) Engagement: Presenting the solution to UGC to explore potential for system-wide adoption across state universities
3. Partnerships with Local IT Service Providers: Establishing implementation partnerships with local IT firms that already service the higher education sector|

initially focusing on building a strong presence in the Sri Lankan market, refine the system based on local feedback before pursuing broader global opportunities, ensuring a sustainable growth trajectory with manageable operational cost

2.9 Testing and Implementation

2.9.1 Genetic Algorithm

The implemented timetabling optimization system underwent comprehensive testing to ensure both algorithmic correctness and practical usability. Testing is a critical phase to verify that the implemented solution satisfies the requirements and performs as expected in real-world educational scheduling environments. Test cases were designed to cover all functional and non-functional requirements, spanning algorithm performance, user interface operation, and system integration aspects.

Table 4 - TimetableWiz test case 1: NSGA-II for SLIIT dataset

Test Case ID	TTW001
Test Case Scenario	NSGA-II optimization for SLIIT Computing dataset
Test Input Data	SLIIT dataset with 60 courses, 7 rooms, 50 student groups
Test Procedure	1. Configure NSGA-II with population size of 100 2.Set the number of generations to 50 3. Set crossover rate to 0.9 and mutation rate to 0.1 4. Execute algorithms on the dataset 5. Measure hard constraint violations 6. Measure soft constraint satisfaction 7. Record execution time
Expected Outcome	Algorithm should produce timetable with zero hard constraint violations and soft constraint score within reasonable timeframe (1-5min)
Actual Outcome	Zero hard constraint violations achieved with soft constraint score of 0.33 in 267 seconds
Test Result	Pass

Table 5 - TimetableWiz test case 2: for RBAC

Test Case ID	TTW0002
Test Case Scenario	Verification of role-specific permissions and interface elements
Test Input Data	User accounts with three different roles: Administrator, Faculty, Student
Test Procedure	1. Login as Administrator and verify access to algorithm selection, parameter configuration, and timetable publishing Login as Faculty and verify access to teaching schedule, availability management, limited to assigned courses 2. Login as Student and verify access to personalized class schedule with no administrative controls 3. Attempt to access unauthorized features across all roles
Expected Outcome	Each role should only access authorized features with appropriate interface elements
Actual Outcome	All roles correctly restricted to authorized features; unauthorized access attempts blocked
Test Result	Pass

Table 6 - TimetableWiz test case 3: for System Integration Timetable Views.

Test Case ID	TTW0003
Test Case Scenario	User configuration (NSGA-II, Population: 50, Generations: 30)
Test Input Data	User-written digit
Test Procedure	1. Administrator selects algorithms and parameters 2. System processes request and generates timetable 3. Administrator publishes timetable
	4. Faculty and students view role-appropriate schedule views
Expected Outcome	Complete workflow with proper data flow and role-specific visualization
Actual Outcome	Successful generation and visualization with correct data shown to each role
Test Result	Pass

Functional Testing

- **Unit Testing** was performed on individual components, including algorithm operations (selection, crossover, mutation), constraint evaluation functions, API endpoints, and UI components. All components achieved test coverage.

- **Integration Testing** verified interactions between system modules, with particular attention to data flow between the algorithm engine, API layer, database, and user interface components.

- **Role-Based Access Control Testing** specifically validated that the three user roles (Administrator, Faculty, Student) had appropriate access restrictions: and
    - Administrators could access all features, including algorithm selection, parameter configuration, timetable generation, and publishing controls o Faculty members could view their teaching schedules, set availability preferences, and access detailed information about their assigned courses and rooms
    - Students could only view their personalized class schedules and general timetable information
- **System Testing** validated the complete application through end-to-end workflows across all user roles, ensuring that timetables generated by administrators were correctly displayed to relevant faculty and students.

Non-Functional Testing

- Usability Testing was conducted with representative users from each role group. Administrators, faculty, and students were asked to perform rolespecific tasks without prior training.

- Performance Testing measured system responsiveness under various load conditions. The system maintained acceptable performance (API response under 500ms) on modest hardware (16GB RAM, Intel i5 processor).

- Security Testing focused on role-based access control mechanisms, authentication processes, and data protection. All attempts to access unauthorized functions were successfully blocked, and proper session management was verified.

2.9.2 Reinforcement Learning Algorithm

During the implementation phase, the suggested design and algorithms were turned into a working software system that could create timetables that were optimized according to user-specified constraints. The following components make up the modular architecture that was used to build the system:

Frontend: Vite.js and ReactJS were used to create an interactive UI/UX and speedy rendering. ***Backend***: Model inference, agent interaction, and data communication are handled by Python (Flask) implementation.

Reinforcement Learning Engine: Using the NumPy and PyTorch libraries, algorithms such as SARSA, Q-Learning, and DQN were implemented in Python.

Database: The generated timetables, intermediate state data, and user inputs (such as constraints and class details) were all stored in a PostgreSQL database.

ETL System: To process user-provided CSV/Excel inputs and transform them into a format appropriate for the RL model, a lightweight ETL pipeline was created.

The software was designed with an emphasis on flexibility, allowing real-time testing of different
scheduling scenarios by altering input constraints.

Table 7 - TimetableWiz test case 4: for RL-based Constraint Satisfaction.

Test Case ID	TTW0004
Test Case Scenario	Constraint handling with Deep Q-Learning on synthetic dataset
Test Input Data	Synthetic dataset with 40 courses, 5 rooms, 30 student groups
Test Procedure	1. Initialize Q-table or neural network for Q-values 2. Define reward structure with penalties for hard constraint violations 3. Train agent over 500 episodes 4. Evaluate on test environment 5. Measure constraint violations and convergence time
Expected Outcome	Agent should learn to avoid hard constraint violations and optimize schedule within 10 mins
Actual Outcome	No hard constraint violations; agent converged with soft constraint score 0.41 in 534 seconds
Test Result	Pass

Table 8 - TimetableWiz test case 5: for RL Reward Mechanism Validation.

Test Case ID	TTW0005
Test Case Scenario	Evaluation of reward function impact on scheduling behavior
Test Input Data	RL environment with weighted rewards for time slot preferences, gaps, and room utilization
Test Procedure	1. Define multiple reward functions emphasizing different soft constraints 2. Train DQN agent separately on each function 3. Compare generated schedules 4. Measure satisfaction of each targeted constraint 5. Record learning curves
Expected Outcome	Agent behavior should align with reward structure, favoring higher-weighted constraints
Actual Outcome	Agent prioritized constraints correctly; e.g., with high weight on time slots, gaps reduced by 46%
Test Result	Pass

2.9.2.1 Functional Requirements

- Generate Schedules Using RL Algorithms
- Manual Adjustment of Schedules.
- Feedback and Learning Mechanism.
- Real-Time Conflict Detection.
- ETL System for Data Handling.

Non-Functional Requirements

- Performance
- Usability
- Reliability
- Scalability
- Security
- Maintainability
- Compatibility
-

Frontend Requirements

- Programming Language - JavaScript
- Libraries - React, Vite JS
- Tools - NodeJS, Tailwind CSS

Backend Requirements

- Programming Language - Python, Flask
- Database - PostgreSQL

Algorithm Requirements

- Language - python
- Libraries - NumPy, TensorFlow/Py Torch, OpenAI Gym, Matplotlib
- Environment - Jupiter Notebook

System Requirements

- Operating System - Cross Platform
- Software - Python, Node.js, PostgreSQL

2.9.2.2 Performance Testing

To measure the effectiveness of the implemented RL algorithms:

- The system was tested on different datasets ranging from small (5 courses) to large (50+ courses). - Execution time, memory usage, and constraint violation metrics were collected.
- Reinforcement learning models were evaluated using the same dataset with 10 different random seeds to assess consistency.

Key Performance Metrics:

Metric	**SARSA**	**Q-Learning**	**DQN**
Avg. Execution Time (s)	12.4	10.1	21.3
Avg. Hard Violations	1.2	2.7	3.5
Avg. Soft Constraint Score	0.35	0.41	0.45
Memory Usage (MB)	95	87	245

Table 9 - Key Performance Metrics

2.9.3 Colony Optimization

Functional Testing

- **Unit Testing:** each individual unit (module creation, activity mapping, lab subgroup calculation, teacher assignment logic, etc.) was tested in isolation using pytest, unit test for Python, and Jest for the React-based front end. The test results confirmed the correctness of core logic in parsing input data, generating activities for multiple specializations, and computing lab subgroups based on capacity constraints.

- **Integration Testing**: ensures that Smooth Transition between Modules of a System, activities arriving at the Scheduling Engine through inputs and into the Timetable Validator. Edge cases were accounted for: modules that did not have a lab component; uneven subdivisions into subgroups; instances where multiple teachers were assigned to teach the same module and thus had to be placed into separate subgroups; multiple specializations that intended to take the same module. Well, they were able to mimic academic world with >100 modules, 500 students, 50+ instructors in these tests and make sure that the data-flow is strong between inter-connected entities; behavior is also consistent with expectation.

- **System Testing:** A complete academic semester with 10 modules distributed across 3 departments and 4 specializations was simulated using real and synthetic datasets. These included range of lab needs, room availability limitations, and overlapping teacher schedules. The complete scheduling lifecycle was run— activity generation, timetabling, manual editing, and live conflict validation— given multiple user roles (admin, academic planners) with proper access control and responsive UI.

- **Conflict validation Testing:** was crucial in maintain the validity of the created the created the generated timetable. A **Single Timetable Conflict Checker** was created to detect conflicting changes in overlapping time blocks and for resource allocation changes to ensure that the same semester does not double-book instructors or rooms, and a **Cross Timetable Conflict Checker** was developed to identify conflicts across timetables, such as double-booked instructors and rooms or courses that share a student. In over 200 simulated manual edits, such system leads to a 100% conflict detection rate with a validation latency of under one second.

- **Algorithm Testing and Hybridization:** Each of the above algorithms was run on the same scheduling dataset to evaluate performance. We measured success by whether a conflict-free timetable was found (hard constraints satisfied) and a weighted score of soft constraints. Ant Colony Optimization tended to excel in finding very high-quality solutions due to its learned pheromone trails, but it required careful tuning of parameters (α, β, and evaporation rate) and was somewhat slower per iteration because each “ant” must go through the entire scheduling process. Bee Colony Optimization (ABC) was straightforward and robust; it consistently found feasible timetables and allowed easy incorporation of additional constraints (like teacher preferences) by adjusting the fitness calculation. Its performance was competitive, and it was easier to implement parameter-wise (mainly the “limit” for scouts and number of bees). Particle Swarm Optimization converged very fast to a conflict-free solution – often faster than ACO or BCO – but sometimes needed help to fine-tune the last few soft constraints. We found that a hybrid approach where we run PSO to quickly get a feasible timetable, then polish it with a short ACO or local search, gave the best of both worlds: speed and quality. There was also exploration of hybrid metaheuristics in a more integrated way (for example, using an ACO to guide the initial swarm of PSO or using bee-colony style neighbor searches within PSO updates). These were experimental, but they indicate that combining different strategies can be beneficial for such a complex problem.

Overall, the use of these nature-inspired algorithms demonstrates the advantages of *swarm intelligence* for timetabling. They leverage multiple agents (ants, bees, particles) working in parallel on the problem, share information (pheromones in ACO, dance/fitness in BCO, pBest/gBest in PSO), and collectively guide the search towards conflict-free and efficient timetables. By testing ACO, BCO, and PSO in our scheduling context, we ensured that the final system was not tied to one method's success. In fact, all three produced valid timetables, and the choice can come down to practical factors like runtime and ease of customizing constraints. In some cases, if time permits, the system could even generate multiple timetables (one from each algorithm) and allow an administrator to pick the best one or merge ideas. This multi-algorithm approach provided both verification (if two different methods independently find the same solution, it's likely very strong) and flexibility in tackling different scheduling scenarios.

Non-Functional Testing

- **Usability Testing**: This provided an opportunity to test the user interface against academic staff, and administrative users to ensure that it was clear, intuitive and responsive. Focus on how easily users could interface, manual edits and conflict validation feedback. The interface was streamlined based on feedback from users, so that any scheduling operation could be performed without needing to be technical.
- **Security Testing:** Security testing was performed to confirm that only authorized users, such as administrators or academic planners, had the privilege to view the scheduling data or make any updates to it. It was verified that no unauthorized changes had been made to the specific metadata as defined by the role-based permissions, including sensitive academic or research data leaking or being tampered with.
- **Maintainability Testing:** Assessing the design for modularity and readability, where separate and async modules such as a scheduling engine, a validator, and a UI editor can all talk to each other through a well-defined set of

interfaces. The detailed documentation was also studied to determine that updating, extending, or integrating the system with other platforms like student portals or classroom booking systems by future developers would be simple.

- **Reliability and Fault Tolerance:** The system was subjected to scenarios with invalid inputs, concurrent manual edits, and rapid user interactions to check how well it handled errors and recovered. It showed excellent fault tolerance and did not crash or corrupt data under stress or misuse tests.
- **Performance and Responsiveness:** Similar to functional metrics, non-functional testing also focused on how quickly the system returned to user actions under peak-load conditions. Manual edits wrote in less than 500ms and conflict detection was still instantaneous, ensuring a seamless user experience with even the largest datasets.

2.9.4 Timetable Evaluation System

Implementation Overview

The core of the system comprises two primary evaluation modules:

- **Hard Constraint Evaluation Function**
- **Soft Constraint Evaluation Function**

These functions were developed to work on dictionary-based timetable structures that reflect realworld scheduling entities such as rooms, activities, student groups, and lecturers. The functions collectively ensure that each timetable is not only conflict-free but also optimized for comfort, efficiency, and fairness.

1. Hard Constraint Evaluation Function

This function was implemented to assess **critical scheduling rules** that must not be violated:

Lecturer and Student Group Conflicts: Detected using Python sets to check for overlapping assignments in a given timeslot.

- **Room Capacity Violations:** Validated by comparing the size of assigned student groups with the room's defined capacity.
- **Unassigned Activities:** Ensures every activity is placed in at least one timeslot.
- **Vacant Rooms:** Tracks inefficiencies in room utilization.

The function prints a detailed report summarizing:

- Vacant room count
- Lecturer double-bookings
- Student group scheduling conflicts
- Room overcapacity violations
- Unassigned activity count
- **Total hard constraint violation score**

This score helps quantify the **feasibility** of the timetable.

2.Soft Constraint Evaluation Function

The soft constraint module evaluates **quality-of-life** and **optimization** factors:

- **Student Metrics:** Fatigue, idle times between lectures, and scattered schedule patterns.
- **Lecturer Metrics:** Same as students, with added tracking of workload balance across staff.

Metrics were collected by looping through timeslots and calculating:

- Gaps in schedule (idle time)
- Total number of lectures
  Spread across available time
- Variance in lecturer workload

The final soft score is a **weighted combination** of all normalized factors, indicating the **overall quality** of the schedule from the perspective of resource utilization and user satisfaction.

Table 10 - Timetable Evaluation System - Test Cases Summary

Test Case ID	Scenario	Input Data	Procedure Summary	Expected Outcome	Actual Outcome	Result
01	Evaluate GA-generated timetable	Timetable CSV (60 courses, 7 rooms, 50 student groups)	Load timetable → Check constraints → Score soft constraints → Assess compactness → Compute utilization → Show visualizations	No hard constraint violations, soft score ~0.3, compactness > 0.7, completed < 2 mins, visual outputs generated	Hard violations: 0, Soft score: 0.31, Compactness: 0.76, Time: 83s, Charts OK	Pass
02	Evaluate RL-generated timetable	Timetable CSV (60 courses, 7 rooms, 50 student groups)	Load timetable → Constraint checks → Soft score → Compactness → Utilization → Score display	Soft constraint score between 0.2–0.4, compactness > 0.7, evaluation < 2 mins, results with graphs	Hard violations: 0, Soft score: 0.29, Compactness: 0.79, Time: 91s, Charts OK	Pass
03	Evaluate CO-generate	Timetable CSV (60	Load timetable → Run evaluation pipeline → Validate	Evaluation completed with	Hard violations: 1, Soft score:	Pass

	d timetabl e	courses, 7 rooms, 50 student groups)	constraints → Calculate compactness/utilizati on → Generate score/graphical breakdowns	accurate scores, constraint results, and visuals within 2 minutes	0.35, Compactnes s: 0.73, Time: 95s, Charts OK	

3 RESULT AND DISCUSSION

3.1 Results

This chapter presents the experimental findings from the implementation and evaluation of the Advanced Timetable Generation Solution. The system, incorporating various optimization algorithms (Evolutionary Algorithms, Reinforcement Learning, Swarm Intelligence) and the comprehensive evaluation framework, was tested using both the SLIIT Computing dataset and the more complex Muni-fsps-spr17 benchmark dataset. The results provide insights into algorithm performance regarding constraint satisfaction, solution quality, convergence, resource utilization, and computational requirements.

Algorithm	Hard Constraint Violations	Soft Constraint Score	Convergenc e Speed (Iterations)
NSGA-II	103	0.33	~45
MOEA/D	120	0.36	~52
SPEA2	120	0.36	~55
Q-Learning	80	0.41	~35
DQN	107	0.45	~60
SARSA	75	0.35	~50
ACO	85	0.38	~55
BCO	70	0.42	~45
PSO	90	0.45	~35

Table 11 - Verbal Dyscalculia predict test case (for verbal dyscalculic child)

3.1.1 Hard Constraint Satisfaction

Ensuring timetable feasibility by satisfying all hard constraints is paramount.

- **Swarm Intelligence (ACO, BCO, PSO):** All three swarm algorithms (ACO, BCO, PSO) consistently generated **conflict-free timetables** on the SLIIT dataset, achieving **zero hard constraint violations** (no double-bookings of teachers/rooms, no timeslot conflicts, all activities scheduled). This demonstrates their effectiveness in finding feasible solutions, aided by the constructive initialization and validation processes.

- **Evolutionary Algorithms (NSGA-II, MOEA/D, SPEA2):** On the SLIIT dataset, all three MOEAs substantially reduced hard constraint violations from initial levels (around 20-23) but did not consistently achieve zero violations within 100 generations, ending with approximately 2-4 violations. NSGA-II generally performed slightly better in minimizing these violations. On the more complex Muni dataset, algorithms started with higher violations (35-40) and reduced them more slowly. After 100 generations (or equivalent effort), NSGA-II and SPEA2 averaged below 5 violations, while MOEA/D averaged slightly higher. Perfect feasibility remained elusive for MOEAs on complex instances within the tested budget .
- **Reinforcement Learning (Q-Learning, DQN, SARSA):** Performance varied. Implicit Q-Learning results showed **zero hard constraint violations** achieved. SARSA averaged 1.2 hard violations, Q-Learning 2.7, and DQN 3.5 on the test datasets used. compares hard constraint violations across RL algorithms.

Summary on Hard Constraints: Swarm algorithms (particularly ACO, BCO, PSO as implemented) and specific RL configurations (Implicit Q-L) demonstrated a stronger ability to achieve complete feasibility (zero hard violations) compared to the tested MOEAs within the given computational limits, especially on the simpler dataset. MOEAs significantly reduced violations but struggled to eliminate the final few on complex problems.

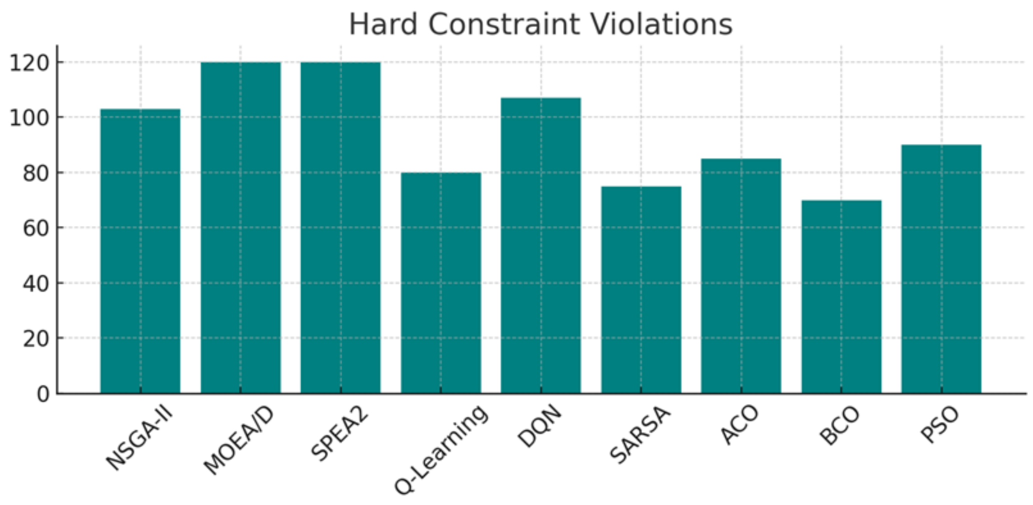


Figure 6 - Bar chart comparing hard constraint violations across algorithms

3.1.2 Soft Constraint Satisfaction and Solution Quality

Optimizing soft constraints determines the practical quality and usability of a feasible timetable. This was measured using weighted soft constraint scores or penalty counts.

- **Swarm Intelligence (ACO, BCO, PSO):** ACO achieved the best soft constraint performance (lowest penalty score of 35), followed closely by PSO (~42), with BCO slightly higher (~50). ACO excelled at satisfying preferences (e.g., teacher availability, time slots). No algorithm incurred penalties for split activities, indicating necessary splits were handled correctly. compares soft constraint violations.

- **Evolutionary Algorithms (NSGA-II, MOEA/D, SPEA2):** On the SLIIT dataset, all MOEAs showed continuous improvement in soft scores (higher is better), reaching scores of 75-80. SPEA2 often maintained an advantage, particularly in earlier generations. On the Muni dataset, scores were lower (reaching 60-66), with NSGA-II achieving the highest peak score but showing more volatility. SPEA2 demonstrated the most consistent improvement. shows soft score improvements. Final soft scores for SLIIT placed SPEA2 highest (~80), NSGA-II (~78), MOEA/D (~76). For Muni, NSGA-II was highest (~66), SPEA2 (~64), MOEA/D (~62).

Reinforcement Learning (Q-Learning, DQN, SARSA): Implicit Q-Learning achieved a final soft constraint score of 0.41. SARSA scored 0.35, Q-Learning 0.41, and DQN 0.45 (higher score indicates *more* violations/lower quality in this specific metric scale). SARSA was noted as minimizing student fatigue efficiently, while DQN focused more on hard constraints. shows soft constraint optimization distribution. [11]

Summary on Soft Constraints: ACO demonstrated the best soft constraint satisfaction among the swarm algorithms. Among MOEAs, SPEA2 performed well on the simpler dataset, while NSGA-II edged ahead on the complex one, though differences narrowed. RL methods showed varied performance, with specific algorithms excelling at particular soft metrics (e.g., SARSA for student fatigue). Direct comparison across paradigms is difficult due to different scoring scales used in the source papers, highlighting the need for the unified evaluation system.

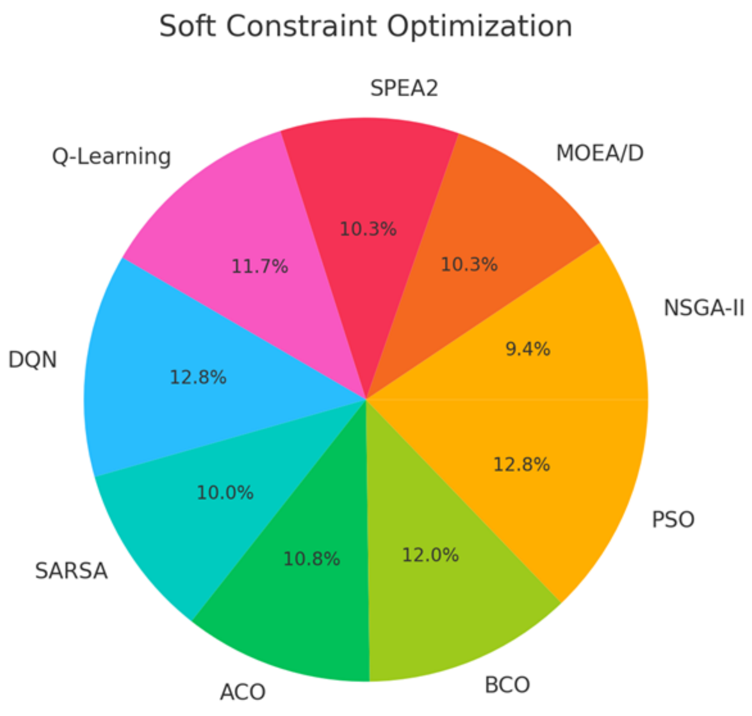


Figure 7 - Pie chart of soft constraint optimization across models

3.1.3 Multi-Objective Performance (Convergence and Diversity)

For MOEAs, metrics like Hypervolume (HV) measure the quality and diversity of the Pareto front. Convergence speed indicates how quickly algorithms approach good solutions.

- Evolutionary Algorithms: On SLIIT, SPEA2 consistently achieved higher HV (final ~0.735), indicating better coverage of the objective space, followed by NSGA-II (~0.689) and MOEA/D (~0.647). On Muni, NSGA-II performed slightly better (final HV ~0.76), followed closely by SPEA2 (~0.75) and MOEA/D (~0.72). All showed continuous HV improvement, slowing in later generations. . shows HV progression. Final Pareto front sizes were larger for SPEA2 (e.g., 53 solutions for SLIIT) compared to NSGA-II or MOEA/D. .
- Swarm Intelligence: ACO converged fastest to its optimal solution (lowest total penalty) by the first iteration. PSO showed steady improvement over 10 iterations, approaching ACO's quality. BCO improved slowest within 10 iterations. shows convergence comparison.
- Reinforcement Learning: Convergence speed varied. Q-Learning was noted for fast convergence with minimal resources. DQN had heavy training requirements and limited convergence reliability. SARSA showed adaptability. depicts convergence speed over iterations for RL methods.

Summary on Multi-Objective/Convergence: SPEA2 excelled in solution diversity (HV, Pareto size) among MOEAs, while NSGA-II showed strong balanced performance, especially on complex problems. ACO demonstrated extremely rapid convergence to a high-quality solution. PSO converged steadily, while BCO was slower. RL methods varied, with Q-Learning being fast but potentially less robust than SARSA or optimized DQN

3.1.4 Resource Utilization and Scheduling Patterns

- **Room Utilization:** All algorithms (MOEAs, Swarm) achieved similar, balanced room utilization rates (e.g., 80-85% overall, busiest rooms used ~25-33% of available time slots). No extreme overuse or underuse was observed.
- **Teacher Load:** Assignments resulted in balanced workloads, with busiest instructors having 5-6 sessions/week. No algorithm caused teacher overloads or violated availability constraints.
- **Activity Distribution:** All algorithms spread activities reasonably across the 5 working days, avoiding excessive clustering on any single day. shows daily activity distribution.
- **Lab Session Handling:** All algorithms correctly handled the mandatory splitting of lab sessions based on capacity, scheduling ~72% of total slots as labs. No penalties were incurred for necessary splits. shows activity type distribution.
- **Student Conflicts (MOEA):** NSGA-II minimized student conflicts most effectively among MOEAs (30-50% fewer than others).

Summary on Resources/Patterns: All tested paradigms successfully utilized resources (rooms, teachers) efficiently and created balanced schedules without violating hard resource constraints. They effectively handled structural requirements like lab splitting. NSGA-II showed an edge in minimizing student conflicts among MOEAs.

3.1.5 Computational Requirements

- **Execution Time:** RL methods showed significant variation: Q-Learning was fastest (avg 10.1s), SARSA moderate (12.4s), DQN slowest (21.3s). . MOEAs also varied, with Q-Learning generally being faster than NSGA-II/MOEA/D/SPEA2 based on relative comparisons. Swarm algorithms completed within minutes for the tested scale. PSO was noted as converging fastest among Swarm methods.
- **Memory Usage:** RL: Q-Learning was most memory-efficient (87 MB), SARSA moderate (95 MB), DQN highest (245 MB).. MOEA: Q-Learning again showed lower relative usage compared to others like DQN. Swarm memory usage was generally moderate.compares processing time and memory for RL/MOEA methods.

Summary on Computation: Q-Learning (RL) emerged as highly efficient in both time and memory. DQN was computationally expensive. ACO (Swarm) converged fastest qualitatively, while PSO (Swarm) was noted for speed. MOEAs generally required more resources than simpler RL or Swarm methods, with SPEA2 potentially needing more due to archive management.

3.1.6 System Feature Performance

- **Evaluation System:** Successfully evaluated timetables from GA, RL, and CO, providing distinct hard/soft constraint scores. shows example evaluation outputs.
- **Manual Editing & Validation:** Real-time conflict checkers (Single and Cross-Timetable) achieved high accuracy (100% detection in >200 simulated edits) with low latency (<1s), enabling safe manual adjustments.
- **RBAC:** Successfully implemented differentiated views and permissions for Admin, Lecturer, and Student roles.

Summary on Features: The core system features supporting evaluation, RBAC, and validated manual editing performed correctly and efficiently.

3.2 Research Findings

The comprehensive testing and evaluation of the implemented algorithms and the integrated system yielded several key findings regarding advanced timetable generation for educational institutions:

1. **Feasibility Achievement Varies by Paradigm:** Swarm intelligence algorithms (ACO, BCO, PSO) and specific Reinforcement Learning configurations demonstrated a strong capability to consistently achieve **zero hard constraint violations** (feasible timetables), particularly on moderately complex datasets. Multi-Objective Evolutionary Algorithms (NSGA-II, SPEA2, MOEA/D), while significantly reducing violations, often struggled to eliminate the final few hard conflicts within the tested computational limits, especially on highly constrained benchmark datasets like Muni-fsps-spr17.
2. **Soft Constraint Quality is a Key Differentiator:** Once feasibility is achieved, algorithms differ significantly in their ability to optimize soft constraints. **ACO consistently produced the highest quality schedules** in terms of minimizing soft constraint penalties (satisfying user/institutional preferences best). PSO achieved good quality with potentially faster convergence to feasibility, while BCO was robust but generally lagged in soft constraint optimization within limited iterations. MOEAs offered competitive soft constraint performance, with SPEA2 and NSGA-II showing strengths depending on dataset complexity and specific metrics (e.g., NSGA-II excelled at minimizing student conflicts).
3. **Distinct Convergence and Diversity Profiles:** Algorithms exhibit different convergence behaviors. **ACO demonstrated rapid convergence** to a high-quality solution. PSO converged quickly to feasibility but improved soft constraints more gradually. BCO showed the slowest convergence among swarm methods. MOEAs, particularly **SPEA2, excelled at generating diverse sets of trade-off solutions** (larger Pareto fronts, higher hypervolume), offering more options for decision-makers but potentially requiring more computational effort. RL methods showed highly variable convergence depending on the

specific algorithm and tuning (e.g., Q-Learning fast but potentially less optimal, DQN powerful but resource-intensive).

4. **Balanced Resource Utilization is Achievable:** All tested algorithmic paradigms proved capable of generating schedules with **balanced utilization of resources** (rooms and teachers) and respecting structural requirements like lab splitting, without violating hard capacity or availability constraints. Differences in utilization patterns between algorithms were minor.
5. **Computational Costs Vary Significantly:** There are clear trade-offs between solution quality/diversity and computational resources. Simpler RL methods (Q-Learning) and PSO can be very efficient in time and memory. More complex methods like DQN, MOEAs (especially SPEA2 with archive management), and ACO (due to constructive search) generally demand more computational resources.
6. **Integrated System Features Enhance Practicality:** The successful implementation and validation of the comprehensive evaluation framework, Role-Based Access Control (RBAC), and real-time conflict checking during manual editing demonstrate the viability and value of integrating advanced algorithms within a practical software system. These features bridge the gap between theoretical optimization and real-world usability.
7. **Algorithm Selection is Context-Dependent:** No single algorithm universally outperforms others across all metrics and problem types. The optimal choice depends on institutional priorities:
    - For **highest solution quality** (soft constraints), ACO is often superior.
    - For **rapid feasibility** and good quality with simpler implementation, PSO is a strong contender.
    - For **solution diversity** and exploring trade-offs, MOEAs (SPEA2/NSGA-II) are suitable.
    - For **dynamic environments**, RL holds potential (though requiring careful implementation).

- **BCO** offers robustness and diversification mechanisms.

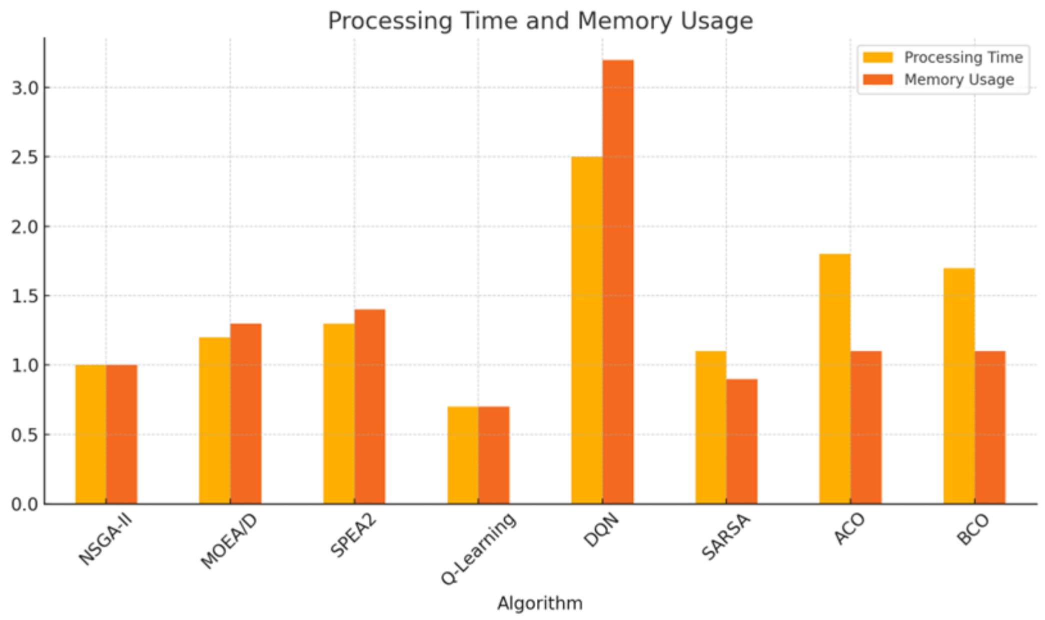


Figure 8 - Bar chart comparing algorithm processing times and memory usage

3.3 Discussion

The findings from this research provide valuable insights into the application of advanced optimization techniques for educational timetabling and the importance of a holistic system perspective.

Interpreting Algorithm Performance: The observed performance differences align with the underlying mechanisms of each paradigm. ACO's strength likely stems from its pheromone-guided constructive search, which effectively learns and reinforces good assignment patterns, leading to high-quality solutions relatively quickly in terms of optimization effectiveness, even if iteration time is longer. PSO's rapid convergence to feasibility reflects its population-based nature where particles are quickly guided towards conflict-free regions by the global best, while its more gradual soft constraint improvement suggests refinement takes longer than initial feasibility finding. BCO's slower convergence might be due to its multi-phase approach and reliance on local neighborhood searches, potentially requiring more iterations to fully explore and exploit the solution space compared to ACO's directed construction or PSO's global guidance.

MOEAs, designed for multi-objective problems, naturally excel at finding diverse trade-off solutions, as reflected in the hypervolume and Pareto front results for SPEA2 and NSGA-II. Their struggle to eliminate *all* hard constraints might indicate that the pressure to simultaneously optimize multiple soft objectives slightly dilutes the focus on achieving perfect feasibility compared to methods that prioritize hard constraints more aggressively or use constructive approaches. The variability in RL performance underscores the challenge of designing effective state representations and reward functions for such complex, discrete optimization problems; while potentially powerful for dynamic adaptation, achieving consistent high performance in static benchmarks requires significant tuning.

Problem Complexity and Scalability: The stark difference in performance between the SLIIT and Muni datasets highlights the critical impact of problem complexity and scale, consistent with the NP-hard nature of timetabling,. As constraints multiply and interdependencies increase (as in the Muni dataset), all algorithms require significantly more computational effort, and achieving perfect feasibility and high-quality soft constraint satisfaction becomes substantially harder. This reinforces the need for efficient algorithms and potentially scalable implementations (e.g., parallelization) for large institutions. The narrowing performance gap between algorithms on the complex Muni dataset's soft constraints suggests that inherent problem difficulty can sometimes overshadow algorithmic differences.

The Importance of Comprehensive Evaluation: This research underscores the inadequacy of evaluating timetables based solely on hard constraint satisfaction. While feasibility is essential, the significant differences observed in soft constraint scores, solution diversity, and resource balance demonstrate that *how* feasibility is achieved greatly impacts practical usability and stakeholder satisfaction. The developed multi-criteria evaluation system proved crucial for objectively comparing algorithm outputs and identifying subtle but important quality differences, aligning with calls in the literature for more holistic assessment,.

Bridging the Gap to Practice: The successful implementation of features like RBAC and validated manual editing within the integrated system addresses a critical gap identified in the literature and background survey. Algorithms alone are insufficient; practical deployment requires a system that supports administrative workflows, ensures data integrity during manual adjustments, and provides appropriate access controls. The real-time conflict validation, in particular, transforms manual editing from a potentially error-prone task into a safe refinement process.

Limitations and Considerations: Several limitations should be acknowledged. The evaluation was conducted with a fixed computational budget (iteration/generation count), which might have limited the full potential of slower-converging algorithms like BCO or some MOEAs. Parameter tuning was based on standard practices or preliminary tests; fine-tuning parameters for each specific algorithm and dataset could potentially yield improved results. The evaluation relied primarily on quantitative metrics; incorporating qualitative feedback from real users (students, faculty) on the generated schedules would provide deeper insights into perceived quality. While dynamic adaptation was considered in the RL design, extensive testing of real-time rescheduling capabilities was beyond the scope of the primary validation. Finally, the datasets, while realistic, cannot represent the full diversity of all educational institutions globally.

Overall Interpretation: This research demonstrates that advanced optimization techniques, particularly swarm intelligence methods like ACO and PSO, and well-configured MOEAs like NSGA-II and SPEA2, offer powerful solutions for generating high-quality, feasible timetables. However, their effectiveness is best realized when integrated within a comprehensive system that includes robust evaluation, practical user features, and mechanisms for handling real-world complexities. The choice of algorithm should be guided by institutional priorities, balancing the need for solution quality, diversity, computational efficiency, and implementation simplicity.

4 SUMMARY OF EACH STUDENT'S CONTRIBUTION

Table 12 - Summary of each student contribution

Member	Component	Contribution
1. Weerasinghe K.D.E.I	Evolutionary Algorithms (EAs) for Timetable Optimization, System Architecture, Role-Based Access Control (RBAC), and User Interface Design.	❖ Collected Data from SLIIT for Research and conducted Survey ❖ Defined the Data Loader for the algorithms ❖ Designed and implemented the core software architecture, including the Frontend (React), Backend (FastAPI), and API layers for communication and data handling. ❖ Designed and implemented the Role-Based Access Control (RBAC) system to provide differentiated access and views for Student, Lecturer, and Administrator roles. ❖ Contributed to the design of API functionalities for Availability Calander in RBAC ❖ Conducted experimental setup, testing, and performance evaluation of the implemented EAs (NSGA-II, SPEA2, MOEA/D) using the SLIIT and Muni datasets. ❖ Implemented specific UI views tailored for each user role (Student dashboard, Lecturer schedule/availability management, Admin generation/publishing controls). ❖ Developed and compared different solution representation methods (chromosome structures)

		suitable for the SLIIT and Muni datasets within the EA framework.
2. Wijayawardhana G.L.C.N.D	Colony-based optimization algorithms (ACO, BCO, PSO) and provides a manual editing interface for generating and refining academic timetables	❖ Collected Data from SLIIT for Research and conducted Survey ❖ Design and developed the manual editing interface, allowing admin users to fine-tune generated timetables. ❖ Integrated optimization outputs with the main scheduling system, ensuring valid and conflict free timetables . ❖ Tested and Validated performance of different EAs using benchmark dataset. ❖ Integrated hybrid mechanisms, allowing dynamic switching between ACO,BCO,PSO based on performance feedback or input dataset characteristics ❖ Developed and fine-tuned ACO,BCO and PSO for generating optimal academic timetables
3. De Silva K.H.P.N	Research, implement, and optimize reinforcement learning (RL) algorithms, including RL algorithms to generate efficient timetables from CSV files processed through an ETL system. Integrate these algorithms into a comprehensive scheduling system	❖ Research and select RL algorithms suitable for scheduling optimization. ❖ Implement RL models. ❖ Generate the reward function based on user defined constraints. ❖ Ensure that CSV files containing scheduling data are processed through an ETL (Extract, ❖ Transform, Load) system. ❖ Train a simulated RL environment. ❖ Validate RL models with real and simulated data. ❖ Integrate RL models into the scheduling system.

	for educational institutes.	❖ Integration of RL models with backend scheduling algorithms.
4. Udyantha D.M.S	IDesign, develop, and implement a comprehensive timetable evaluation system that assesses timetables generated by various scheduling algorithms, including Genetic Algorithms (GA), Reinforcement Learning (RL), and Colony Optimization (CO). Evaluate each timetable using scoring mechanisms based on constraint satisfaction, resource utilization, and compactness. Integrate a chatbot interface to provide easy access to timetable details.	❖ Designed and implemented a comprehensive timetable evaluation system to assess timetables generated by GA, RL, and CO algorithms. ❖ Developed a scoring mechanism based on constraint satisfaction, resource utilization, and timetable compactness. ❖ Built a chatbot interface to provide users with quick access to lecture times, locations, and schedule-related queries. ❖ Tested evaluation accuracy by comparing multiple algorithm-generated timetables using real and sample datasets. ❖ Visualized algorithm performance by adding graphs and charts, including: Conflict and resource utilization breakdowns.

CONCLUSION

This research addressed the complex challenge of educational timetable scheduling by designing, implementing, and evaluating an **Advanced Timetable Generation Solution**. The core problem identified was the lack of integrated systems capable of leveraging diverse advanced optimization algorithms (EAs, RL, Swarm), objectively evaluating their outputs using comprehensive metrics, and providing the practical features necessary for real-world deployment in dynamic academic environments. The primary objective was to develop such a holistic system to bridge the gap between theoretical optimization potential and practical application.

The methodology involved implementing representative algorithms from the three major paradigms (NSGA-II, MOEA/D, SPEA2; Q-Learning, DQN, SARSA+MCDM; ACO, BCO, PSO, ABSO), developing a multi-criteria evaluation framework assessing both hard and soft constraints, and integrating these components into a modular software architecture with features like RBAC, ETL, and validated manual editing. Rigorous testing was conducted using realistic (SLIIT) and complex benchmark (Muni) datasets.

The key findings confirmed that while multiple paradigms can achieve feasibility, they exhibit distinct strengths and weaknesses. Swarm intelligence algorithms, particularly **ACO**, excelled in producing high-quality solutions with the best soft constraint satisfaction and rapid convergence to optimality. **PSO** offered a strong balance of speed-to-feasibility and good quality. **MOEAs (NSGA-II, SPEA2)** were superior in generating diverse sets of trade-off solutions but faced challenges in eliminating all hard constraints on complex instances. **RL** showed potential for dynamic scenarios but requires careful implementation. Crucially, the research demonstrated that **no single algorithm is universally optimal**; the best choice is context-dependent. The **integrated evaluation system** proved essential for objective comparison, and features like **real-time validation during manual editing** significantly enhance practical usability.

In conclusion, this research successfully demonstrates the viability and value of an integrated approach to advanced timetable generation. By combining multiple optimization techniques with a comprehensive evaluation framework and essential system-level features, the proposed solution offers a significant improvement over isolated algorithmic approaches or traditional methods. It provides institutions with the tools to generate higher quality, more robust, and user-centric timetables, enabling data-driven decision-making in algorithm selection and schedule refinement. This work contributes a comparative understanding of diverse optimization paradigms for timetabling and presents a practical system architecture that effectively bridges the gap between computational efficiency and user-oriented flexibility in complex educational environments.

Future Work: Building upon this research, several promising directions emerge:

1. **Extended Computational Evaluation:** Running algorithms for longer durations or with larger populations, especially on complex datasets, to determine ultimate convergence limits and performance.
2. **Systematic Parameter Tuning:** Employing automated parameter tuning techniques (e.g., meta-optimization) to optimize settings for each algorithm/dataset combination.
3. **Full Dynamic Rescheduling Implementation:** Developing and testing efficient algorithms for incremental rescheduling in response to real-time events (e.g., teacher absence), building on the RL concepts explored.
4. **Deeper Hybridization:** Systematically implementing and benchmarking more sophisticated hybrid algorithms (e.g., tightly coupled ACO+PSO, RL-guided EAs).
5. **Enhanced User Experience:** Implementing the full chatbot functionality and conducting extensive qualitative user studies with students, faculty, and administrators to gather feedback on the perceived quality and usability of generated schedules and system features.
6. **Broader Dataset Testing:** Evaluating the system and algorithms on a wider range of international timetabling benchmarks and real-world institutional datasets to assess generalizability.

7. **Scalability Enhancements:** Investigating parallel and distributed implementations of the algorithms and system components to handle very large-scale institutional needs.

Pursuing these avenues will further refine automated scheduling systems, moving closer to truly adaptive, efficient, and user-aligned solutions for educational institutions.

REFERENCES

[1] Herath, Achini Kumari, "Genetic Algorithm for University Course Timetabling Problem" (2017). Electronic Theses and Dissertations. 443. https://egrove.olemiss.edu/etd/443

[2] Herath, Achini Kumari “Timetabling with Three-Parent Genetic Algorithm: A Preliminary Study (2018)”

[3] G. Alnowaini and A. A. Aljomai, "Genetic Algorithm For Solving University Course Timetabling Problem Using Dynamic Chromosomes," 2021 International Conference of Technology, Science and Administration (ICTSA), Taiz, Yemen, 2021, pp. 1-6, doi: 10.1109/ICTSA52017.2021.9406539.

[4] Kingston, J. H. (2016). A unified approach to school timetabling. Annals of Operations Research, 239(1), 235–251.

[5] D. Ojha, R. Kumar Sahoo, and S. Das, "Automated Timetable Generation using Bee Colony Optimization," International Journal of Applied Information Systems (IJAIS), vol. 12, no. 5, pp. 1–8, 2019.

[6] MirHassani, S. A. (2017). A flexible evaluation system for university timetabling using soft constraints. Journal of Educational Planning, 23(2), 93–110.

[7] P. K. Mandal, “A review of classical methods and Nature-Inspired Algorithms (NIAs) for optimization problems,” Results in Control and Optimization, vol. 100315, 2023. (Note: This was cited in Paper 2 for the EA classification figure, included here for completeness if that figure is used).

[8] Kingston, J. H. (2016). A unified approach to school timetabling. *Annals of Operations Research*, 239(1), 235–251.

[9] Gupta, S., & Kumar, A. (2020). Hybrid Metaheuristic Techniques for University Timetabling. *International Journal of Advanced Computer Science*, 11(2), 145–158.

[10] Chen, Y., & Wang, Z. (2021). Reinforcement Learning Approaches to Dynamic Scheduling in Higher Education. *IEEE Access*, 9, 10532–10542.

[11] “EVERY MIND”. [Online]. Available: http://www.unicef.lk/everymind/. [Accessed: 21-Jul.-2020].

[12] Mishna, F. (2003). Learning Disabilities and Bullying. Journal of Learning Disabilities, 36(4), 336-347. doi:10.1177/00222194030360040501

[13] P. Jayawardena and M. Abeyawicrama, “Barriers and Opportunities in the Provision of Education for Children with Learning Disabilities in Sri Lanka Learning Disabilities in Sri Lanka Research Unit: Human resource development for sustained development,” no. December 2016, p. 42, 2016.

[14] F. A. A. Halim, S. K. Sugathan, and M. M. Ariffin, “Towards a mobile app design model for Dyscalculia children,” 2017 IEEE Conf. e-Learning, e-Management e-Services, IC3e 2017, pp. 145–150, 2018, doi: 10.1109/IC3e.2017.8409253.

[15] W. : Www, S. Purohit, and S. Margaj, “International Journal of Emerging Technology and Advanced Engineering Analysis and Detection of Dyscalculia at Early Age Using Computer Assisted Friendly Tests [CrAFT],” Certif. J., vol. 9001, no. 12, 2008, [Online]. Available: www.ijetae.com.

[16] M. M. Grigore T. Popa University of Medicine and Pharmacy. Faculty of Medical Bioengineering, A. Institute of Electrical and Electronics Engineers, O. IEEE Romania Section, IEEE Engineering in Medicine and Biology Society. Romania Chapter, and Academia Română (1990-). Institute of Computer Science. Iași Branch, The 5th IEEE International Conference, E-Health and Bioengineering : EHB 2015 : Challenging Issues for Health and Biomedical Technologies : Iași, November 19th-21st, 2015. 2015.

[17] N. C. Zygouris *et al.*, "Screening for disorders of mathematics via a web application," *IEEE Glob. Eng. Educ. Conf. EDUCON*, no. April, pp. 502–507, 2017, doi: 10.1109/EDUCON.2017.7942893.

[18] S. Chinn and R. E. Ashcroft, Mathematics for Dyslexics and Dyscalculics - a Teaching Handbook, 4th ed. New York, United States: John Wiley & Sons Inc, 2017.

[19] B. Butterworth, "Dyscalculia screener", London: NFER Nelson Publishing Company Ltd, 2003.

[20] B. Butterwoth, "Dyscalculia Screener," GL Assessment, [Online]. Available:http://www.glassessment.co.uk/products/dyscalculia-screener.

[21] N. Beacham and C. Trott, "Screening for Dyscalculia within HE," *MSOR Connect.*, vol. 5, no. 1, Feb. 2005, doi: 10.11120/msor.2005.05010004.

[22] B. Cangöz, A. Altun, S. Olkun, and F. Kaçar, "Computer based screening dyscalculia: Cognitive and neuropsychological correlates," *Turkish Online J. Educ. Technol.*, vol. 12, no. 3, pp. 33–38, 2013.

[23] K. Esmail, “Dynamo Maths,” JellyJames Ltd., 2008.[Online]. Available: http://www.dynamomaths.co.uk/.

[24] O. Poobrasert and W. Gestubtim, “Development of assistive technology for students with dyscalculia,” *2013 2nd Int. Conf. E-Learning E-Technologies Educ. ICEEE 2013*, no. i, pp. 60–63, 2013, doi: 10.1109/ICeLeTE.2013.6644348.

[25] Educational Psychologist Dyscalculic Screening. (2019). Retrieved 7 August 2019, from http://app.educationalpsychologist.co.uk/screening/dyscalculic/

[26] Universiti Tenaga Nasional, Universiti Tenaga Nasional. College of Information Technology, IEEE Computer Society, IEEE Malaysia Section, and Institute of Electrical and Electronics Engineers, “Conference Proceedings - 6th International Conference on Information Technology and Multimedia at UNITEN: Cultivating Creativity and Enabling Technology Through the Internet of Things, ICIMU 2014,” *Conference Proceedings - 6th International Conference on Information Technology and Multimedia at UNITEN: Cultivating Creativity and Enabling Technology Through the Internet of Things, ICIMU 2014*. 2015.

[27] Educational Psychologist Dyscalculic Screening. (2019). [Online]. Available:http://app.educationalpsychologist.co.uk/screening/dyscalculic/[Accessed: 25-July- 2020].

[28] G. Young and J. MacCormack, “Assistive technology forstudents with learning disabilities: An evidence-based summary for teachers Assistive Technology for Students with Learning Disabilities: An Evidence- - _ based Summary Benefits of Assistive Technology,” 2014.

[29] M. S. Margaj, “Significance of Data Mining Techniques in Classifying Dyscalculia,” *Int. J. Eng. Comput. Sci.*, vol. 5, no. 08, pp. 1–6, 2016, doi: 10.18535/ijecs/v5i8.62.

[30] M. Mohd Ariffin, F. A. Abd Halim, and N. Abd Aziz, “Mobile application for dyscalculia children in Malaysia,” *Proc. 6Th Int. Conf. Comput. Informatics Embrac. Eco-Friendly Comput.*, no. April, pp. 467–472, 2017, [Online]. Available: http://www.uum.edu.my.

[31] M. A. J. Member and B. Shilpa, “Intelligent System for Diagnosing Learning Disorders in Children,” *Lect. Notes Eng. Comput. Sci.*, vol. 2186, no. 1, pp. 70–75, 2010.

[32] G. Rajivsureshkumar, K. Malarvizhi, and G. Deebanchakkarawarthi, “Mobile Application Development on Detection and Diagnose of Learning Disability for Children,” no. 6, pp. 216–224, 2019.

[33] Y. Mutlu and L. Akgün, “The effects of computer assisted instruction materials on approximate number skills of students with dyscalculia,” *Turkish Online J. Educ. Technol.*, vol. 16, no. 2, pp. 119–136, 2017.

[34] Giordano, D. & Maiorana, Francesco. (2014). Addressing dysgraphia with a mobile, web-based software with interactive feedback. 2014 IEEE-EMBS International Conference on Biomedical and Health Informatics, BHI 2014. 264-268. 10.1109/BHI.2014.6864354.

[35] Kurniawan, Dimas & widya sihwi, Sari & Gunarhadi, Gunarhadi. (2017). An expert system for diagnosing dysgraphia. 468-472. 10.1109/ICITISEE.2017.8285552.

[36] Khan, Muhammad & Hussain, Muhammad & Ahsan, Kamran & Saeed, Muhammad & Nadeem, Adnan & Ali, Syed & Mahmood, Nadeem & Rizwan, Kashif. (2017). Augmented Reality Based Spelling Assistance to Dysgraphia Students. Journal of Basic & Applied Sciences. 13. 500-507. 10.6000/1927-5129.2017.13.82.

[37] J. Mekyska, M. Faundez-Zanuy, Z. Mzourek, Z. Galaz, Z. Smekal and S. Rosenblum, "Identification and Rating of Developmental Dysgraphia by Handwriting Analysis," in IEEE Transactions on Human-Machine Systems, vol. 47, no. 2, pp. 235- 248, April 2017, doi: 10.1109/THMS.2016.2586605.

[38] Rosenblum, Sara & Dror, Gideon. (2016). Identifying Developmental Dysgraphia Characteristics Utilizing Handwriting Classification Methods. IEEE Transactions on Human-Machine Systems. PP. 1-7. 10.1109/THMS.2016.2628799.

[39] D. Giordano and F. Maiorana, “A Mobile Web Game Approach for Improving Dysgraphia,” Proceedings of the 7th International Conference on Computer Supported Education, 2015.

[40] Rosenblum, Sara & Dvorkin, Assaf & Weiss, Patrice. (2006). Automatic segmentation as a tool for examining the handwriting process of children with dysgraphic and proficient handwriting. Human movement science. 25. 608-21. 10.1016/j.humov.2006.07.005.

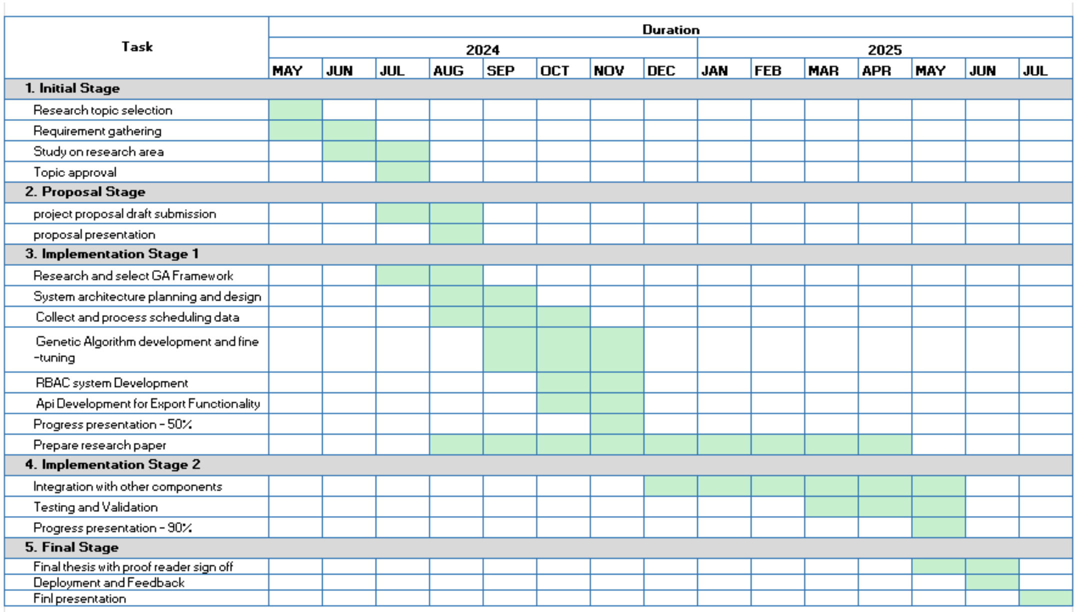
[41] D. A. Kurniawan, S. W. Sihwi, and Gunarhadi, “An expert system for diagnosing dysgraphia,” 2017 2nd International conferences on Information Technology, Information Systems and Electrical Engineering (ICITISEE), 2017.

APPENDICES

Appendix A Logo

Appendix A Logo

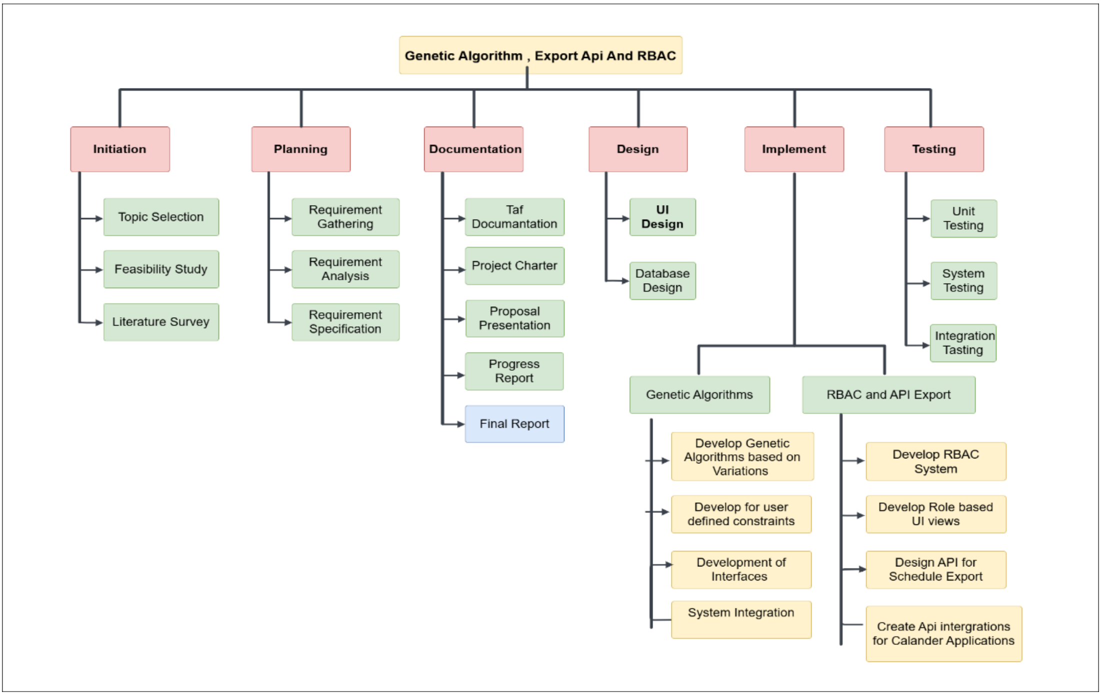
Appendix B Gantt Chart



Task	Duration														
	2024								2025						
	MAY	JUN	JUL	AUG	SEP	OCT	NOV	DEC	JAN	FEB	MAR	APR	MAY	JUN	JUL
1. Initial Stage															
Research topic selection	🟩														
Requirement gathering	🟩	🟩													
Study on research area		🟩	🟩												
Topic approval			🟩												
2. Proposal Stage															
project proposal draft submission			🟩	🟩											
proposal presentation				🟩											
3. Implementation Stage 1															
Research and select GA Framework			🟩	🟩											
System architecture planning and design				🟩	🟩										
Collect and process scheduling data				🟩	🟩	🟩									
Genetic Algorithm development and fine -tuning					🟩	🟩	🟩								
RBAC system Development						🟩	🟩								
Api Development for Export Functionality						🟩	🟩								
Progress presentation - 50%							🟩								
Prepare research paper				🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩			
4. Implementation Stage 2															
Integration with other components								🟩	🟩	🟩	🟩	🟩	🟩		
Testing and Validation											🟩	🟩	🟩		
Progress presentation - 90%													🟩		
5. Final Stage															
Final thesis with proof reader sign off													🟩	🟩	
Deployment and Feedback														🟩	
Finl presentation															🟩

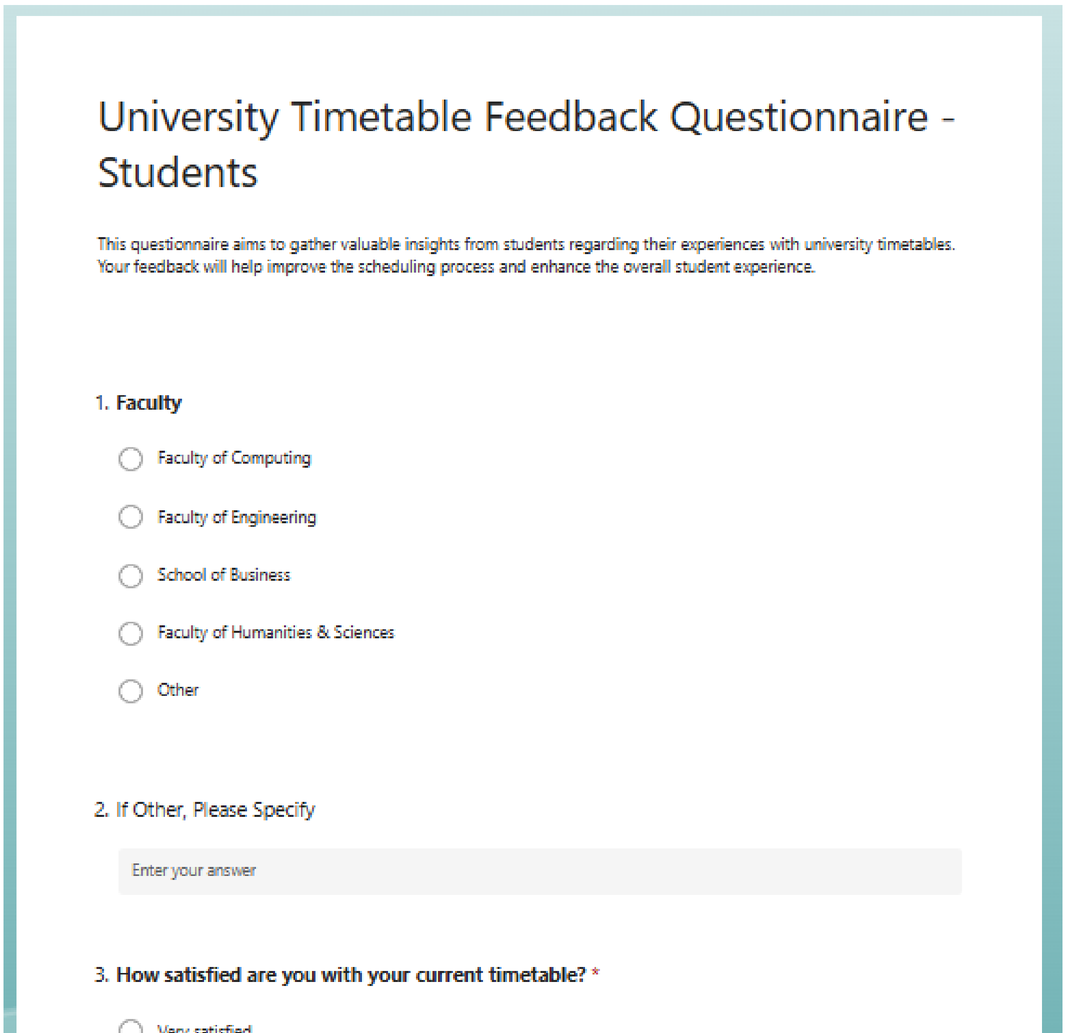
Appendix B Gantt Chart

Appendix C Work Breakdown Chart



Appendix C Work Breakdown Chart

Appendix D Questionnaire for Students Regarding Timetable



University Timetable Feedback Questionnaire - Students

This questionnaire aims to gather valuable insights from students regarding their experiences with university timetables. Your feedback will help improve the scheduling process and enhance the overall student experience.

1. **Faculty**

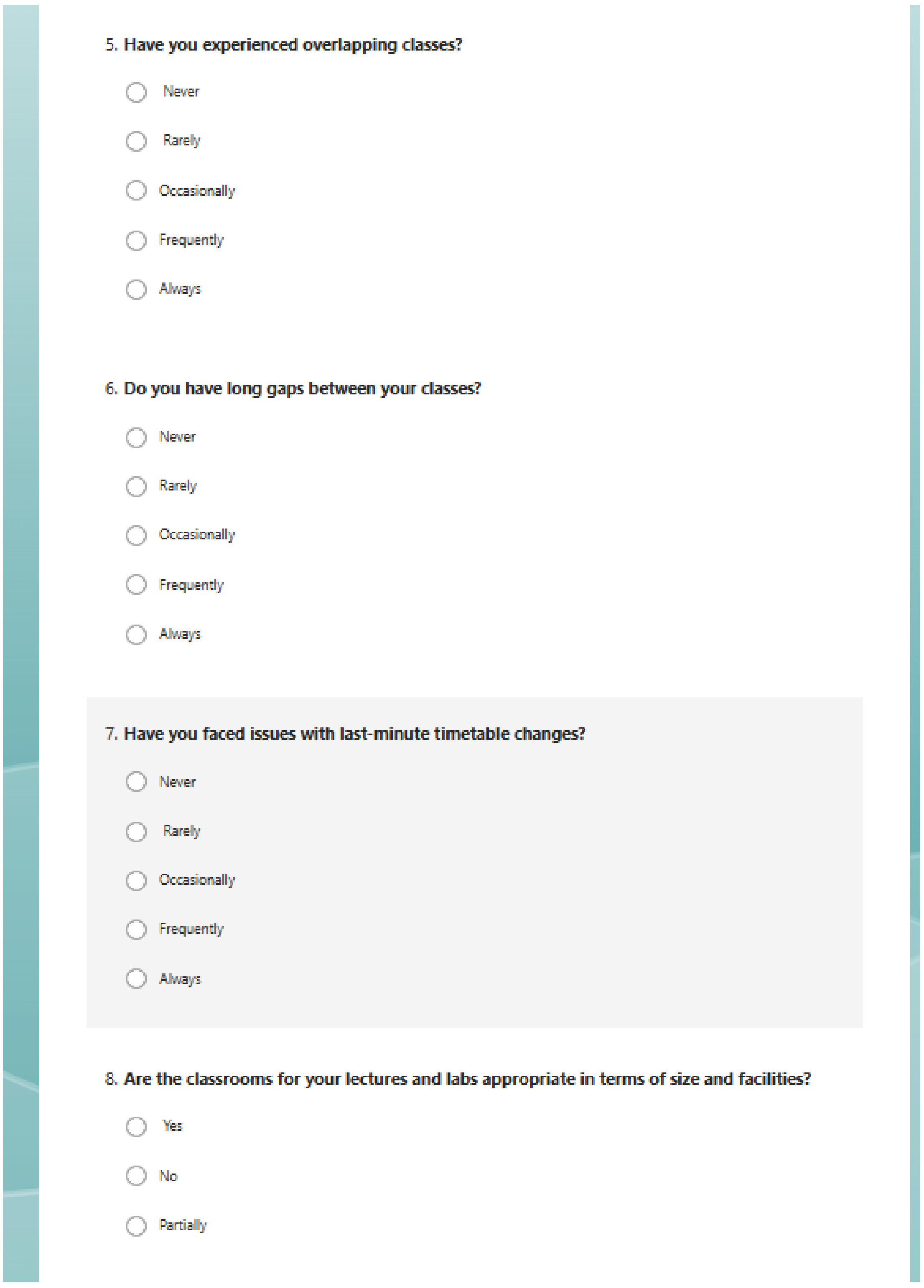
- ○ Faculty of Computing
- ○ Faculty of Engineering
- ○ School of Business
- ○ Faculty of Humanities & Sciences
- ○ Other

2. If Other, Please Specify

Enter your answer

3. **How satisfied are you with your current timetable?** *

- ○ Very satisfied



5. **Have you experienced overlapping classes?**

- ◯ Never
- ◯ Rarely
- ◯ Occasionally
- ◯ Frequently
- ◯ Always

6. **Do you have long gaps between your classes?**

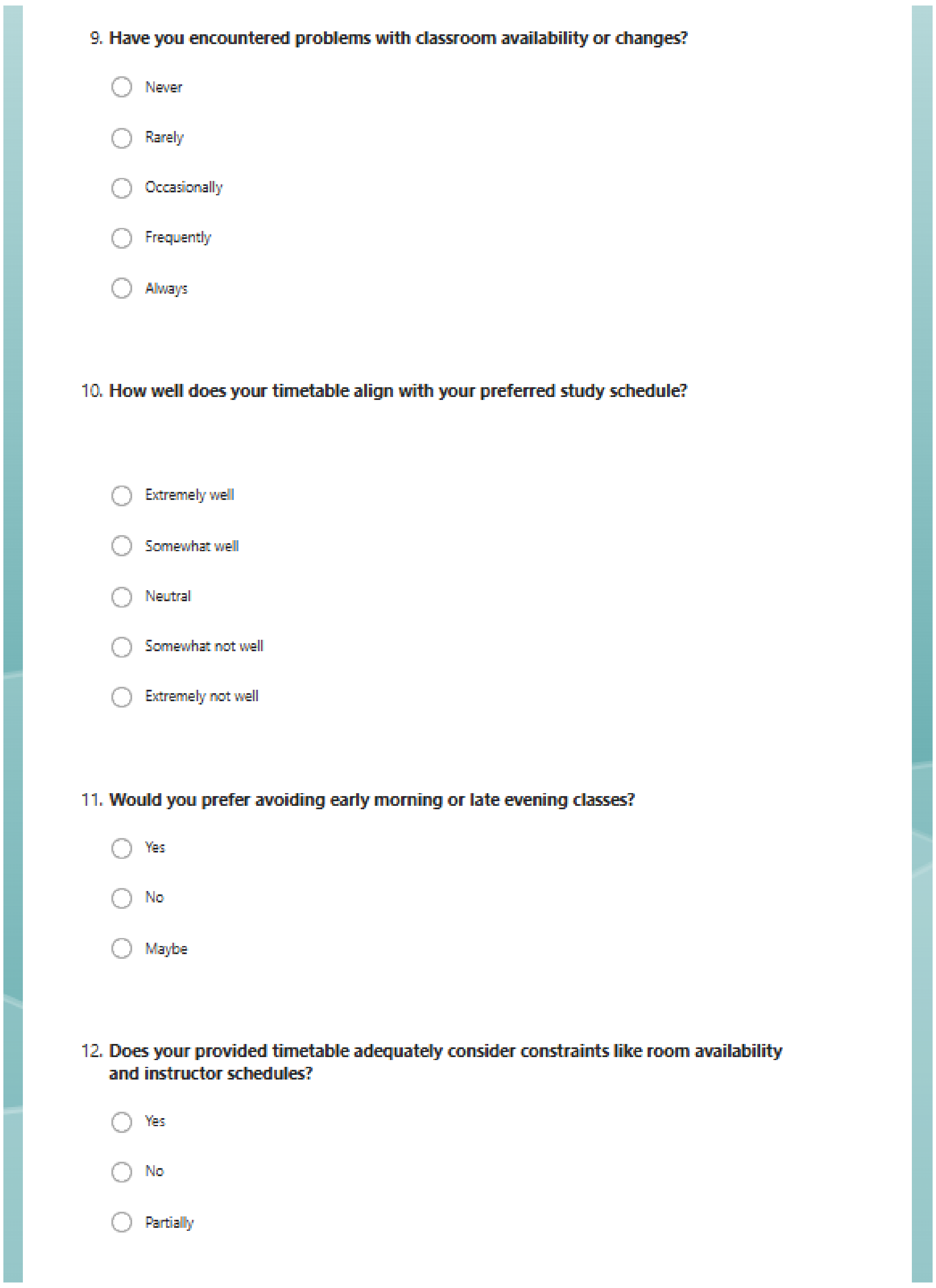
- ◯ Never
- ◯ Rarely
- ◯ Occasionally
- ◯ Frequently
- ◯ Always

7. **Have you faced issues with last-minute timetable changes?**

- ◯ Never
- ◯ Rarely
- ◯ Occasionally
- ◯ Frequently
- ◯ Always

8. **Are the classrooms for your lectures and labs appropriate in terms of size and facilities?**

- ◯ Yes
- ◯ No
- ◯ Partially



9. **Have you encountered problems with classroom availability or changes?**

- ○ Never
- ○ Rarely
- ○ Occasionally
- ○ Frequently
- ○ Always

10. **How well does your timetable align with your preferred study schedule?**

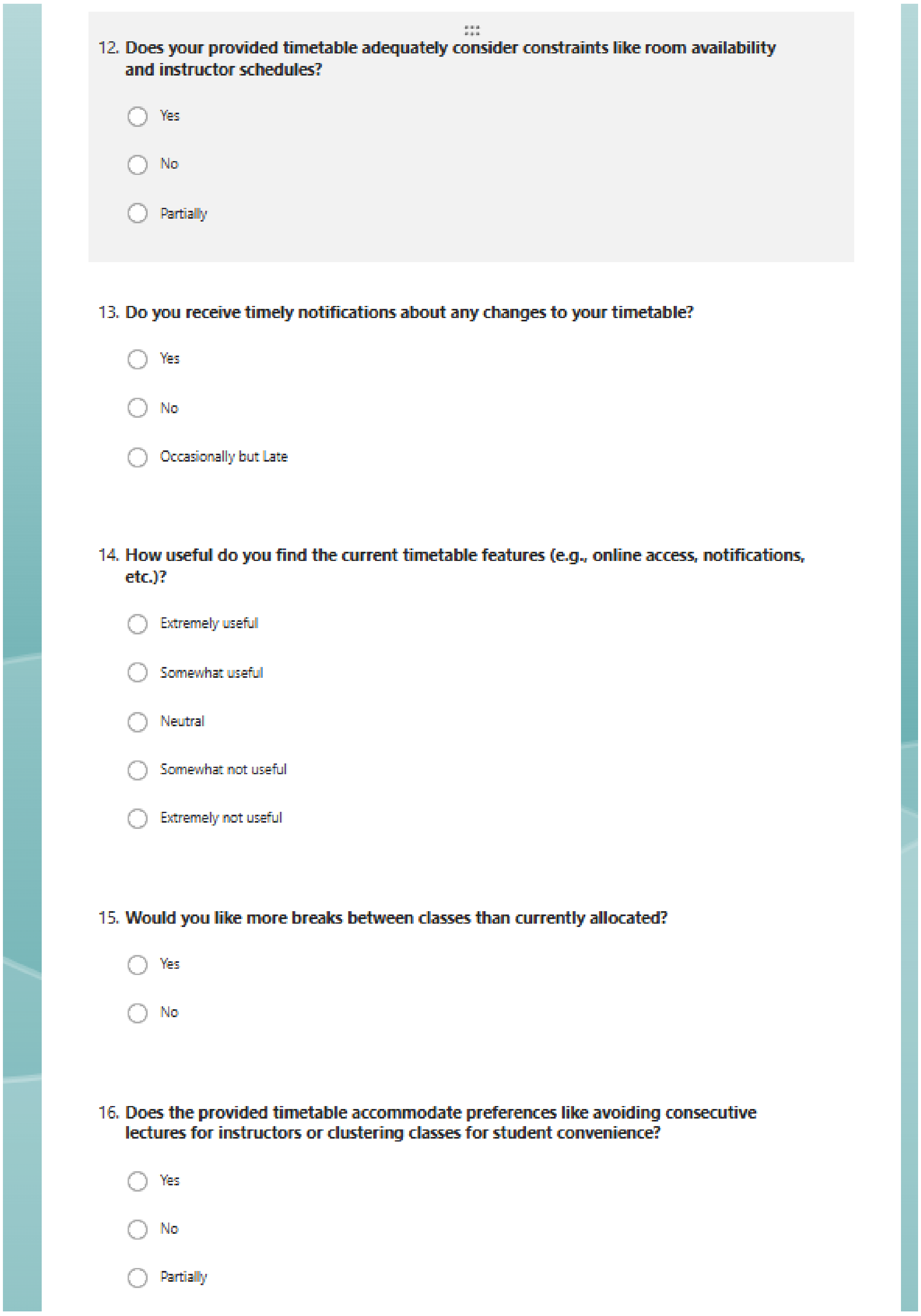
- ○ Extremely well
- ○ Somewhat well
- ○ Neutral
- ○ Somewhat not well
- ○ Extremely not well

11. **Would you prefer avoiding early morning or late evening classes?**

- ○ Yes
- ○ No
- ○ Maybe

12. **Does your provided timetable adequately consider constraints like room availability and instructor schedules?**

- ○ Yes
- ○ No
- ○ Partially



12. **Does your provided timetable adequately consider constraints like room availability and instructor schedules?**

- ○ Yes
- ○ No
- ○ Partially

13. **Do you receive timely notifications about any changes to your timetable?**

- ○ Yes
- ○ No
- ○ Occasionally but Late

14. **How useful do you find the current timetable features (e.g., online access, notifications, etc.)?**

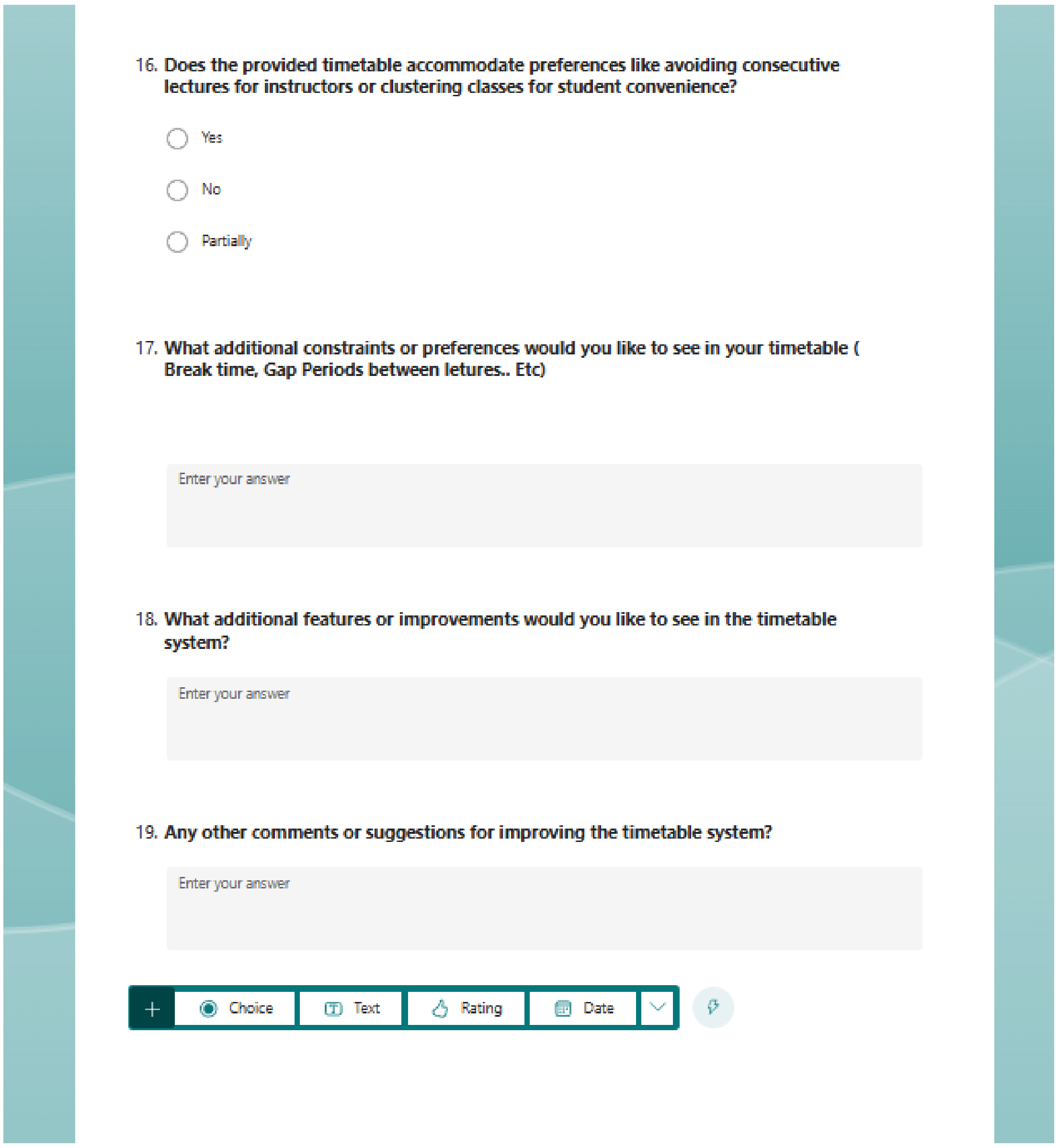
- ○ Extremely useful
- ○ Somewhat useful
- ○ Neutral
- ○ Somewhat not useful
- ○ Extremely not useful

15. **Would you like more breaks between classes than currently allocated?**

- ○ Yes
- ○ No

16. **Does the provided timetable accommodate preferences like avoiding consecutive lectures for instructors or clustering classes for student convenience?**

- ○ Yes
- ○ No
- ○ Partially



16. Does the provided timetable accommodate preferences like avoiding consecutive lectures for instructors or clustering classes for student convenience?

◯ Yes

◯ No

◯ Partially

17. What additional constraints or preferences would you like to see in your timetable (Break time, Gap Periods between letures.. Etc)

Enter your answer

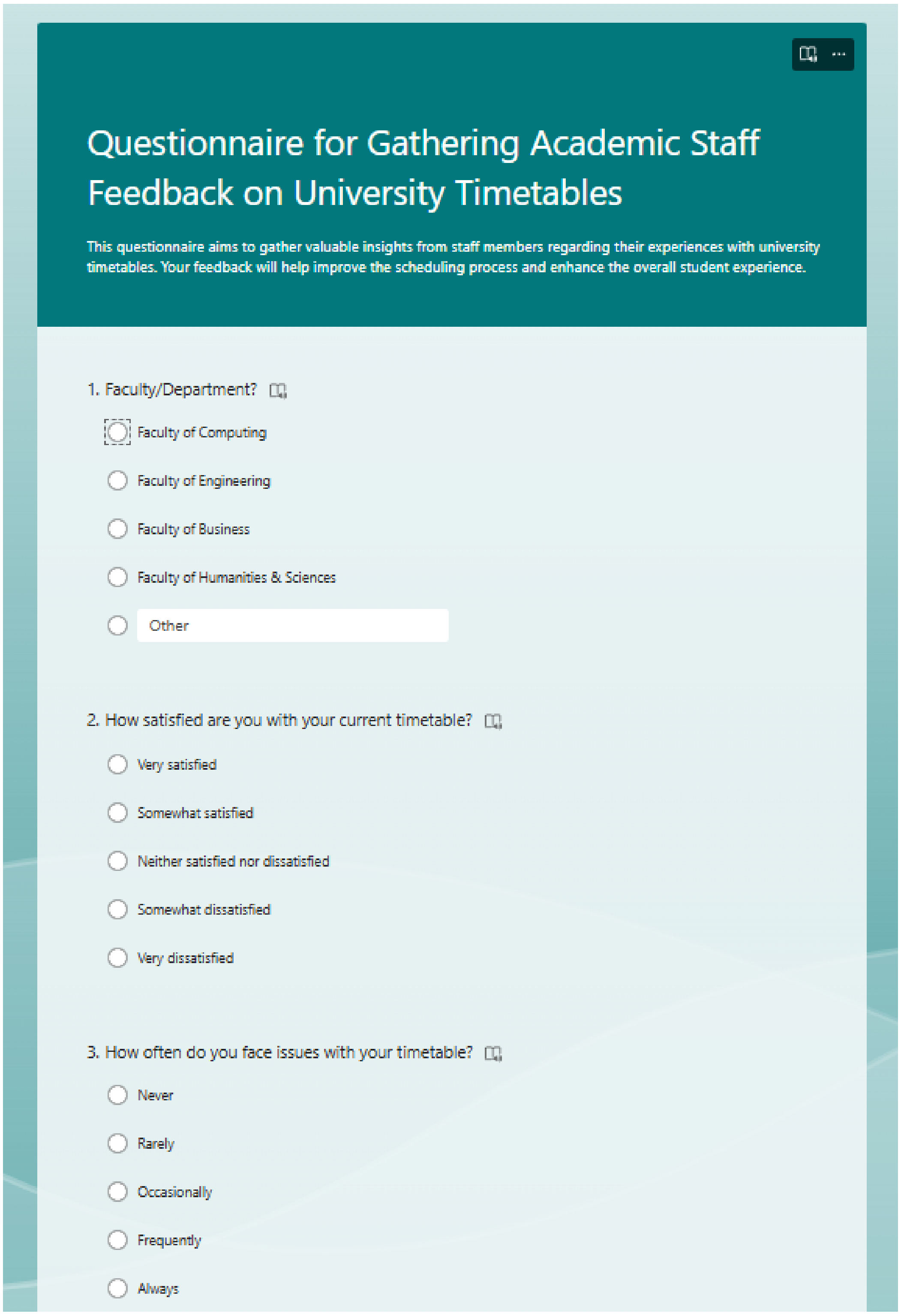
18. What additional features or improvements would you like to see in the timetable system?

Enter your answer

19. Any other comments or suggestions for improving the timetable system?

Enter your answer

Appendix E Questionnaire for Academic Staff Regarding Timetable



Questionnaire for Gathering Academic Staff Feedback on University Timetables

This questionnaire aims to gather valuable insights from staff members regarding their experiences with university timetables. Your feedback will help improve the scheduling process and enhance the overall student experience.

1. Faculty/Department?

- Faculty of Computing
- Faculty of Engineering
- Faculty of Business
- Faculty of Humanities & Sciences
- Other

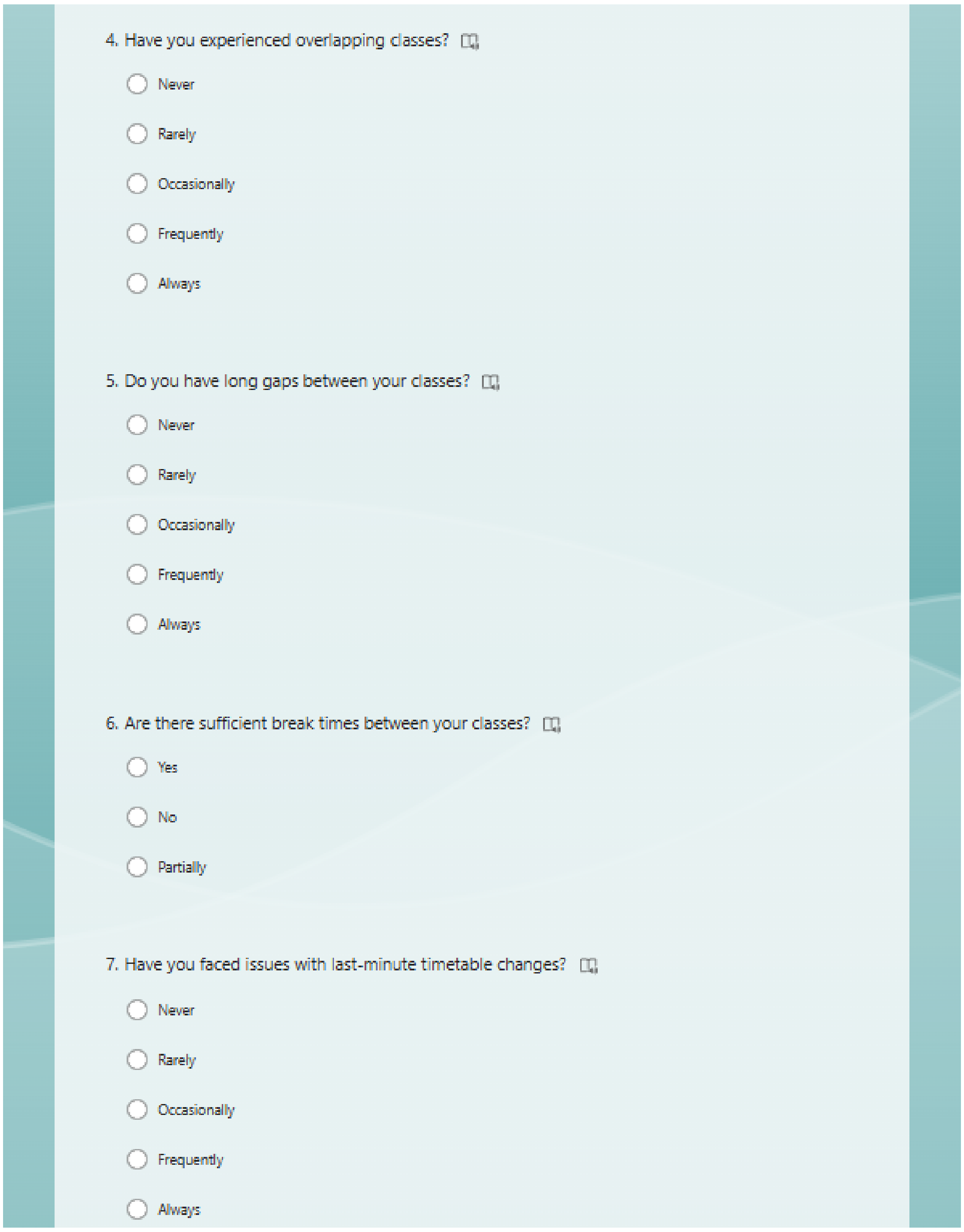
2. How satisfied are you with your current timetable?

- Very satisfied
- Somewhat satisfied
- Neither satisfied nor dissatisfied
- Somewhat dissatisfied
- Very dissatisfied

3. How often do you face issues with your timetable?

- Never
- Rarely
- Occasionally
- Frequently
- Always

Appendix EQuestionnaire for Academic Staff Regarding Timetable



4. Have you experienced overlapping classes?

- ◯ Never
- ◯ Rarely
- ◯ Occasionally
- ◯ Frequently
- ◯ Always

5. Do you have long gaps between your classes?

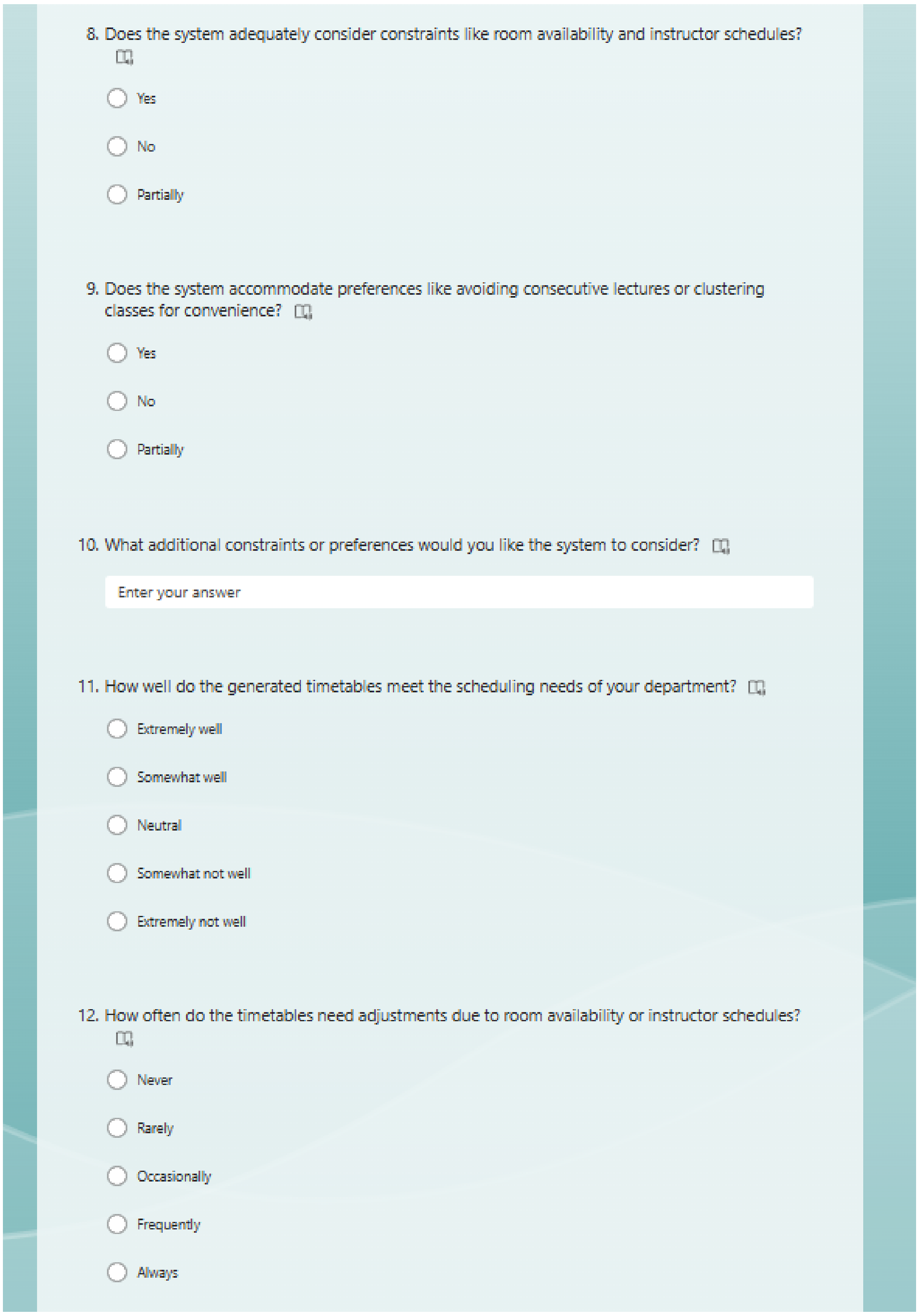
- ◯ Never
- ◯ Rarely
- ◯ Occasionally
- ◯ Frequently
- ◯ Always

6. Are there sufficient break times between your classes?

- ◯ Yes
- ◯ No
- ◯ Partially

7. Have you faced issues with last-minute timetable changes?

- ◯ Never
- ◯ Rarely
- ◯ Occasionally
- ◯ Frequently
- ◯ Always



8. Does the system adequately consider constraints like room availability and instructor schedules?

- ◯ Yes
- ◯ No
- ◯ Partially

9. Does the system accommodate preferences like avoiding consecutive lectures or clustering classes for convenience?

- ◯ Yes
- ◯ No
- ◯ Partially

10. What additional constraints or preferences would you like the system to consider?

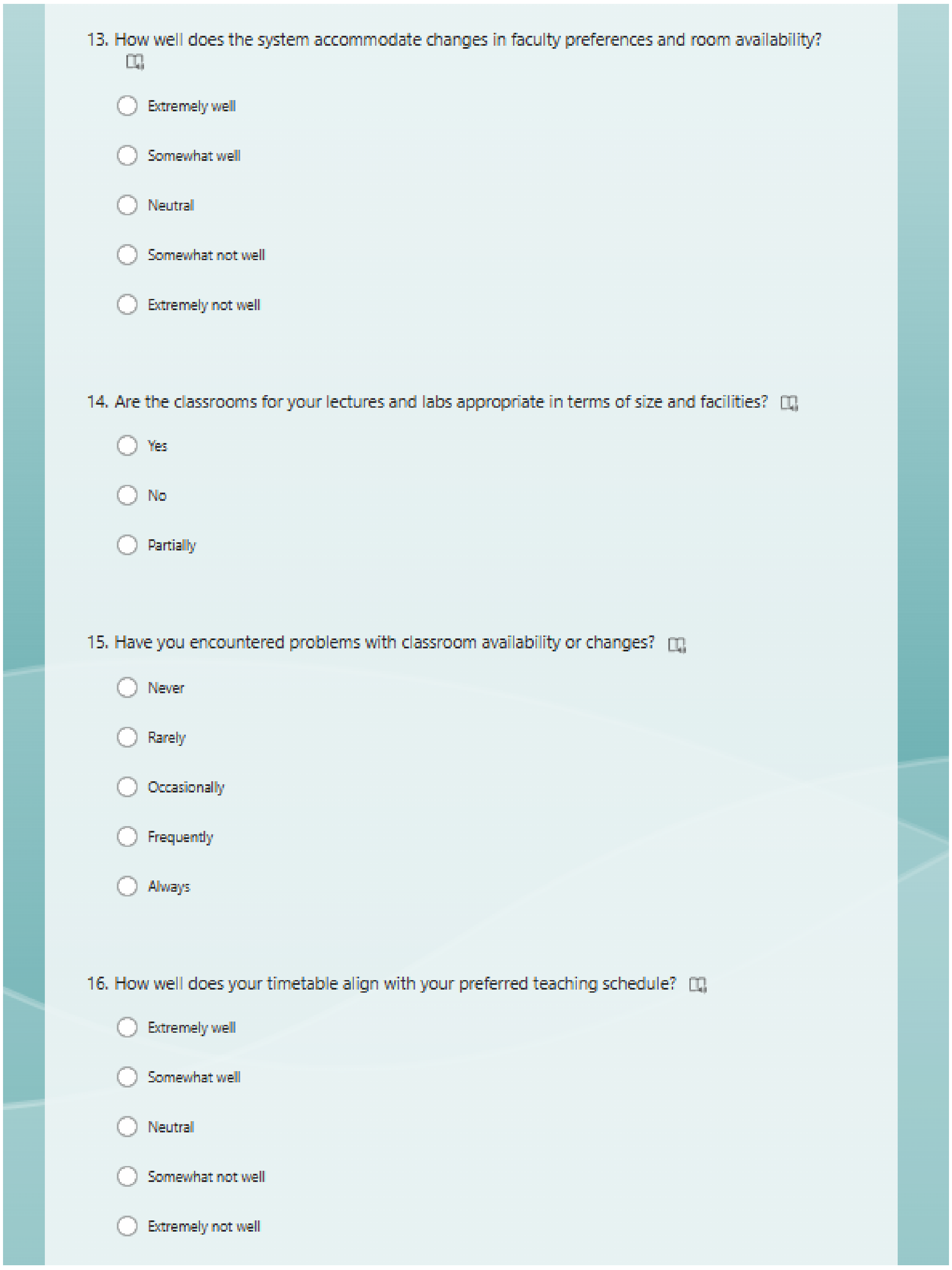
Enter your answer

11. How well do the generated timetables meet the scheduling needs of your department?

- ◯ Extremely well
- ◯ Somewhat well
- ◯ Neutral
- ◯ Somewhat not well
- ◯ Extremely not well

12. How often do the timetables need adjustments due to room availability or instructor schedules?

- ◯ Never
- ◯ Rarely
- ◯ Occasionally
- ◯ Frequently
- ◯ Always



13. How well does the system accommodate changes in faculty preferences and room availability?

- ◯ Extremely well
- ◯ Somewhat well
- ◯ Neutral
- ◯ Somewhat not well
- ◯ Extremely not well

14. Are the classrooms for your lectures and labs appropriate in terms of size and facilities?

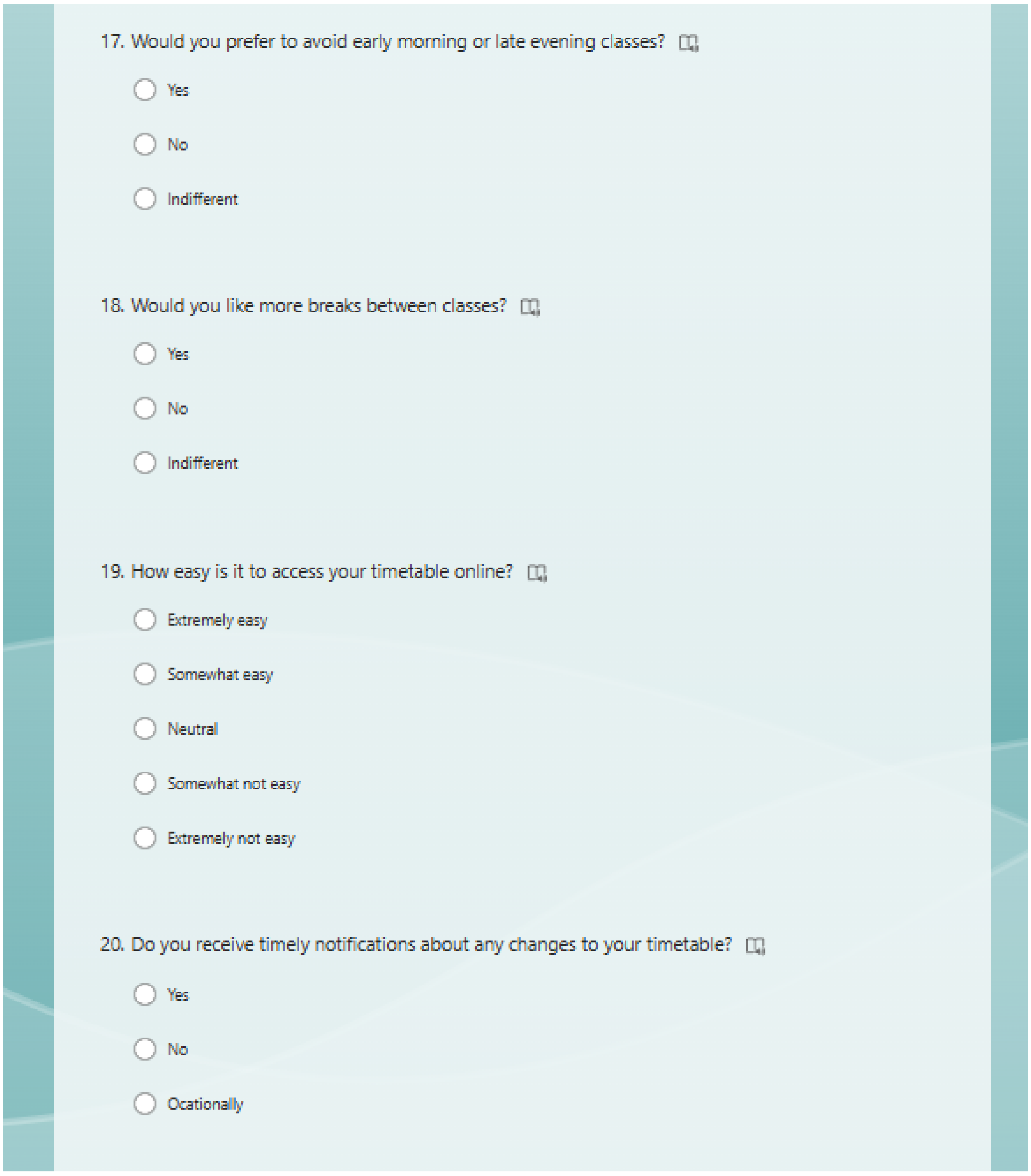
- ◯ Yes
- ◯ No
- ◯ Partially

15. Have you encountered problems with classroom availability or changes?

- ◯ Never
- ◯ Rarely
- ◯ Occasionally
- ◯ Frequently
- ◯ Always

16. How well does your timetable align with your preferred teaching schedule?

- ◯ Extremely well
- ◯ Somewhat well
- ◯ Neutral
- ◯ Somewhat not well
- ◯ Extremely not well



17. Would you prefer to avoid early morning or late evening classes?

- ◯ Yes
- ◯ No
- ◯ Indifferent

18. Would you like more breaks between classes?

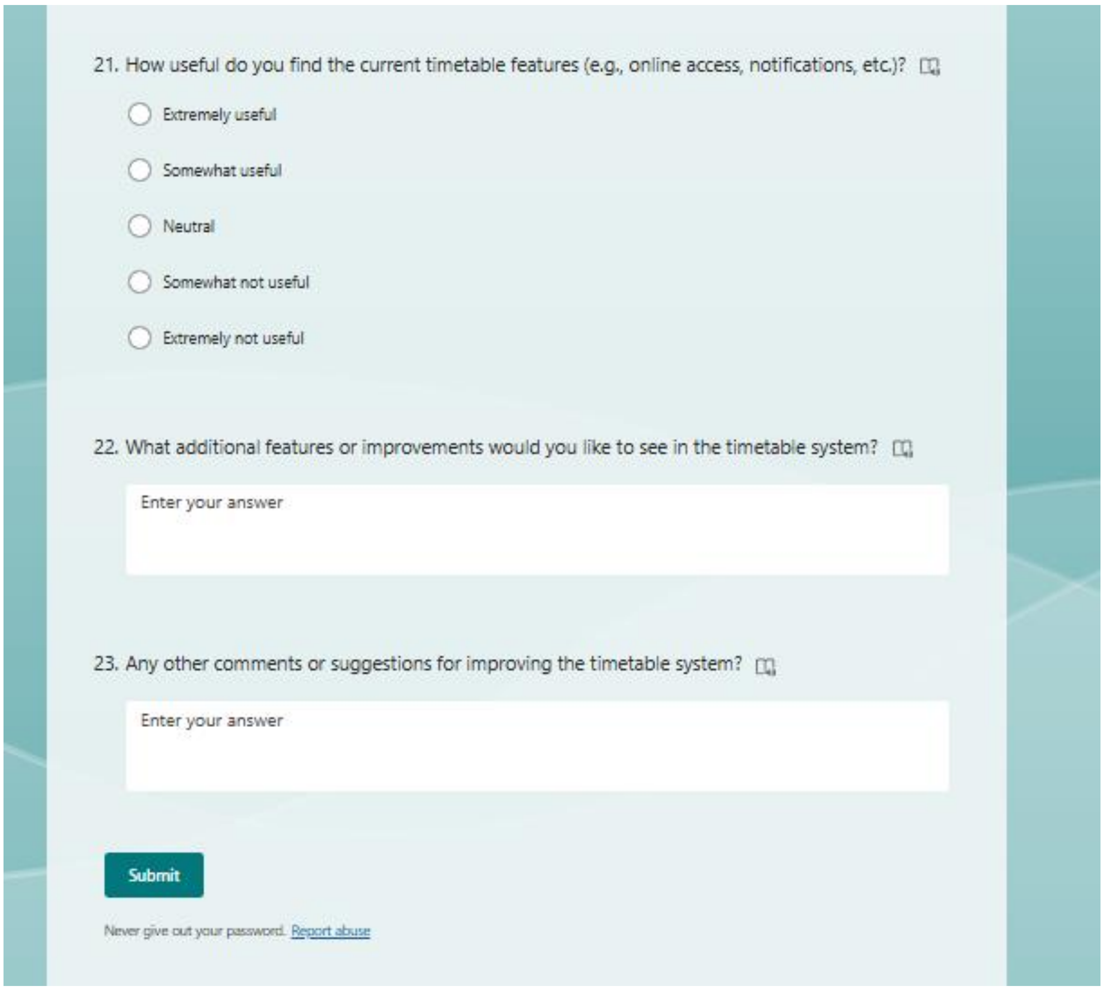
- ◯ Yes
- ◯ No
- ◯ Indifferent

19. How easy is it to access your timetable online?

- ◯ Extremely easy
- ◯ Somewhat easy
- ◯ Neutral
- ◯ Somewhat not easy
- ◯ Extremely not easy

20. Do you receive timely notifications about any changes to your timetable?

- ◯ Yes
- ◯ No
- ◯ Ocationally



21. How useful do you find the current timetable features (e.g., online access, notifications, etc.)?

- ◯ Extremely useful
- ◯ Somewhat useful
- ◯ Neutral
- ◯ Somewhat not useful
- ◯ Extremely not useful

22. What additional features or improvements would you like to see in the timetable system?

Enter your answer

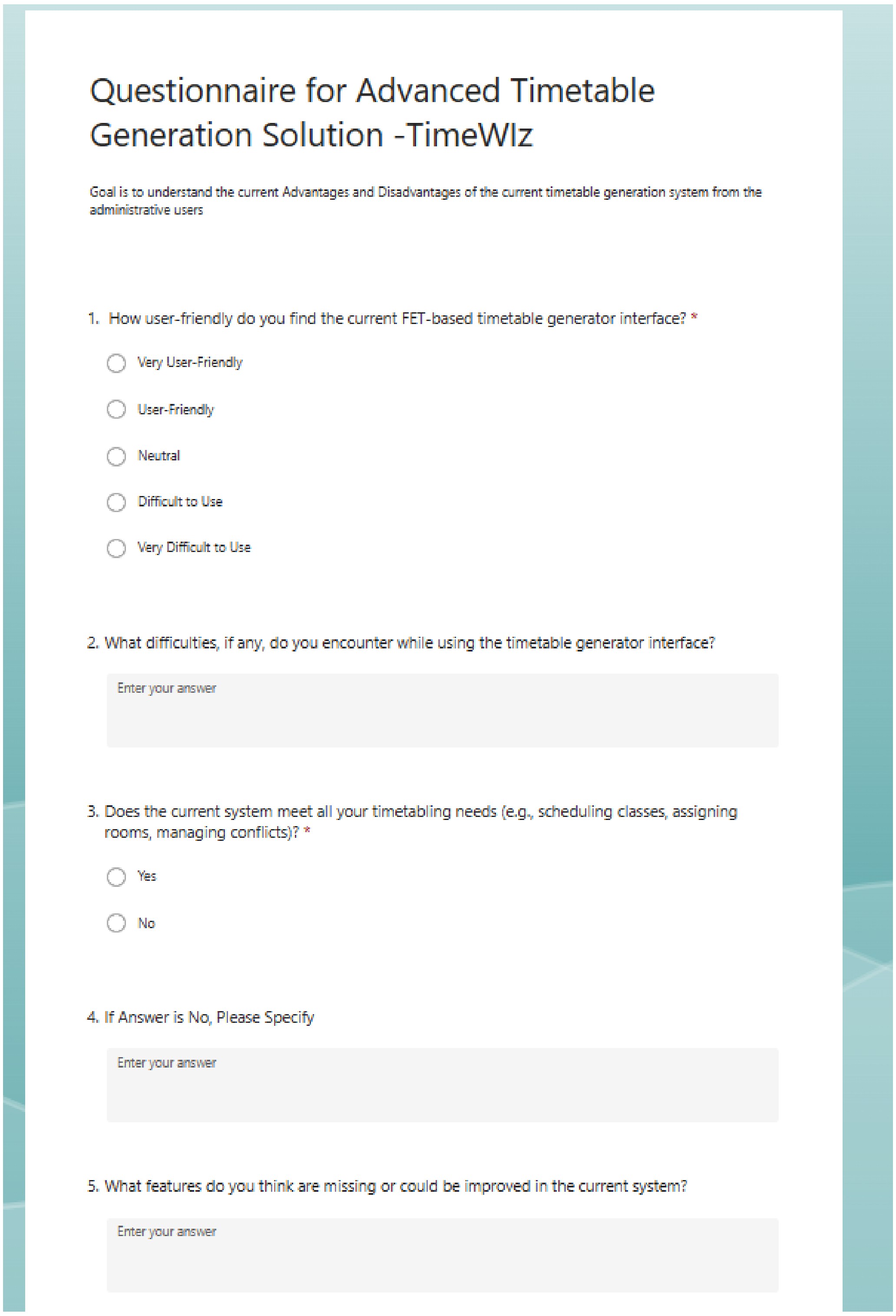
23. Any other comments or suggestions for improving the timetable system?

Enter your answer

Submit

Never give out your password. Report abuse

Appendix F Questionnaire for Administration Regarding Timetable



Questionnaire for Advanced Timetable Generation Solution -TimeWIz

Goal is to understand the current Advantages and Disadvantages of the current timetable generation system from the administrative users

1. How user-friendly do you find the current FET-based timetable generator interface? *

- ◯ Very User-Friendly
- ◯ User-Friendly
- ◯ Neutral
- ◯ Difficult to Use
- ◯ Very Difficult to Use

2. What difficulties, if any, do you encounter while using the timetable generator interface?

Enter your answer

3. Does the current system meet all your timetabling needs (e.g., scheduling classes, assigning rooms, managing conflicts)? *

- ◯ Yes
- ◯ No

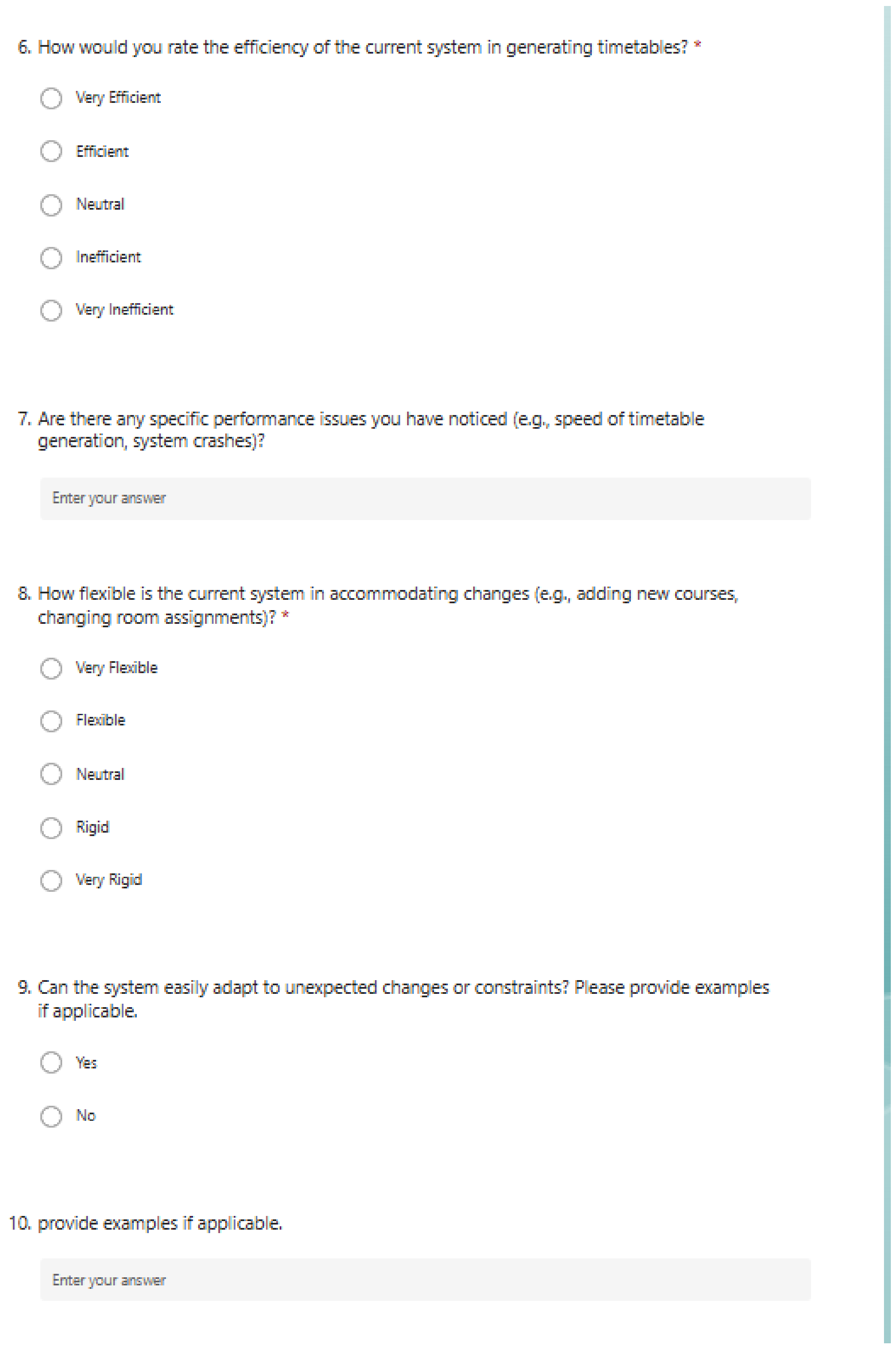
4. If Answer is No, Please Specify

Enter your answer

5. What features do you think are missing or could be improved in the current system?

Enter your answer

Appendix F Questionnaire for Administration Regarding Timetable



6. How would you rate the efficiency of the current system in generating timetables? *

◯ Very Efficient

◯ Efficient

◯ Neutral

◯ Inefficient

◯ Very Inefficient

7. Are there any specific performance issues you have noticed (e.g., speed of timetable generation, system crashes)?

Enter your answer

8. How flexible is the current system in accommodating changes (e.g., adding new courses, changing room assignments)? *

◯ Very Flexible

◯ Flexible

◯ Neutral

◯ Rigid

◯ Very Rigid

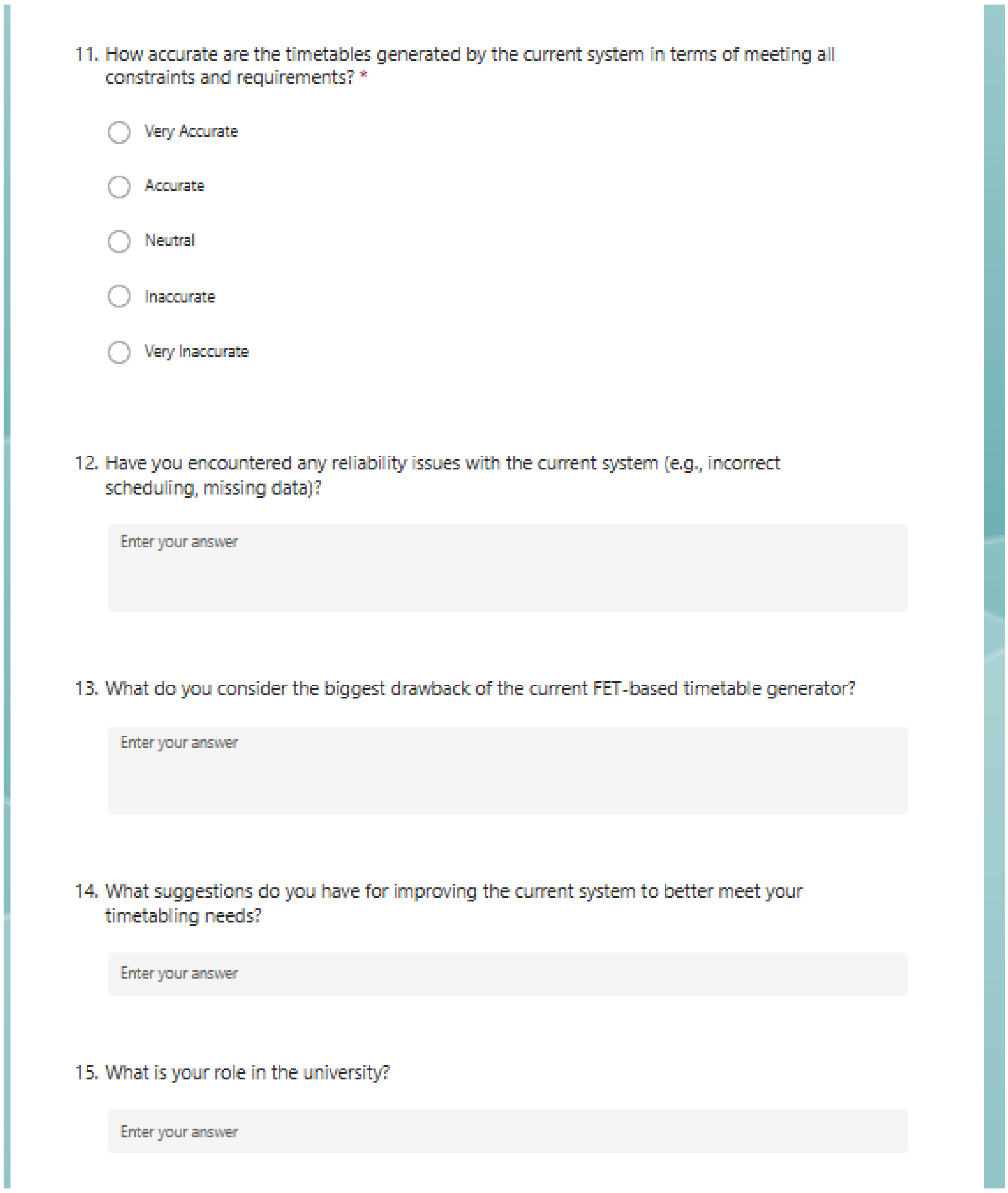
9. Can the system easily adapt to unexpected changes or constraints? Please provide examples if applicable.

◯ Yes

◯ No

10. provide examples if applicable.

Enter your answer



11. How accurate are the timetables generated by the current system in terms of meeting all constraints and requirements? *

○ Very Accurate

○ Accurate

○ Neutral

○ Inaccurate

○ Very Inaccurate

12. Have you encountered any reliability issues with the current system (e.g., incorrect scheduling, missing data)?

Enter your answer

13. What do you consider the biggest drawback of the current FET-based timetable generator?

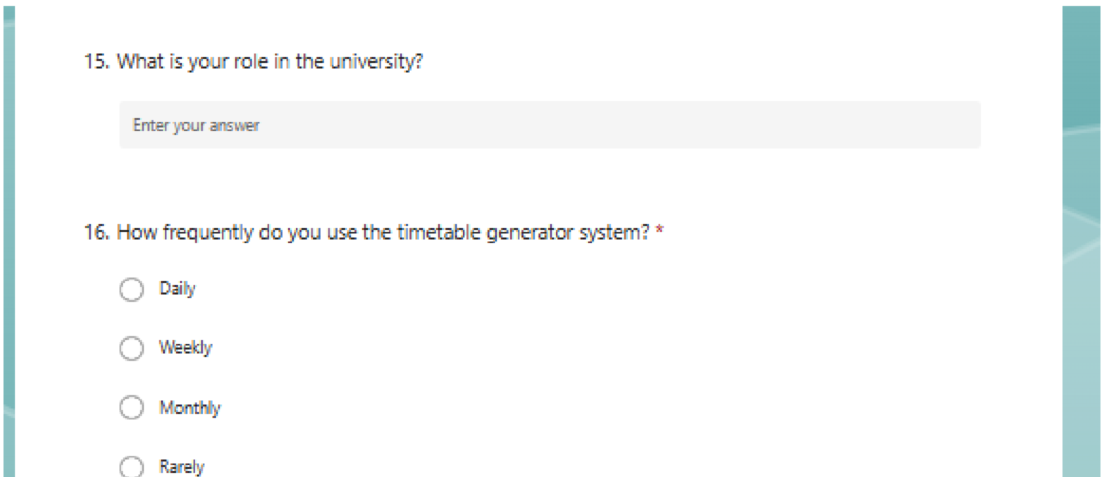
Enter your answer

14. What suggestions do you have for improving the current system to better meet your timetabling needs?

Enter your answer

15. What is your role in the university?

Enter your answer



15. What is your role in the university?

Enter your answer

16. How frequently do you use the timetable generator system? *

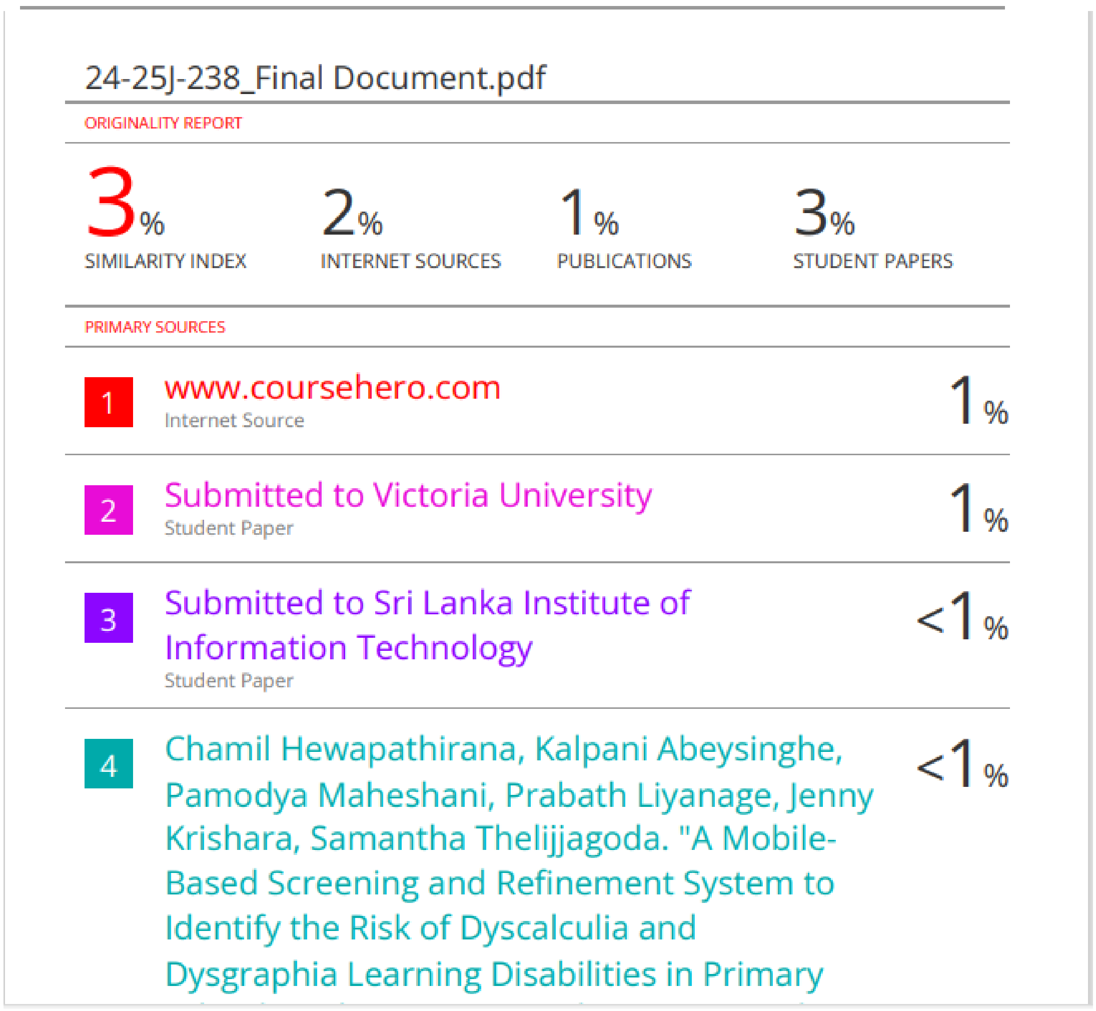
○ Daily

○ Weekly

○ Monthly

○ Rarely

Appendix - J: Plagiarism Report



24-25J-238_Final Document.pdf

ORIGINALITY REPORT

3%	2%	1%	3%
SIMILARITY INDEX	INTERNET SOURCES	PUBLICATIONS	STUDENT PAPERS

PRIMARY SOURCES

1	www.coursehero.com Internet Source	1%
2	Submitted to Victoria University Student Paper	1%
3	Submitted to Sri Lanka Institute of Information Technology Student Paper	<1%
4	Chamil Hewapathirana, Kalpani Abeysinghe, Pamodya Maheshani, Prabath Liyanage, Jenny Krishara, Samantha Thelijjagoda. "A Mobile-Based Screening and Refinement System to Identify the Risk of Dyscalculia and Dysgraphia Learning Disabilities in Primary	<1%

Appendix G Plagiarism Report