Machine Reasoning and Reasoning System Project Report

Intelligent Teaching Assistant System

ITAS

TEAM MEMBERS

A0198535N LI XINLIN

A0198491M XU JIACHEN

MASTER OF TECHNOLOGY

Contents

1.0 Executive Summary

In the tide of globalization, the rapid flow of people, goods, information, and funds not only brought about the precision of the division of labor, but also caused the competition for resources among countries. Education competition is one of the most important part of it, all countries are facing unprecedented fierce competition in educational resources nowadays. To be specific, the competition for entering world-famous universities has changed from nationwide to global wide, which leads to the educational training institutions springing up.

Our team, consists of two Chinese students, all shuttled between different educational training institutions from childhood since Chinese education competition is one of the most fierce one in the worldwide. Although we do benefit a lot from it, we found that these institutions still have some common problems. Due to the large number of students, they tend to arrange same course length for different students which is unreasonable. Besides in some small institutions, teachers always need to do not only teaching but also arranging the curriculum schedule. Some of these institutions try to solve these questions by recruiting teaching assistant from college, since those teaching assistants always not profession enough, the effect is often not very good. Hence, we want to build an online intelligent teaching system for these educational training institutions.

The system is the Intelligent Teaching Assistant System (ITAS). Since it is built on a web-based application, both teacher and student can register and log in to the system. For teacher, he/she can set up the class, approve the students' appointments for the class, trigger the scheduling process and check both all lessons arrangement and one specific student's calendar. For students, they can apply for the class by submit their basic information and they can get the recommendation for the proper number of lessons generated by the system based on their existing skills and goals, and they can also check their own calendar after teacher finished all the process mentioned before.

To build the Intelligent Teaching Assistant System, which is a hybrid solution using the Vue.js, Flask, Spring Boot, Orange and OptaPlanner. We first extract the possible types of personal information which may influence the course length of one student should take to achieve the goal and design the questionnaire to collect the data for decision tree training, based on the rules generate, we finished built the recommendation part. For the function of reasonable scheduling, we build the domain model to further refine the acquired knowledge from interviewing the institution's teacher and identify the crux of requirements firstly, then we choose OptaPlanner as the constraint satisfaction engine to realize the scheduling process.

In this project, our team applied the knowledge and techniques acquired from the lectures and workshops into the reality business application scenario, and we would recommend our intelligent teaching assistant systems to every small and medium sized institutions, hoping it can help them improve the teaching efficiency and quality, help the students get better arrangement.

2.0 Problem Description

Complied to the education trend, educational training institutions are springing up recent years in China. They can provide different types of teaching and training services based on different requirements, include language skills training, professional skills training, entrance examination training and so on.

Jenny institution is a small-sized language training institutions, providing IELTS and TOEFL one-to-one tutoring services, to help students improve English ability and achieve their target scores in IELTS or TOEFL examination.

Jenny institution offers two categories, IELTS and TOEFL, and 4 types, speaking, writing, listening and reading classes in each category. The class of this institution is established by time period, and the student number of each course depends on the time period, maybe from 50 in spring course to 500 in summer course. Besides, different students have different English foundations and goals that better be arranged different course length between each type and each student. In addition, teacher and students may have some personal requirements, like not teaching or attending classes on weekend or other similar requirements.

With all of these requirements, it is very troublesome for a person to come out with all of those students' personalized recommendation of course length and all of those courses' reasonable schedule. We hope to help both teacher and students getting a proper arrangement.

2.1 Project Objective

Based on the background, our team aims to build an intelligent teaching assistant system which can be used by both teacher and students.

From the perspective of the life cycle of a course,

- Firstly, teacher can login in to set up the course based on the start time of the course, the end time of the course and the deadline of accepting the application from students, and make a request for her need at the same time, like not teaching on weekends.
- After teacher finished setting, students can login to make appointments to join the course, after every student typing in his/her personal information, our system can make personalized recommendation about the lengths of the lectures the student should take based on the individual information.
- Then the teacher can view all students' appointments and decide to approve or not. And teacher can close the applying anytime she wants. When all the applications have been processed and the application channel has been closed, the teacher can get the course schedule by just clicking the "scheduling" button. And the system will do scheduling not only based on the course duration and student size, but also considering both the teacher's personal requirements and the students' personal requirements.
- Both teacher and students can check their schedule whenever they want and they can also check their whole schedule in the calendar view.

For the student part, they can login in to obtain the personalized recommendation for each types of courses on how many courses they should attend to achieve their goals based on their individual information. Besides, they can apply for joining the course set by teacher and submit their personal requirements, like not attending the course on Monday. Once all of their applications are processed by teacher, those whose applications are approved can check their course schedule whenever they want.

For the teacher part, teacher can login in to set up the course based on the start time of the course, the end time of the course and the deadline of accepting the application from students, and make a request for her need at the same time, like not teaching on weekends. The teacher can view all students' appointments and decide to approve or not. And teacher can close the applying channel anytime she wants. When all the applications have been processed and the application channel has been closed, the teacher can get the course schedule by just clicking the "scheduling" button and can check the arrange anytime she wants.

2.2 Project Team

The project members' name and work items are listed below:

Name	Work Items
LI XINLIN	Knowledge Modelling, OptaPlanner Solver Development, Backend Development, Documentation, Video
XU JIACHEN	Application Architect, Frontend Development, Backend Development, Documentation, Video

Table 1. Project contribution

Our team met twice a week to discuss project status and updates. Tasks were assigned and areas of responsibilities were clearly outlined for each member.

3.0 Knowledge Modeling

For this project, knowledge modeling consists of two main aspects:

- Knowledge identification and acquisition
- Knowledge representation

Based on the knowledge, our team mainly realize two core functions of our system:

- make personalized recommendation about the lengths of the lectures the student should take based on the individual information
- do scheduling not only based on the course duration and student size, but also considering both the teacher's personal requirements and the students' personal requirements.

3.1 Knowledge Identification and Acquisition

Based on the problem description and problem objectives, we have identified two main source data that are useful to our system.

Source of Information	Insights from Information Sources	Knowledge Acquisition Technique
Jenny Institution Expert	The teacher with many years teaching experience in institutions will be able to: 1. Identify the relevant information type which may influence the course length of one student should take 2. Identify and explain the considerations and constraints when plan class schedule.	Elicitation of tacit knowledge through the conduct of interview
Students used studied at Jenny Institution	The impact of different types of information on the length of the course required to achieve the goal, extracting the rules about the course length recommendation	Data gathering from documented information collecting through questionnaire

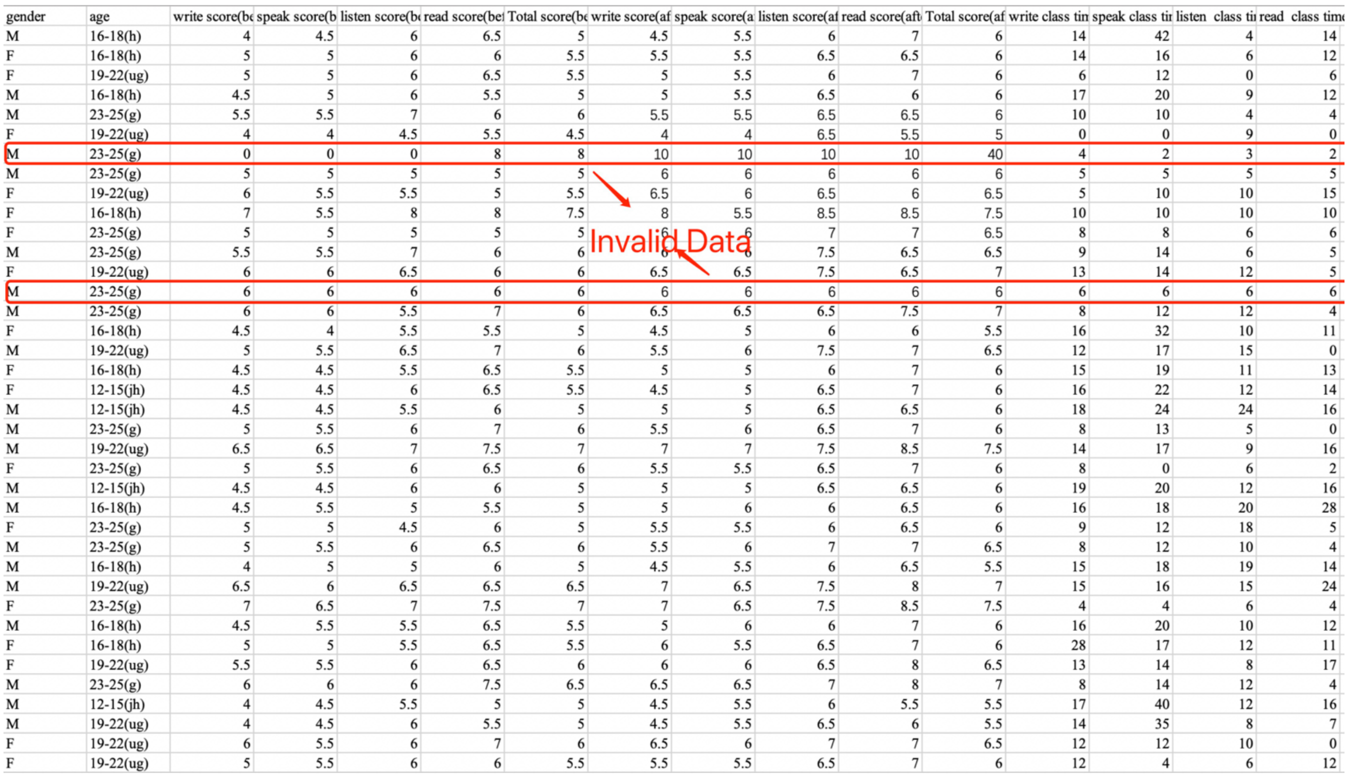
Table 2. Knowledge Source and Insights

3.2 Knowledge Representation

3.2.1 Recommendation Part

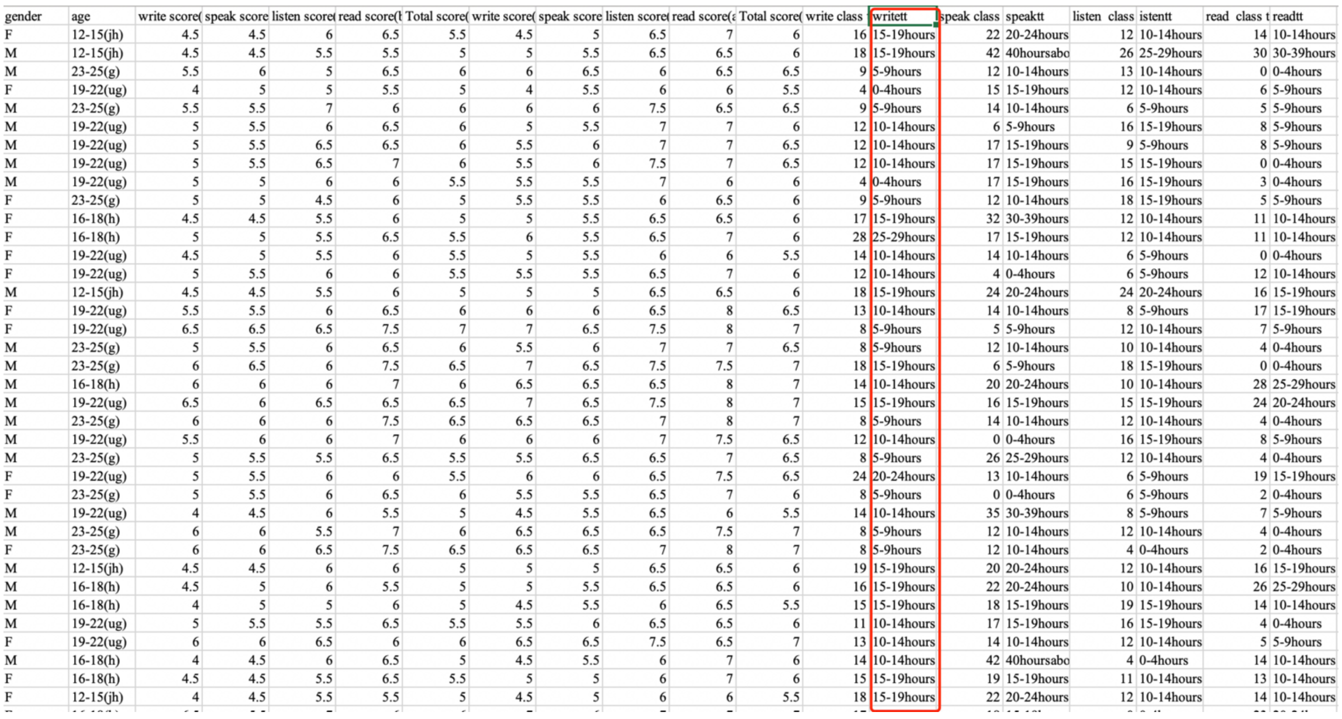
Firstly, we extract the possible types of personal information which may influence the course length of one student should take to achieve the goal from interviewing Jenny, who is founder of this institution. Based on these, we designed the questionnaire and sent the questionnaire to all the students who have attended classes at this institution before. We Stopped collecting when the number of questionnaires is enough for training the decision tree.

Before training started, we tidied up the data first by transforming the course quantity into different classes, detecting the irrelevant or unnormal data, dropping out the invalid data and some other preprocessing works.


Invalid Data

gender	age	write score(be	speak score(b	listen score(b	read score(bef	Total score(be	write score(af	speak score(a	listen score(af	read score(aft	Total score(af	write class tim	speak class tim	listen class tim	read class time
M	16-18(h)	4	4.5	6	6.5	5	4.5	5.5	6	7	6	14	42	4	14
F	16-18(h)	5	5	6	6	5.5	5.5	5.5	6.5	6.5	6	14	16	6	12
F	19-22(ug)	5	5	6	6.5	5.5	5	5.5	6	7	6	6	12	0	6
M	16-18(h)	4.5	5	6	5.5	5	5	5.5	6.5	6	6	17	20	9	12
M	23-25(g)	5.5	5.5	7	6	6	5.5	5.5	6.5	6.5	6	10	10	4	4
F	19-22(ug)	4	4	4.5	5.5	4.5	4	4	6.5	5.5	5	0	0	9	0
M	23-25(g)	0	0	0	8	8	10	10	10	10	40	4	2	3	2
M	23-25(g)	5	5	5	5	5	6	6	6	6	6	5	5	5	5
F	19-22(ug)	6	5.5	5.5	5	5.5	6.5	6	6.5	6	6.5	5	10	10	15
F	16-18(h)	7	5.5	8	8	7.5	8	5.5	8.5	8.5	7.5	10	10	10	10
F	23-25(g)	5	5	5	5	5	6	6	7	7	6.5	8	8	6	6
M	23-25(g)	5.5	5.5	7	6	6	6	6	7.5	6.5	6.5	9	14	6	5
F	19-22(ug)	6	6	6.5	6	6	6.5	6.5	7.5	6.5	7	13	14	12	5
M	23-25(g)	6	6	6	6	6	6	6	6	6	6	6	6	6	6
M	23-25(g)	6	6	5.5	7	6	6.5	6.5	6.5	7.5	7	8	12	12	4
F	16-18(h)	4.5	4	5.5	5.5	5	4.5	5	6	6	5.5	16	32	10	11
M	19-22(ug)	5	5.5	6.5	7	6	5.5	6	7.5	7	6.5	12	17	15	0
F	16-18(h)	4.5	4.5	5.5	6.5	5.5	5	5	6	7	6	15	19	11	13
F	12-15(jh)	4.5	4.5	6	6.5	5.5	4.5	5	6.5	7	6	16	22	12	14
M	12-15(jh)	4.5	4.5	5.5	6	5	5	5	6.5	6.5	6	18	24	24	16
M	23-25(g)	5	5.5	6	7	6	5.5	6	6.5	7	6	8	13	5	0
M	19-22(ug)	6.5	6.5	7	7.5	7	7	7	7.5	8.5	7.5	14	17	9	16
F	23-25(g)	5	5.5	6	6.5	6	5.5	5.5	6.5	7	6	8	0	6	2
M	12-15(jh)	4.5	4.5	6	6	5	5	5	6.5	6.5	6	19	20	12	16
M	16-18(h)	4.5	5.5	5	5.5	5	5	6	6	6.5	6	16	18	20	28
F	23-25(g)	5	5	4.5	6	5	5.5	5.5	6	6.5	6	9	12	18	5
M	23-25(g)	5	5.5	6	6.5	6	5.5	6	7	7	6.5	8	12	10	4
M	16-18(h)	4	5	5	6	5	4.5	5.5	6	6.5	5.5	15	18	19	14
M	19-22(ug)	6.5	6	6.5	6.5	6.5	7	6.5	7.5	8	7	15	16	15	24
F	23-25(g)	7	6.5	7	7.5	7	7	6.5	7.5	8.5	7.5	4	4	6	4
M	16-18(h)	4.5	5.5	5.5	6.5	5.5	5	6	6	7	6	16	20	10	12
F	16-18(h)	5	5	5.5	6.5	5.5	6	5.5	6.5	7	6	28	17	12	11
F	19-22(ug)	5.5	5.5	6	6.5	6	6	6	6.5	8	6.5	13	14	8	17
M	23-25(g)	6	6	6	7.5	6.5	6.5	6.5	7	8	7	8	14	12	4
M	12-15(jh)	4	4.5	5.5	5	5	4.5	5.5	6	5.5	5.5	17	40	12	16
M	19-22(ug)	4	4.5	6	5.5	5	4.5	5.5	6.5	6	5.5	14	35	8	7
F	19-22(ug)	6	5.5	6	7	6	6.5	6	7	7	6.5	12	12	10	0
F	19-22(ug)	5	5.5	6	6	5.5	5.5	5.5	6.5	7	6	12	4	6	12

Figure 1. Original data and some invalid examples



gender	age	write score(	speak score	listen score(	read score(b	Total score(	write score(	speak score	listen score(	read score(a	Total score(	write class t	writett	speak class	speaktt	listen class	istentt	read class t	readtt
F	12-15(jh)	4.5	4.5	6	6.5	5.5	4.5	5	6.5	7	6	16	15-19hours	22	20-24hours	12	10-14hours	14	10-14hours
M	12-15(jh)	4.5	4.5	5.5	5.5	5	5	5.5	6.5	6.5	6	18	15-19hours	42	40hoursabo	26	25-29hours	30	30-39hours
M	23-25(g)	5.5	6	5	6.5	6	6	6.5	6	6.5	6.5	9	5-9hours	12	10-14hours	13	10-14hours	0	0-4hours
F	19-22(ug)	4	5	5	5.5	5	4	5.5	6	6	5.5	4	0-4hours	15	15-19hours	12	10-14hours	6	5-9hours
M	23-25(g)	5.5	5.5	7	6	6	6	6	7.5	6.5	6.5	9	5-9hours	14	10-14hours	6	5-9hours	5	5-9hours
M	19-22(ug)	5	5.5	6	6.5	6	5	5.5	7	7	6	12	10-14hours	6	5-9hours	16	15-19hours	8	5-9hours
M	19-22(ug)	5	5.5	6.5	6.5	6	5.5	6	7	7	6.5	12	10-14hours	17	15-19hours	9	5-9hours	8	5-9hours
M	19-22(ug)	5	5.5	6.5	7	6	5.5	6	7.5	7	6.5	12	10-14hours	17	15-19hours	15	15-19hours	0	0-4hours
M	19-22(ug)	5	5	6	6	5.5	5.5	5.5	7	6	6	4	0-4hours	17	15-19hours	16	15-19hours	3	0-4hours
F	23-25(g)	5	5	4.5	6	5	5.5	5.5	6	6.5	6	9	5-9hours	12	10-14hours	18	15-19hours	5	5-9hours
F	16-18(h)	4.5	4.5	5.5	6	5	5	5.5	6.5	6.5	6	17	15-19hours	32	30-39hours	12	10-14hours	11	10-14hours
F	16-18(h)	5	5	5.5	6.5	5.5	6	5.5	6.5	7	6	28	25-29hours	17	15-19hours	12	10-14hours	11	10-14hours
F	19-22(ug)	4.5	5	5.5	6	5.5	5	5.5	6	6	5.5	14	10-14hours	14	10-14hours	6	5-9hours	0	0-4hours
F	19-22(ug)	5	5.5	6	6	5.5	5.5	5.5	6.5	7	6	12	10-14hours	4	0-4hours	6	5-9hours	12	10-14hours
M	12-15(jh)	4.5	4.5	5.5	6	5	5	5	6.5	6.5	6	18	15-19hours	24	20-24hours	24	20-24hours	16	15-19hours
F	19-22(ug)	5.5	5.5	6	6.5	6	6	6	6.5	8	6.5	13	10-14hours	14	10-14hours	8	5-9hours	17	15-19hours
F	19-22(ug)	6.5	6.5	6.5	7.5	7	7	6.5	7.5	8	7	8	5-9hours	5	5-9hours	12	10-14hours	7	5-9hours
M	23-25(g)	5	5.5	6	6.5	6	5.5	6	7	7	6.5	8	5-9hours	12	10-14hours	10	10-14hours	4	0-4hours
M	23-25(g)	6	6.5	6	7.5	6.5	7	6.5	7.5	7.5	7	18	15-19hours	6	5-9hours	18	15-19hours	0	0-4hours
M	16-18(h)	6	6	6	7	6	6.5	6.5	6.5	8	7	14	10-14hours	20	20-24hours	10	10-14hours	28	25-29hours
M	19-22(ug)	6.5	6	6.5	6.5	6.5	7	6.5	7.5	8	7	15	15-19hours	16	15-19hours	15	15-19hours	24	20-24hours
M	23-25(g)	6	6	6	7.5	6.5	6.5	6.5	7	8	7	8	5-9hours	14	10-14hours	12	10-14hours	4	0-4hours
M	19-22(ug)	5.5	6	6	7	6	6	6	7	7.5	6.5	12	10-14hours	0	0-4hours	16	15-19hours	8	5-9hours
M	23-25(g)	5	5.5	5.5	6.5	5.5	5.5	6.5	6.5	7	6.5	8	5-9hours	26	25-29hours	12	10-14hours	4	0-4hours
F	19-22(ug)	5	5.5	6	6	5.5	6	6	6.5	7.5	6.5	24	20-24hours	13	10-14hours	6	5-9hours	19	15-19hours
F	23-25(g)	5	5.5	6	6.5	6	5.5	5.5	6.5	7	6	8	5-9hours	0	0-4hours	6	5-9hours	2	0-4hours
M	19-22(ug)	4	4.5	6	5.5	5	4.5	5.5	6.5	6	5.5	14	10-14hours	35	30-39hours	8	5-9hours	7	5-9hours
M	23-25(g)	6	6	5.5	7	6	6.5	6.5	6.5	7.5	7	8	5-9hours	12	10-14hours	12	10-14hours	4	0-4hours
F	23-25(g)	6	6	6.5	7.5	6.5	6.5	6.5	7	8	7	8	5-9hours	12	10-14hours	4	0-4hours	2	0-4hours
M	12-15(jh)	4.5	4.5	6	6	5	5	5	6.5	6.5	6	19	15-19hours	20	20-24hours	12	10-14hours	16	15-19hours
M	16-18(h)	4.5	5	6	5.5	5	5	5.5	6.5	6.5	6	16	15-19hours	22	20-24hours	10	10-14hours	26	25-29hours
M	16-18(h)	4	5	5	6	5	4.5	5.5	6	6.5	5.5	15	15-19hours	18	15-19hours	19	15-19hours	14	10-14hours
M	19-22(ug)	5	5.5	5.5	6.5	5.5	5.5	6	6.5	6.5	6	11	10-14hours	17	15-19hours	16	15-19hours	4	0-4hours
F	19-22(ug)	6	6	6.5	6	6	6.5	6.5	7.5	6.5	7	13	10-14hours	14	10-14hours	12	10-14hours	5	5-9hours
M	16-18(h)	4	4.5	6	6.5	5	4.5	5.5	6	7	6	14	10-14hours	42	40hoursabo	4	0-4hours	14	10-14hours
F	16-18(h)	4.5	4.5	5.5	6.5	5.5	5	5	6	7	6	15	15-19hours	19	15-19hours	11	10-14hours	13	10-14hours
F	12-15(jh)	4	4.5	5.5	5.5	5	4.5	5	6	6	5.5	18	15-19hours	22	20-24hours	12	10-14hours	14	10-14hours

Figure 2. Data after preprocessing and some transforming examples

Then we built the decision tree model by using the Oranges, and we obtained good rules by adjusting the algorithms, the minimum number of instances in leaves, the smallest split subsets size, the maximal tree depth and some other parameters.

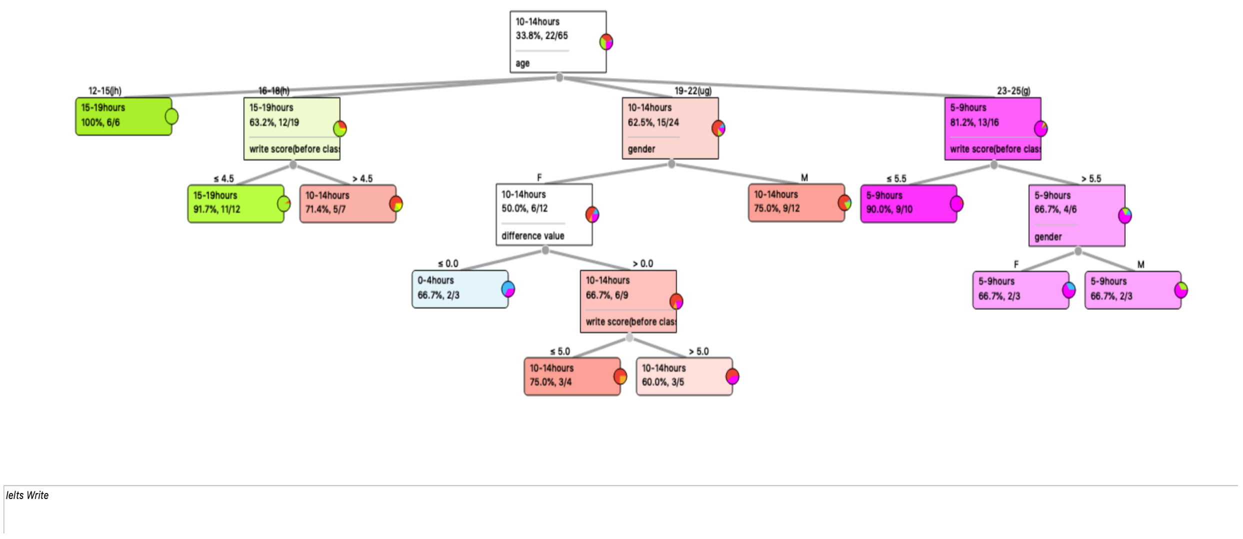


Figure 3. Decision tree result example (IELTS writing)

3.2.2 Scheduling Part – problem modeling

For the function of reasonable scheduling, it's very important to further refine the acquired knowledge to identify the crux of requirements and represented in the knowledge model. This process is conducted and presented by using the domain diagram as shown below.

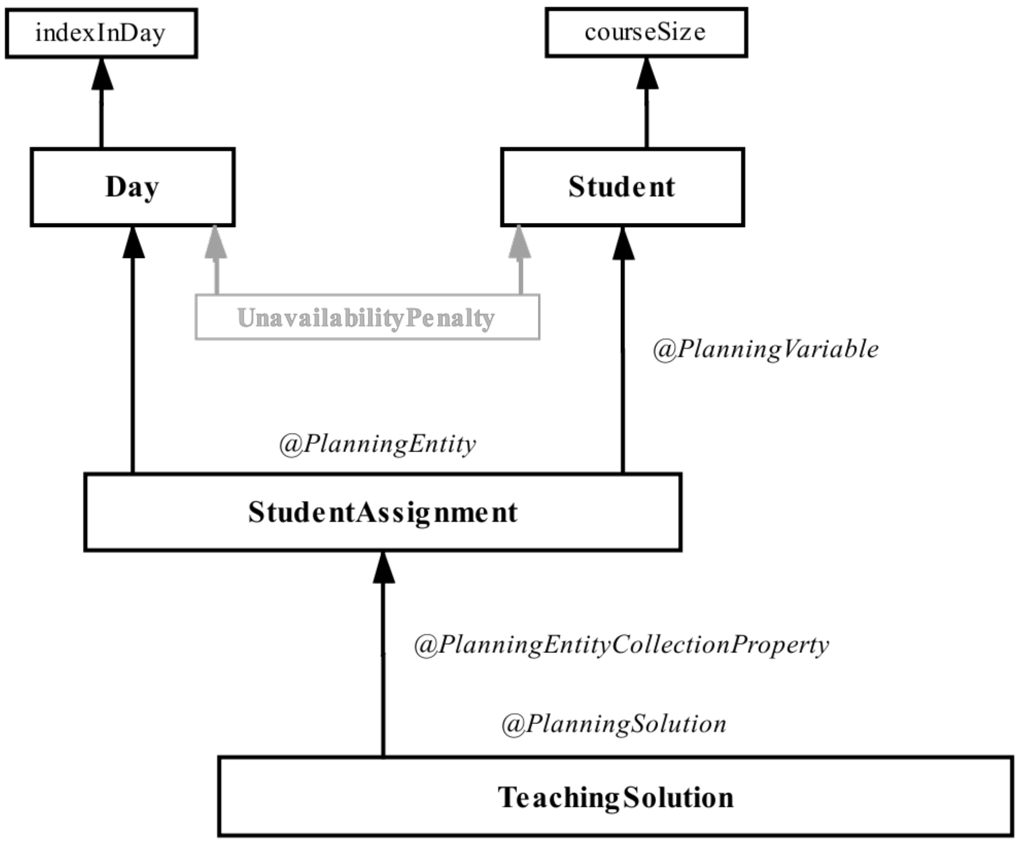


Figure 4. Domain Model

The domain model shows all objects, relationships and directions clearly. There are four main classes that define the domain:

- Day
- Student
- UnavailabilityPenalty
- StudentAssignment
- TeachingSolution

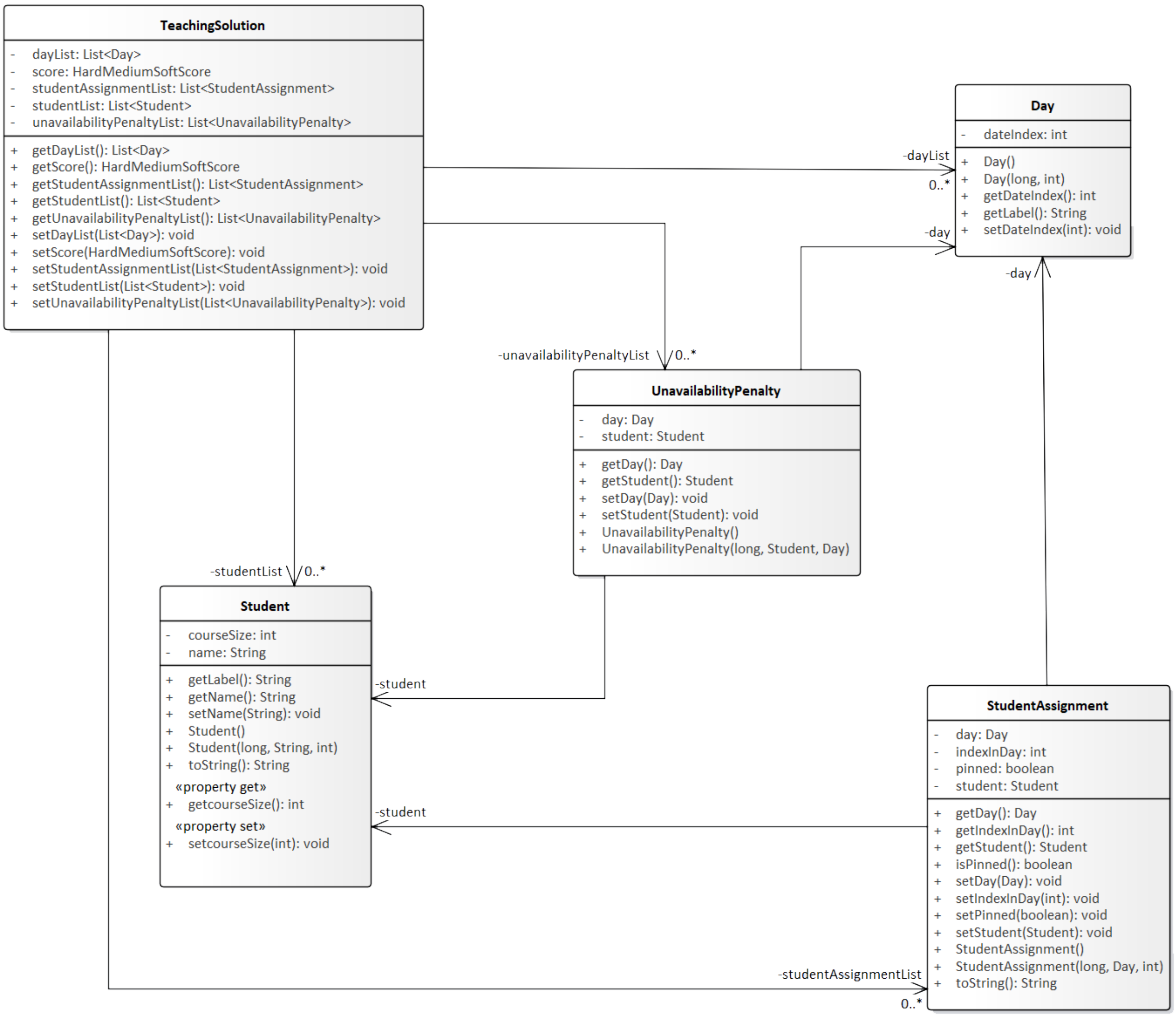

TeachingSolution
- dayList: List<Day>
- score: HardMediumSoftScore
- studentAssignmentList: List<StudentAssignment>
- studentList: List<Student>
- unavailabilityPenaltyList: List<UnavailabilityPenalty>
+ getDayList(): List<Day>
+ getScore(): HardMediumSoftScore
+ getStudentAssignmentList(): List<StudentAssignment>
+ getStudentList(): List<Student>
+ getUnavailabilityPenaltyList(): List<UnavailabilityPenalty>
+ setDayList(List<Day>): void
+ setScore(HardMediumSoftScore): void
+ setStudentAssignmentList(List<StudentAssignment>): void
+ setStudentList(List<Student>): void
+ setUnavailabilityPenaltyList(List<UnavailabilityPenalty>): void
Day
- dateIndex: int
+ Day()
+ Day(long, int)
+ getDateIndex(): int
+ getLabel(): String
+ setDateIndex(int): void
-dayList
0..*
-day
-day
-unavailabilityPenaltyList
0..*
UnavailabilityPenalty
- day: Day
- student: Student
+ getDay(): Day
+ getStudent(): Student
+ setDay(Day): void
+ setStudent(Student): void
+ UnavailabilityPenalty()
+ UnavailabilityPenalty(long, Student, Day)
-studentList
0..*
Student
- courseSize: int
- name: String
+ getLabel(): String
+ getName(): String
+ setName(String): void
+ Student()
+ Student(long, String, int)
+ toString(): String
«property get»
+ getcourseSize(): int
«property set»
+ setcourseSize(int): void
-student
-student
StudentAssignment
- day: Day
- indexInDay: int
- pinned: boolean
- student: Student
+ getDay(): Day
+ getIndexInDay(): int
+ getStudent(): Student
+ isPinned(): boolean
+ setDay(Day): void
+ setIndexInDay(int): void
+ setPinned(boolean): void
+ setStudent(Student): void
+ StudentAssignment()
+ StudentAssignment(long, Day, int)
+ toString(): String
-studentAssignmentList
0..*

Figure 5. UML-Class Diagram

Day define the date of class, **Student** define a student to be assigned to the day and specific time period respectively which is the planning variable. They are the basic classes that encapsulate this real-world problem's core entities: the students and the dates of class they should attend.

Day

- dateindex: date identifier

Student

- name: the students' name
- courseSize: the course size the student should attend to achieve his/her goal, which is generated by our system's recommendation part mentioned before

UnavailabilityPenalty define the student's adjustments which represent someone student cannot attend lesson in some specific day.

UnavailabilityPenalty

- student: the student who cannot attend
- day: the date the student cannot attend

All planning variables have a one-to-many relationship with the planning entity **StudentAssignment**, and the **TeachingSolution** class contains the full set of planning entities.

StudentAssignment

- day: the date of class
- indexInDay: the specific time period in one day
- pinned: to make some student assignments immovable
- student: the student to be assigned to the day and specific time period

TeachingSolution

- studentList: all student need to be assigned in this problem
- dayList: all date of the entire class set by the teacher
- unavailabilitypenaltyList: the date and student pair list, represents someone student cannot attend lesson in some specific day, i.e. the adjustments required by the students
- studentAssignment: the date, the period in this day and the student pair, represents the assignment result
- score: measurement criteria of the solution according to the rules

It clearly highlights the goals of the system and can be used directly to start implementation of knowledge- based system.

3.2.3 Scheduling Part – measurement criteria

According to the interview with teacher and the analysis of project objectives, we extract three rules listed below:

Rule name	Type	Description	score
oneAssignmentPerDayPerStudent	Hard	One student can only take one lesson in a day	Once not staisfied, Hard score -1
enoughStudentCourseSize	Medium	The student should take enough number of lessons recommended by our system according to the student's personal information and specific goal	Once not staisfied, Medium score -1
unavailabilityPenalty	Soft	Student cannot attend lesson the day he/she	Once not staisfied, Soft score -1

		is not available (adjustments submit when they apply for the class)	

Table 3. Drools rules and score’s function

4.0 Solution

4.1 System Architecture

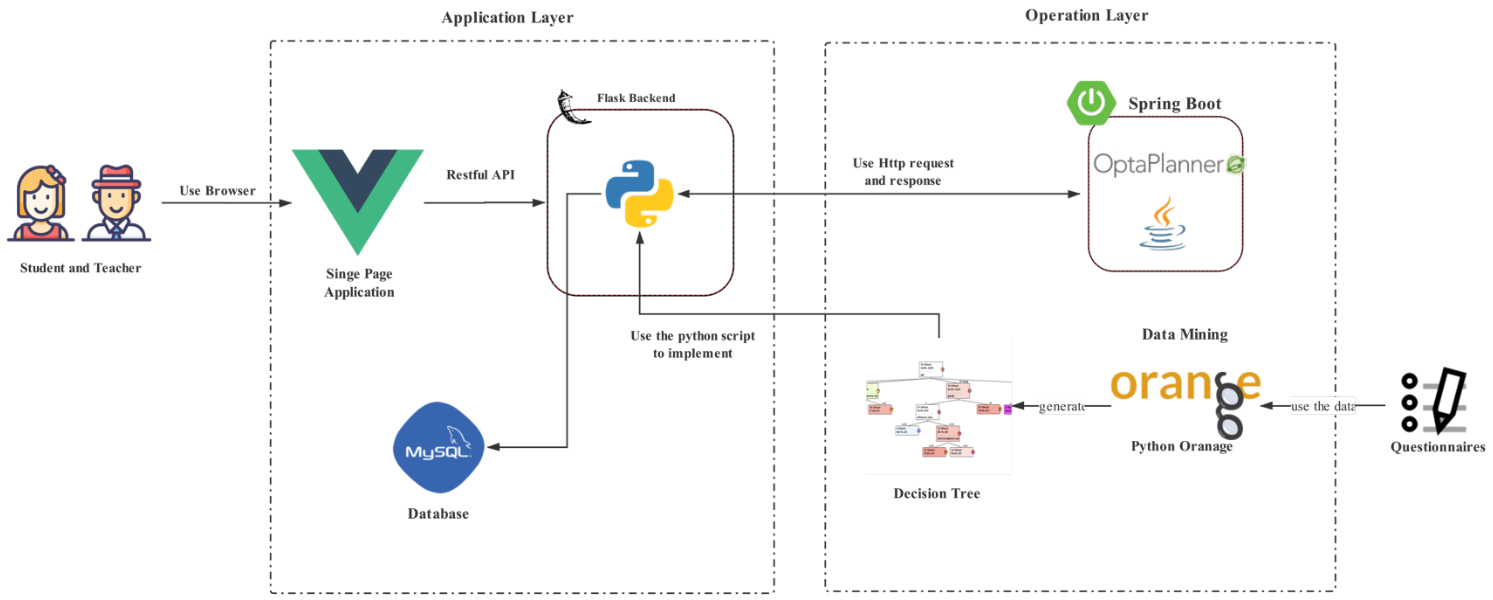


Figure 6. ITAS -- System Architecture

The Intelligent Teaching Assistant System is a hybrid solution using the Vue.js, Flask, Spring Boot, Orange and OptaPlanner.

The system is designed following the Frontend and Backend Separation Architecture, which makes the system highly componentized with high scalability and maintainability. And we also use the docker container as the deployment, which make every part of the system can be updated independently and easy to update and maintain the system.

Based on function, our team split our system into two parts, the application layer and the operation layer.

The application layer of our system uses the separation architecture of frontend and backend, the front stage is a single-page application developed based on vue.js and element.js, and the back stage is a background service that provides RESTful API calls based on Flask development.

The operation layer consists of two parts, one part is responsible for generating the recommendation's rules based on the decision tree, the other part is responsible for scheduling based on the OptaPlanner solver, built in Java8, following the Springboot framework.

For the communication between the application layer and the operation layer, we use the Python multithreading and HTTP request and response to do the asynchronous scheduling and database update.

For the storage of the scheduling result, courses data and students' and teacher's personal profile, we use the MySQL as the database of our system.

4.2 System implementation

4.2.1 Database Design

Considering the requirements, we design a database for our system.

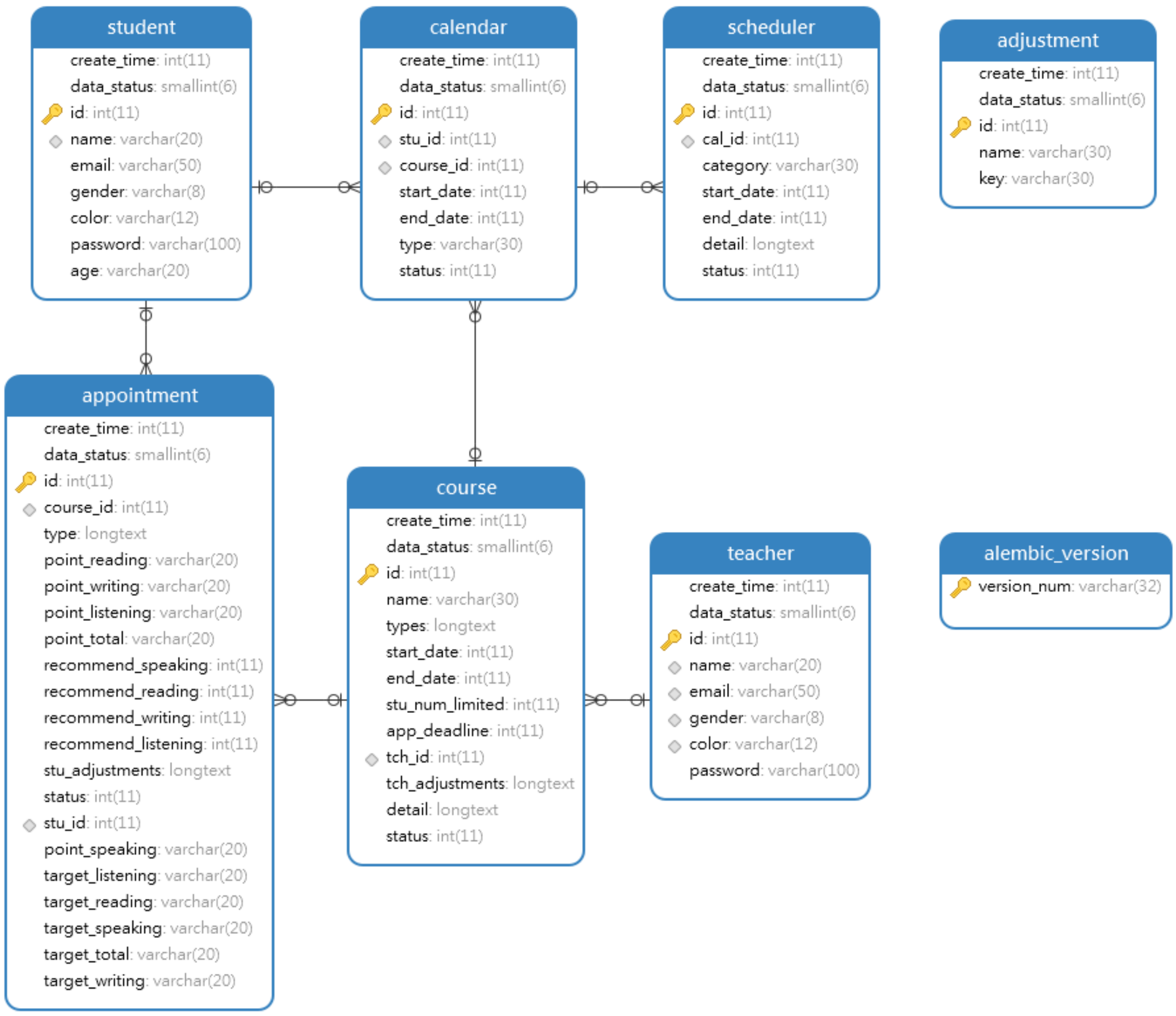


Figure 7. Database ER

Student: student table is designed to store students' profile, including the id, name, age password(encrypted), email, gender and color (used in the personal course schedule)

Teacher: teacher table is designed to store teacher's profile, including the id, name, email, gender, color, password(encrypted)

Course: course tables is designed to store the information of course, including the id, name, types, start_date, end_date, stu_num_limited, app_deadline(appointment collection deadline), tch_id, tch_adjustments, detail and status

Appointment: appointment table is designed to store students' appointment info for target course, including the id, course_id, point_reading, point_writing, point_listening, point_listening, point_total, recommend_speaking, recommend_writing, recommend_listening, stu_adjustments, stu_id, target_listening, target_reading, target_speaking, target_writing, target_total and status.

Calendar: calendar table is designed to store students' calendar info for target course, including id, stu_id, course_id, start_date, end_date, type and status

Scheduler: scheduler table is designed to store students' scheduler for every single lectures, including id, cal_id, category, start_data, end_data, detail and status.

4.2.2 Application Layer

To implement the Application Layer of Intelligent Teaching Assistant System requires three parts of design and programming

- Design and implement the frontend based on the Vue.js and Node.js
- Design and implement the backend based on the Flask

Because we follow the Frontend and Backend Separation Architecture, our system's frontend and backend have their own development and deployment environment. The frontend's development environment is Node.js and the deployment environment is using the Nginx. The backend's development and deployment environment are both Python 3.6.

For the security and user permissions, our system has carried out authorization and restriction of permissions based on pages and API requests respectively in the front and back, ensuring the Security of the system.

1. Design and implement the frontend based on the Vue.js and Node.js

For the design and implementation of the frontend interface, we use the most popular progressive JavaScript framework, Vue.js, and the Vue-element-admin Frontend Solution.

a) UI Design

We use the Vue-element-admin Frontend Solution for our UI design which provides many useful and components and tools. The UI design is mainly designed by using Element.js, which is a Vue 2.0 based component library. We also use other JavaScript component libraries, such as TOAST Calendar UI for scheduling result rendering, which is used in our course and daily scheduler display page.

b) Frontend Architecture and Security

Because of using Vue.js, our frontend Architecture is following the MVVM Architecture, it allows separating the user interface logic from the business logic. And each of our pages is composed of different components, which make the frontend system can be very scalable.

In terms of security, we use the page security authentication based on user's action scope and present different pages through different action scope of each user, which is to ensure that there will be no unauthorized operation. At the same time, we also use Token as the authentication way to interact with the background to ensure that the background Restful API will not be called without permission

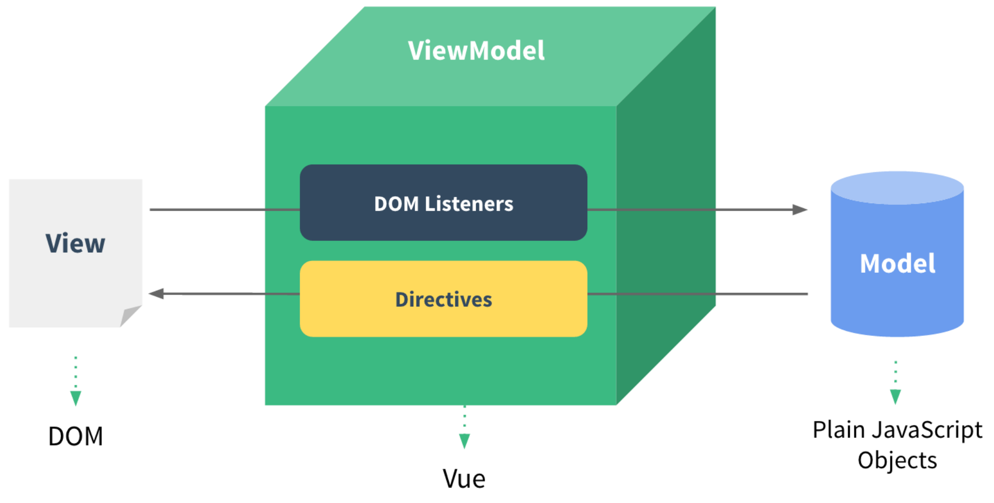


Figure 8. MVVM Framework

2. Design and implement the backend based on the Flask

Our system's backend uses the most popular web application python library, flask to build a Restful API web service, which is very suitable for backend web applications that are separated from the frontend

a) RESTful API and its Security

In the development of backend, we use the combination of the Blueprint and Redprint (modified Blueprint) to build the Restful API and security authentication based on user permissions to ensure the security of Restful API calls

b) Architecture of the backend

In our system, the backend is mainly responsible for providing the RESTful API service for the frontend to get data, communicating with the database to store and select students', teachers' and courses' relevant data, calling the Operation Layer, sending problem and getting the solution.

We use the SqlAlchenmy ORM framework to connect with the database for easily storing and selecting data, the ViewModel to modify the data selected from the database for frontend to display and the multithreading and http request to interact with the Operation Layer making sure this operation will not affect the user experience of the frontend.

4.2.3 Operation Layer

To implement the Operation Layer of Intelligent Teaching Assistant System requires three parts of design and programming

- Implement the recommendation part based on python
- Implement the scheduling part based on OptaPlanner, Drools, Java8, Springboot framewoke

1. Implement the recommendation part based on python

Based on 3.2.1 in knowledge modeling chapter, we interpreted the rules obtained from the decision tree showed before into python language, then our system can do personalized recommendation based on the different input typed in the system by the students. Here shows one example of code:

```
def ielts_writing(self,gender,age,module_point,module_target_point):
  difference_value = module_target_point - module_point
  if age >= 12 and age <=15:
      return (15,19)
  if age >= 16 and age <=18:
      if module_point >= 4.5:
        return (15,19)
      else:
        return (10,14)
  if age >= 19 and age <=22:
    if gender == "male":
      return (10,14)
    elif gender == "female":
      if difference_value <= 0.0:
        return (0,4)
      else:
        if module_point <= 5.0:
          return (10,14)
        else:
          return (10,14)
  if age >=23 and age <=25:
    if module_point <= 5.5:
      return (5,9)
    else:
      if gender == "female":
        return (5,9)
      else:
        return (5,9)
```

Figure 9. Backend decisiontree.py example

2. Implement the scheduling part based on OptaPlanner, Drools, Java8, Spring Boot Framework

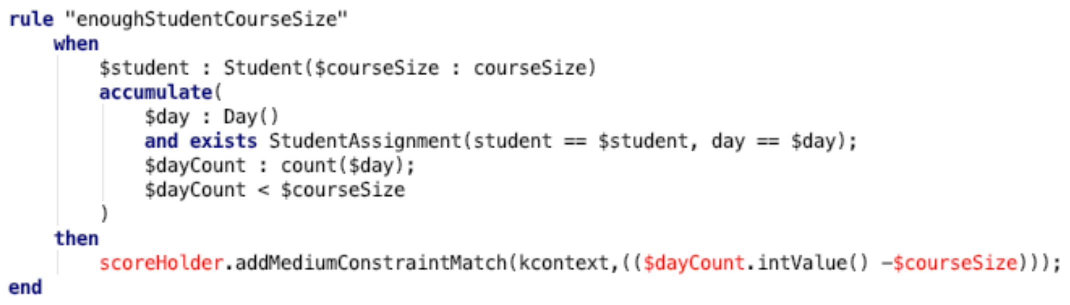
Based on 3.2.2 and 3.2.3 in knowledge modeling chapter, we finished the problem modeling, i.e. the design of the scheduling solution.

a) constraint solver

Our team chooses OptaPlanner as the constraint satisfaction engine because it combines optimization heuristics and metaheuristics with very efficient score calculation. It is a built-in rule engine to accommodate the ever-changing business rules with the Score-based solver (Drools) to assign staff to shifts.

Rules and scores configuration

The rules extracted in chapter 3.2.3 are implemented with Drools, here shows one example of codes:



```
rule "enoughStudentCourseSize"
    when
        $student : Student($courseSize : courseSize)
        accumulate(
            $day : Day()
            and exists StudentAssignment(student == $student, day == $day);
            $dayCount : count($day);
            $dayCount < $courseSize
        )
    then
        scoreHolder.addMediumConstraintMatch(kcontext,(($dayCount.intValue() -$courseSize)));
end
```

Figure 10. Optaplanner teachingScoreRules.drl example

Optimization algorithms

We use Late Acceptance Hill Climbing Search, which is a heuristic search algorithm and accepts non-improving moves when a candidate cost function is better than it was a number of iterations before. LAHC works by maintaining a "late acceptance size", and we adjusted the size to obtain a good search perform.

b) Spring Boot Framework

We use the Spring Boot framework to provide the web interface services for our OptaPlanner solver. By using this framework, we can send the request and the problem file (xml file) to be solved sent by the application layer to the OptaPlanner solver, and then send the solution file generated by OptaPlanner back to the application layer.

4.2.4 Integration

In our system, the calling OptaPlanner part and get the solution part must be asynchronous, because if both calling for solution and responding the frontend request are in the main thread, backend will lock and can't make and respond for the frontend until the Rule Engine give back the solution.

When we used multithreading to interact with the Operation Layer, we encountered a problem that we cannot update and insert the data into the database outside the Flask Content. The problem took us a lot of time to solve, because we were not very familiar with the Python multithreading Programming. After doing the research of the Python and Flask multithreading programming, we successfully used the Flask Content in the other thread and complete the update database operations just after the Rule Engine sending back the solutions.

4.2.5 Deployment

Our system uses docker the container orchestration tool of docker compose to deploy. Containers can interact with each other through the network of docker. Different components can also be used to deploy on different servers (like docker swarm clusters).

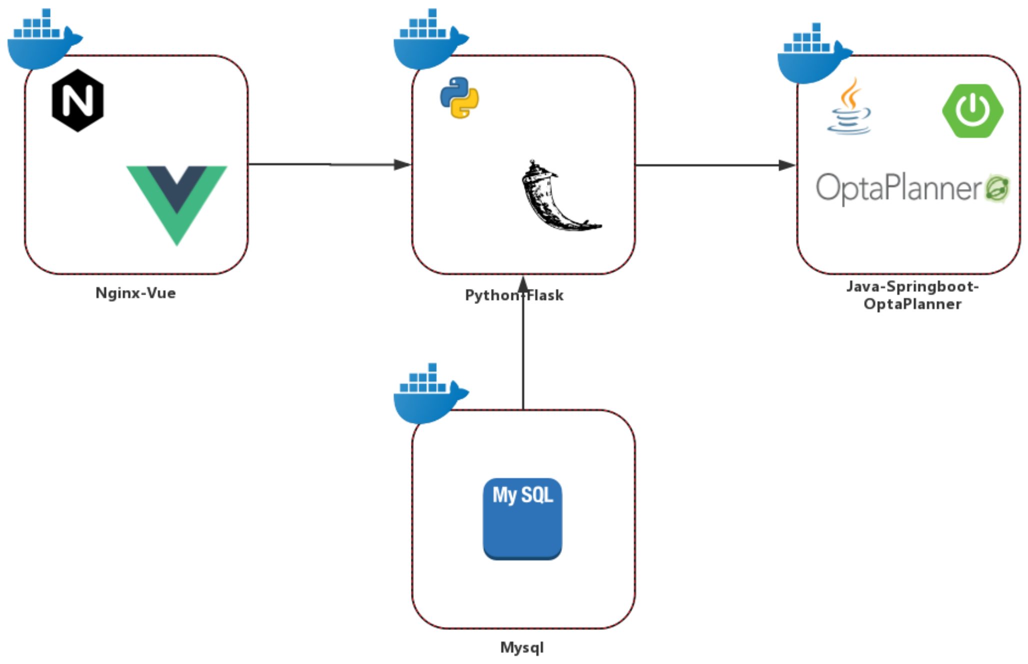


Figure 11. Deployment structure

4.3 Process Flow

The general process flow of our system is described below, and the details are written in the user guide.

1. Register and log in
2. Teacher logs in to the system to create a new class. He/she can set the begin time, end time and applying deadline of the class. And he/she can also add the personal adjustments based on his/her own need, like not teaching at every Saturday.
3. During the application period, student can log in and create application for the class, after typing in their personal information, our system can recommend the number of lessons the student should take to achieve the goal. And students can also add their personal adjustments
4. During the application period, the teacher who set up the class can log in to the system to view and approve the students' applications at any time, or close the application channel at any time.
5. At the end of the application period, the teacher can get the course schedule by just click the button.
6. After our system finished the scheduling process, both teacher and students can log in to the system to check their own calendar.

4.4 System's Features

Ease of Access

The system is both easy to get and easy to use. ITAS is built on a web-based application. After the deployment on a server, the user can easily access it from a browser with Internet connection.

Constraints Configurable

Most of our constraint variables are user configurable so that users can adjust them accordingly to meet their business needs. These variables will then be used by the constraint solver engine.

Highly flexible

Teacher can set up many classes at different time, and the student number will always change at different time period. Depends on the time period, the class size maybe from 50 students in spring course to 500 students in summer course, our system is flexible enough to handle all of these situations.

4.5 Limitations and Future Enhancements

Because of the limited time, our system is designed to accept the adjustments (like not attending or teaching class at weekend and so on) only before the start of the course, to be specific, the time of class set up for teacher or appointment submission for students.

Since our operation layer supports dynamic change restrictions, it is possible for our system to add the function of real-time adjustment submission.

5.0 Conclusion

In this project, our team built a hybrid machine reasoning system combining the knowledge discovery techniques, data mining techniques, reasoning techniques and optimization techniques, which can provide intelligent service throughout the course cycle, like personalized recommendation and constraint satisfaction scheduling.

Through this project, our team now has a better understanding of how data mining techniques like decision tree help to discover knowledge in database, how to use search method to optimize the business resource with the help of OptaPlanner—one of the most popular constraint solver, how to use the Spring Boot framework to provide the web interface service for the OptaPlanner solver and the Python and Flask multithreading programming.

During the whole process, we have encountered many challenges, like when we do the process of generating the problem's xml file which will be sent to OptaPlanner, parsing the returned solution's xml file and saving the solution into the database, we found since the time's format respectively in OptaPlanner, database and front-end are all different, especially for the adjustments part, the teacher's and students' adjustments are weekly based and are saved in string format, we should calculate the accurate date index based on the string saved in the database and send it to OptaPlanner, it took us quite a lot of time to overcome this challenge. But when we overcome this difficulty and get the perfect schedule meets all the requirements asked by teacher and students, all efforts are worthwhile.

6.0 Bibliography

OptaPlanner User Guide

https://docs.optaplanner.org/7.24.0.Final/optaplanner-docs/html_single/index.html#plannerIntroduction

OptaPlanner Workbench and Execution Server User Guide

https://docs.optaplanner.org/7.1.0.Final/optaplanner-wb-es-docs/html_single/#_optaplanner_engine

TOAST UI Calendar

https://nhn.github.io/tui.calendar/latest/

Vue

https://vuejs.org/index.html

Vue-element-admin

https://panjiachen.github.io/vue-element-admin-site/

Flask

https://flask.palletsprojects.com/en/1.1.x/

7.0 APPENDICES

APPENDIX A

Questionnaire

Due to the complexity of the questionnaire structure and hard to intuitionistic display, we provide only the link to the questionnaire

https://www.wenjuan.com/project/share/5d3e41c892beb5426e4787f2/?new=0