



AiCE Undergraduate Research Project Final Report

Fall 2025 Semester

ScheDool

Team Members

Nunthatinn Veerapaiboon, Poon, nveerap@cmkl.ac.th
Atchariyapat Sirijirakarnjaroen, Beam, asiriji@cmkl.ac.th
Nachayada Pattaratichakonkul, May, npattar@cmkl.ac.th
Petch Suwapun, Diamond, psuwan@cmkl.ac.th
Thanawin Pattanaphol, Win, tpattan@cmkl.ac.th

Advisor

Dr. Charnon Pattiyanon, charnon@cmkl.ac.th

12 December 2025

Table of Contents

1.1 Problem Statement	4
1.2 Project Solution Approach	5
1.3 Project Objectives	5
Genetic Algorithm	6
Constraints Programming	6
Web Application	6
LLM	6
TrWIN	7
2.1 Fundamental Theory and Concepts	7
2.1.1 School Scheduling Pipeline	8
2.1.1.1 Preparation and Data Pre-load	8
2.1.1.2 Schedule Generation and Optimization	8
2.1.1.3 Publication and Ongoing Maintenance	8
2.2 Technologies	9
2.2.1 Langchain	9
2.2.2 Ollama	9
2.2. React.js	9
2.3 Related Research	11
Genetic Algorithms	11
2.4 Market Analysis	12
3.1 Survey / Requirements Gathering	14
3.2 Standard Input Format	16
3.3 Designing UX / UI	17
3.4 Design Algorithms (Backend Core Logic)	18
3.4.1 Pre-Scheduling Data Processor	18
3.4.2 Genetic Algorithm (GA) Core Scheduler	18
3.5 Implementation (Programming)	19
3.5.1 Backend and Core Logic	19
3.5.2 Frontend (User Interface)	19
3.5.3 Data Storage Layer	19
4.1 Sample Dataset	21
4.2 User Journeys	22
Admin Flow	22
Teacher's Flow	27
4.3 Backend Implementation	35
4.3.1 Pre-Scheduler	35
4.3.2 Genetic Algorithm Solver	35
5.1 Summary of Accomplishments	36
5.2 Issues and Obstacles	36
5.3 Future Directions	36
5.4 Lessons Learned	37
References	38

Chapter 1

Introduction

1.1 Problem Statement

In recent years, several schools across Thailand face problems with timetable scheduling, especially public schools of the Office of the Basic Education Commission (OBEC) known in Thai as สำนักงานคณะกรรมการการศึกษาขั้นพื้นฐาน (สพฐ.), such as Bodindecha (Singha-Singhaseni) School. These schools require a team of no fewer than eight faculty members and up to two weeks to prepare the entire school's academic timetable of the semester.

For the case of most major Thai public schools, since there are two semesters in an academic year and more than 300 teachers to schedule, the task of creating a school-wide class schedule is an extremely complex task. Apart from being a time consuming and tedious process for the faculty, in order to create a large-scale school-wide timetable, many other factors are needed to be considered, such as, assigning teachers' teaching duties and responsibilities, surveying teachers' constraints, and creating a timetable that satisfies many other future constraints that may arise as the academic year progresses.

From our field surveys and interviews with the school faculty members, we found that they do have systems that assist with scheduling, however, said systems are limited in functionality and faculty members still need to spend a significant amount of their time in manually planning and rearranging the schedules. [See Figures 1, 2, and 3] This demonstrates how current scheduling solutions that are used in some schools are not mitigating manual tasks as much as they might have been intended to be.

Realizing these problems, we decided to choose the topic of “class timetabling / scheduling” for our URD project. We hope this project will help make timetable creation easier, faster and more efficient by using AI technology in assisting with scheduling, reducing preparation time from two weeks to just several hours and reducing manual work by at least 70%. This way, faculty members can spend less time on logistics and more time on their core role as the nation's educators.

1.2 Project Solution Approach

Our project aims to approach the inefficiency of manual school timetable generation through the development of a web-based Software-as-a-Service (SaaS) scheduling system; the core problem of an inefficient, time-consuming, and confusing manual way of scheduling will be solved through automation, reducing inefficiencies and the teachers' workload.

Our solution will employ Genetic Algorithm (GA), to automatically generate school timetables, ensuring all essential hard constraints (e.g. avoiding teacher/classroom conflicts) are always met and respected. The core scheduling process (GA) will be complemented by using a Large Language Model (LLM) during the preprocessing stage to interpret and structure diverse, often ambiguous, school requirements and rules into a machine-readable format for the GA. This approach ensures maximum flexibility in adapting to the unique needs of various institutions. The automation will simultaneously optimize for soft constraints that significantly improve schedule quality and fairness (e.g., balancing teacher workloads, minimizing consecutive teaching blocks, and accommodating teacher preferences).

Most importantly, the platform is designed for flexibility and scalability to accommodate the diverse and unique requirements of different educational institutions and curricula across Thailand. By offering role-based access for administrators, teachers, and students, the system will not only generate schedules but also streamline day-to-day operations like schedule viewing, managing substitutions, and providing clear communication, ultimately cutting down manual workload and establishing a new, efficient standard for the Thai educational system.

1.3 Project Objectives

The primary objective of this project is to create and deliver a comprehensive, adaptable, and efficient web-based school scheduling system. The most important outcomes and deliverables are summarized below:

1. **Automated Timetable Generation Engine:** We plan to develop a sophisticated scheduling engine that automates the creation and adjustment of school timetables. This engine will be capable of efficiently handling a large number of hard constraints (preventing conflicts) and soft constraints (improving quality and fairness, such as balancing workloads and respecting preferences). It will be based on a selected algorithmic approach (e.g., Genetic Algorithms) to ensure flexibility, scalability, and high-quality results.
2. **Multi-Role Web-Based SaaS Platform:** The project will deliver a fully functional, role-based web application accessible via desktop and tablet browsers, operating as a Software-as-a-Service (SaaS) platform. This platform will provide distinct access levels for **Administrators** (for generation and finalization), **Teachers** (for viewing schedules and managing substitutions/exchanges), and **Students** (for viewing personalized weekly schedules). This accessibility ensures ease of use and rapid deployment without requiring installation on school servers.
3. **System Adaptability and Integration Toolkit:** A key deliverable is a system designed to adapt to the unique constraints of diverse Thai schools (e.g., fixed periods, team-teaching, specific room allocations). To ensure smooth integration with existing school management workflows, the platform will include essential functionalities like data export options to standard formats such as CSV and Excel. This will enable schools to seamlessly adopt the new timetable system without major upheaval to their current administrative processes.

Chapter 2

Background

Genetic Algorithm

Genetic algorithm is a type of a metaheuristic search algorithm, modeled after Charles Darwin's theory of natural selection. Genetic algorithms are commonly used to generate high-quality solutions to optimization and search problems, in which the algorithm modifies a population of individual solutions. In each iteration or "generation", the genetic algorithm selects individuals (solutions) from the current population to be parents and uses them to produce the children (newer solutions) for the next generation. The algorithm iterates this process over successive generations and as it goes through, the population of solutions evolves towards an "optimal solution". (*What Is the Genetic Algorithm?*, n.d.)

Constraints Programming

Constraints Programming, also known as "constraint optimization" is a paradigm for identifying "feasible solutions" out of a very large set of candidates, in which the problem can be modeled in terms of arbitrary constraints. The paradigm is based on feasibility rather than optimization, as in, the paradigm is more focused towards finding a practicable solution rather than the *perfect* solution, it focuses on the constraints and variables rather than the objective function. ("CP-SAT Solver," n.d.)

Web Application

A web application is software that is run on a remote (or a local) server and is served over the web browser, removing the need for the software to be installed on the system itself. (*What Is a Web App? - Web Application Explained - AWS*, n.d.) They are generally compatible with most computers. Most web applications are built on the basis of the Hypertext Markup Language (HTML), Cascading Style Sheet (CSS) and JavaScript (JS). However, some web applications are built with more sophisticated frameworks, such as, [React.js](#), [Vue.js](#), and others.

LLM

Large Language Models is a type of language model that is trained via self-supervised machine learning on a significantly large amount of text, designed for natural language processing tasks, specifically, text / language generation. (*Contributors to Wikimedia projects, 2025*) A very prevalent and popular example of such technique are ChatGPT, Gemini, and Copilot; these are considered "generative pre-trained transformers" or GPTs. (*Stryker, n.d.*) LLMs can be fine-tuned for specific tasks or guided by entering a user prompt; said LLMs mostly run on datacenters, sometimes owned by the company that provides the LLM itself, e.g. OpenAI, Google, Microsoft, and etc.

TrWIN

Trwin is a Windows-based application designed to automate and manage complex academic timetabling. The software supports both Standalone and Local Area Network (LAN) architectures, allowing for multi-user collaborative scheduling to reduce processing time. It features a robust logic engine capable of handling up to 7 days/14 periods per day, managing complex constraints (locked periods, double blocks, rotating schedules), and performing real-time conflict detection for teachers, students, and classrooms. The current version (6.09) supports data export to Microsoft Excel and integration with third-party PDF drivers for digital reporting. (*โปรแกรมจัดการตารางสอน, n.d.*)

2.1 Fundamental Theory and Concepts

2.1.1 School Scheduling Pipeline

In every semester, Thai public schools undertake a common, multi-stage pipeline to create a comprehensive schedule for students, teachers, and rooms. While curriculum requirements and specific constraints vary widely across institutions, the core workflow remains consistent and can be broken down into three main phases: Preparation, Generation, and Maintenance.

2.1.1.1 Preparation and Data Pre-load

This initial phase focuses on defining all necessary inputs for the scheduling process.

Administrators and department heads collaborate to establish the 'Pre-Load' data. This involves:

- Curriculum Design and Adjustment: Finalizing the subject list and overall structure for the upcoming semester.
- Teacher and Workload Assignment: Determining teacher hiring needs, assigning specific teachers to subjects and student classes, and specifying detailed workload constraints.
- Information Gathering: Collecting essential requirements from every department (e.g., Math, Science, Social Studies, Foreign Languages), including fixed periods, required room types, and other specific teaching needs.

A significant pain point in this phase is the inconsistent data format. Information is often documented in diverse and unstructured ways across different departments, requiring administrators to spend considerable time manually standardizing and consolidating this data before scheduling can begin.

2.1.1.2 Schedule Generation and Optimization

Once all the structured data is collected, the core scheduling activity commences. Administrators must iteratively build and adjust the timetable until all requirements and constraints are met:

- Constraint Satisfaction: Admins meet to brainstorm and repeatedly add and adjust activities, primarily focusing on fitting all necessary classes without generating hard conflicts (e.g., one teacher or room double-booked).
- Quality Optimization: Beyond just avoiding conflicts, this phase requires balancing the schedule according to soft constraints to ensure fairness and quality. This includes distributing teacher workloads evenly across the week and avoiding long blocks of consecutive teaching.

The primary challenge here is the sheer volume of activities and the manual, repetitive nature of the trial-and-error process. Admins must constantly adjust and reassess the schedule, which is labor-intensive and highly prone to human error, especially when trying to balance curriculum needs with fairness requirements.

2.1.1.3 Publication and Ongoing Maintenance

The final phase involves deploying the complete schedule and managing necessary changes throughout the semester:

- Publication: The finalized schedule is published, making it accessible for teachers and students to view their respective timetables.
- Maintenance: This is a continuous process where administrators handle unexpected events, such as teacher absences or unexpected school activities, which necessitate schedule shifts. A critical task within this phase is managing teacher substitution and other real-time adjustments to ensure minimal disruption to learning activities.

2.2 Technologies

2.2.1 Langchain

LangChain, a product created by LangChain (the company), is an open-source framework which provides “a pre-built architecture” that is able to integrate with any large language models or tools. Essentially, the framework simplifies the process of developing applications that use LLMs, e.g. chatbots and AI agents. The product is available in Python and JavaScript based libraries. (*LangChain Overview, n.d.*)

Since LLMs are not standalone applications: they are pre-trained statistical models that *must* be used with an application. A prime example of this is ChatGPT. ChatGPT, in its entirety, is an application that essentially wraps around GPT models, e.g. GPT-3.5, GPT-4, and GPT-5 models. The LLM itself does not provide an interface for the user, rather, the application provides the interface and passes on the input to the models. (*Bergmann & Stryker, n.d.*)

In the context of a development environment, LangChain plays the role of the application, which provides a way for the developer to interact with the models themselves through programmed functions.

2.2.2 Ollama

As mentioned earlier in the fundamental theory or concepts section, Large Language Models or LLMs mostly run on large datacenters around the size of warehouses, however, it is also possible to run them on local machines, such as, laptops or mini-PCs; it may sound counterintuitive since LLMs require a lot of computing power to run, but on a smaller scale, it is able to be ran on a laptop.

In order to run these models locally, it is recommended to use Ollama, a community-driven open source project, licensed under the MIT license, which aims to provide a “lightweight, extensible framework for building and running language models on the local machine” (*Ollama, n.d.*). Ollama provides a simple interface for developers to run various LLMs, e.g. gpt-oss, DeepSeek-R1, Gemma 3, etc. under a couple of minutes.

In the context of programming, Ollama provides several ways for developers to interface with it programmatically. One of those methods is through REST APIs on localhost; enabling the developers to send GET and POST requests to interact with the LLM. This makes LLM integration much easier and simpler.

2.2. React.js

React is a JavaScript framework developed by Meta that lets developers build user interfaces (UIs) in a simpler manner through the introduction of reusable components, such as, custom buttons, textboxes, and other elements; making projects easier to maintain and organize.

React.js is currently licensed under the MIT license, therefore, is free and open source, meaning they are free to use without any restrictions. This provides developers with more flexibility and freedom to use, modify, adapt React into their projects.

Additionally, React also provides developers with great performance and efficiency, i.e., it provides fast rendering by employing the Virtual DOM (Document Object Model) and the reconciliation algorithm; minimizing unnecessary re-renderings of pages, provides good responsiveness for applications, better performance optimization through code splitting and lazy loading, along with cross-platform capabilities through its sibling: React Native. (*GeeksforGeeks*, 2022)

2.3 Related Research

Genetic Algorithms

Our project's core scheduling agent, the "solver", named after its predecessor that was implemented using Google's CP-SAT solver, employs the use of Genetic Algorithm for "completing" the schedule that were provided by the pre-scheduling agent, in which, filled in some of the hard constraints to alleviate the workload of the main solver. For us to come to the conclusion of using Genetic Algorithms, we researched through several projects and papers to help formulate our possible solutions.

A. S. Budiarto, N. M. Satvika Iswari and E. M. Dharma focuses on refining GA parameters for university course timetabling. Researchers tested population size, crossover rates, and mutation probabilities on real datasets from a Vietnamese university, achieving a 25% reduction in scheduling conflicts compared to manual methods. The optimized GA generated feasible timetables in under 5 minutes, demonstrating scalability for large semesters. (*Budiarto et al., 2024*)

Lingzhi Yan, Hua Deng. proposes a hybrid GA for classroom assignment in higher education. It incorporates local search operators to handle soft constraints like instructor preferences, evaluated on benchmark datasets from European universities. Results showed a 15-20% improvement in solution quality over standard GAs, with near-optimal schedules produced in reasonable computation times. (*Yan & Deng, 2025*)

Nasien, Dewi & Andi applies GA to K-12 school scheduling in Indonesia, modeling chromosomes as timetable grids with fitness functions penalizing overlaps. Experiments on a 30-class dataset yielded conflict-free schedules 92% faster than heuristic approaches. The method proved robust for dynamic adjustments, like teacher absences. (*Nasien & Andi, 2022*)

Our solution most resembles Nasien, Dewi & Andi's implementation as they implement their genetic algorithm-based system specifically for school timetabling generation. What they also have in common to our system is their simplification of their dynamic constraints; both encompasses a static model, i.e., both implement a fitness function that penalizes conflicts, such as, class overlaps, while lacking the way to handle extra constraints e.g. teacher absences, enrollment shifts, etc. Thus, both encompass similar limitations. (*Nasien & Andi, 2022*)

During the next term, we plan to tackle the aforementioned limitations directly by examining or researching ways that we can adapt our fitness function to, not only fit our current "Schedule validation framework" that covers the three aspects of: Completeness, Fairness, and Correctness, but to also support other ways in implementing our genetic algorithm-based solver; e.g. similar to the other two implementations that focuses on parameter tuning and a hybrid model + local searches.

2.4 Market Analysis

Feature / Capability	TRWIN (โปรแกรมจัดตารางสอน, n.d.)	NextSchool (Products – Nextschool, n.d.)	ScheDool
Platform Type	Windows desktop (stand-alone or LAN)	Web-based school system	Cloud-based scheduling system
Automatic Timetable Generation	✗ Not documented	✗ Not documented	✓ Core AI feature
Manual Timetable Creation	✓ Yes	✓ Basic creation	✗ Not required (auto-generated)
Constraint Handling	✓ Complex constraint setup (teacher/room/class rules)	✓ Can define groups, subjects, rooms, electives	✓ AI handles constraints automatically
Conflict Detection	✓ Has conflict-checking features	✗ Not documented	✓ Conflict-free by design
Real-Time Timetable Access	✗ No	✗ Not documented	✓ Yes (teachers + students)

Timetable Updates Mid-Semester	Manual adjustments possible	Cancel, substitution, room change supported	✓ Fully flexible automatic updates
Multi-User Support	LAN multi-user	Web multi-user	✓ Cloud multi-user
Target Users	School scheduling staff	School administrators & teachers	Scheduling admins, OBEC large schools
Strength for Large Schools	✓ Proven with 1,934 schools	✓ Proven with 37 schools	✓ Purpose-built for complex schools

Chapter 3

Methodology

3.1 Survey / Requirements Gathering

The first and most important step in our project was figuring out the exact problem we needed to solve. We started by talking to different school administrators, and they all agreed on one major difficulty: **the complex and time-consuming process of creating the school timetable.**

To understand this challenge in detail, we moved on to a deeper investigation of the current scheduling pipeline. We conducted focused interviews with the administrators who are directly responsible for making the master schedule. Our questions were designed to collect specific information about three key parts of their work:

1. **Gathering and Preparing Data:** We carefully recorded how administrators collect all the starting information, such as the list of classes, who teaches them, and which rooms are needed. A main issue we found was that this information often comes in many different and messy formats from various departments. This means the administrators have to spend a lot of time manually cleaning up and organizing this data before they can even begin scheduling.
2. **The Step-by-Step Scheduling Process:** We asked them to explain their entire scheduling process, step by step. This helped us map out the necessary rules they must follow (**hard constraints**, like preventing conflicts) and the helpful rules they try to follow to make a good schedule (**soft constraints**, like balancing how much teachers work). We focused on finding the tasks they do repeatedly that waste the most time or cause errors, so our future system can automate them.
3. **The Reasons Behind Decisions (Logics):** We explored the unwritten rules and judgment they use when finalizing a timetable. This included why they place certain subjects at specific times, how they decide if a teacher's workload is '**fair**,' and what techniques they use to improve the overall quality of the schedule. Furthermore, we discovered that some schools have special teaching methods or requirements that are **not officially documented in the curriculum**. Only the teachers within that school know these specific details. Understanding these unique, informal constraints is crucial to making sure our system creates a timetable that is truly usable and respected by the school staff.

3.2 Standard Input Format

Following the requirements gathering phase, a major finding from our interviews was the **lack of any existing standardized data format** across or even within individual schools. Administrators are currently forced to manually create a final, consolidated document from various sources before they can even begin the scheduling process.

To directly address this crucial pain point, our objective is to define and implement a **unified data format** that will serve as the single, consistent input for our automated scheduling system.

This standard format is being developed by carefully analyzing and synthesizing the most common and essential fields found in the existing, non-standard documents we collected from each school. The final format will be designed to ensure two main things:

1. **Data Consistency and Integrity:** By requiring all scheduling information to conform to a strict structure, we guarantee that the data used by the system is clean, complete, and reliable.
2. **Completeness:** The format will be comprehensive, including all necessary information that administrators require for the *entire* scheduling task—from teacher workloads and classroom constraints to special curriculum requirements.

To maintain the quality of the input, the system will also include a **format validator (or input validation tool)**. This tool will automatically check the data files submitted by the school against our standardized rules, immediately flagging any errors or inconsistencies. This step eliminates the administrative burden of manual data cleanup, ensuring a smooth transition to the automated scheduling process.

3.3 Designing UX / UI

We want to ensure the design is modern and user-friendly so the learning curve stays low. Our main color palette is blue, with additional primary colors like green, red, and orange. The goal is to keep it similar to the existing tool so users can adapt quickly, while making it more visually pleasing and easier to navigate. We identified which parts teachers and administration are already familiar with and kept those work flow patterns intact while updating the look and feel to be more visually appealing and easier to navigate. This way users don't have to relearn everything from abstract, they can just transfer their editing knowledge to a new system.

We organize the interface to present information clearly using consistent spacing and typography and since scheduling invoice juggling a lot of data all at once we make sure users can quickly scan and understand what they are looking at. We also hide advanced features until needed, keeping the main interface clean and focused.

Our color system provides immediate visual feedback: green indicates success states and available slots, red highlights conflicts and errors that need fixing, and orange flags warnings requiring attention. This allows administrators to identify issues at a glance without reading detailed messages.

The dashboard displays scheduling progress, pending conflicts, and quick access to frequently used functions. The data upload interface supports drag-and-drop with real-time validation feedback, showing exactly where errors occur and allowing users to preview data before processing.

For the class exchange workflow, we built a dedicated teacher portal where teachers can submit swap requests with a visual calendar showing available options. Administrators review requests through a side-by-side comparison view, with the system automatically checking that exchanges won't create new conflicts.

The main scheduling interface features a grid-based timetable with drag-and-drop functionality, color-coded teacher workload indicators, and multiple view modes (by class, teacher, or room). Hovering over any slot reveals detailed constraint information, helping administrators make informed scheduling decisions.

3.4 Design Algorithms (Backend Core Logic)

This phase focuses on developing the core computational engine responsible for automated timetable generation. We conducted research and comparative analysis on various solver approaches, including traditional Constraint Programming (CP) and Constraint Satisfaction Problems (CSP). While CP and CSP are excellent for proving exact solutions, we found they often become inefficient or computationally prohibitive when dealing with the extremely **large dataset and high volume of complex, weighted soft constraints** characteristic of real-world school scheduling. Therefore, we selected the **Genetic Algorithm (GA)** as the primary method. The GA is better suited for this problem because of its robust ability to efficiently find near-optimal solutions by prioritizing the satisfaction of both hard constraints and the numerous, often conflicting soft constraints (like fairness and teacher preferences).

The GA approach will be implemented in two distinct, sequential sub-processes:

3.4.1 Pre-Scheduling Data Processor

This is the initial, non-iterative phase designed to handle all data entry, cleaning, and processing before the main GA run. This stage is crucial for efficiency and for processing the unique human-defined constraints we identified:

- **Data Cleaning and Standardization:** It validates the input against our **Standard Input Format (Section 3.2)**, ensuring data consistency and integrity.
- **Handling Fixed Constraints:** This process permanently places activities that won't change, no matter how many generations the GA runs. Examples include fixed appointments (like a staff meeting every Monday at 8 AM), non-negotiable teacher availability slots, and activities that require a specific time slot (like a remedial class).
- **Human-to-Machine Constraint Translation (LLM):** We specifically designed this process to leverage the **Large Language Model (LLM)**. The LLM's role is to interpret and process the natural language constraints gathered from users (e.g., "Teacher X prefers late mornings" or "Complex subjects should be spread evenly") and transform them into precise, weighted, machine-readable parameters and constraints that the Genetic Algorithm can then mathematically process.

3.4.2 Genetic Algorithm (GA) Core Scheduler

This is the main, iterative optimization engine that generates the final timetable:

- **Constraint Modeling:** All constraints, both hard and the LLM-processed soft constraints, are translated into a formal mathematical model.
- **Fitness Function Development:** The core of the GA is the **Fitness Function**, which evaluates the quality of every potential timetable. It strictly penalizes violations of hard constraints (resulting in an infeasible solution) and rewards high scores for satisfying the weighted soft constraints.
- **GA Mechanism Design:** We will define the specific GA operators: **Representation** (how a schedule is encoded), **Selection** (choosing fit schedules), **Crossover** (combining features of fit schedules), and **Mutation** (introducing random changes) to efficiently search for the highest fitness score (the optimal schedule).

3.5 Implementation (Programming)

This phase details our plan for building the complete web-based system (**Software-as-a-Service**, or **SaaS**). The system will be built using three main technical components: the Backend, the Frontend, and the Data Storage Layer.

3.5.1 Backend and Core Logic

The **Backend** serves as the operational hub, housing all the scheduling intelligence. For top speed and efficiency, the core scheduling service—which combines the data processing and the GA—will be implemented as a **single, fast service** using the programming language **Rust**. Rust is chosen for its focus on reliable, high-speed performance during the complex, iterative process. This service handles the entire scheduling pipeline seamlessly: it first performs the **Pre-scheduling and Data Processing** (including the LLM translation of human constraints), and then immediately moves to the **Core Scheduling Engine** to run the Genetic Algorithm (GA) and generate the optimized timetable. This unified approach ensures high throughput and consistency for the critical task of timetable generation.

Crucially, this service will also expose a dedicated **Schedule Evaluation API endpoint**. This API can be called by the frontend whenever an administrator makes a **manual change** to the schedule output (e.g., dragging and dropping a class). The API will immediately re-run the fitness calculation on the modified schedule, providing real-time feedback to the user on whether the change created any conflicts (hard constraint violations) or negatively impacted the overall quality score (soft constraint adherence).

Other essential services within the backend are separated to maintain a scalable and modular architecture. These supporting services include the **Data Access Services (DAS)**, which manage efficient communication and interaction with the storage layer; a **DB ORM (Object-Relational Mapper)**, which helps abstract and standardize complex database commands; and a dedicated **Authentication Service**, responsible for managing user logins, roles, and maintaining system security. Separating these functions allows for focused development and easier updates to security or data handling protocols without impacting the core scheduling engine.

3.5.2 Frontend (User Interface)

The **Frontend** is what the users see and interact with. It will be developed using the **React** framework. This lets us build a fast, responsive website that works well on both desktop computers and tablets. We will focus on designing easy-to-use interfaces for the different roles (**Admin, Teacher, Student**). This includes simple ways for admins to put in data, adjust the schedule (like dragging and dropping classes), and clear views of the weekly timetables for everyone.

3.5.3 Data Storage Layer

The system will use a mix of storage types. All the structured, changing data (like teacher names, classes, and current requirements) will be stored in a reliable **SQL Database**, specifically **PostgreSQL (PSQL)**. However, for **full-scale deployments and long-term storage of documents and static files** (e.g., historical schedule exports, reports, and large input data files), we plan to use an **Object Storage solution like R2**. This combination offers good performance for daily tasks while providing the scalability and low cost needed for large-scale, future deployments.

3.6 Testing & Iterative Improvement

Testing is split into two critical areas. First, **Core Algorithm Testing (Verification)**: We use the **Surawiwat sample dataset (Section 4.1)** to test the core scheduling service. The primary check is to verify that the **Genetic Algorithm (GA)** always produces **valid schedules** (no hard conflicts) and achieves a high quality score (optimal soft constraint satisfaction) within fast processing times. Second, **User Acceptance Testing (UAT) (Validation)**: The complete web platform will be tested by administrators and teachers from Surawiwat School. This UAT focuses on the system's **usability, performance, and functionality** in a real setting. We validate the ease of data input, the responsiveness of the adjustment features, and the clarity of feedback from the **Schedule Evaluation API**.

Based on all testing feedback, we use a continuous **Iterative Improvement Cycle** to refine the product before full deployment. This cycle involves several steps: Any constraint errors or low-quality scores found in the GA will lead to an immediate **Logic Refinement** and adjustment of the constraint model and the LLM's data translation rules. And issues found during UAT will result in quick **Usability Enhancement** modifications to the React frontend and supporting services (e.g., simplifying the Schedule Evaluation API's feedback display). This focused approach ensures the final system is technically sound, easy to use, and directly solves the problems faced by school staff.

Chapter 4

Results

4.1 Sample Dataset

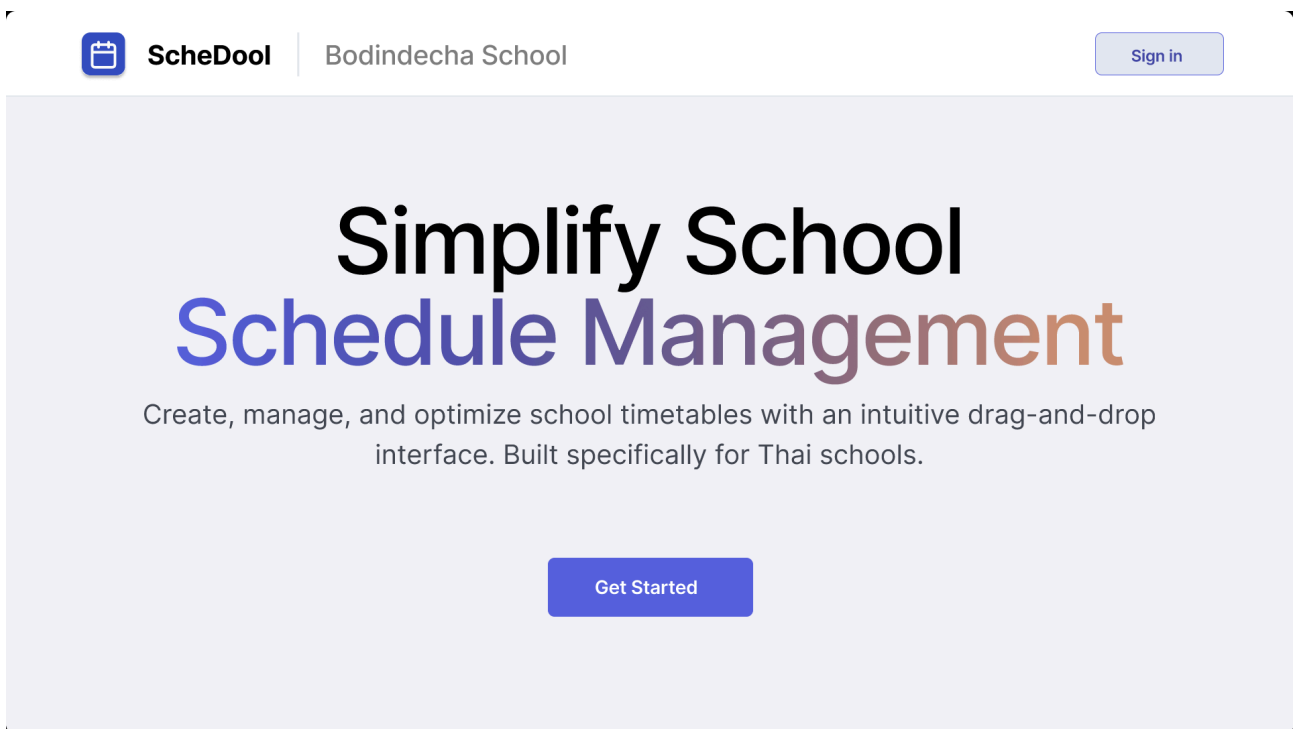
For the current phase of testing and development, we have established a collaborative partnership with a local public school to collect a representative sample dataset. The current dataset that we are utilizing is provided by Surawiwat School, specifically reflecting the scheduling requirements for the academic period of Year 2025 (2568 B.E.), Semester 2. This dataset is the most recent set available, which is highly beneficial as it ensures the constraints and operational requirements are up-to-date and accurately reflect current practices. Furthermore, working with a recent dataset allows for easy and rapid clarification with Surawiwat administrators in the case of any ambiguity in the data arising during development.

The dataset is currently managed and accessed collaboratively via the following Google Sheet link: https://docs.google.com/spreadsheets/d/11VfgJt_jgqUHW0X0a6VjAxQVzrXc31xsu4HUyAotQMA/edit?usp=sharing

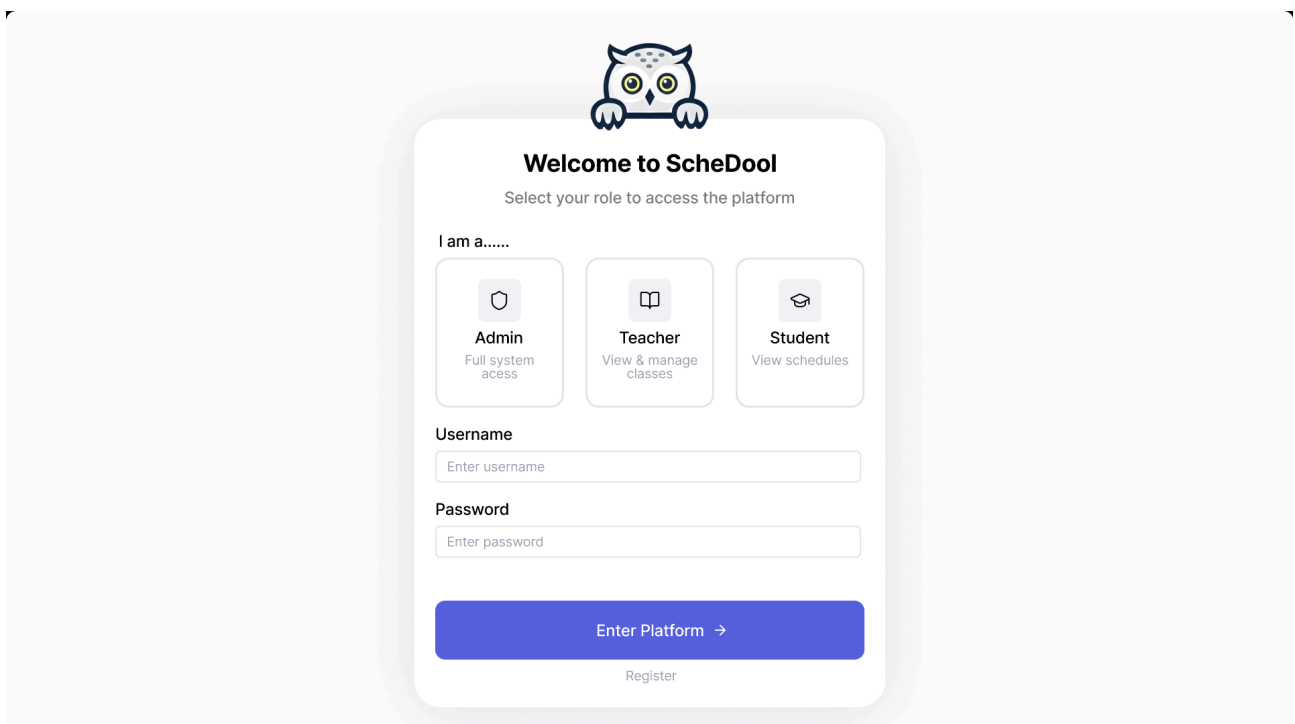
The structure of this input data strictly adheres to our [Standard Input Format](#) defined in Section 3.2. This consistency is vital for validating our data processing pipeline. Importantly, during our discussions with the Surawiwat administrative team, they provided positive feedback regarding our format, noting that its design, being derived from an analysis of their existing documents, made it particularly easy to apply, understand, and integrate into their workflow. This positive feedback confirms the practical usability of our standardized data approach.

4.2 User Journeys

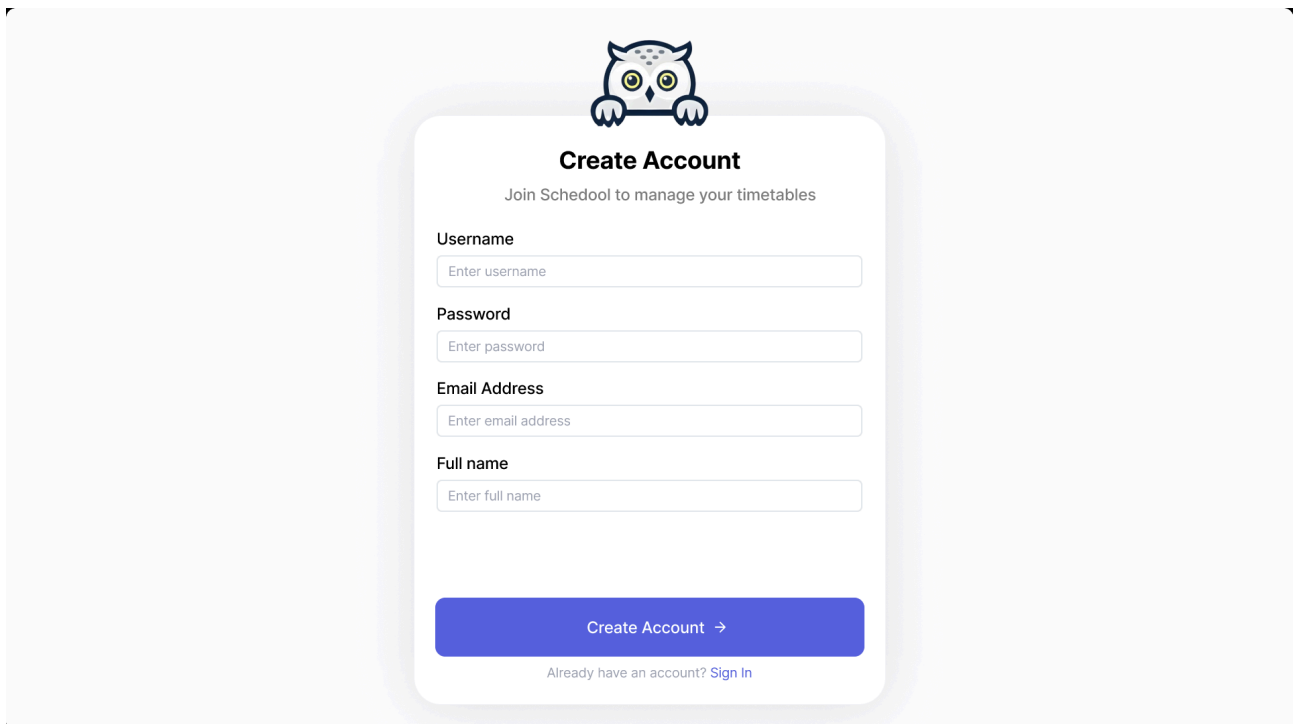
Admin Flow



Homepage: When entering our application, the user would first arrive at the homepage of ScheDool. In this page, there is a button that, when pressed, will take you to begin the process of using the application.

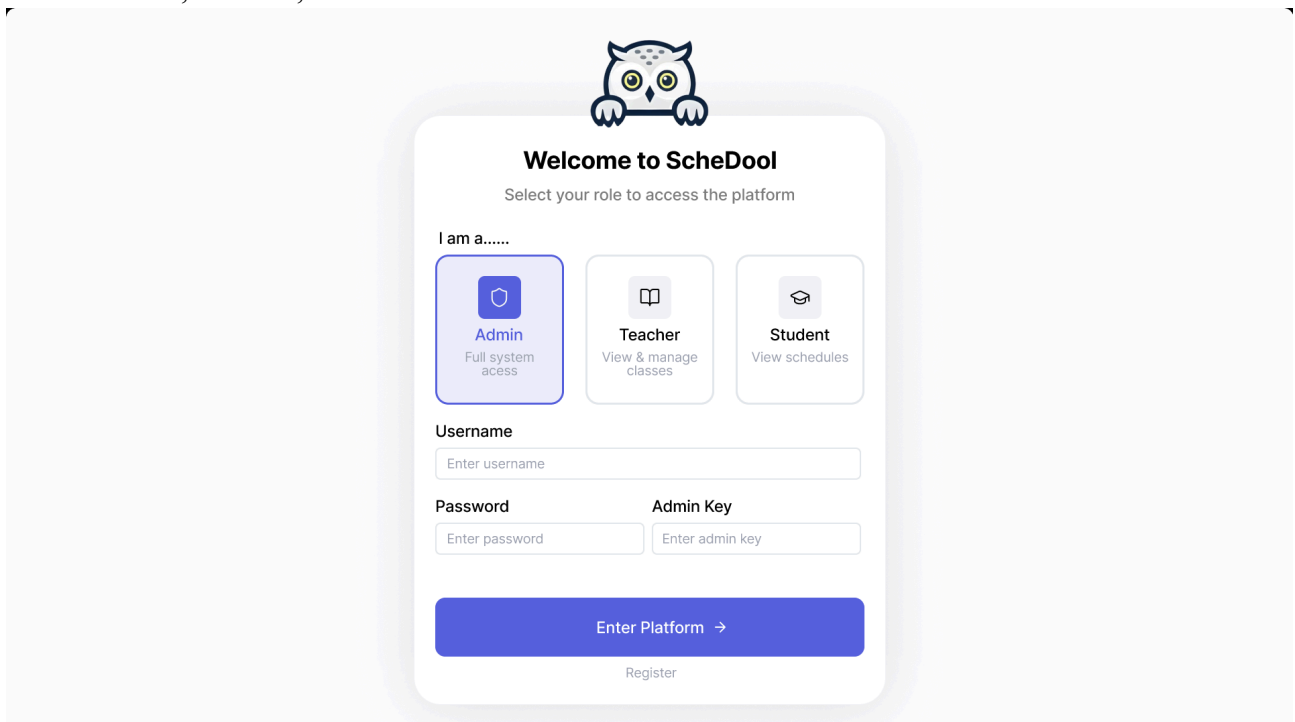


Login Page: After the user has pressed the “Get Started” button, the site will take the user to this page, where users are asked to fill in their credentials, i.e., email and password, in order to login. However, if the user does not have an account, they can create one by pressing on the “Register” text below the “Enter Platform” button.



The image shows a 'Create Account' form for 'Schedool'. At the top is an owl logo. The title 'Create Account' is centered, followed by the subtitle 'Join Schedool to manage your timetables'. The form contains four input fields: 'Username' (placeholder: 'Enter username'), 'Password' (placeholder: 'Enter password'), 'Email Address' (placeholder: 'Enter email address'), and 'Full name' (placeholder: 'Enter full name'). Below these fields is a large blue button labeled 'Create Account →'. At the bottom, there is a link: 'Already have an account? [Sign In](#)'.

Registration Page: In the case that the user does not have an account and has decided to create one, they will be presented with the registration form. This will then prompt the user to fill in their credentials, i.e., username, password, email, and their full name in order to create the account. Additionally, the form gives the user the ability to choose on their account’s role, i.e., Administrator, Teacher, or Student.



The image shows a 'Welcome to Schedool' registration form. At the top is an owl logo. The title 'Welcome to Schedool' is centered, followed by the subtitle 'Select your role to access the platform'. Below this is a section titled 'I am a.....' with three selectable roles: 'Admin' (with a shield icon and description 'Full system access'), 'Teacher' (with a book icon and description 'View & manage classes'), and 'Student' (with a graduation cap icon and description 'View schedules'). The 'Admin' role is currently selected. Below the role selection are three input fields: 'Username' (placeholder: 'Enter username'), 'Password' (placeholder: 'Enter password'), and 'Admin Key' (placeholder: 'Enter admin key'). At the bottom is a large blue button labeled 'Enter Platform →'. Below the button is a link: 'Register'.

Login Page (Admin): Now that the user has created an administrative account, they can login to our platform as usual with their credentials along with the Admin Key.


The screenshot shows the Admin Login Page of the ScheDool platform. At the top, there is a header with the ScheDool logo, the school name "Bodindecha School", and an "Admin" button next to a user profile icon. Below the header, the main section is titled "Schedules" with the subtitle "Manage your school timetables". A large blue button with a plus icon and the text "Create New Schedule" is prominently displayed. Below this, a section titled "All Schedules" contains a table listing existing schedules.

Schedule Name	Semester	Last Edited	Status	Actions
Schedule Semester 2/ 2025	2/2025	11/8/2025	Draft	⋮
Schedule Semester 1/ 2025	1/2025	11/1/2025	Published	⋮
Schedule Semester 2/ 2024	2/2024	11/8/2024	Published	⋮
Schedule Semester 1/ 2024	1/2024	11/1/2024	Published	⋮
Schedule Semester 2/ 2023	2/2023	11/8/2023	Published	⋮
Schedule Semester 1/ 2023	1/2023	11/1/2023	Published	⋮
Schedule Semester 2/ 2022	2/2022	11/8/2022	Published	⋮

Schedule List Page: After the user has logged in, the user will be presented with the schedule page, which lets the user create a new schedule via the big blue button or banner and view or edit existing schedules.


The screenshot shows the "Create Schedule Page" in the ScheDool platform. The page is part of a 10-step setup process, currently on "Step 1 out of 10" (10% complete). The left sidebar lists the setup steps: Curriculum (1), Teacher (2), Elective (3), Scout (4), Period (5), Student (6), Room (7), Constraint (8), Related files (9), and Generate (10). The main content area is titled "Schedule information" and contains three input fields: "Schedule name", "Year", and "Semester". Below these is a section for "Curriculum" with a link to "example_curriculum.csv" and a large upload area with a plus icon and the text "Upload Curriculum Data". At the bottom, there are "Back" and "Next" buttons.

Create Schedule Page: In the case that the user decided to create a schedule, they will be met with this page, which lets the user fill in the necessary information about the schedule along with uploading the CSV files needed to create that schedule in each step, i.e., curriculum, teachers, electives, etc.

 **ScheDool**

Bodindecha School

Admin



SETUP STEP

Curriculum

Teacher

Elective

Scout

Period

Student

Room

Constraint

Related files

Generate

12

Step 1 out of 10

10% complete

Schedule information

Schedule name

Enter schedule name

Year

Enter year

Semester

Enter semester



Curriculum

Upload the curriculum data file

[example_curriculum.csv](#)

Curriculum.csv


105.1 KB

← Back


Next →

Create Schedule Page (Error Detection): In case if the csv file that was uploaded is in an invalid format, the page will display a red warning / pop-up indicating that the uploaded file is invalid.

 **ScheDool**

Bodindecha School

Admin



SETUP STEP

Curriculum

Teacher

Elective

Scout

Period

Student

Room

Constraint

Related files

Generate

12

Step 1 out of 10

10% complete

Schedule information

Schedule name

Enter schedule name

Year

Enter year

Semester

Enter semester



Curriculum

Upload the curriculum data file

[example_curriculum.csv](#)

Curriculum.csv

105.1 KB

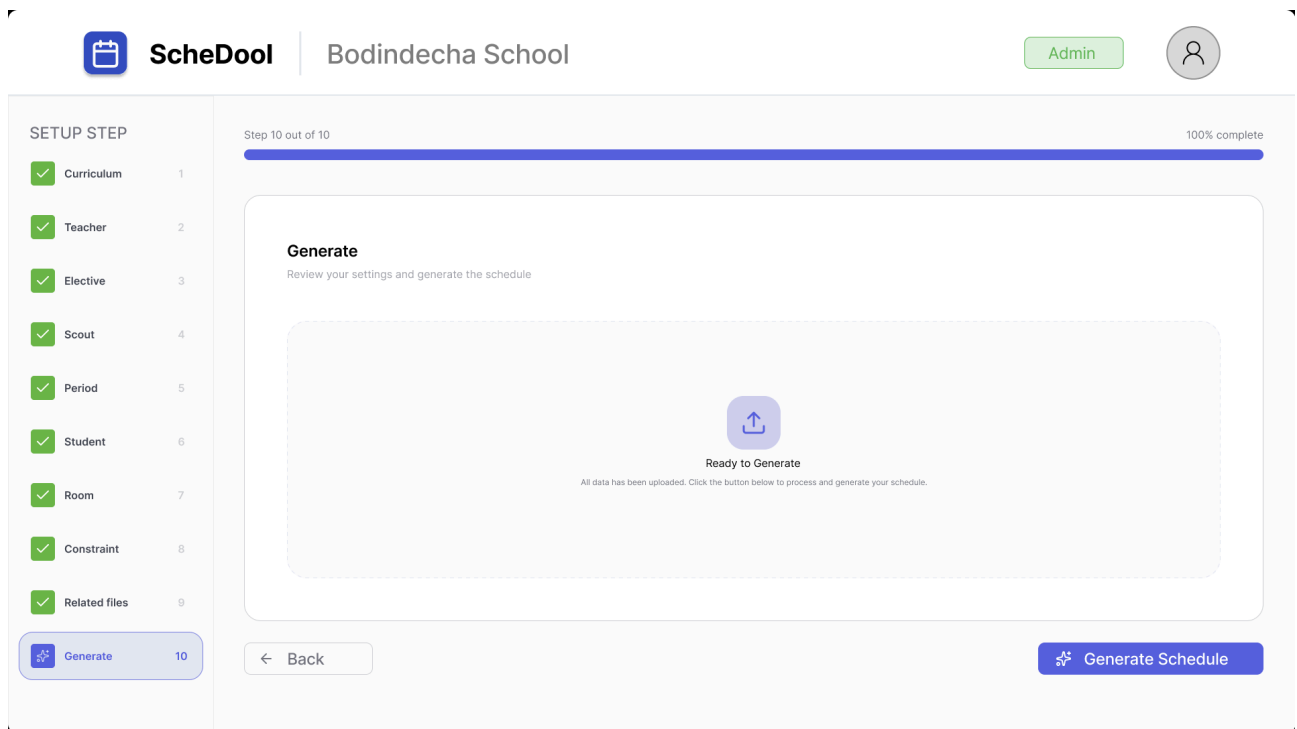
 

← Back

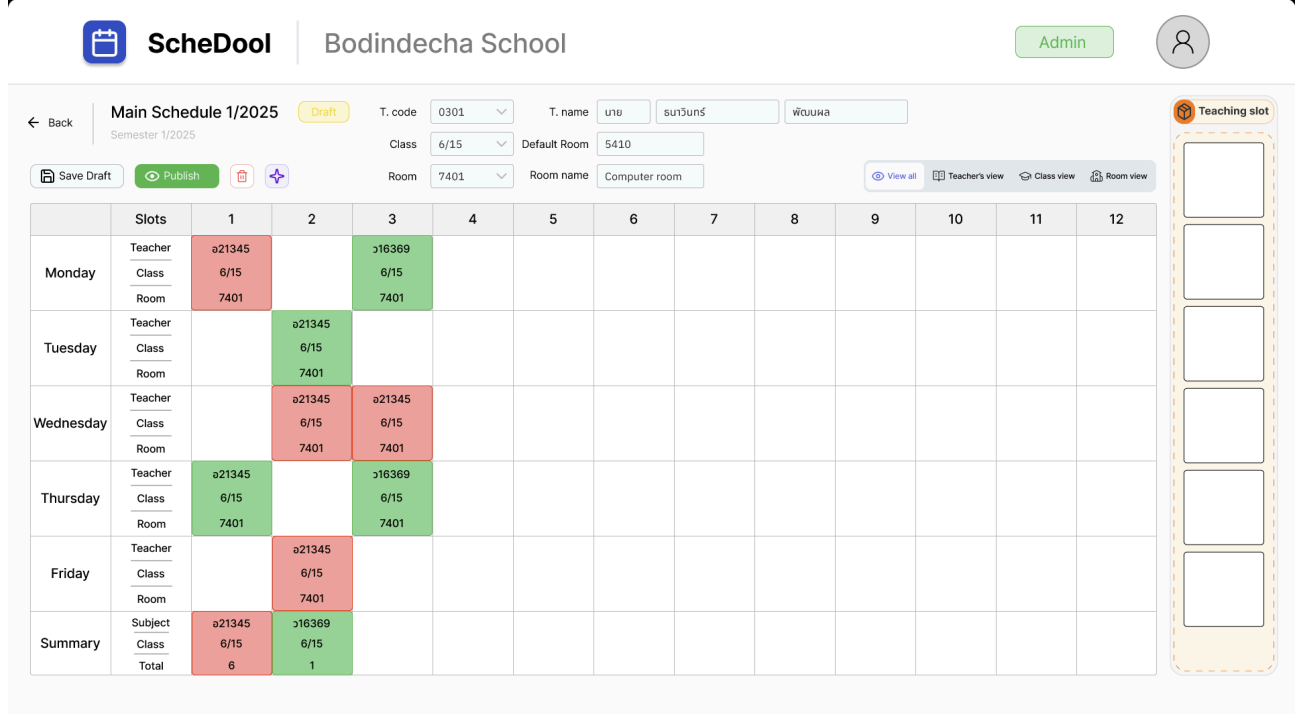
Next →

Create Schedule Page (Success): In the case that the uploaded CSV file is in the correct format, the page displays a green pop-up, indicating that the uploaded CSV file is valid.

25



Generate Schedule Page: Now that the user has uploaded the required files, the users can then request ScheDool’s AI engine to generate a schedule.

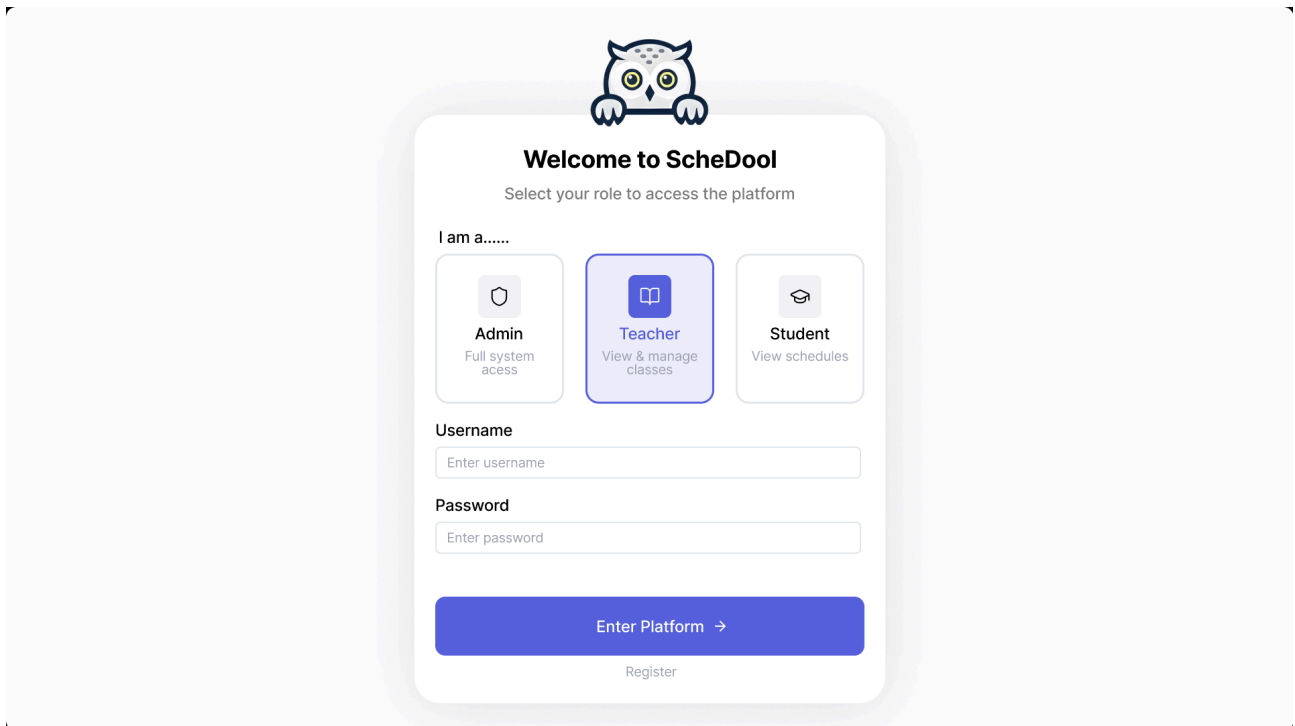


Schedule Edit Page: When the ScheDool AI engine finishes generating, the site will take the user to this page, where the user has the ability to: drag and drop, view teacher’s view, room’s view, class’s view, and view all. There is also an additional filter where the user can scope down what they would like to focus on, e.g., specific teachers, specific rooms, or specific classes.

The user can then save it as draft in case of future edits or modifications, or publish it if the schedule is finalized (can still be edited if needed), delete it if the schedule does not follow any of the user’s expectations, in which, the user can later regenerate the schedule using the AI engine.

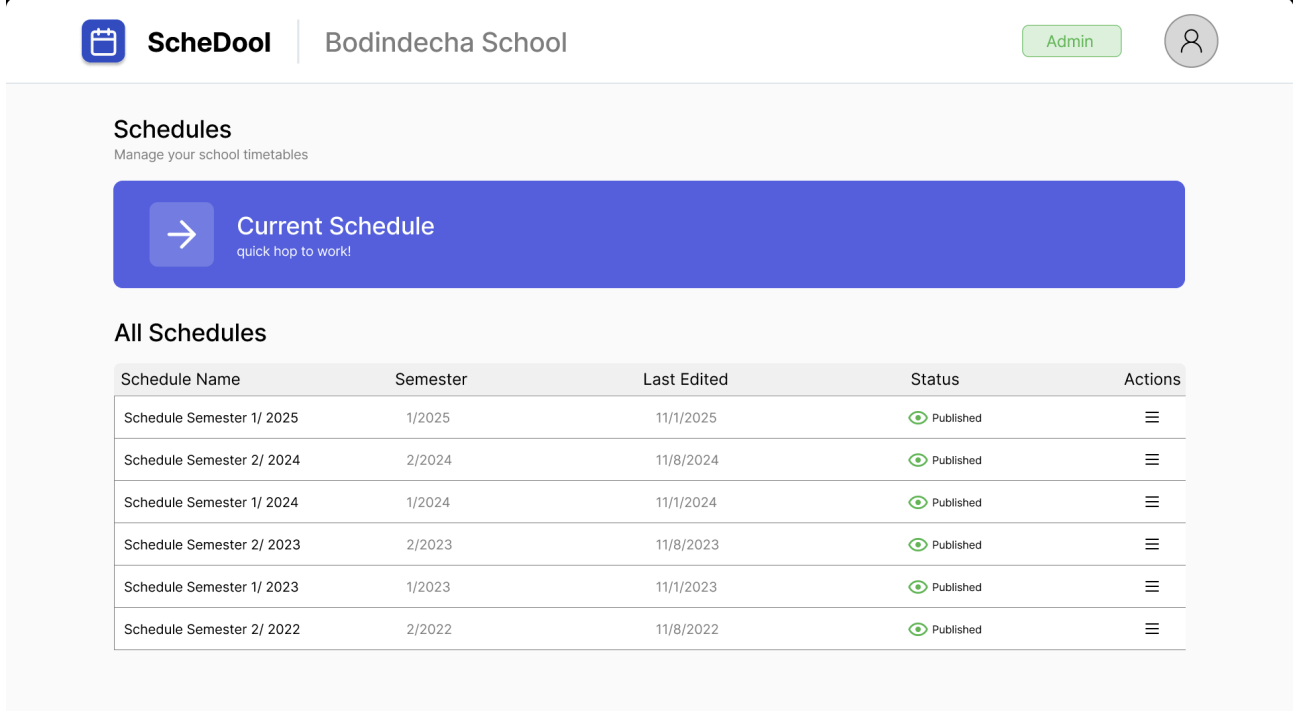
Moreover, if the user wanted to delete a specific period but wanted to add it again later, the user can then drag it into the right where there would be a slot where the user can place the period into a preset that can be used later on.

Teacher's Flow



The login page features a central white card with a blue owl logo at the top. Below the logo, the text "Welcome to ScheDool" is displayed, followed by "Select your role to access the platform". Underneath, there are three role selection buttons: "Admin" (Full system access), "Teacher" (View & manage classes), and "Student" (View schedules). The "Teacher" button is highlighted with a blue border. Below these buttons are input fields for "Username" and "Password", each with a placeholder "Enter username" and "Enter password" respectively. A large blue button labeled "Enter Platform →" is positioned below the input fields, and a smaller "Register" link is at the bottom.

Login Page: In this scenario, the user is a teacher, which the user can simply select the “Teacher” box in the middle, followed by entering their username and password, and logging in.



The interface shows the "ScheDool" logo and "Bodindecha School" in the top header. A green "Admin" button and a user profile icon are on the right. The main content area is titled "Schedules" with the subtitle "Manage your school timetables". A blue button labeled "Current Schedule" with a right arrow icon and the text "quick hop to work!" is prominent. Below this, the "All Schedules" section contains a table with the following data:

Schedule Name	Semester	Last Edited	Status	Actions
Schedule Semester 1/ 2025	1/2025	11/1/2025	Published	
Schedule Semester 2/ 2024	2/2024	11/8/2024	Published	
Schedule Semester 1/ 2024	1/2024	11/1/2024	Published	
Schedule Semester 2/ 2023	2/2023	11/8/2023	Published	
Schedule Semester 1/ 2023	1/2023	11/1/2023	Published	
Schedule Semester 2/ 2022	2/2022	11/8/2022	Published	

Schedule List Page: In this page, the user will be able to view the current schedule in use or view older / past schedules.

[← Back](#)

Main Schedule 1/2025

Published

T. code

0301

T. name

นาย

สมวิบูรณ์

พินิจ

[Teacher's view](#)[Class view](#)[Room view](#)

	Slots	1	2	3	4	5	6	7	8	9	10	11	12
Monday	Subject	๑21345		๖16369									
	Class	6/15		6/15									
	Room	7401		7401									
Tuesday	Subject		๑21345										
	Class		6/15										
	Room		7401										
Wednesday	Subject		๑21345	๑21345									
	Class		6/15	6/15									
	Room		7401	7401									
Thursday	Subject	๑21345		๖16369									
	Class	6/15		6/15									
	Room	7401		7401									
Friday	Subject		๑21345										
	Class		6/15										
	Room		7401										
Summary	Subject	๑21345	๖16369										
	Class	6/15	6/15										
	Total	6	1										

Schedule View Page: Here, the user is able to view the schedule itself. Note: Since the user in this scenario has the “Teacher” role, they would have less abilities in interacting with this page than the administrators.

[← Back](#)

Main Schedule 1/2025

Published

T. code

0301

T. name

นาย

สมวิบูรณ์

พินิจ

[Teacher's view](#)[Class view](#)[Room view](#)

	Slots	1	2	3	4	5	6	7	8	9	10	11	12
Monday	Subject	๑21345		๖16369									
	Class	6/15		6/15									
	Room	7401		7401									
Tuesday	Subject		๑21345										
	Class		6/15										
	Room		7401										
Wednesday	Subject		๑21345	๑21345									
	Class		6/15	6/15									
	Room		7401	7401									
Thursday	Subject	๑21345		๖16369									
	Class	6/15		6/15									
	Room	7401		7401									
Friday	Subject		๑21345										
	Class		6/15										
	Room		7401										
Summary	Subject	๑21345	๖16369										
	Class	6/15	6/15										
	Total	6	1										

Slot 1 : Monday

Slots Info

Subject ๑21345

Class 6/15

Room 7401

[← Back](#)[Select](#)

Period Information: In the case that the user would want to view the information of a period / slot, they can press on the period / slot, and the pop-up will display the information of that period.

In the case that the user would want to exchange classes, they would press the “Select” button.

[← Back](#)

Main Schedule 1/2025

[Publish](#)

T. code

0301

T. name

นาย

สมวันรุ่ง

พิจูณา

[Teacher's view](#)[Class view](#)[Room view](#)

	Slots	1	2	3	4	5	6	7	8	9	10	11	12
Monday	Subject	๑21345		๑16369									
	Class	6/15	Teacher Poon	6/15	Teacher Beam	Teacher Chanon							
	Room	7401		7401									
Tuesday	Subject		๑21345										
	Class	Teacher Beam	6/15	Teacher Poon	Teacher May								
	Room		7401										
Wednesday	Subject		๑21345	๑21345									
	Class	Teacher Diamond	6/15	6/15	Teacher May								
	Room		7401	7401									
Thursday	Subject	๑21345		๑16369									
	Class	6/15	Teacher Poon	6/15	Teacher May								
	Room	7401		7401									
Friday	Subject		๑21345										
	Class	Teacher Poon	6/15	Teacher Poon	Teacher Beam								
	Room		7401										
Summary	Subject	๑21345	๑16369										
	Class	6/15	6/15										
	Total	6	1										

Period Information (Exchange Mode): To exchange classes, the user can then click on the period that they want to exchange.

[← Back](#)

Main Schedule 1/2025

[Publish](#)

T. code

0301

T. name

นาย

สมวันรุ่ง

พิจูณา

[Teacher's view](#)[Class view](#)[Room view](#)

	Slots	1	2	3	4	5	6	7	8	9	10	11	12
Monday	Subject	๑21345		๑16369									
	Class	6/15	Teacher Poon	6/15	Teacher Beam								
	Room	7401		7401									
Tuesday	Subject		๑21345										
	Class	Teacher Beam	6/15	Teacher Poon	Teacher May								
	Room		7401										
Wednesday	Subject		๑21345	๑21345									
	Class	Teacher Diamond	6/15	6/15	Teacher May								
	Room		7401	7401									
Thursday	Subject	๑21345		๑16369									
	Class	6/15	Teacher Poon	6/15	Teacher May								
	Room	7401		7401									
Friday	Subject		๑21345										
	Class	Teacher Poon	6/15	Teacher Poon	Teacher Beam								
	Room		7401										
Summary	Subject	๑21345	๑16369										
	Class	6/15	6/15										
	Total	6	1										

Slot 1 : Monday

Exchanged Slots Info

Subject ๑21345
 Teacher Atcharyapat S.
 Class 6/15
 Room 7401

[← Back](#)[Select](#)

Period Information (Exchange Mode): The user can click on the period which will show the information again then press “Select” to start exchanging the period.



← Back

Main Schedule 1/2025

Publish

T. code

0301

T. name

นาย

สมวันตร์

พิชญพา

Semester 1/2025

Teacher's view

Class view

Room view

	Slots	1	2	3	4	5	6	7	8	9	10	11	12
Monday	Subject	๑21345		๑16369	Pending								
	Class	6/15		6/15									
	Room	7401		7401									
Tuesday	Subject		๑21345										
	Class		6/15										
	Room		7401										
Wednesday	Subject		๑21345	๑21345									
	Class		6/15	6/15									
	Room		7401	7401									
Thursday	Subject	๑21345		๑16369									
	Class	6/15		6/15									
	Room	7401		7401									
Friday	Subject		๑21345										
	Class		6/15										
	Room		7401										
Summary	Subject	๑21345	๑16369										
	Class	6/15	6/15										
	Total	6	1										

Period Information (Exchange Mode): The user will then wait for the exchange request to be accepted, which in the meantime will be displayed as such.

Inbox

Sender's Name	Topic	Since
Thanawin P.	Class exchanging : signing	1.23

← Back

Inbox Pop-up: The other user who was registered for the slot that was chosen in the earlier step will receive a notification in their inbox, notifying them of a class exchange request.

[Back](#)

Main Schedule 1/2025

Publish

T. code

0301

T. name

นาย

สมวิบูลย์

พัฒนา

Done

Semester 1/2025



2.101.4

แบบการขอแลกเปลี่ยนคาบสอน/มอบหมายหน้าที่

วันที่.....เดือน.....พ.ศ.....

เรื่อง ขออนุญาตแลกเปลี่ยนคาบสอน/มอบหมายหน้าที่

เรียน ผู้อำนวยการโรงเรียนดินทรเดชา (สิงห์ สิงหเสนี)

ด้วยข้าพเจ้า.....ตำแหน่ง.....

กลุ่มสาระการเรียนรู้.....มีความจำเป็น.....

ระหว่างเวลา.....น. ถึง.....น. จึงขอแลกเปลี่ยนคาบสอน/มอบหมายหน้าที่ ดังนี้

1.งานสอน ☐ ไม่มีคาบสอน ☐ มีคาบสอน.....คาบ แลกเปลี่ยนคาบดังนี้

ลำดับ ที่	คาบสอนปกติ					คาบที่ขอแลก				ลงชื่อผู้รับแลก/ผู้สอนแทน	
	ว/ด/ป	คาบที่	ม...../.....	รหัสวิชา	สถานที่	ว/ด/ป	คาบที่	รหัสวิชา	สถานที่	ตัวบรรจง	ลงชื่อ

Period Exchange Form: Since our current stakeholder still requires the need of an official form in order to exchange classes, the user who made the request to exchange classes will be required to fill in the form.

[Back](#)

Main Schedule 1/2025

Publish

T. code

0301

T. name

นาย

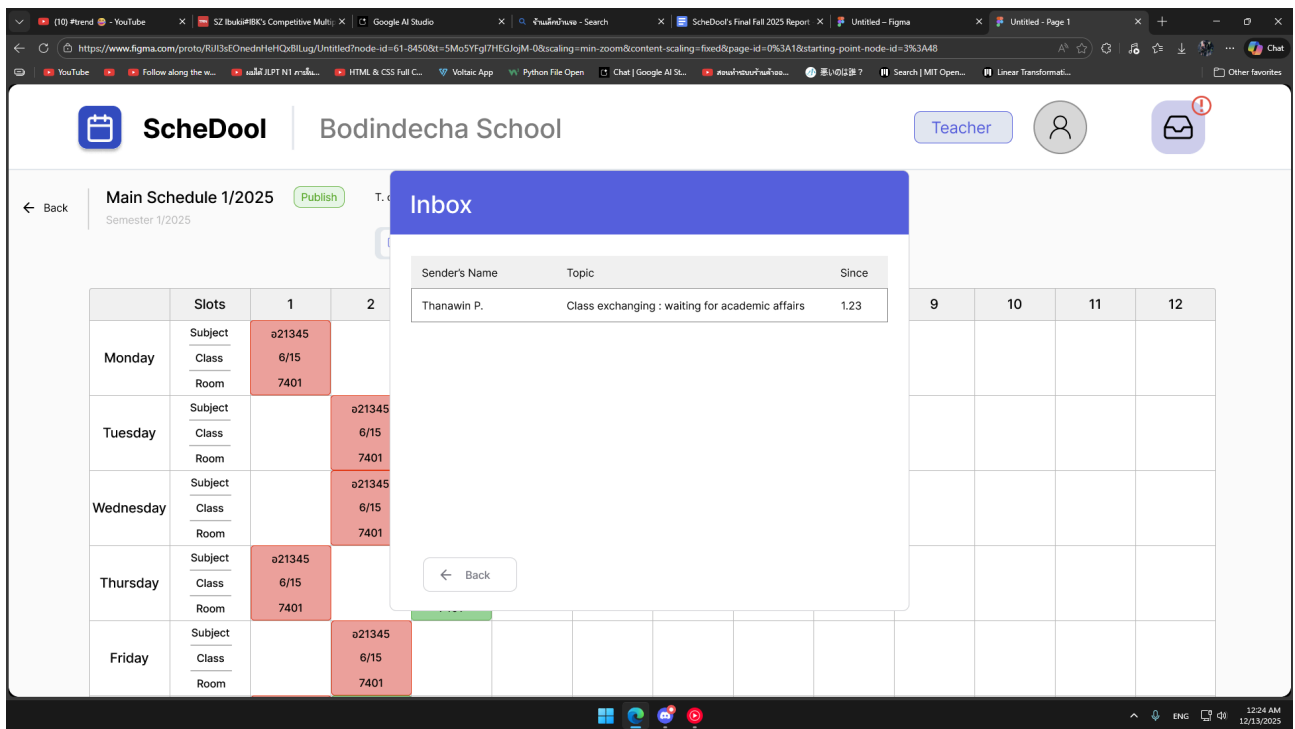
สมวิบูลย์

พัฒนา

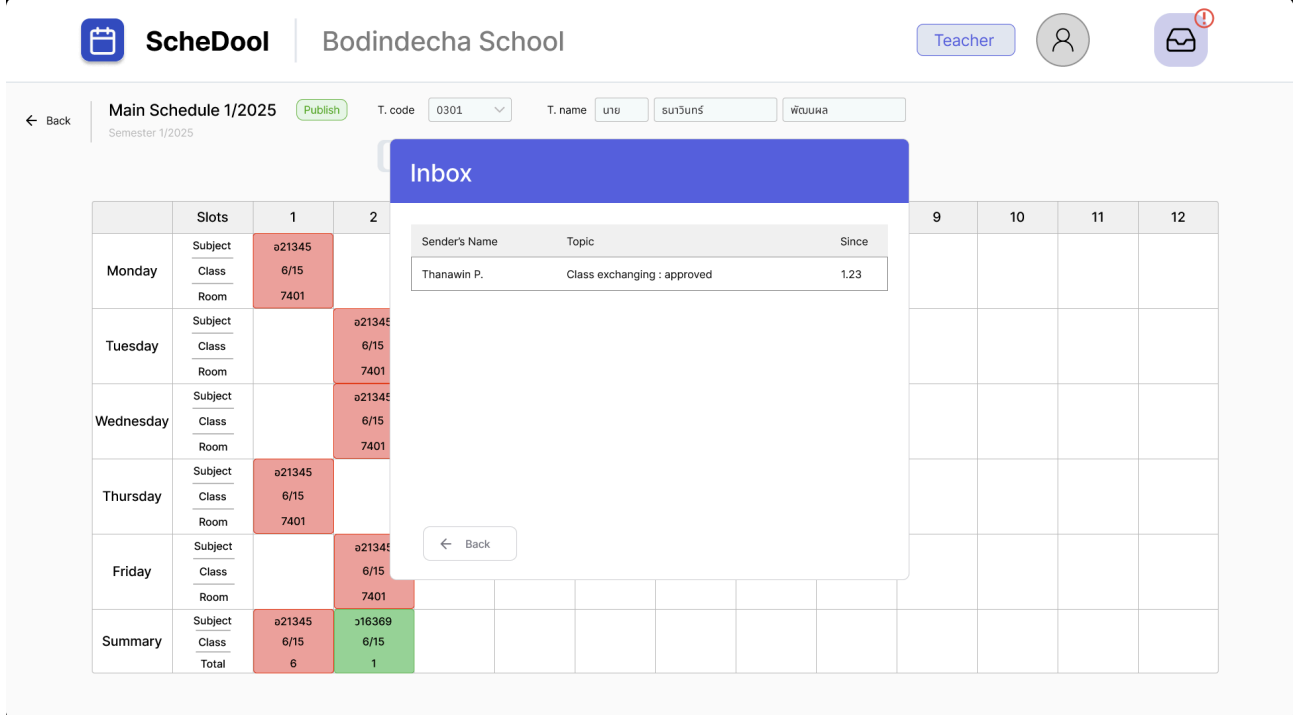
Semester 1/2025

Inbox

	Slots	1	2		9	10	11	12
Monday	Subject	๑21345		<div> <div>Sender's Name</div> <div>Topic</div> <div>Since</div> </div> <div> <div>Thanawin P.</div> <div>Class exchanging : receiver signing</div> <div>1.40</div> </div>				
	Class	6/15						
	Room	7401						
Tuesday	Subject		๑21345					
	Class		6/15					
	Room		7401					
Wednesday	Subject		๑21345					
	Class		6/15					
	Room		7401					
Thursday	Subject	๑21345						
	Class	6/15						
	Room	7401						
Friday	Subject		๑21345					
	Class		6/15					
	Room		7401					
Summary	Subject	๑21345	๑16369					
	Class	6/15	6/15					
	Total	6	1					



Period Exchange (Inbox Dialog): The index notification updates in real time on the status of the class exchange, in this case, the user would then need to wait for academic affairs to approve the exchange request.



[← Back](#)

Main Schedule 1/2025

[Publish](#)

T. code

0301

T. name

นาย

สมวันต์

พินิจภา

[Teacher's view](#)[Class view](#)[Room view](#)

	Slots	1	2	3	4	5	6	7	8	9	10	11	12
Monday	Subject			๖16369	๖21345								
	Class			6/15	6/15								
	Room			7401	7401								
Tuesday	Subject		๖21345										
	Class		6/15										
	Room		7401										
Wednesday	Subject		๖21345	๖21345									
	Class		6/15	6/15									
	Room		7401	7401									
Thursday	Subject	๖21345		๖16369									
	Class	6/15		6/15									
	Room	7401		7401									
Friday	Subject		๖21345										
	Class		6/15										
	Room		7401										
Summary	Subject	๖21345	๖16369										
	Class	6/15	6/15										
	Total	6	1										

View Schedule Page: Now that the exchange request was approved, the user will then see the changes in the schedule, that is, the exchange was successful.

4.3 Backend Implementation

4.3.1 Pre-Scheduler

The implementation of the *Pre-scheduling and Data Processing Service* was an important step in ensuring the core Genetic Algorithm (GA) solver receives a clean, standardized, and machine-ready input. Our current test implementation of the prototype version is written in Python.

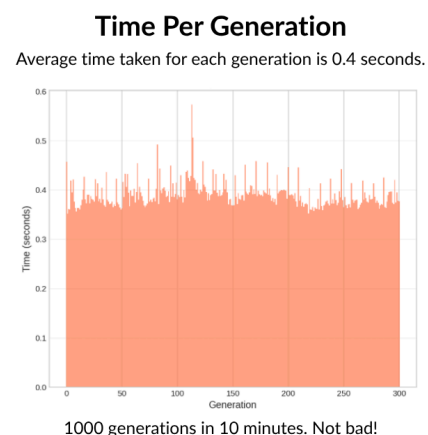
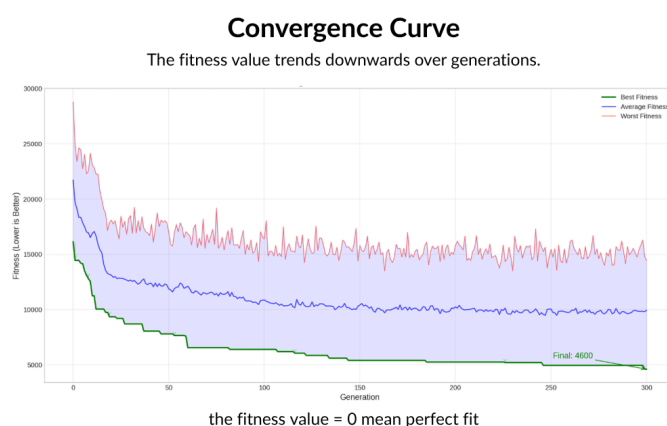
We successfully implemented the logic for strict data validation based on the [Standard Input Format \(Section 3.2\)](#), which automatically flags or rejects inconsistent data. This directly solves the problem of messy data formats found in schools. Furthermore, the service successfully handles *fixed constraint placement* (locking in non-negotiable activities, e.g., permanent meetings, non-available slots) before the GA runs. This process significantly reduces the complexity for the subsequent GA run, improving overall efficiency. This service ensures robust input quality for the iterative optimization phase.

The implementation of the **Large Language Model (LLM) constraint translation** is currently **under development** and is not yet fully implemented. This feature, once complete, will accurately process unique, informal human constraints (like teacher time preferences) and convert them into the necessary weighted parameters for the GA's optimization.

4.3.2 Genetic Algorithm Solver

As mentioned in the previous section, the pre-scheduler service / agent provides an output of a partially-filled schedule with fixed constraints, the output then flows into the *main scheduler agent*. The main scheduler agent, or colloquially called “GA Solver”, takes in the partially-filled schedule and uses genetic algorithm to find an optimal solution to complete that schedule.

In our current development stage, we have successfully implemented the basis of our main scheduling solver / GA solver in Python from scratch, in which, we can simulate the scheduling process by sending in a demonstrative output from the pre-scheduler agent to the solver agent, the GA solver takes in the input and forms an optimal solution from the input, and provides an output of a schedule. In its current form, the genetic algorithm’s fitness function is defined to penalize any conflicts that arise, [refer to Section 2.3 Related Research](#), therefore, it does not currently account for some constraints and can still output schedules that may not respect teacher’s preferences and such.



Chapter 5

Conclusions

5.1 Summary of Accomplishments

During this semester, on the frontend aspect, we were able to finish the entirety of our *user journey*, therefore, we can proceed to focus on developing our prototypes for our stakeholders to use in their next semester and fully implement for the entire next semester. Keeping parts of the designs where the users are already familiar with, while further customizing and refining the appearance of our frontend; ensuring a beautiful user interface and a great user experience. Additionally, we mapped out the various use cases of administrators and teachers that happen on a day-to-day basis, especially the class exchanging feature. Lastly, we also built a file upload validation tool that validates the uploaded files instantly and warns users of the specific errors that the files consist of.

On the other hand, we have also accomplished several milestones in our backend development aspect of the project. Our system's main genetic algorithm-based solver's development process is underway and so far, has produced a working containerized prototype that provides REST API endpoints which can then be easily integrated into the overall backend architecture of the system. We have developed our scheduling validation framework that will be finalized in the upcoming weeks.

5.2 Issues and Obstacles

Frontend - We took a long time to figure out what would be the best user journey for our users since there many ways but we also want the most efficient one but after long discussion and referring back to old existing software we are able to come up with simplest way to make users or average people use it at ease. Figuring out the best user journey was harder than we expected. There were many different ways we could have designed it, and we kept debating which would actually be easiest for real people to use. We didn't want to just pick something that looked good; it had to work for administrators who might not be tech-savvy. After lots of team discussions and going back to look at how the current system works, we finally landed on an approach that felt natural. We kept the familiar parts that users already understand and just made the confusing bits simpler.

5.3 Future Directions

Following the initial implementation with our stakeholders, the next phase will focus on systematic feedback collection from teachers, administrators, and scheduling staff. This input will guide iterative improvements to ensure that the platform aligns with real operational needs and constraints. Insights gathered from actual school usage will drive refinement of the scheduling engine, user interface, and workflow design. Ultimately, the goal is to evolve the initial prototype into a market-ready solution that not only meets stakeholder expectations but also scales effectively across different school sizes and contexts.

5.4 Lessons Learned

The clear lesson that we have learned is to always ensure clear communication skills. Due to the fact that the members of our team have a diverse set of skills and that some of the work that we need to do may not align with our strengths. We still should be able to signal whenever we are or are not able to do a certain task; lacking this skill could lead to problems specific to that task to expand. Potentially lengthen the time needed to be spent on fixing it.

We also learned that we must go through a major overhaul on our team management structure, as in many instances, even if every single member of the team has excellent communication skills, i.e., good communication, able to articulate their thoughts well, etc., if the team structure, agreements, roadmaps, responsibilities are not *clearly* defined, inefficiencies and confusion will inevitably occur and can potentially lead to more severe consequences, e.g., arguments, quarrels, etc.

References

- Bergmann, D., & Stryker, C. (n.d.). LangChain. *IBM*. Retrieved December 13, 2025, from <https://www.ibm.com/think/topics/langchain>
- Budiarto, A. S., Satvika Iswari, N. M., & Dharma, E. M. (2024). Application of genetic algorithm for school timetable scheduling. *2024 International Conference on Informatics Electrical and Electronics (ICIEE)*, 1–6. <https://doi.org/10.1109/iciee63403.2024.10920446>
- Contributors to Wikimedia projects. (2025, December 10). *Large language model*. Wikipedia. https://en.wikipedia.org/wiki/Large_language_model
- CP-SAT solver. (n.d.). *Google for Developers*. Retrieved December 13, 2025, from https://developers.google.com/optimization/cp/cp_solver
- GeeksforGeeks. (2022, February 10). What are the advantages of React.js ? *GeeksforGeeks*. <https://www.geeksforgeeks.org/reactjs/what-are-the-advantages-of-react-js/>
- LangChain overview*. (n.d.). Docs by LangChain. Retrieved December 13, 2025, from <https://docs.langchain.com/oss/python/langchain/overview>
- Nasien, D., & Andi, A. (2022). Optimization of genetic algorithm in courses scheduling. *IT Journal Research and Development*, 151–161. <https://doi.org/10.25299/itjrd.2022.7896>
- Ollama*. (n.d.). Retrieved December 13, 2025, from <https://ollama.com/>
- Products – nextschool*. (n.d.). Retrieved December 13, 2025, from <https://www.nextschool.io/products/>
- Stryker, C. (n.d.). Large language models. *IBM*. Retrieved December 13, 2025, from <https://www.ibm.com/think/topics/large-language-models>
- What is a Web App? - Web Application Explained - AWS*. (n.d.). Amazon Web Services, Inc. Retrieved December 13, 2025, from <https://aws.amazon.com/what-is/web-application/>
- What is the genetic algorithm?* (n.d.). MATLAB & Simulink. Retrieved December 13, 2025, from <https://www.mathworks.com/help/gads/what-is-the-genetic-algorithm.html>

Yan, L., & Deng, H. (2025). Adaptive genetic algorithm for computer course scheduling system under the background of educational informatization. *Systems and Soft Computing*, 7, 200324. <https://doi.org/10.1016/j.sasc.2025.200324>

โปรแกรมจัดตารางสอน. (n.d.). Retrieved December 13, 2025, from <https://www.teesoftware.com/Tr/Html/Trangson.htm>