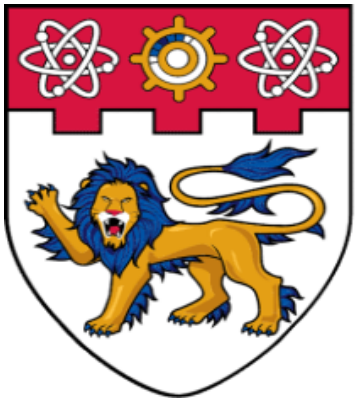

NANYANG TECHNOLOGICAL UNIVERSITY
SINGAPORE

BC2410/BC2411 Prescriptive Analytics

Group Project

Group 1

Name	Matriculation Number
Cao Tingsen	U2210840C
Foo Yee Zuan, Ferlyn (Fu Yixuan)	U2310374E
Isabelle Lim Xin Yin (Lin XingYing)	U2310318C
Tan Pang Boon	U2340262G
Zhang Yawen	N2402219G

Table of Contents:

1. Business Problem: Inefficiencies in Student Timetable Planning at NTU

1.1 Background & Problem

Before the start of every semester, first-year NTU students receive a pre-arranged course timetable from their school, while students in later years are allowed to plan and register their own timetables based on personal preferences using the NTU STARS system. However, whether it's a school-arranged schedule or a self-arranged one, scheduling conflicts and inefficiencies continue to crop up. Back-to-back lessons, long breaks between lectures, awkward early morning or late night sessions, and trouble with Co-Curricular Activities (CCA), part-time jobs, and personal obligations are just a few examples of the inefficient schedules that can result from this time-consuming and frustrating process. For example, imagine a student with classes at 8.30am and the next class happening at 4.30pm, this will leave students with a long awkwards break with nothing to do in between. Due to the lack of an integrated optimization tool, students are sometimes left to make do with less-than-ideal timetables.

1.2 Who It Affects

All NTU students are negatively impacted by this problem, but those who wish to maximize their time are most affected. Many students struggle to find a schedule that works best for them, whether they are attempting to squeeze in CCAs, part-time employment, or just less time on campus. Students frequently wind up with schedules that don't suit their needs or preferences.

1.3 How It Impacts Students

The impact of an inefficient timetable is more than just inconvenience. Long breaks between classes mean hours spent idly waiting on campus, while poorly structured schedules make it harder to balance academic and personal life. Some students end up with packed days that leave them exhausted. For instance, with back-to-back lessons, students do not even have enough time to travel from one classroom to another, while others may have scattered classes that make their weeks inefficient. With course registration already being competitive, students don't always get the best options, and manually fixing a bad timetable is frustrating.

1.4 Current Solutions

To get an ideal schedule, students have two options. Firstly, non-first-year students can pre-plan their timetables and secure their desired indices as soon as registration opens. For first-year students, or for those dissatisfied with their initial schedule, the add/drop period allows modifications by adding or dropping courses, changing indices, or swapping indices with friends. However, this requires long hours of staring into the screen, trying to piece together the right combination from a maze of index options in search of the most optimal arrangement.

1.5 Relevant Assumptions

a. All lessons occur weekly without any variation (i.e. odd and even weeks).
b. All classes will adhere to a set one-hour time window; there won't be any 30-minute or erratic lesson lengths.

2. Our Solution

In this project, we will develop a **Best Timetable Generator** using optimization methods to automatically create the most efficient schedule based on real NTU module timetable data and students' personal preferences. In our solution, the objective is to minimize both the number of days on campus and the total time spent in school attending lessons. Additionally, students can choose to prioritize fewer days on campus or less overall time spent, allowing more flexibility.

We also take factors such as CCAs, work commitments, and preferred time slots into account by integrating them into the constraints. For example, students can block out certain periods so that no classes are scheduled during those times. This means that instead of spending hours trying to manually create the "perfect" schedule, students would instantly get the top three best options by default (with the flexibility to adjust that number as needed), helping them make smarter, stress-free decisions and making their time at NTU more productive.

In our project, we will only be using modules from Year 2 Business (BA) students. However, the model is intended for future use across all NTU students.

3. Model Setting

3.1. Data Preparation and Cleaning

Course and timetable data are scraped from the official NTU scheduling portal (NTU, n.d.). The data is filtered to include Year 2 Business courses. However, the format provided on the portal is not suitable (figure 2), and attempts to obtain a structured version from the school were unsuccessful due to confidentiality concerns (figure 1). As a result, basic Python preprocessing was employed to resolve inconsistencies in case formatting and spacing (figure 3). The data is structured into a standardized table format to ensure each module and its corresponding schedule details can be accurately interpreted by the optimization model.

The table include the following information:

Course Code:	All possible course codes
Index:	Unique index number for each course
Type:	Type of lesson, e.g. SEM or Lecture
Group:	Course Index Group
Day:	day where classes occur, ['MON', 'TUE', 'WED', 'THU', 'FRI']
Time:	Time where classes occur, START-END

3.2. Model Setting Formulation

Refer to **Appendix B** for Description of the Decision Variables and Parameters

Objective: **Minimize**

$$\alpha \cdot \sum_{d \in Days} y_d + \beta \cdot \sum_{d \in Days} z_d^{\text{total}}$$

The objective is to minimize both the number of school days and the total time spent on campus. α and β act as weights to balance priorities. α controls the importance of minimizing school days, while β controls the importance of minimizing total time spent on campus. Users can adjust them based on their preferences. Since α and β are measured on different scales, we multiply α by a scaling factor to make them comparable which will be explained in Section 3.3.1.

Note: time spent on campus includes lessons and breaks, since lessons will be constant, this is the same as minimizing break time.

Decision Variables Constraint

$$x_i \in \{0,1\} \quad \forall i \in Index$$

$$y_d \in \{0,1\} \quad \forall d \in Days$$

$$class_{d,t} \in \{0,1\} \quad \forall t \in Timeslot \quad \forall d \in Days$$

$$z\_start_d,\ z\_end_d \in [0830, 2230] \quad \forall d \in Days$$

$$z\_total_d \geq 0 \quad \forall d \in Days$$

Constraints 1: Course Index Selection

$$\sum_{i \in Index_c} x_i = 1 \quad \forall c \in Course$$

While a single course may offer multiple indexes, only one index should be chosen. Thus this constraint ensures exactly one x_i is selected among all indices belonging to course c, using a sum of binary variables over that course.

Constraints 2: No Overlapping Classes

Prevents overlapping classes by ensuring that no two selected indexes share the same time slot

$$x_{i_1} + x_{i_2} \leq 1 \quad \text{if } i_1, i_2 \text{ overlap on the same day and time}$$

This constraint prevents two course indexes with overlapping time slots from being selected simultaneously. Since the decision variables are binary, setting $x_{i1} + x_{i2} \leq 1$ ensures that at most one of the conflicting indexes can be chosen.

Note: How do we know if i1 and i2 overlap? using a nested loop and checks every unique pair of course indices (id1, id2)

Constraints 3: Active School Day Tracking

$$x_i \leq y_d \quad \forall (d, s, e) \in Interval_i, \quad \forall i \in \mathcal{I}ndex$$

For any selected class $x_i = 1$, the corresponding day variable y_d must also be at least 1. Since both are binary, this forces $y_d = 1$ when $x_i = 1$.

Constraints 4: Total Time in School per Day

To calculate the total time a student spends in school on a given day, we must determine the earliest start and latest end times across all selected classes.

First and last class of each day:

$$z_d^{\text{start}} \leq s + M(1 - x_i), \quad \forall (d', s, e) \in Interval_i, \quad \forall i \in \mathcal{I}ndex, \quad \forall d \in \mathcal{D}ays \quad s.t. \quad d' = d$$

$$z_d^{\text{end}} \geq e \cdot x_i, \quad \forall (d', s, e) \in Interval_i, \quad \forall i \in \mathcal{I}ndex, \quad \forall d \in \mathcal{D}ays \quad s.t. \quad d' = d$$

Why is $(1\text{-}x_i)$ required? Some classes start early, but if it's not selected, we don't want to include it. Thus we add a M so that the constraint can relax. For example for z_start_d, when $x_i = 1$, this becomes $z\_start_d \leq s$, the earliest start time is no later than this class's start. When $x_i = 0$, the big-M term relaxes the constraint. Similar logic for z_end_d.

This allows us to directly compute the total time spent on campus per day by subtracting the earliest start time from the latest end time.

$$z_d^{\text{total}} = z_d^{\text{end}} - z_d^{\text{start}} \quad \forall d \in \mathcal{D}ays$$

Constraints 5: Blocked Time Enforcement

Prevent classes from being scheduled in time slots or days that the student has blocked out

$$y_d = 0, \quad \forall d \in blocked_{(d,t)} \quad \text{if user block whole day}$$

$$x_i = 0 \quad \forall (d, s, e) \in interval_i, \quad \forall i \in \mathcal{I}ndex, \quad \forall d, t \in blocked_{(d,t)} \quad if \quad s \leq t < e$$

Forces day d to have no classes (all x_i of that day must be zero implicitly). For each class that overlaps a blocked time t, its decision variable is forced to zero.

Constraints 6: Max Consecutive Hours

Limits the number of consecutive hours a student can be in class without a break

$$\text{class}_{d,t} = \sum_{\substack{i \in \mathcal{I}ndex \\ (d,s_i,e_i) \in interval_i \\ s.t. \\ s_i \leq t < e_i}} x_i \quad t \in TimeSlot, \quad \forall d \in Days$$

$$\sum_{t \in \text{window}} \text{class}_{d,t} \leq maxH \quad window \in T_{maxC}, \quad \forall d \in Days$$

For each day, we look at each timeslot. We look at all possible indexes that have class in this timeslot. Taking the sum of x_i gives a binary result of 1 or 0 since only one class can happen at each timeslot. This information will be stored in Class_(d,t).

We create this sliding window based on the max consecutive hours from the user, then we check through every possible time interval to ensure that there are at most maxH consecutive occupied slots in any continuous block of maxH+1.

3.3 Computational Results with Analysis and Interpretation

3.3.1 User Inputs and Preferences

Upon running the script, students can use the search box to select the course codes they need, along with the following customizable preferences (Figure 5):

- **Maximum Consecutive Hours**: A slider allows students to define the longest acceptable stretch of back-to-back classes in a day (ranging from 3 to 13 hours).
- **Blocked Times**: Students can mark specific time slots and days they are unavailable for lessons (e.g. no classes after 5:30 PM on Wednesdays).
- **Preference Weighting**: 2 priority sliders are used to control the emphasis between days on campus (α) and total time spent in school (β).
- **Number of timetables to generate**: the parameter that controls the number of different output solutions.
- **Check Box for Duplicate timeslot**: When checked, this option prevents returning timetables that have the same timeslot with different indexes.

α and β are on different numerical scales. α measures days (usually 1–5, in days), while β measures time (usually 1800–2500, in Z_total). To make a fair comparison, we multiply α by 700 to match the scale of β. 700 is derived based on assumptions that a business student takes an average of 17 AUs per semester (Figure 4). This roughly translates to 17 hours + 4 hours of breaks (assumed) = 21 hours/week. Spread over an ideal and realistic schedule of 3 school days, this averages to around 21 hrs divided by 3 days = 7 hours/day or 700 in terms of Z_total.

After selecting all parameters and submitting the optimizer, the students can get the solution immediately on the same page to avoid consistently browsing different screens (Figure 6)

3.3.2 Output Interpretation

Our output consists of 2 main parts:

1. **Timetable Solutions:**

   The first part presents the unique solution in the form of a well-formatted weekly timetable (Figure 7), covering the schedule from Monday 08:30 to Friday 21:30. This table visually displays course allocations for each hour, making it easy for students to view their weekly commitments.

2. **Summary Stats:**

   The second part provides a detailed breakdown of the selected courses, including the course code, index, and group, and the following statistics informations (Figure 8):

   Active School Days: A binary indicator per day, where 0 represents a day off and 1 indicates the student has classes.

   Time in School each day: The cumulative total of hours spent on campus throughout the day.

   Total time in school: The cumulative total of hours spent on campus throughout the week.

In addition, students can navigate through the top right tabs to see the different solutions. To avoid identical time, we filter out structurally identical solutions by comparing their class times, and we add constraints to exclude repeats. So the outputs can offer more diverse solutions to students.

If there is no feasible solution, the output will indicate the reason. For instance, while choosing AB2008 and blockout time on Thursday morning, the output will show the infeasible block time and course code (Figure 9):

Infeasible constraint: block_time_THU_830

Infeasible constraint: one_index_for_AB2008

4. Analysis of Solution

4.1. Limitations of Solution

Despite the various strengths and practicality of our proposed "Best Timetable Generator", there are several limitations that are worth acknowledging:

A. Dependency on NTU STARS database and registration system

The current optimisation model is highly dependent on the NTU STARS database. Currently, our model assumes that students will be able to register for any course index included in the generated timetables. However, this does not reflect the competitive nature of NTU STARS registration system. When the STARS system "opens up" at the allocated time slot for each student, popular indices and modules often fill up within seconds of opening. The model does not account for index demand or real-time availability which may lead to the generation of ideal timetables that are not feasible during the actual registration.

Additionally, the model operates independently of NTU STARS and students are required to manually input their selected indices into the actual system. There is currently no real-time integration to check the slot availability during registration, which will reduce practicality.

Moreover, NTU's class schedules change at every academic term - change in index number, timeslots, classroom locations. This requires the dataset powering the model to be updated regularly. Without timely updates, the model may generate outdated or invalid timetables.

Lastly, the model does not consider NTU's exam schedule when generating timetables. Although exam timetables are typically available during the STARS registration, they are not included in the extracted database as the respective faculties do not publish them publicly due to privacy concerns. As a result, students may end up with multiple exams scheduled closely together which may lead to potential stress and workload imbalance during the examination period.

B. Limited module availability across days

Some modules offered at NTU are only available on specific days of the week. For instance, certain elective courses might only be available on Mondays and Thursdays. If a student chooses to block out these days due to personal commitments, the model will treat these restrictions as hard constraints. As a result, the model may be unable to generate feasible timetables if the selected modules and block-out timings conflict. As mentioned above, in such cases, the model will return an infeasibility message indicating

the specific constraints that caused the issue (e.g., "Infeasible constraint: block_time_THU_830"). This ensures clarity but may also limit the flexibility for students with strict availability constraints.

4.2. Recommendations

To enhance the functionality, usability and scalability of the "Best Timetable Generator", several improvements could be considered. These recommendations are aimed to address the current limitations and expand on the model's practical applicability in real-world settings:

A. Enhance model robustness against NTU database and registration constraints

To address the current model's dependency on NTU STARS database and registration system, several enhancements could be considered to further improve the practicality, accuracy and usability of our model during the course registration process. Firstly, to overcome the challenge of popular indices being oversubscribed during the registration period, historical data on index fill rates could be collected (through surveys, etc). This would help incorporate demand weighting. Recently, NTU has launched a new initiative to improve student's course registration and plan resources more efficiently (Figure 10). By assigning a weight or probability to each index based on its demand, the model can prioritise more accessible indices which will increase the chances of students securing their optimal timetables.

Secondly, direct real-time integration with the NTU STARS platform may not be feasible due to system access restrictions. However, the model could be implemented with an upload function where the users are able to input real-time availability snapshots. This would allow the model to filter out unavailable indices which will improve its relevance and applicability during the live registration.

Next, to address the need for regular database updates, a web scraping pipeline could be developed to automate the extraction of each semester's module and index information directly from NTU's scheduling portal. If this is not viable, having a structured data input template (e.g., Google Sheet or CSV template) could be designed. This would enable administrators or students to update the database easily and conveniently. Ultimately, reducing human error and ensuring smoother updates between each semester.

Lastly, future iterations of the model could include exam scheduling information once available to avoid back-to-back or same day exams. This could be implemented through additional constraints or warnings to flag such occurrences which would allow the students to make more informed decisions during the registration and add/drop period.

B. Implement soft constraints or trade-off suggestions for blocked days

To improve the experience and support decision-making for our users, soft constraints or trade-off suggestions could be introduced when the infeasibility occurs due to blocked days. Instead of only reporting the infeasibility constraints message, the model could recommend possible adjustments. This includes relaxing specific blocked time slots to accommodate essential or preferred modules. Alternatively, flagged warnings or optional solutions could be provided whereby certain blocked days are partially relaxed (e.g., attending a single class on a blocked day). This allows students to evaluate their options in order to secure their desired timetable. With this feature, it would increase transparency and provide students with more control over their scheduling decisions.

5. Conclusion

Planning the course timetable has long been a headache for all NTU students before the start of each semester. With so many indices to choose from, students often spend an unnecessarily long time manually planning their timetables, only to end up with clashing course slots or less-than-ideal class schedules. But now, with our Best Timetable Generator, we help NTU students not only simplify the planning process but also create optimized schedules that match with their personal preferences. As a result, this tool will help students reduce stress and anxiety before course registration. Additionally, in the new semester, students can achieve better time management, eventually improving their academic performance.

In future iterations, we will work on solving the limitations in our current generator. Enhancements will focus on syncing the model with NTU STARS's latest dataset and adding more course information, such as indices competition and real-time availability. We also plan to develop an app for running our program in order to make the generator more accessible to every NTU student.

To achieve these goals, we recognize that further support from NTU is necessary. Hence, we will reach out to the respective faculties to collaborate on the dataset synchronization and promote wider adoption of the model. Additionally, we will collect student feedback in order to fix any potential problems and enhance the model’s practicality. These future improvements will enable our generator to create more optimized solutions that reflect the real registration scenarios, ultimately helping every NTU student tackle their own “STARS WAR” ahead.

6. References

NTU. (n.d.). Class Schedule. https://wish.wis.ntu.edu.sg/webexe/owa/aus_schedule.main

ChatGPT chatlog:

Model: https://chatgpt.com/share/67fd0596-5db0-8011-990f-5b0f1d8d6f52

User Interface: https://chatgpt.com/share/67fd0fd5-b7c4-8011-94e6-e02485efed51

Appendix A

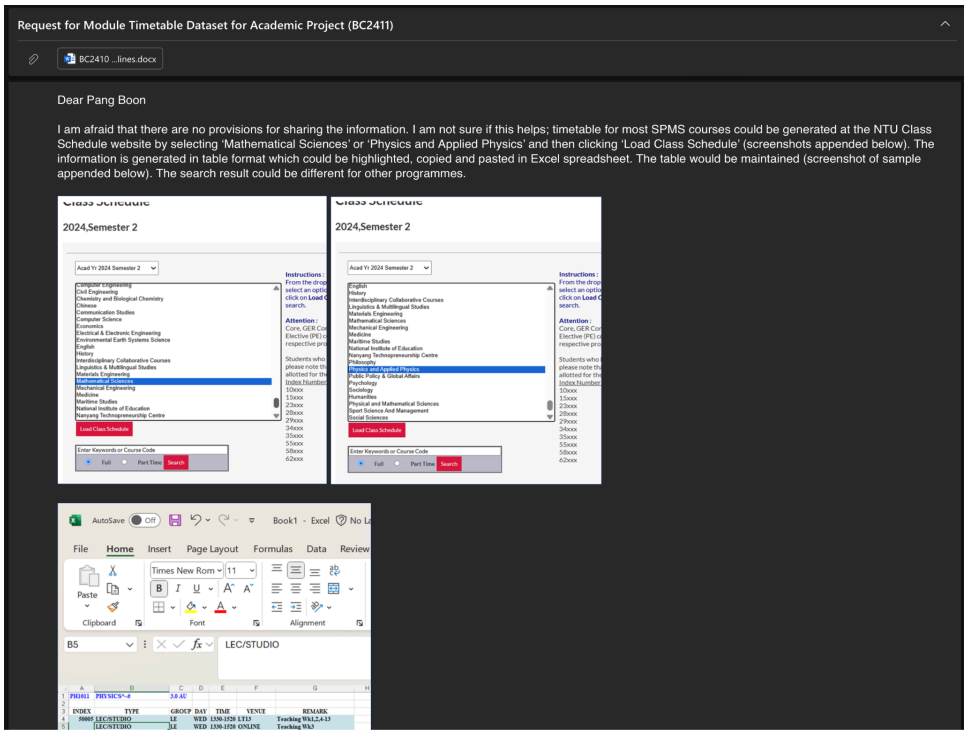

Figure 1: Request for structured data was denied due to confidentiality

	A	B	C	D	E	F	G	H
1	PH1011	PHYSICS*~#	3.0 AU					
2								
3	INDEX	TYPE	GROUP	DAY	TIME	VENUE	REMARK	
4	50005	LEC/STUDIO	LE	WED	1330-1520	LT13	Teaching Wk1,2,4-13	
5		LEC/STUDIO	LE	WED	1330-1520	ONLINE	Teaching Wk3	
6		TUT	C01	MON	1230-1320	LT13	Teaching Wk2,5-13	
7		TUT	C01	MON	1230-1320	ONLINE	Teaching Wk3	
8		TUT	C01	MON	1230-1320	LT19	Teaching Wk4	
9	58086	LEC/STUDIO	LE	WED	1330-1520	LT13	Teaching Wk1,2,4-13	
10		LEC/STUDIO	LE	WED	1330-1520	ONLINE	Teaching Wk3	
11		TUT	T1	WED	1530-1620	ONLINE	Teaching Wk3	
12		TUT	T1	WED	1530-1620	LT13	Teaching Wk2,4-13	
13	58087	LEC/STUDIO	LE	WED	1330-1520	LT13	Teaching Wk1,2,4-13	
14		LEC/STUDIO	LE	WED	1330-1520	ONLINE	Teaching Wk3	
15		TUT	T2	WED	1630-1720	ONLINE	Teaching Wk3	
16		TUT	T2	WED	1630-1720	LT13	Teaching Wk2,4-13	
17								
18								
19								
20	PH1012	PHYSICS A*~#	4.0 AU					
21	Remark: For students who A level Physics(applicable to FIT students)							
22								
23	INDEX	TYPE	GROUP	DAY	TIME	VENUE	REMARK	
24	50006	LEC/STUDIO	LEC	THU	0830-0950	LT25	Teaching Wk1,2,4-13	
25		LEC/STUDIO	LEC	THU	0830-0950	ONLINE	Teaching Wk3	
26		LEC/STUDIO	LEC	MON	1630-1720	ONLINE	Teaching Wk3	
27		LEC/STUDIO	LEC	MON	1630-1720	LT20	Teaching Wk1,2,4-13	
28		TUT	C01	THU	1330-1450	ONLINE	Teaching Wk3	
29		TUT	C01	THU	1330-1450	LHS-TR+52	Teaching Wk2,4-13	
30	58088	LEC/STUDIO	LEC	THU	0830-0950	ONLINE	Teaching Wk3	
31		LEC/STUDIO	LEC	THU	0830-0950	LT25	Teaching Wk1,2,4-13	
32		LEC/STUDIO	LEC	MON	1630-1720	LT20	Teaching Wk1,2,4-13	
33		LEC/STUDIO	LEC	MON	1630-1720	ONLINE	Teaching Wk3	

Figure 2: format before data preprocessing

Course Code	INDEX	TYPE	GROUP	DAY	TIME
AB0403	540	SEM	1	MON	0830-1020
AB0403	541	SEM	2	MON	1030-1220
AB0403	542	SEM	3	MON	1430-1620
AB0403	543	SEM	4	MON	1630-1820
AB0403	544	SEM	5	TUE	0830-1020
AB0403	545	SEM	6	TUE	1030-1220

Figure 3: format after data preprocessing

ACADEMIC UNITS REQUIRED FOR GRADUATION – BATCH ADMITTED IN AY2023

BACHELOR OF BUSINESS - BUSINESS ANALYTICS (BA)

Year of Study	Number of Academic Units (AUs)					
	Major Requirement		Interdisciplinary Collaborative Core		Broadening and Deepening Electives (BDE)	Total
	Core	Major Prescribed Electives	Common Core	Foundational Core		
1	10/12		4/4	3/0		17/16
	12/10		4/4	5/8		21/22
2	17		3			20
	4	3	6	5		18
3		3			12	15
	3	3			6	12
Total	46	9	17	13	18	103

Figure 4: overview of a typical NTU Business Analytics student. Showing average of 17 AU per Semester

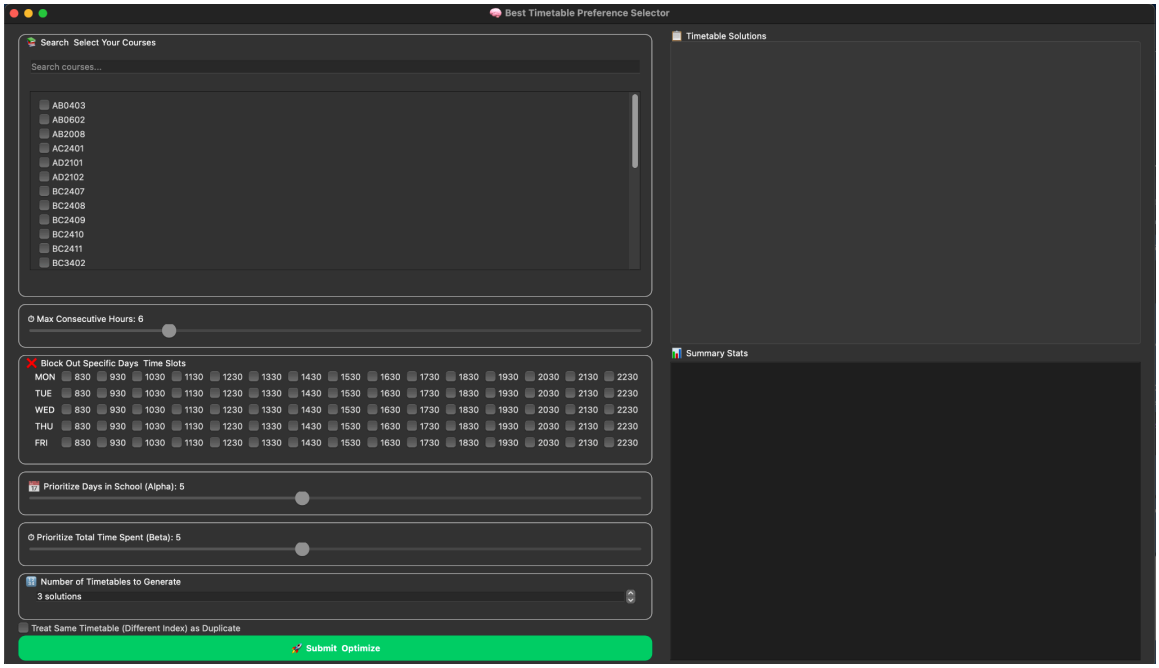

Figure 5: Appearance when start

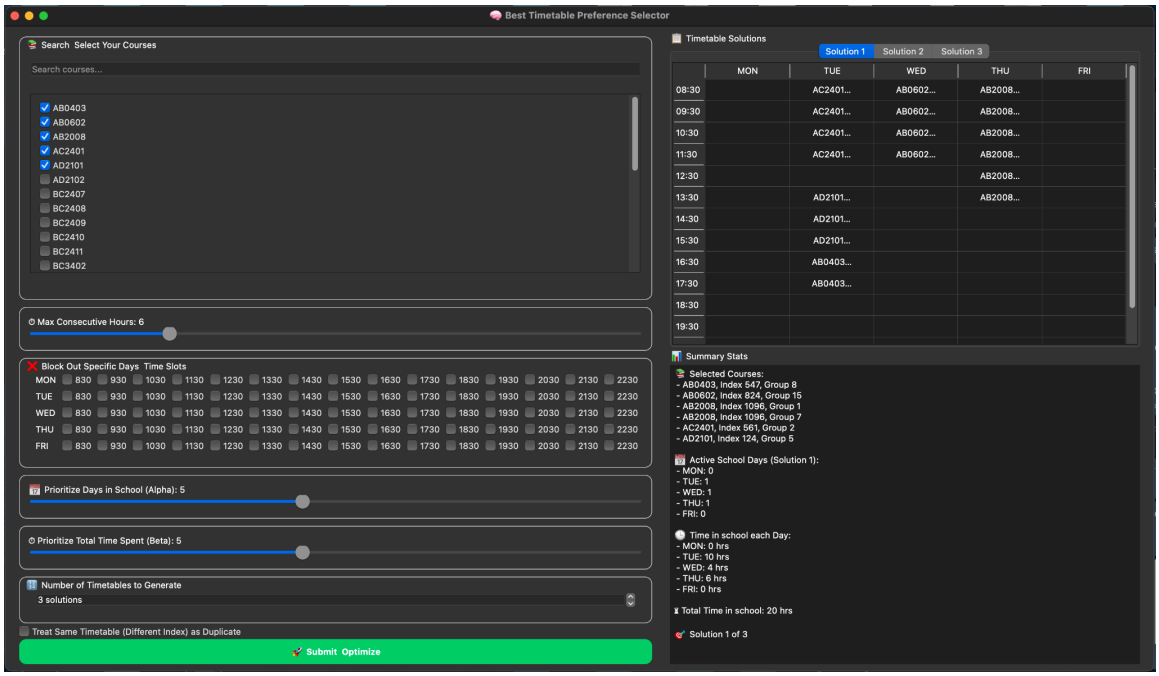

Figure 6: Appearance after user optimise

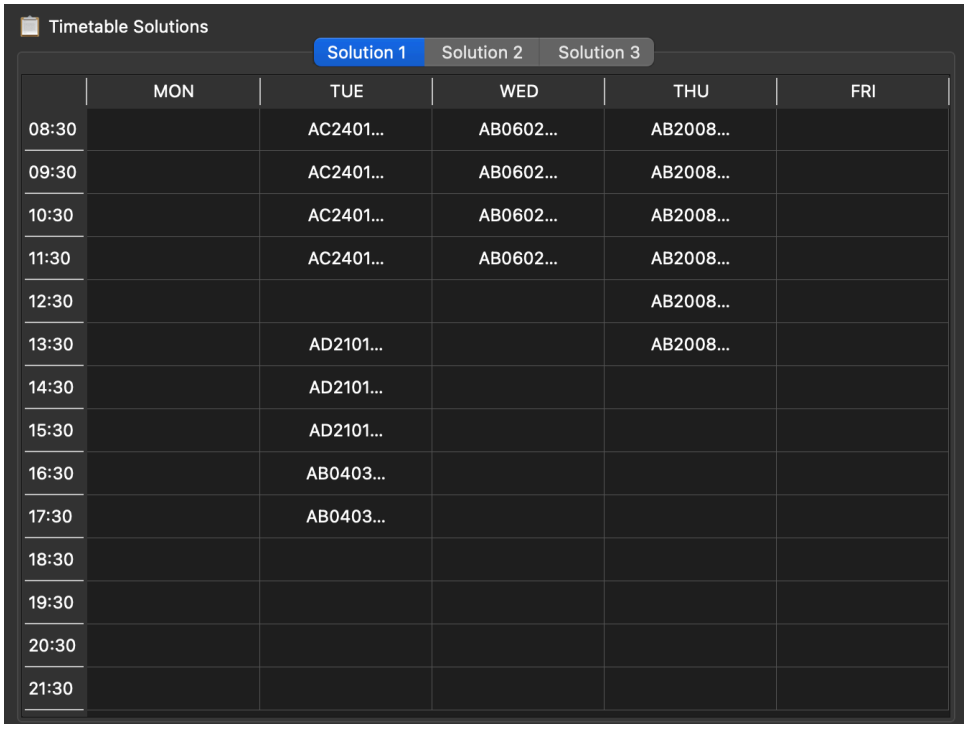

Figure 7: Timetable Panel

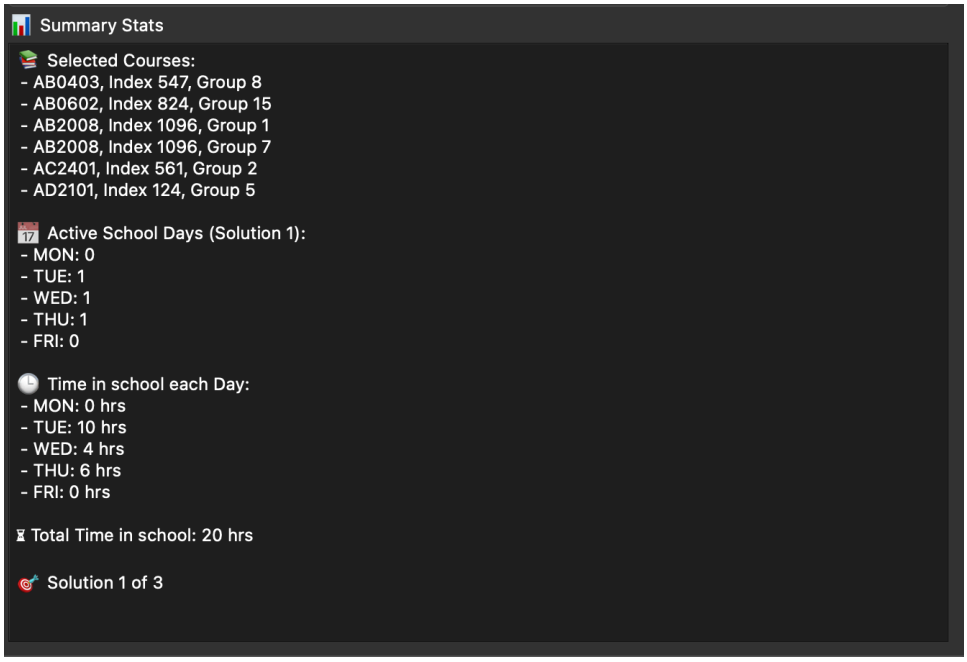

Figure 8: Summary Stats with feasible solution

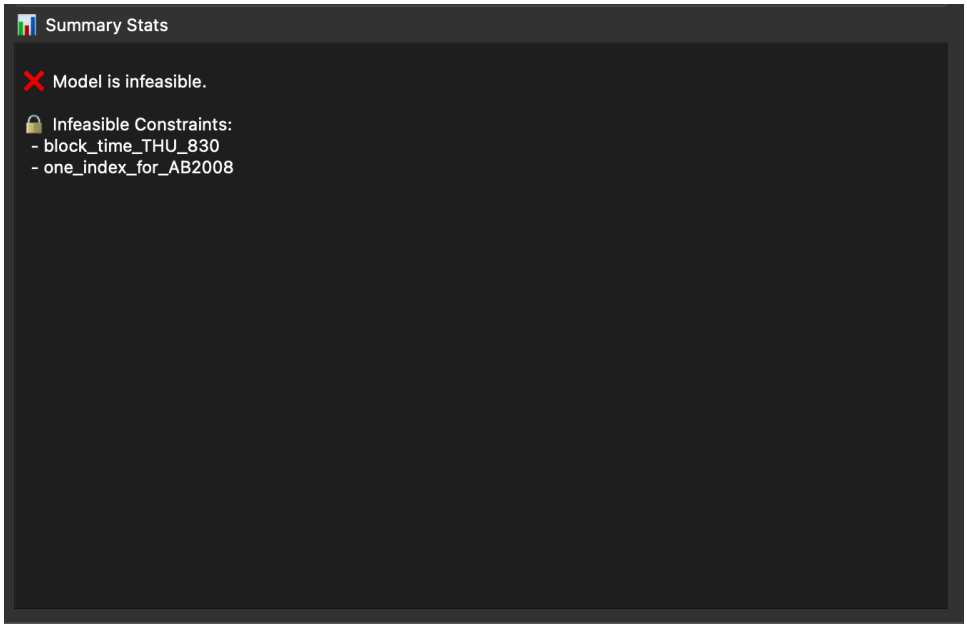

*Figure 9: Summary Stats with **no** feasible solution **with reasons***

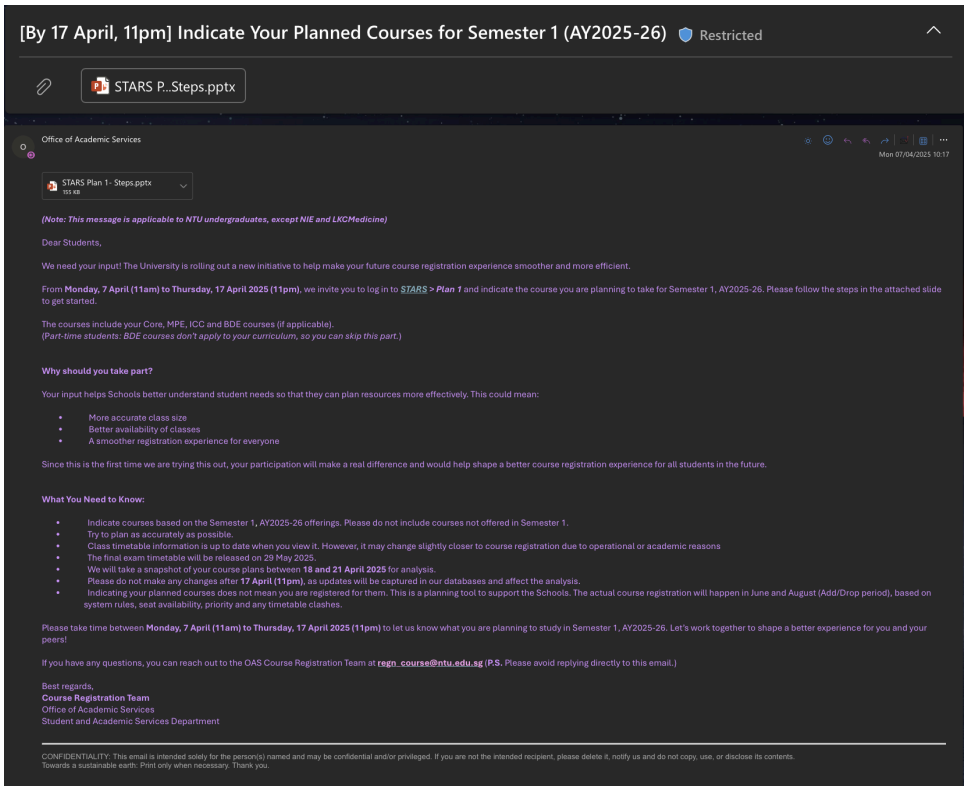

Figure 10: New initiative by NTU

Appendix B

Decision Variables	Type	Description
x_i	Binary	1 if index i is selected, 0 otherwise
y_d	Binary	1 if any class occurs on day d, 0 otherwise
z_start_d	Integer	Start time of the earliest class on day d
z_end_d	Integer	End time of the latest class on day d
z_total_d	Integer	Total duration spent in school on day d
$class_{d,t}$	Binary	1 if a class is scheduled on day d at time slot t, 0 otherwise

Parameters	Description
Index	Set of all index IDs
$Index_c$	Set of all index belonging to course c
Course	Set of all courses
Days	Set of all days in the week = {MON, TUE, WED, THU, FRI}
Timeslot	Set of all possible time slot (e.g., 0830, 0930, …, 2230)
α	Weight for minimizing number of school days
β	Weight for minimizing total time spent in school
$Interval_i$	List of tuples (d,s,e) = (day, start, end) for index i
$Block_{(d,t)}$	Set of blocked time slots on day d and time t
T_{maxC}	(Number of possible time_slots) - (max consecutive hours)
window	A continuous block of max_consecutive_hours + 1 time slots
maxH	Maximum allowed consecutive hours