Design requirements

for

**Schedule Sharp**

by

Richie Chan 108850198,

Minjung Lee 108512030,

Chaehoon Lim 109039844,

& Nikhil Nair 109307897

**Table of Contents**

1. General Information
   1. Overview………………………..……………………....... 1
   2. Assumptions………………………..…………………….. 1
   3. Functionality for Administrators…………………………. 2
   4. Functionality for Students………………………..………. 2
   5. Other Requirements………………………..……….…….. 3
2. Use Case Diagrams
   1. Administrators………………………..……….………….. 5
   2. Log in………………………..……….…………………... 6
   3. Students………………………..……….………………… 6
3. Use Cases
   1. Administrators………………………..……….………….. 8
   2. Students………………………..……….………………… 9
4. Contributions………………………..……….……………….. 12

1

**Overview**

(As distributed by Professor Stoller on Blackboard)

This system helps high school students create course schedules that include desired courses with preferred instructors and overlap with their friends’ schedules.

There are several course schedule planning systems for university students, but few for high school students. There are several differences. Many high schools do not publish complete course schedules: they simply send each student his/her course schedule, based on the student's course preferences. By aggregating their course schedules, students can infer the complete course schedule, and use it to make informed requests for schedule changes. On average, there are more sections of each high school course than each university course, so optimizing the choice of sections is more important. High school students are present during the entire school day, so it is irrelevant to select courses or sections that are clustered on a few days of the week or in consecutive timeslots. The choice of sections is therefore based on other criteria, for example, the instructor or coordination with friends.

**Assumptions**

The academic year is divided into S units. S must be in the range of 1 to 4. For convenience, these units are called "semesters" in this document, regardless of the number (even though they may be academic years, trimesters, or quarters).

Within each semester, the class schedule repeats every D days. D must be in the range of 1 to 7. For example, a school might have a 5-day schedule (i.e., a different schedule on each weekday) or a 2-day schedule (i.e., two daily schedules that alternates).

The number P of periods in each day of the schedule is the same. P must be in the range of 6 to 12. The system does not need to know the start and end times of the periods.

The schedule is organized into "schedule blocks". A schedule block is a set of timeslots in the same period. Schedule blocks may overlap. For example, in a school with a 2-day schedule, there might be a schedule block <period 1; days 1,2> for daily classes, and schedule blocks <period 1; day 1> and <period 1; day 2> for classes that meet on alternate days.

Each course is scheduled in one schedule block in some range of semesters. For example, in a school with 4 semesters/year, a course might be in a schedule block in semester 1, or semesters 1-2, or semesters 1-4, etc.

Each course has a unique identifier (e.g., "PHY101" or "0159") and a name.

Each section of a course is identified by the range of semesters and schedule block in which it is scheduled. For example, a section of PHY101 might be offered in (and uniquely identified by) semesters = 1-2 and schedule block = <period 1; days 1,2>.

Lunch is offered in the same range of periods (for example, periods 3 through 5) every day. Students may have lunch on any subset of the schedule days, not necessarily every day.

The system ignores co-requisite requirements. For example, Chemistry and Chemistry Lab might be co-requisites, but the system treats them as separate courses, each in its own schedule block.

2

The system ignores study halls: it treats them as equivalent to empty timeslots.

The system uses only information available to students. It does not have access to an authoritative database of course offerings. It creates its own database of course offerings by aggregating the assigned course schedules entered by students.

**Functionality for Administrators**

3.1. ADD OR DELETE SCHOOL. The administrator specifies the following information for each school and academic year: the number S of semesters in an academic year, the number D of days in the schedule, the number P of periods in each day, the range of periods in which lunch is offered, and the set of all legal schedule blocks.

3.2. MANAGE STUDENT ACCOUNT REQUESTS. The system displays a list of pending student account requests. The administrator can approve or delete a pending request, or leave it in the list of pending requests. For convenience, there is an "Approve all pending requests" function. When an account request is approved, the system sends a notification to the student by email.

3.3. DELETE STUDENT ACCOUNTS. A specified student account is deleted.

Comment: Administrator accounts are created by direct modification of the account database. The system does not need to provide an interface for this.

**Functionality for Students**

The system keeps track of two course schedules for each student for each academic year: (1) an assigned course schedule, created by the school and entered in this system by the student, and (2) a desired course schedule, generated by this system and possibly edited by the student.

4.1. REQUEST ACCOUNT. The student fills out a form with fields for school, student name, email address, password, etc.

4.2. MANAGE FRIENDS. The student enters a first name and last name. If an account with that name exists, a friend request is sent to the other user. If the other user accepts the request, then the two users are friends. The friend relation is symmetric. For privacy, students cannot browse the list of user accounts or do wildcard searches on it. A student can unfriend a friend.

4.3. DISPLAY FRIENDS. The system displays the student's friends and their assigned and desired course schedules for a specified school and academic year.

4.4. ENTER, VIEW, AND EDIT ASSIGNED COURSE SCHEDULE. The student enters his assigned course schedule for a specified school and academic year, by entering, for each course that appears in his/her schedule, the range of semesters, the schedule block, and the instructor. The student does not need to enter study halls, because the system ignores them, or lunches, because the system already knows when lunch is offered. When entering a course, the student first enters the course identifier. If the system has seen that course identifier in any student's assigned course schedule for the same school and academic year, then some student has already entered the course name, and the system displays it. Otherwise, the system asks the student to confirm or enter the name of the course; specifically, if the system saw the course identifier for the same school in the previous academic year, it displays the associated course name, and the student needs to enter the course name only if it has changed. Similarly, if the system has seen that course identifier in any student's assigned course schedule for the same school, academic year,

3

range of semesters, and schedule block, then some student has already entered the instructor, and the system displays it; otherwise, the system the system asks the student to enter the name of the instructor.

The assigned course schedule can be viewed with or without friends’ schedules. "With friends' schedules" means that each entry includes the names of friends in the class.

4.5. VIEW COURSE OFFERINGS. The system lists all courses offered by the specified school in the specified academic year, sorted by course identifier, and with full details about each section of the course, including the number of students who listed that section in their assigned course schedule.

4.6. GENERATE DESIRED COURSE SCHEDULE. The desired course schedule is generated from scratch (not derived from the assigned course schedule), based on the following criteria entered by the student: (1) list of courses to include in the schedule, each optionally with a set of sections to exclude, and optionally with a preferred instructor; (2) whether lunch is desired, for each day in the schedule. Note that the system does not know whether a section is open (i.e., has available seats) or closed. A student who knows that a section is closed can exclude it.

The system stores each student's schedule generation criteria, so a student can later edit them and re-generate the desired course schedule.

The system attempts to generate a conflict-free schedule that includes all of the specified courses and lunches and does not contain any excluded sections. For brevity, a schedule satisfying these conditions is said to be "acceptable". If this is impossible, the system generates a list of all minimal-size sets of changes to the list of desired courses and lunches that lead to an acceptable schedule, where each change is removal of one course or all lunches.

If an acceptable schedule exists, the system generates an optimal one. A schedule is optimal if (C1) it has the maximal number of courses with preferred instructors, and (C2) among schedules satisfying condition C1, it has maximal overlap with friends' schedules. The overlap is measured by summing the number of friends in each section of a course in the student's

schedule.

4.7 VIEW DESIRED COURSE SCHEDULE. The desired course schedule can be viewed with or without friends' schedules.

4.8. EXPORT DESIRED COURSE SCHEDULE. The desired generated course schedule is exported in some standard format, such as a plain text file an HTML file. It can be exported with or without friends' schedules.

**Other requirements**

5.1 AUTHENTICATION. All access to the system, other than student account request, requires authentication with a password.

5.2 NETWORK SECURITY. Communication is secured using HTTPS or SSL.

5.3 CONCURRENCY. Multiple administrators and multiple students on different computers may use the system concurrently. Synchronization is used to ensure sensible behavior. The system may prevent the

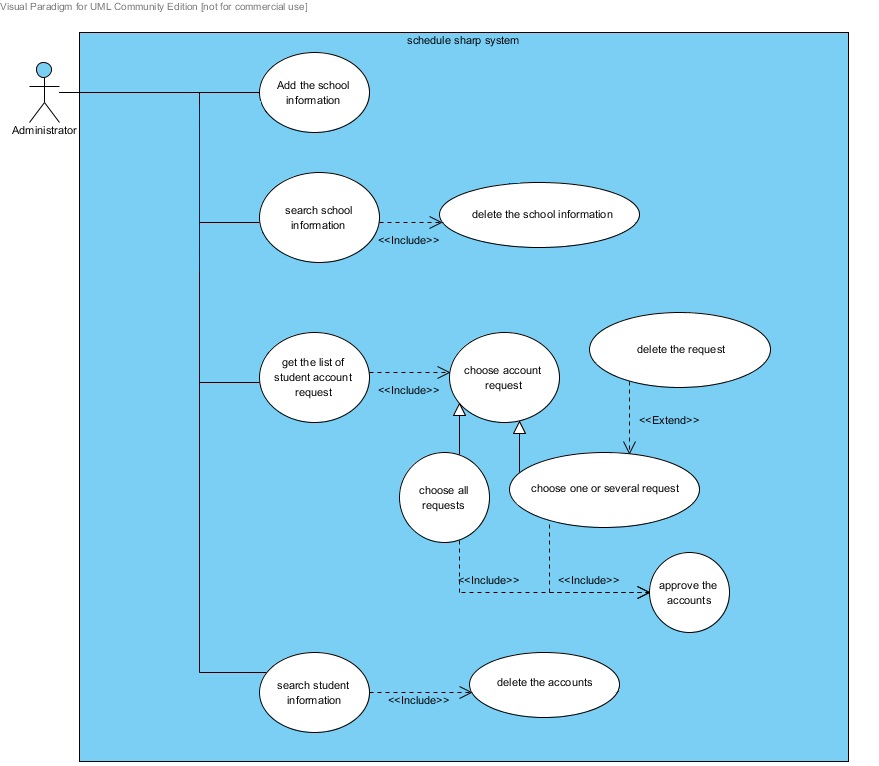
4

same user from having multiple concurrent login sessions; if it does not prevent this, it should ensure sensible behavior.

5

**Use Case Diagrams**

**Administrator**

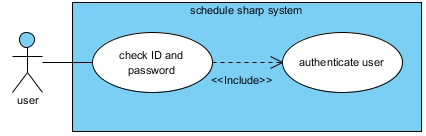
**

*Use Case Diagram 1 - Illustrates the actions that an administrator may undertake.*

The administrator’s (admin) role in the system is critical, as he/she has to perform actions that are important for the system to remain active. The admin’s responsibilities begin with adding schools to the system; when a new school is added, the information is relayed to an internal database that would allow students of the certain school to request an account. After adding a new school, the admin may also search for a school to view and/or delete schools for various purposes. Additionally, the admin has the authority to accept or delete the requests from students to join a school in the system. Two choices are present in this scenario - choose one or all the student requests; the admin may decide to approve all the student accounts, while the choice to delete the student accounts is also a possibility (therefore the “extend”). Finally the admin has the capability to search for student information, whereby they may delete an existing student account or perform no action.

6

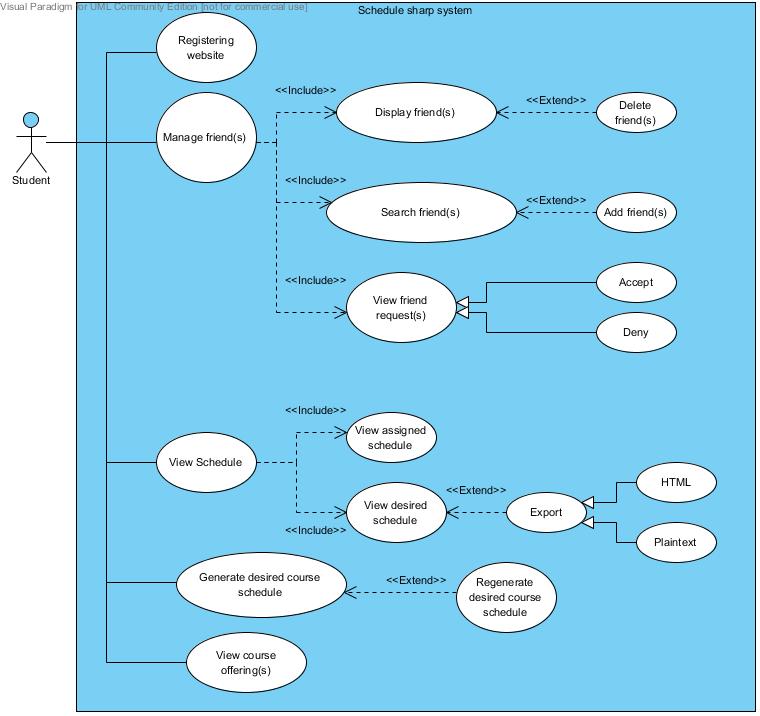
**Log in for Administrator or Student**

**

*Use Case Diagram 2 – Illustrates the actions that administrator and student login the system*

Since the actions taken by both the student and the administrator will remain the same for login, they can be generalized as a user.

**Student**



*Use Case Diagram 3 - Illustrates the actions that a student may perform.*

7

The student is the user base being targeted by the system. Initially, the only action that a student may perform is to submit a request on the registration page; then, the student awaits an email response after the admin has accepted the request. Thereafter, the student may perform various tasks including viewing their friends list and friend requests, and delete a friend. The user may also search for a first and last name, view the results and then choose to add a friend if they prefer (therefore, it’s an “extend”). Additionally, the student may view the two types of course schedules - assigned and desired. While viewing the course schedules, the student may decide view whether they share the schedules with any friends; though it’s useful, the feature has been omitted from the use case diagram as it’s a minute detail of a large system. Lastly, the user may generate a desired course schedule based on the schedules of their friends.

8

**Use Case**

**Administrator**

**Use case: Login to system**

Primary actor: Administrator or Student

Secondary actor: None

Primary flow:

1. User inputs id and password
2. The system check the id and password
3. The system prompts user for the main page

**Use case: Add school information (3.1)**

Primary actor: Administrator

Secondary actor: None

Primary flow:

1. Administrator enters the information for a school.
2. Administrator submit the information and the system saves the information.

**Use case: Delete school information (3.1)**

Primary actor: Administrator

Secondary actor: None

Primary flow:

1. Administrator enters the information for a school.
2. Administrator submits the information and the system deletes the school information.

Secondary flow: Administrator does not delete the school information

2.1. Administrator cancels the action and the system prompts administrator for the main page

**Use case: Manage student account request (3.2)**

Primary actor: Administrator

Secondary actor: None

Primary flow:

1. The system prompts administrator for the list of student information that is requested.
2. Administrator chooses one or several pending requests.
3. Administrator approves the students account.

Alternative flow: Approve few students on the list

2.1. Administrator chooses all students.

2.2. Administrator approves the student account.

Alternative flow: Delete pending requests

3.1. Administrator deletes the student account request.

9

Alternative flow: Administrator does not delete the student request

3.1. Administrator cancel and the system prompts administrator for the main page

**Use case: Delete student account system (3.3)**

Primary actor: Administrator

Secondary actor: Primary flow:

1. Administrator enters the information about a student on the search engine.
2. Administrator submits the information and the system deletes the student account.

Secondary flow: Administrator does not delete the student account

* 1. Administrator cancels and the system prompts administrator for the main page.

## Students

**Use case: Request Account (4.1)**

Primary Actor: Student

Secondary Actor: None

Primary flow:

1. Student visits the registration page.
2. Student completes the information on the page.
3. Finally, the student submits the form.

Alternative flow: Password is too short

3.1. Website displays “Your password must be at least 8 characters long”

Alternative flow: Email address is invalid

3.1. Website displays “Invalid email address”

**Use case: Manage Friends (4.2)**

Primary Actor: Student

Secondary Actor: None

Primary flow:

1. Student enters first and last name, and searches for possible profiles with a matching name.
2. Student requests other student as a friend.

Alternative flow: Name does not exist

2.1. Website displays “A profile with that name does not exist”

**Use case: Display Friends (4.3)**

Primary Actor: Student

Secondary Actor: System

Primary flow:

1. Student enters a certain school and year to filter the results.
2. Student can view a friend’s assigned and desired course schedules for a specified school and academic year.

10

Alternative flow: No friend(s)

* 1. Website displays “Please add a friend first”

Alternative flow: Invalid school or academic year

2.1. Website displays “Invalid school or academic year”

**Use case: Enter Assigned Course Schedule (4.4)**

Primary Actor: Student

Secondary Actor: None

Primary flow of events:

* 1. Student begins to enter the course identifier.
  2. System auto-fills the course identifier if it has been entered by any student in the same school.
  3. Students fills the required fields: schedule block, range of semesters and academic year.
  4. System auto-fills the instructor field if the required fields matches any course from previous semesters.
  5. The form is submitted by the student.
  6. The assigned course schedule is saved in the system.

Alternative flow of events: Confirm the course identifier

2.1 The student receives a prompt to confirm the course identifier if it does not exist already.

2.2 Student confirms or cancels the new course identifier.

Alternative flow of events: Enter the instructor name

4.1 Prompt the user to enter the instructor name.

**Use case: View/Edit Assigned Course Schedule (4.4)**

Primary Actor: Student

Secondary Actor: System

Primary flow of events:

* 1. Student views the course schedule.
  2. Student chooses to edit the schedule.
  3. Once the edits are complete, student submits the new schedule.
  4. The edited course schedule is saved in the system.

**Use case: View friends’ schedules (4.4)**

Primary Actor: Student

Secondary Actor: None

Primary flow of events:

* 1. Student views the course schedule.
  2. Student chooses to view the courses of friends’ for the semester.

Alternative flow of events: No friends

2.1 Notify the user that they have not added any friends, yet.

**Use Case: View course offerings (4.5)**

Primary Actor: Student

Secondary Actor: System

11

Primary Flow:

1. Student selects school.
2. Student selects academic year.
3. Student submits information.
4. Course offerings are displayed and sorted.

**Use Case: Generate desired course schedule (4.6)**

Primary Actor: Student

Secondary Actor: System

Primary Flow:

1. Student enters criteria for generating schedule.
2. Student submits form.
3. Webpage saves the criteria to the database.
4. The system generates an optimal schedule based on the criteria.

Alternative Flow: Schedule impossible

1. System generates list of minimal-size sets of changes to list.
2. Student changes criteria.
3. Student submits form.
4. Go to Step 3 (Primary Flow).

**Use Case: View desired course schedule (4.7)**

Primary Actor: Student

Secondary Actor: System

Primary Flow:

1. Student selects course view
2. System displays the corresponding view.

**Use Case: Export desired course schedule (4.8)**

Primary Actor: Student

Secondary Actor: System

Primary Flow:

1. Student requests to export his generated schedule.
2. System prompts student to select a format
3. System generates the requested format

12

**Contributions**

Richie Chan - Completed uses cases 4.5, 4.6, 4.7, & 4.8.

Minjung Lee - Completed use cases 3.1, 3.2 & 3.3. Also completed use case diagram for administrator.

Chaehoon Lim - Completed use cases 4.1, 4.2 & 4.3.

Nikhil Nair - Completed use case 4.4, and formatted the documentation.

The student use case diagram was conceptualized and completed by Richie, Chaehoon and Nikhil.