Project:High-School Schedule Planner (HISP)

Requirements Document

CSE 308 Software Engineering

Section 01

Professor Scott Stoller

Team: Seriously?

Team Members:

Thang Bui (thang.bui@stonybrook.edu)

Hung Pham (hung.pham@stonybrook.edu)

Kevin Qi (kevin.qi@stonybrook.edu)

Wei Tang (wei.tang@stonybrook.edu)

Contents

1 [Overview](https://docs.google.com/a/stonybrook.edu/document/d/1sDXDaGlVY6Jh2DdnY207I4cX-S47MNS1oYt2vB7SMbo/edit#heading=h.qafigsun3guw) *3*

2 [Functional Requirements](https://docs.google.com/a/stonybrook.edu/document/d/1sDXDaGlVY6Jh2DdnY207I4cX-S47MNS1oYt2vB7SMbo/edit#heading=h.vyzkzyrhie82) *4*

2.1 [Functionality for Administrators](https://docs.google.com/a/stonybrook.edu/document/d/1sDXDaGlVY6Jh2DdnY207I4cX-S47MNS1oYt2vB7SMbo/edit#heading=h.bu0hqoyq3bq8) *4*

2.2 [Functionality for Students](https://docs.google.com/a/stonybrook.edu/document/d/1sDXDaGlVY6Jh2DdnY207I4cX-S47MNS1oYt2vB7SMbo/edit#heading=h.4lqmbrkizcgl) *5*

3 [Other Requirements](https://docs.google.com/a/stonybrook.edu/document/d/1sDXDaGlVY6Jh2DdnY207I4cX-S47MNS1oYt2vB7SMbo/edit#heading=h.vcomii2d6nz6) *9*

4 [Use Case Diagrams](https://docs.google.com/a/stonybrook.edu/document/d/1sDXDaGlVY6Jh2DdnY207I4cX-S47MNS1oYt2vB7SMbo/edit#heading=h.fqs7ond5vgt) *10*

5 [Use Case Descriptions](https://docs.google.com/a/stonybrook.edu/document/d/1sDXDaGlVY6Jh2DdnY207I4cX-S47MNS1oYt2vB7SMbo/edit#heading=h.d6d0x76ybgi4) *13*

6 System Architecture *22*

7 Object Model *23*

8 Graphical User Interface/System Flow *24*

9 Dynamic Model *43*

10 [Languages, Technologies, and Tools](https://docs.google.com/a/stonybrook.edu/document/d/1sDXDaGlVY6Jh2DdnY207I4cX-S47MNS1oYt2vB7SMbo/edit#heading=h.pbkjhxifv55b) *48*

# **1. Overview**

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 Many high schools do not publish complete course schedules: they simply send each student his/her course schedule, based on the student's course preferences.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.

# **2. Functional Requirements**

## 2.1 Functionality for Administrators

*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.  
  
Comment: To enter the set of legal schedule blocks, a text-based interface  
might be convenient. For example, the user could provide a single chunk of  
text describing all schedule blocks, each expressed using a syntax such as  
the one above, or some other convenient syntax you define. This provides  
an easy way to copy and modify the set of schedule blocks from the previous  
academic year (assuming the system displays the data in the same syntax  
that it was entered): just copy, paste, and edit the text.  
  
*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*. 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.

**2.2 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.  
  
*1*. REQUEST ACCOUNT. The student fills out a form with fields for school,  
student name, email address, password, etc.  
  
*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.  
  
*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*. 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, 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.  
  
*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.  
  
*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.  
  
*7*. VIEW DESIRED COURSE SCHEDULE. The desired course schedule can be  
viewed with or without friends' schedules.  
  
*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.

# 

# **3. Other Requirements**

*1*. AUTHENTICATION. All access to the system, other than student account  
request, requires authentication with a password.  
  
*2*. NETWORK SECURITY. Communication is secured using HTTPS or SSL.  
  
*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 same user  
from having multiple concurrent login sessions; if it does not prevent  
this, it should ensure sensible behavior.

# **4. Use Case Diagrams**

# CSE 308 Project diagram.jpg

Figure 1: Administrator use case diagram

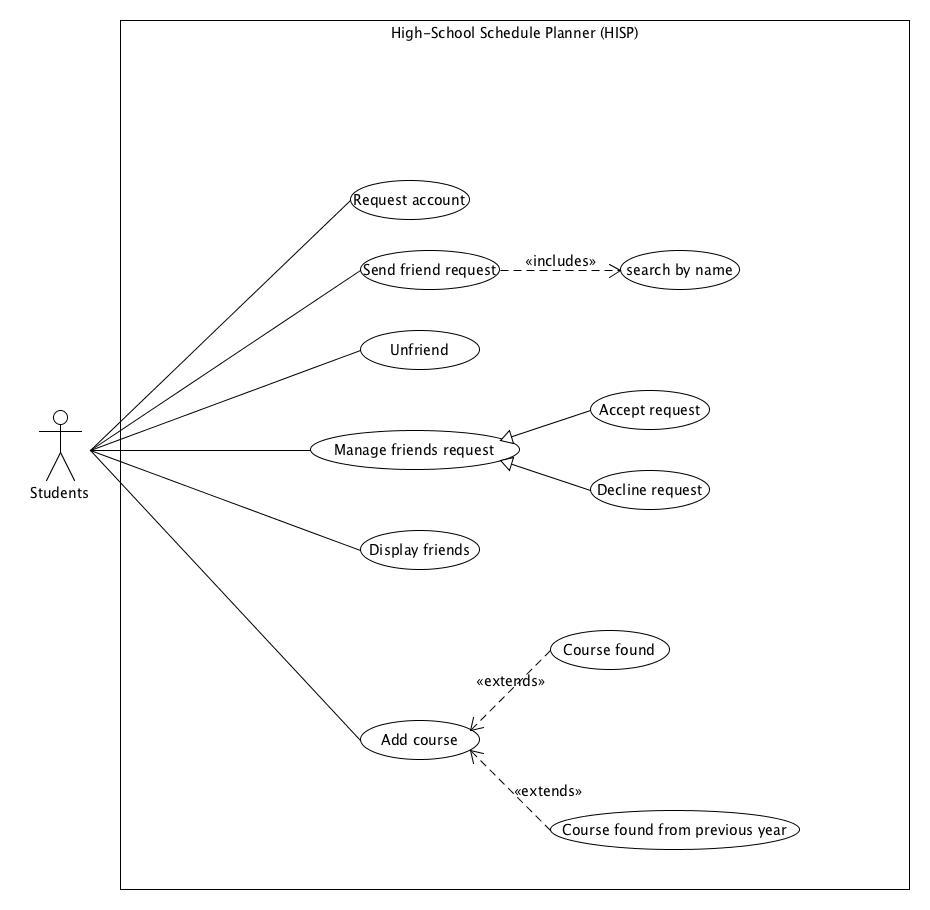


Figure 2. Student use case diagram

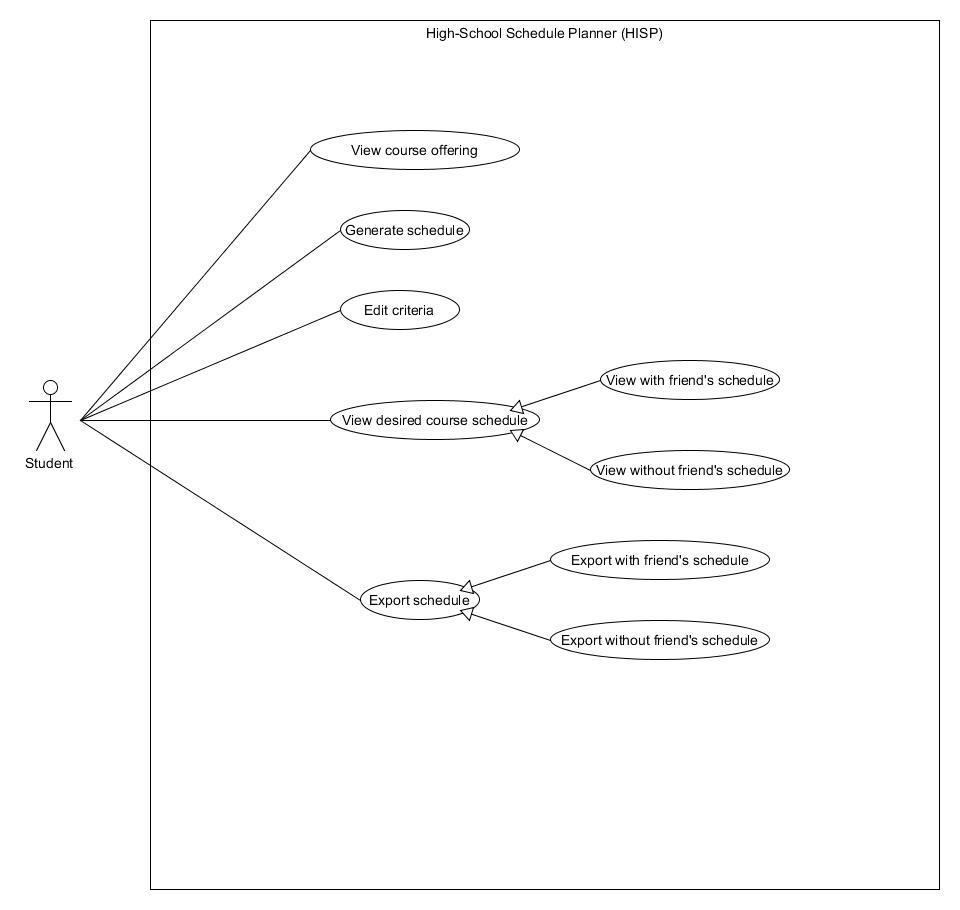


Figure 3: Student use case diagram (cont.)

# 

# **5 Use Case Descriptions**

**Use Case**: Add School

**Primary Actor:** Administrator

**Secondary Actor**: N/A

**Primary Flow:**

1. Admin specifies information for the school to be added
2. Admin clicks ‘Add School’
3. School is added to the system successfully
4. System displays ‘School added successfully!’

**Alternative Flow:** School already exists in the system

3.1 The school is not added to the system

3.2 System displays ‘School already exists!’

**Precondition:** Administrator is logged in

**Use Case**: Delete School

**Primary Actor:** Administrator

**Secondary Actor**: N/A

**Primary Flow:**

1. Admin enters school to be deleted
2. Admin clicks ‘Delete School’
3. The school is in the system and is deleted successfully
4. System displays ‘School successfully deleted!’

**Alternative Flow:** School does not exist in the system

3.1 The school is not found in the system

3.2 System displays ‘School does not exist!’

**Precondition:** Administrator is logged in

**Use Case**: Approve Students Account request

**Primary Actor:** Administrator

**Secondary Actor**: N/A

**Primary Flow:**

1. System displays all pending students account request.
2. Administrator can approve student’s pending account request.
3. System will send notification to approved student.

**Precondition:** Administrator is logged in

**Use Case**: Delete Student Account request

**Primary Actor:** Administrator

**Secondary Actor**: N/A

**Primary Flow:**

1. System displays all pending students account request
2. Administrator deletes student’s pending account request
3. System will send notification to deleted student

**Precondition:** Administrator is logged in

**Use Case**: Delete Student Account

**Primary Actor:** Administrator

**Secondary Actor**: N/A

**Primary Flow:**

1. Execute ‘Search by Name’
2. Administrator searches for the account to be deleted in result set and clicks ‘Delete’ to delete the student account
3. System will prompt a confirmation message to the Administrator
4. Administrator clicks Confirm to finish the action
5. The system will send a notification email to the the account that has been deleted

**Precondition:** Administrator is logged in

**Use Case** : Student request account

**Primary Actor**: Student

**Secondary Actor**: N/A

**Primary Flow:**

1. Student clicks on ‘Request New Account’
2. System prompts student with a form containing fields for related information such as school, student’s name, email address, password, etc.
3. Student fills out the form
4. Student clicks submit to send the request
5. System approves or denies request and notifies student

**Use Case** : Send friend request

**Primary Actor**: Student

**Secondary Actor**: N/A

**Primary Flow:**

1. Execute ‘Search by Name’
2. Student searches for friend in result set and clicks ‘Add Friend’ to add the specified friend to buddy list
3. Friend request will be sent to the specified friend

**Precondition:** Student is logged in with an approved account

**Use Case** : Search by Name

**Primary Actor**: Student

**Secondary Actor**: N/A

**Primary Flow:**

1. Student clicks on ‘Search’
2. Student enters a first and last name
3. System will show the list of accounts with the same first and last name entered

**Precondition:** Student is logged in with an approved account

**Use Case** : Unfriend

**Primary Actor**: Student

**Secondary Actor**: N/A

**Primary Flow:**

1. Student clicks on ‘Friends’
2. List of all the student’s friends will be displayed
3. Student clicks ‘Unfriend’ to unfriend a specified person

**Precondition:** Student is logged in with an approved account

**Use Case** : Accept Request

**Primary Actor**: Student

**Secondary Actor**: N/A

**Primary Flow:**

1. A friend request will be prompted if someone sends the student a friend request
2. Student clicks ‘Accept’ to be friends with the account
3. The system will send a notification to sender if the friend request is accepted

**Precondition:** Student is logged in with an approved account

**Use Case** : Decline Request

**Primary Actor**: Student

**Secondary Actor**: N/A

**Primary Flow:**

1. A friend request will be prompted if someone sends the student a friend request
2. Student clicks ‘Decline’ to decline the request

**Precondition:** Student is logged in with an approved account

**Use Case** : Display friends

**Primary Actor**: Student

**Secondary Actor**: N/A

**Primary Flow:**

1. Student clicks on ‘Friends’
2. List of all of the student’s friends will be displayed
3. Student clicks on the friend’s account to check the friend’s schedules
4. Student clicks to choose the school from the list of the friend’s school
5. Student clicks to choose the academic year
6. The friend’s assigned schedule and desired schedule will be displayed for the chosen school and academic year

**Precondition:** Student is logged in with an approved account

**Use Case**: Add course

**Primary Actor**: Student

**Secondary actor**: N/A

**Primary Flow**:

1. Student enters his assigned course identifier, school and academic year.
2. System prompts student to enter new course name, schedule block, and instructor’s name
3. Student fills out the information
4. Student clicks on ‘Submit’ button to add his assigned course

**Alternative Flow**: Course found for the corresponding academic year

2.1. The system display the course information. Student can click “View with friends” buttons to reveal their friends taking that course.

2.2. Student clicks on ‘confirm’ button to add his assigned course

**Alternative Flow**: Course not found in that corresponding academic year but from previous year.

2.1. The system displays it. Student can click “View with friends” buttons to reveal their friends taking that course.

2.2. The student has the options to edit course name and instructor’s name

2.3. Student clicks on ‘confirm’ button to add his assigned course

**Precondition:** Student is logged in with an approved account

**Use Case** : View Course Offerings

**Primary Actor**: Student

**Secondary Actor**: N/A

**Primary Flow:**

1. Students enters specific academic year and school
2. Student clicks ‘View Course Offering’
3. System lists all courses offered by the specified school in the specified academic year including the number of students who listed that section in their assigned course schedule

**Precondition:** Student is logged in with an approved account.

**Use Case:** Generate Desired Course Schedule

**Primary Actor:** Student

**Secondary Actor**: N/A

**Primary Flow:**

1. Student enters schedule criteria(list of courses and sections, preferred instructor, whether lunch is desired)
2. Criteria is stored in system
3. Student edits criteria
4. Student presses ‘Generate schedule’
5. System generates the optimal schedule and outputs it to the screen

**Precondition:** Student is logged in with an approved account.

**Use Case :** View Desired Course Schedule with Friends’ Schedules

**Primary Actor:** Student

**Secondary Actor:** N/A

**Primary Flow:**

1. Student clicks on ‘View Desired Schedule’
2. Student clicks on “With Friends’ Schedules” to view own schedule and his/her friends’ schedules
3. System displays desired course schedule with friends’ schedules

**Precondition:** Student is logged in with an approved account.

**Use Case :** View Desired Course Schedule without Friends’ Schedules

**Primary Actor:** Student

**Secondary Actor:** N/A

**Primary Flow:**

1. Student clicks on ‘View Desired Schedule’
2. Student chooses to view desired course schedule without friends’ schedules
3. System displays desired course schedule without friends’ schedules

**Precondition:** Student is logged in with an approved account.

**Use Case** : Export Desired Course Schedule with Friends’ Schedules

**Primary Actor:** Student

**Secondary Actor:** N/A

**Primary Flow**:

1. Student chooses to export desired course schedule
2. The desired schedule of the student and his/her friends are exported in specified format

**Precondition:** Student is logged in with an approved account.

**Use Case** : Export Desired Course Schedule without Friends’ Schedules

**Primary Actor:** Student

**Secondary Actor:** N/A

**Primary Flow**:

1. Student chooses to export desired course schedule without friends’ schedules
2. The desired schedule of student is exported in specified format

**Precondition:** Student is logged in with an approved account.

**6. System Architecture**

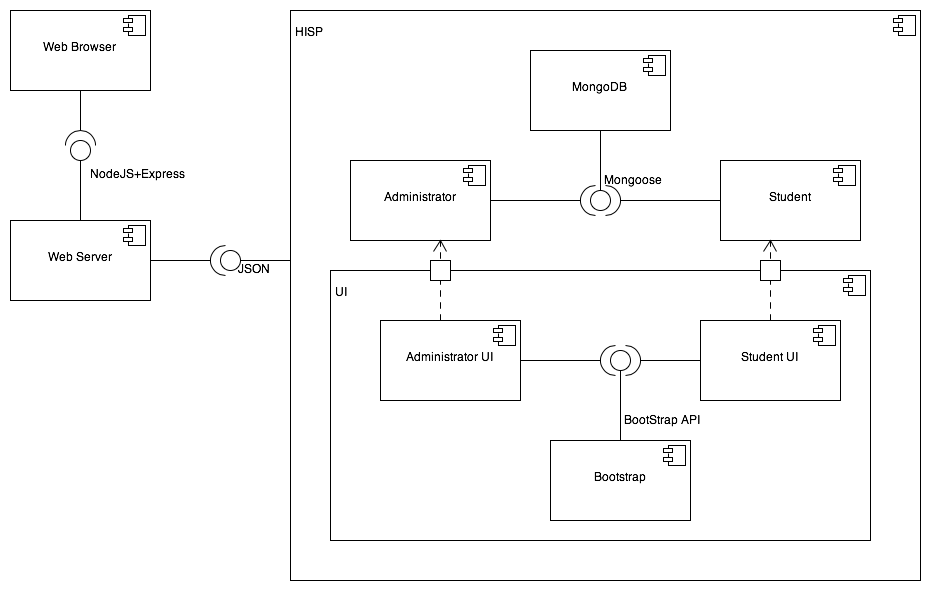


Figure 6.1 Component diagram

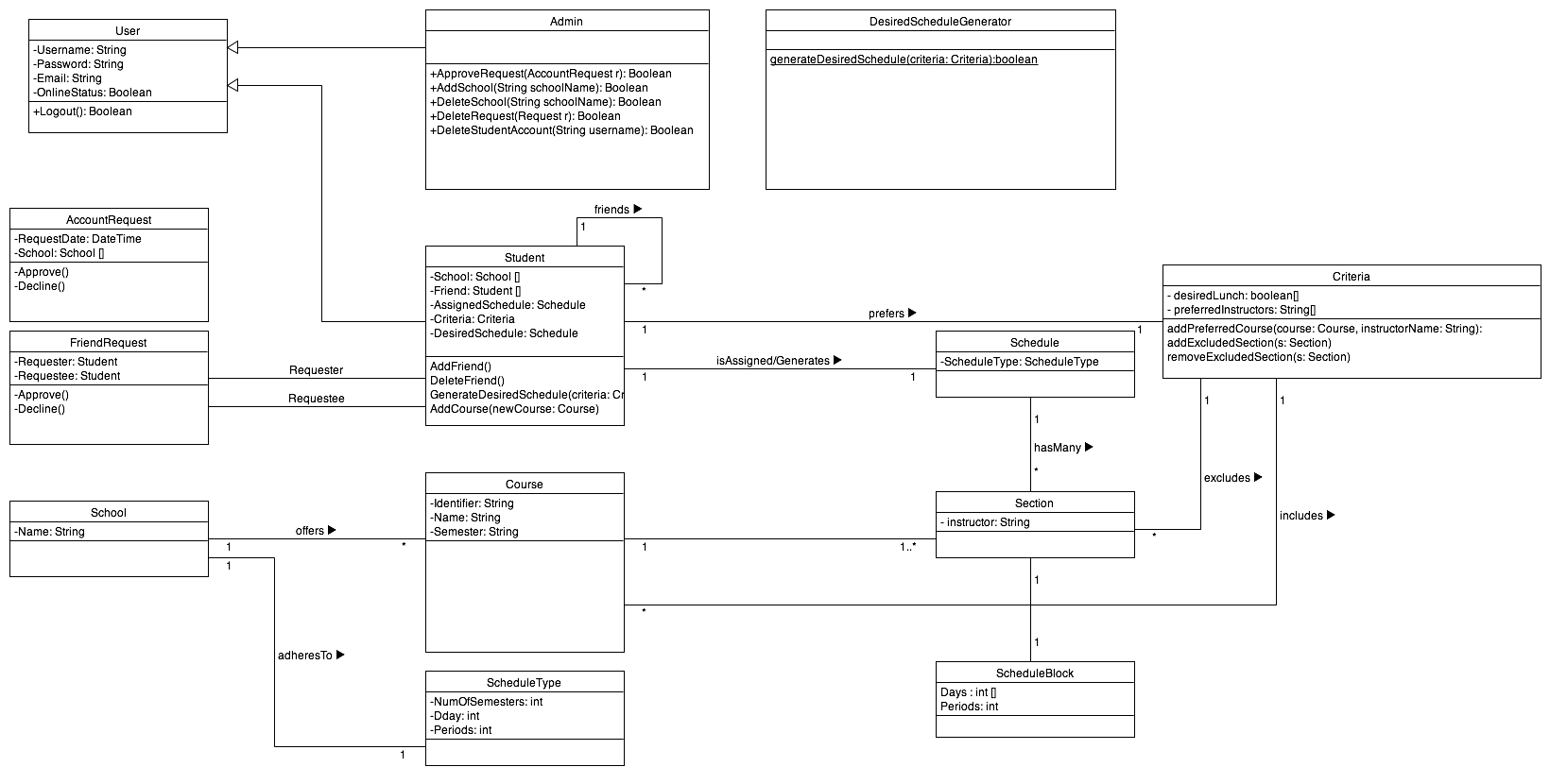
**7. Object Model**

Figure 7.1 Class diagram

**8. Graphical User Interface/System Flow**

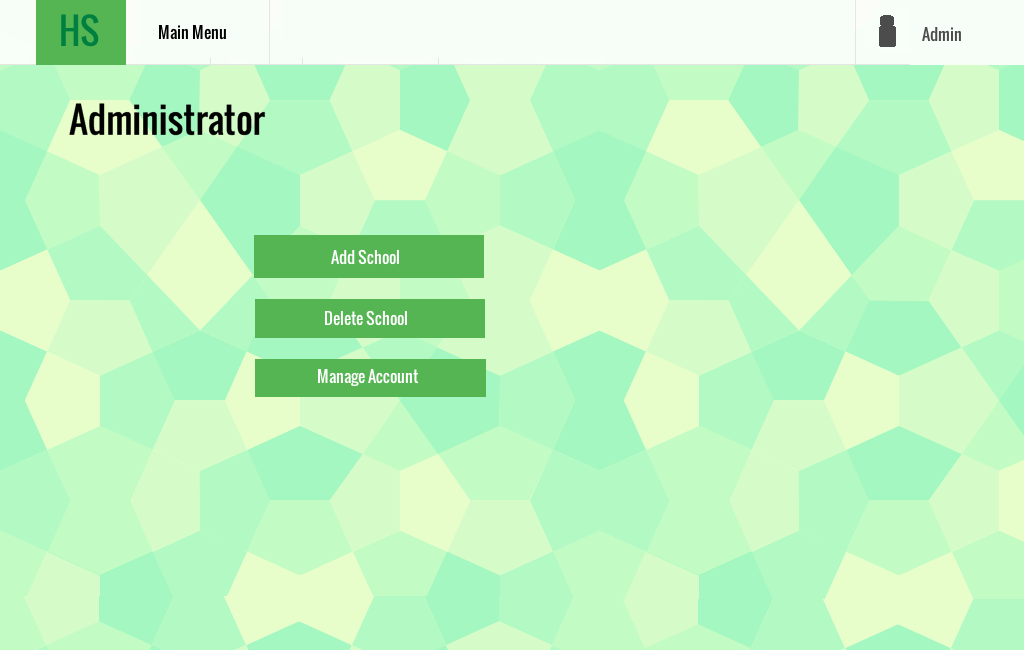


Figure 8.1 . Administrator Menu

The <Add School> button leads to the “Administrator add school” screen (Figure 8.2)

The <Delete School> button leads to the “Delete School” screen (Figure 8.3)

The <Manage Account> button leads to the “Administrator Manage Account” screen (Figure 8.4)

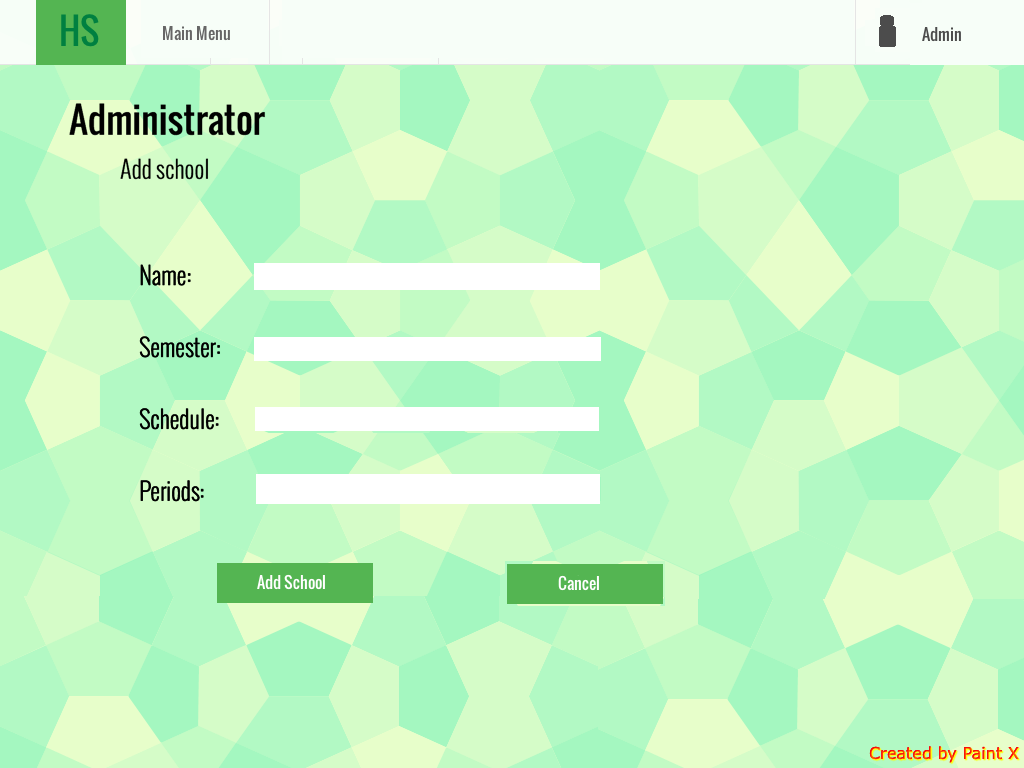


Figure 8.2. Administrator add school screen

The <Add School> button will add a new school to the database.

The <Cancel> button will go back to “Administrator Menu” (Figure 8.1) and no school will be added.

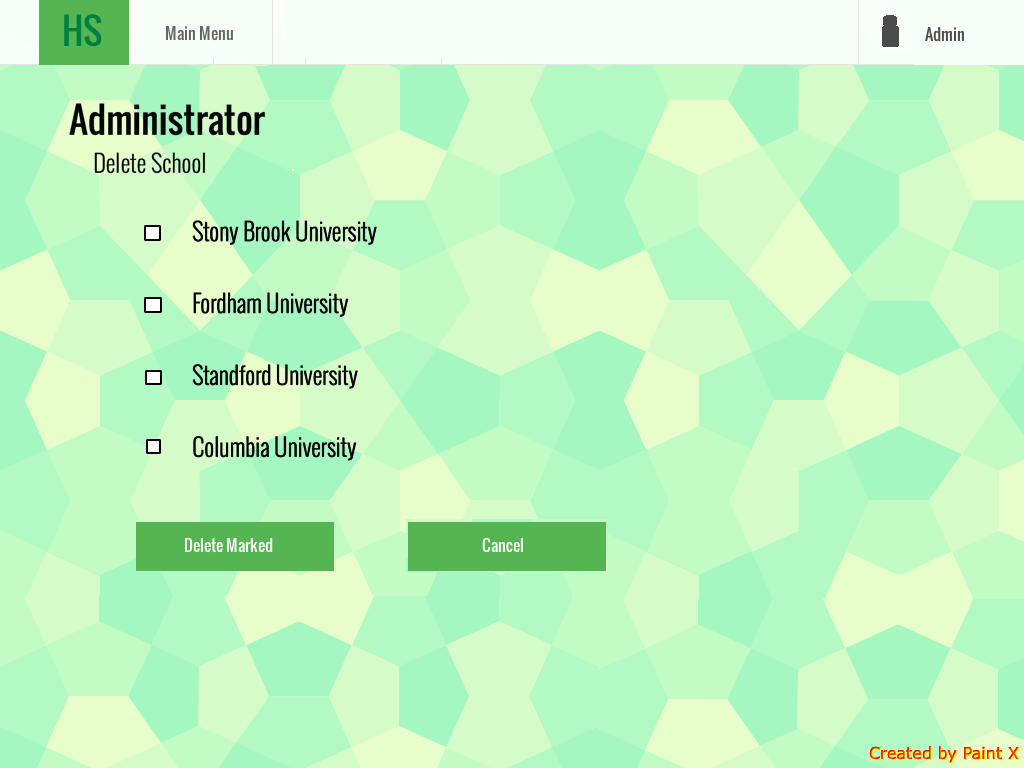


Figure 8.3. Administrator delete school screen

The <Delete Marked> button will delete checked school from database.

The<Cancel> button will go back to “Administrator Menu” screen and no schools will be deleted.

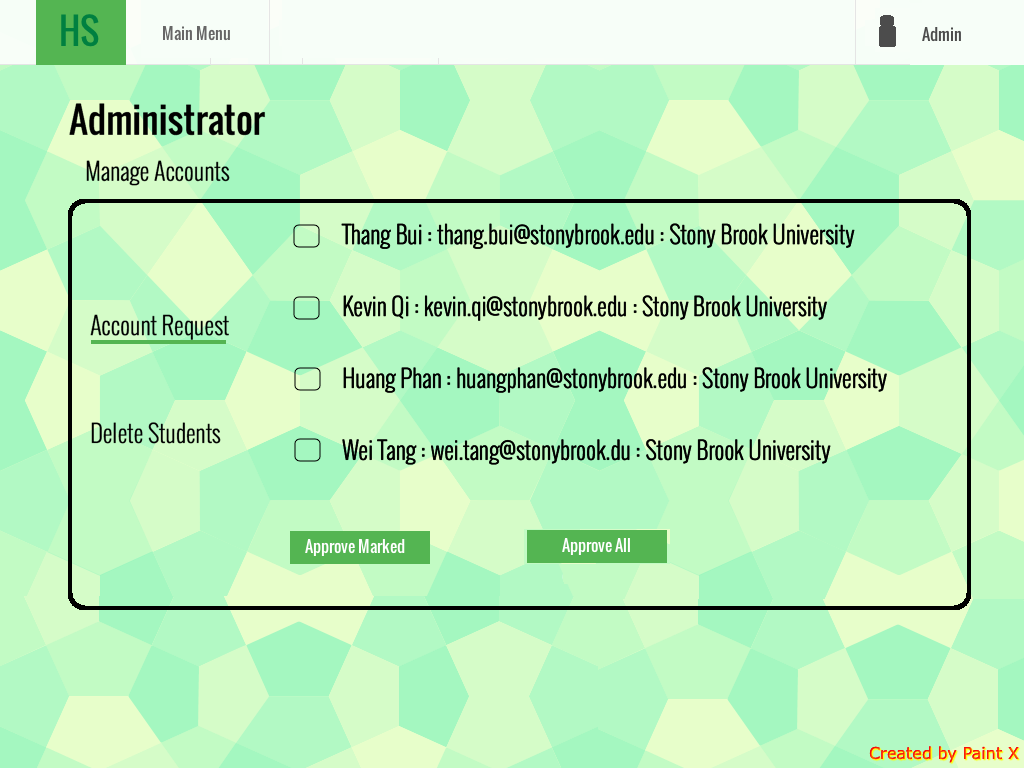


Figure 8.4. Administrator manage student account request screen

The administrator can click on a checkbox to choose which accounts to approve. Once selected, the admin clicks ‘Approve Marked’ to approve selected accounts. The admin can also choose to approve all account requests by clicking ‘Approve All’.

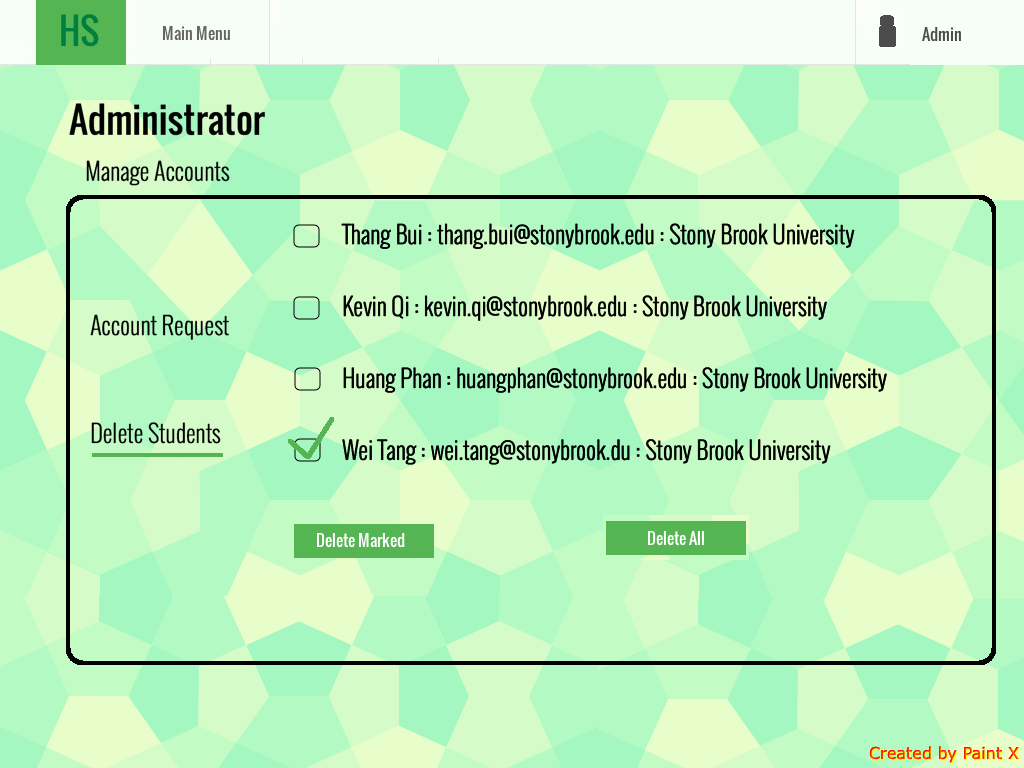


Figure 8.5 . Administrator delete students account page

Similar to approve account requests in figure 8.4.

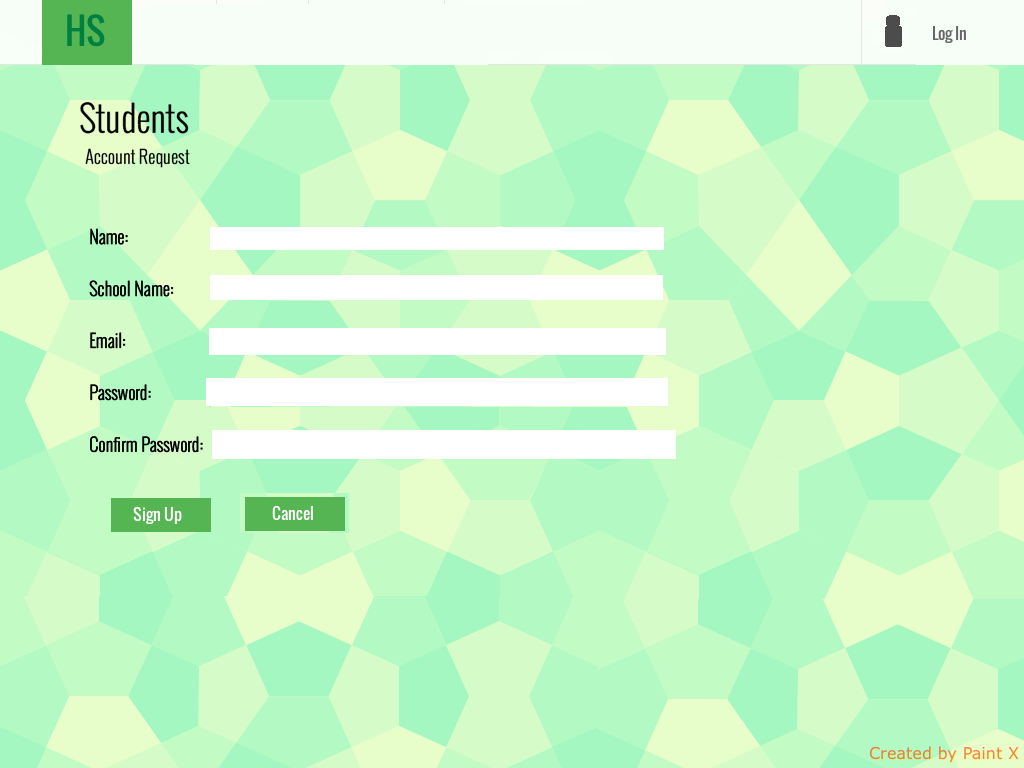


Figure 8.6 . New student requests account

If a student is a new user, he/she can request a new account. The student enters the required information and clicks ‘Sign Up’. Then the administrator will choose whether to approve the account request as seen in figure 8.4.

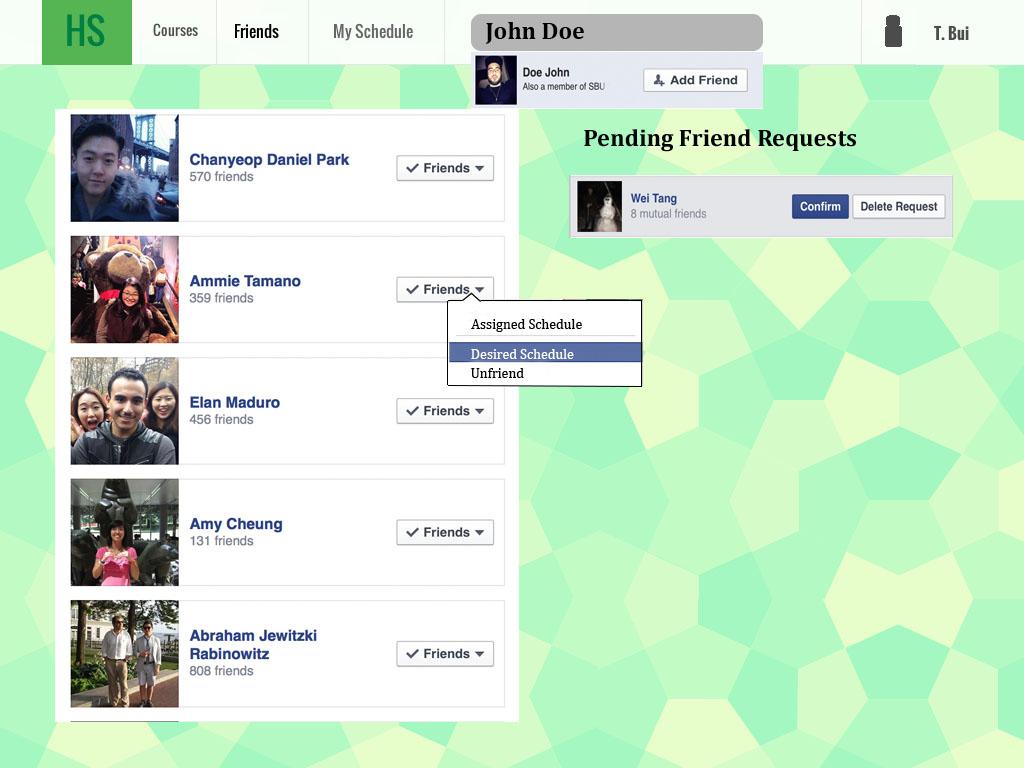


Figure 8.7 View List of Friends

When the user presses the ‘Friends’ heading, a list of the student’s friends will appear. If the student enters a name in the search bar above, then matching students’ profiles will appear, and each profile will have an ‘Add Friend’ button allowing the searcher to add that student as their friend. Clicking on the ‘Friends’ will drop down three choices for the student to choose from. Clicking ‘Desired Schedule’ or ‘Assigned Schedule’ will show the desired schedule (only if the friend already generated one) (Figure 8.8) and assigned schedule of the selected friend respectively. The current student can also choose to unfriend someone. All pending friend requests will appear on the right side, student can accept or decline the requests by clicking on the corresponding buttons.

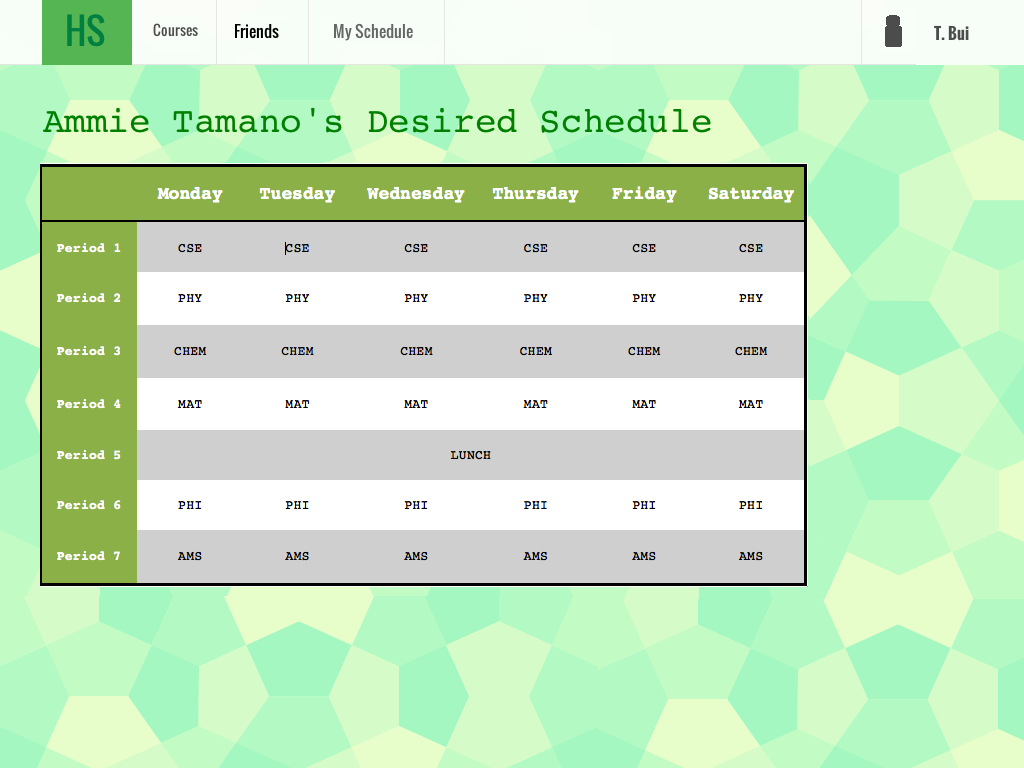
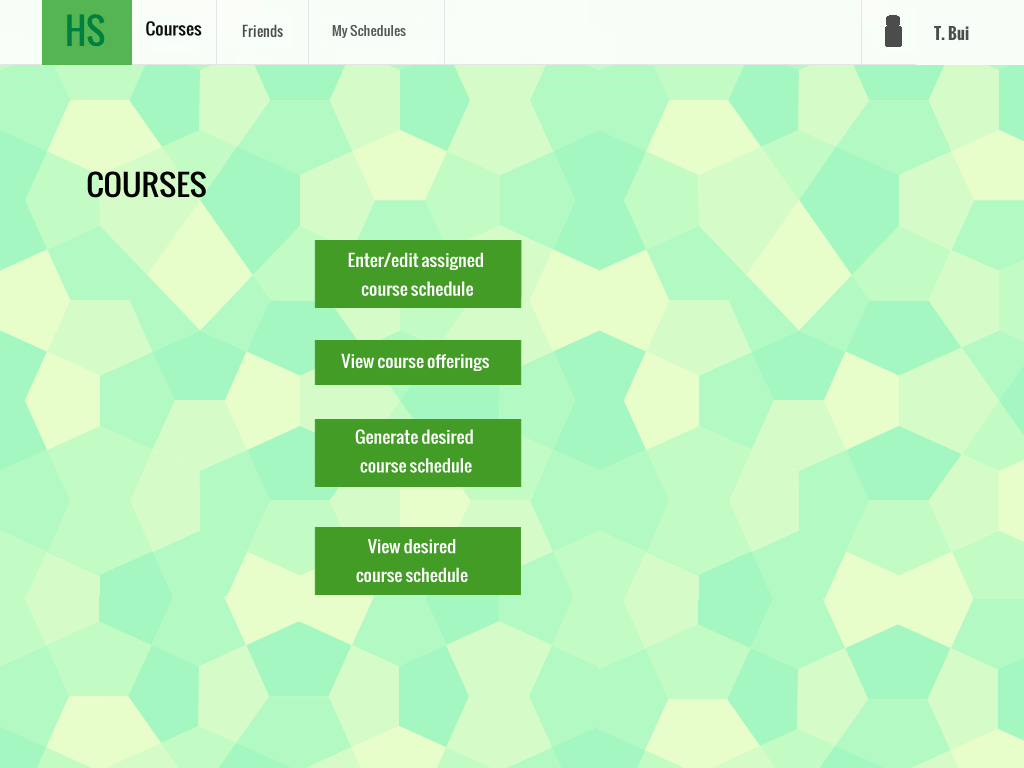


Figure 8.8: Friend’s Schedule

Figure 8.9: Course page’s menu

This is the user interface when entering the course. The student has 4 options: ‘Enter/edit assigned course schedule’; ‘View course offering; generate desired course schedule’; and ‘View desired course schedule’ (the one generated and saved before)

Below is when the student clicks on ‘Enter/edit assigned course schedule’ button:

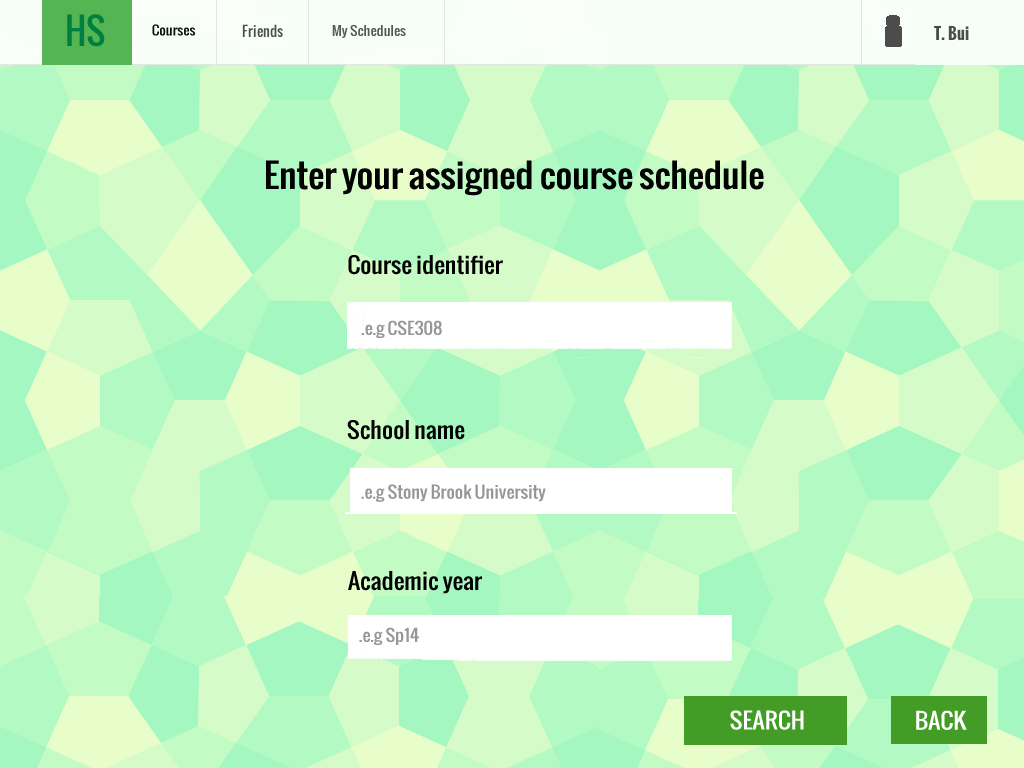


Figure 8.10: Enter assigned schedule initial interface

The student will enter the course identifier, school name, and the academic year, then click the search button to search for that course. He can also click back to go back previous page.

There are several possible flows from this point onward:

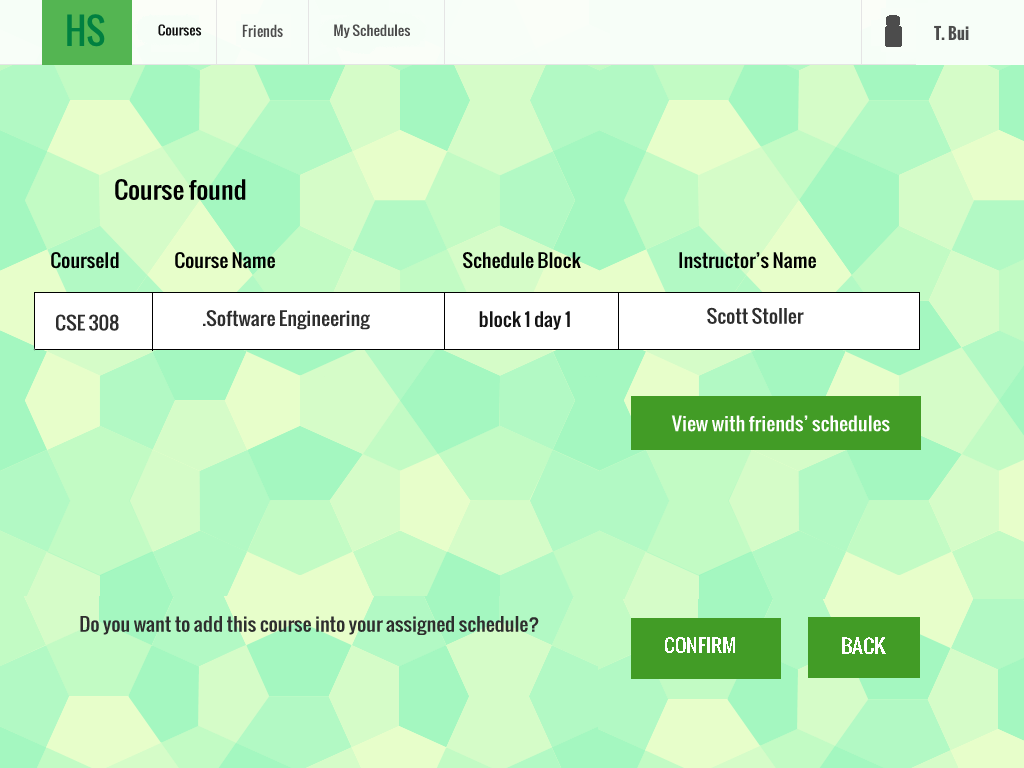


Figure 8.11: Course found case

Flow1: The course is already entered into the database by others:

The system says ‘Course found’ and displays the course. The student is then prompted to confirm the course or go back.

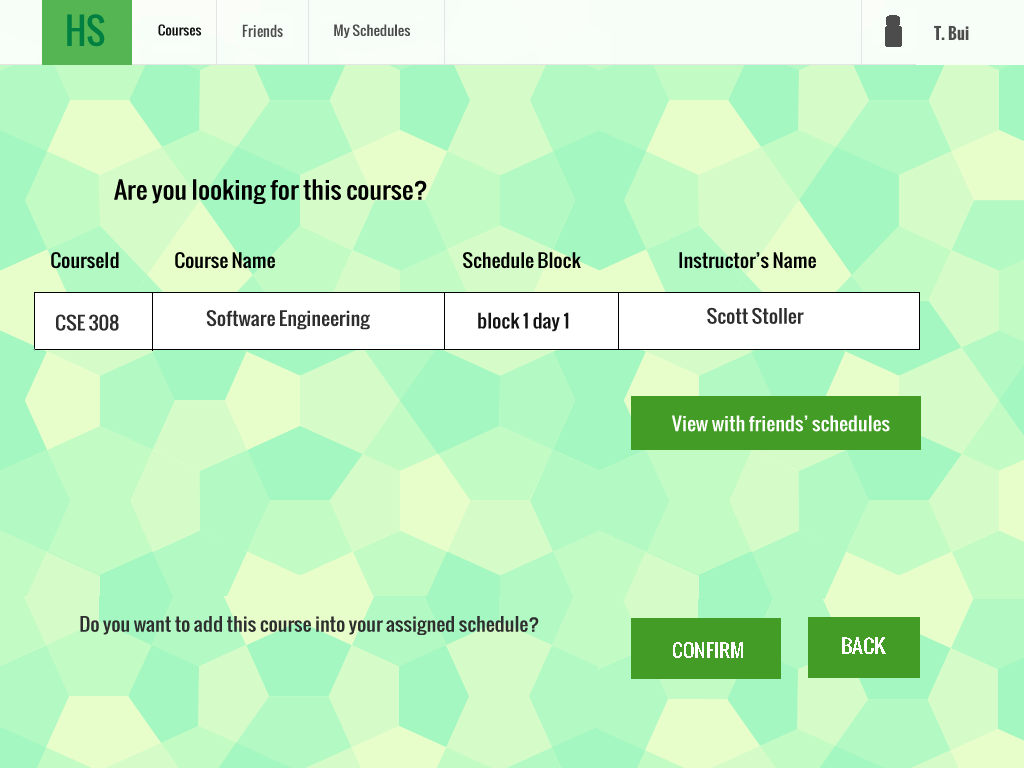


Figure 8.12: Course found from previous year case

Flow 2: The course isn’t found for this academic year, but found from the previous academic year.

The system asks the student to confirm that this is the course he is looking for. If the student hovers his mouse over the course/instructor name, edit buttons will appear so that the student can edit the course/instructor name.

After that, the student is prompted to confirm to add that course into his assigned schedule, or he can go back.

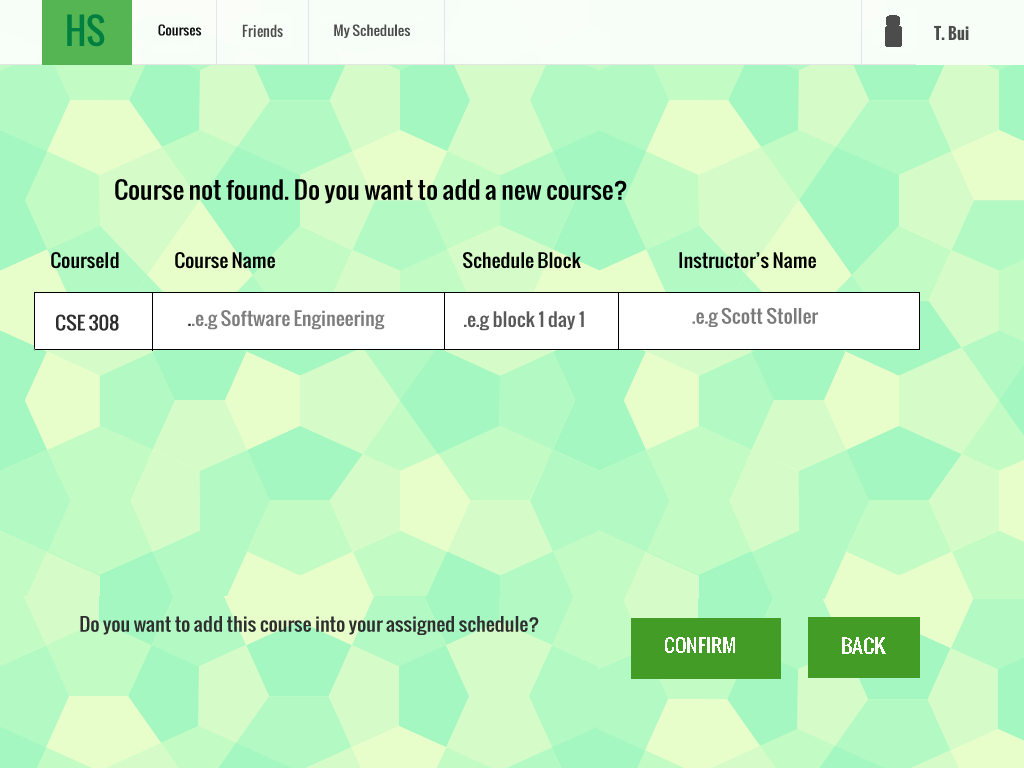


Figure 8.13: Course not found in database case

Flow 3: Course not found in database.

The student can then enter the information for a completely new course. Then he is prompted to confirm to add that course into his assigned schedule, or go back.

Notice that in the case of flow 1 and 2, we have an option that the student can view the course with his friends inside that class. Below is the scenario when the “view with friend’s schedules” button is pressed:

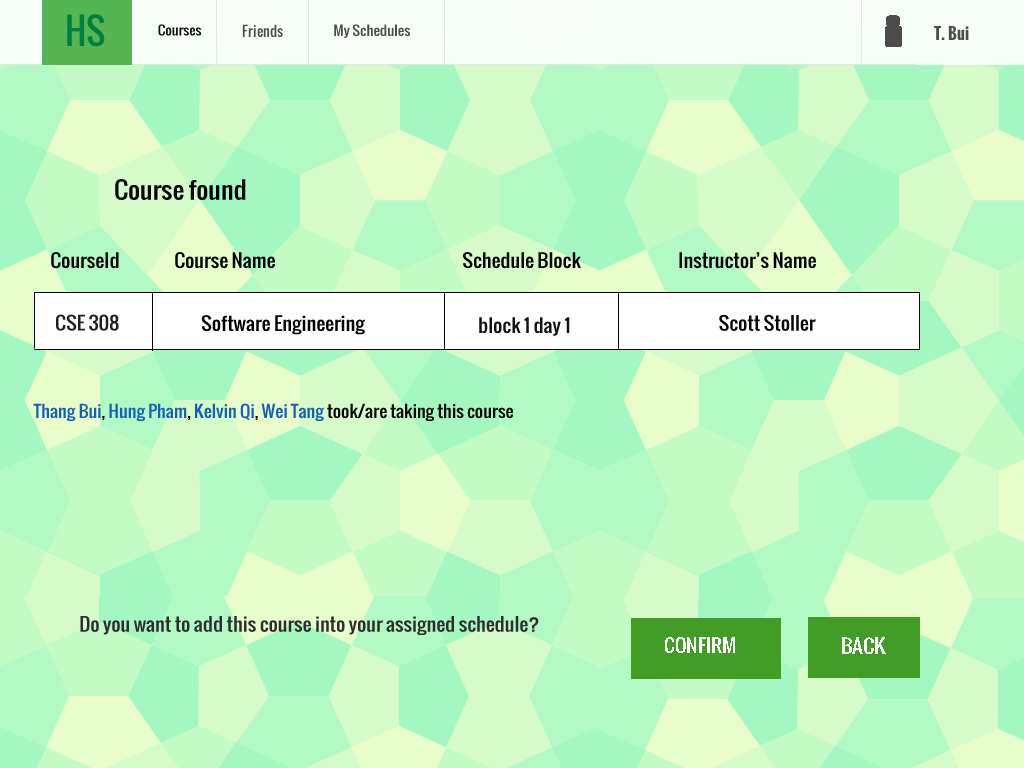


Figure 8.14: View with friend’s schedule

The list of friends who are taking that course will appear.

View course offering: From the initial course page, if the student clicks on “View course offering”:

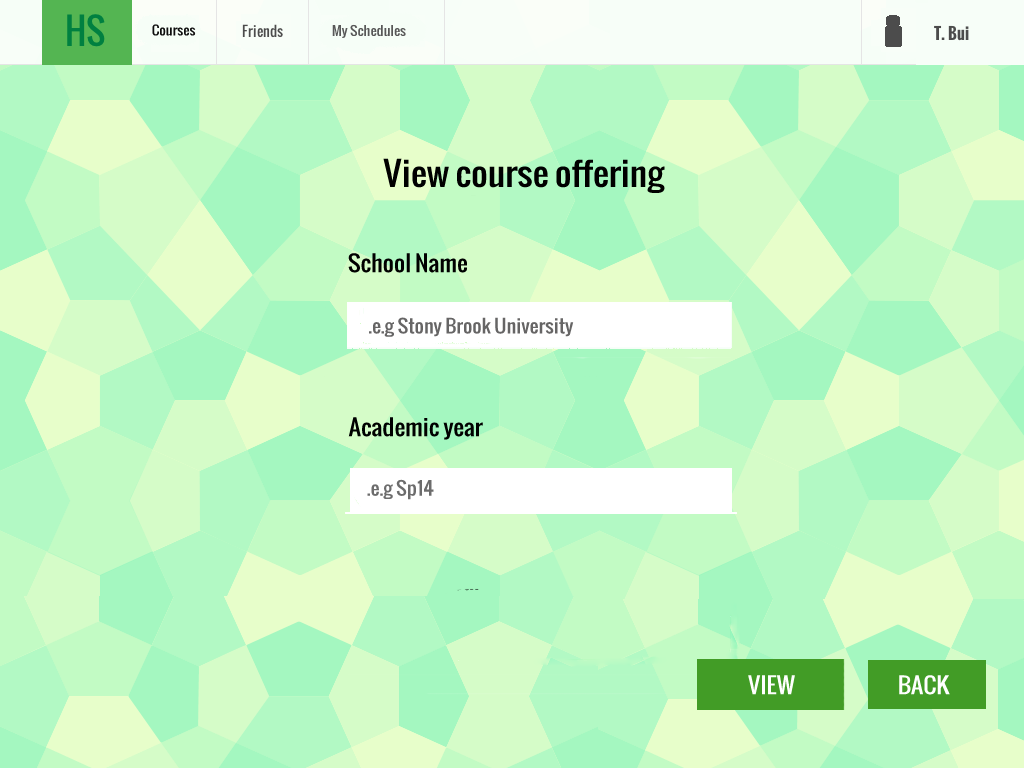


Figure 8.15: View course offering input page

The student is asked to enter the school and the academic year, then he can click ‘View’ button to see the result (Figure 8.16).

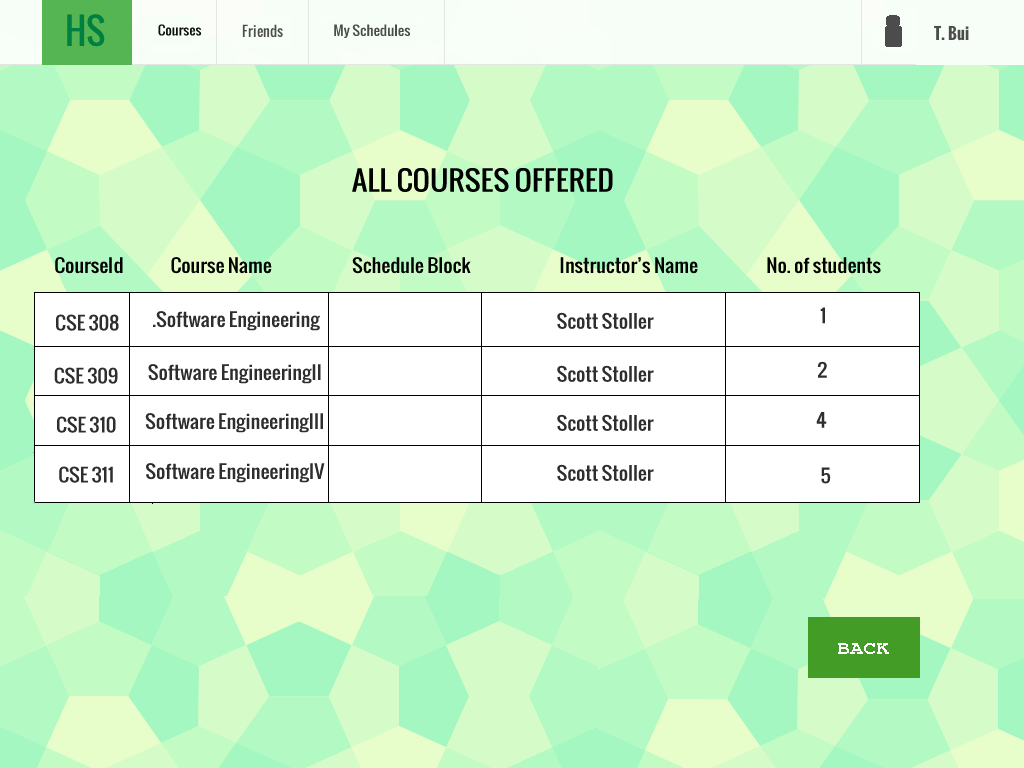


Figure 8.16: View course offering.

When the student clicks ‘View, the system lists all the course offered by that school in that academic year, along with the number of students taking that course.

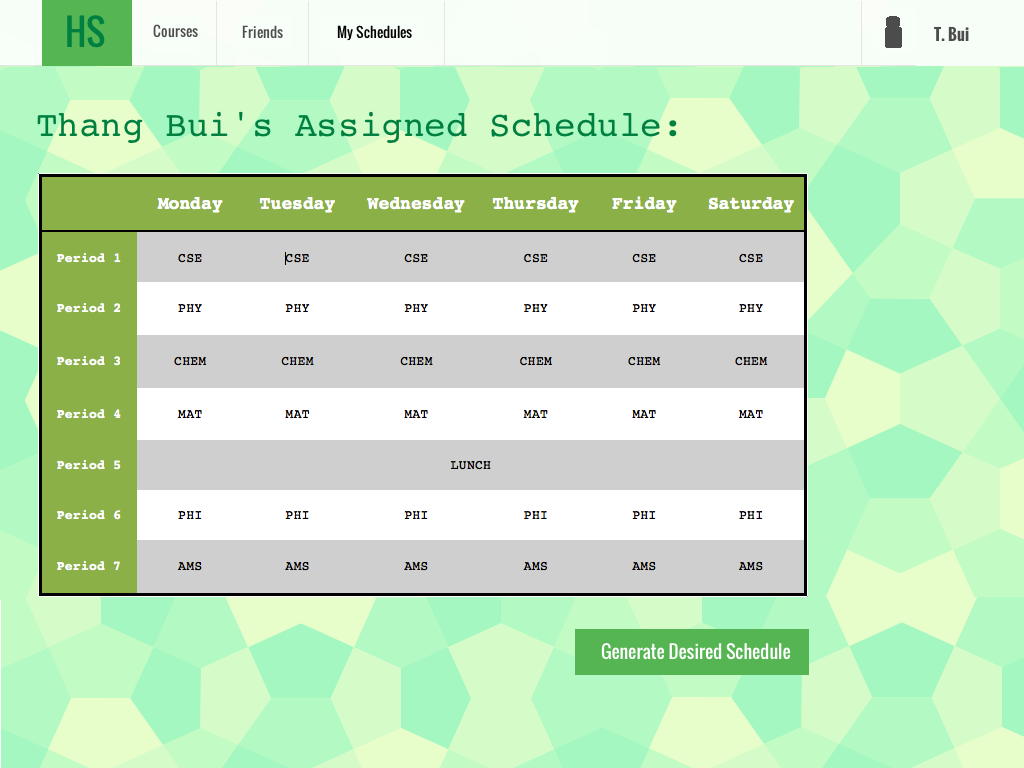


Figure 8.17 Student’s assigned schedule

When the student clicks ‘My Schedules’ tab on top, one or two schedules will appear on the page. Only the assigned schedule will appear if the student has not generated a desired schedule yet. The student can generate desire schedule by clicking on ‘Generate Desired Schedule’ button below the assigned schedule (If the student has not generated any desired schedule before) . It will redirect to new page for entering the desired schedule criteria as shown in figure 8.18

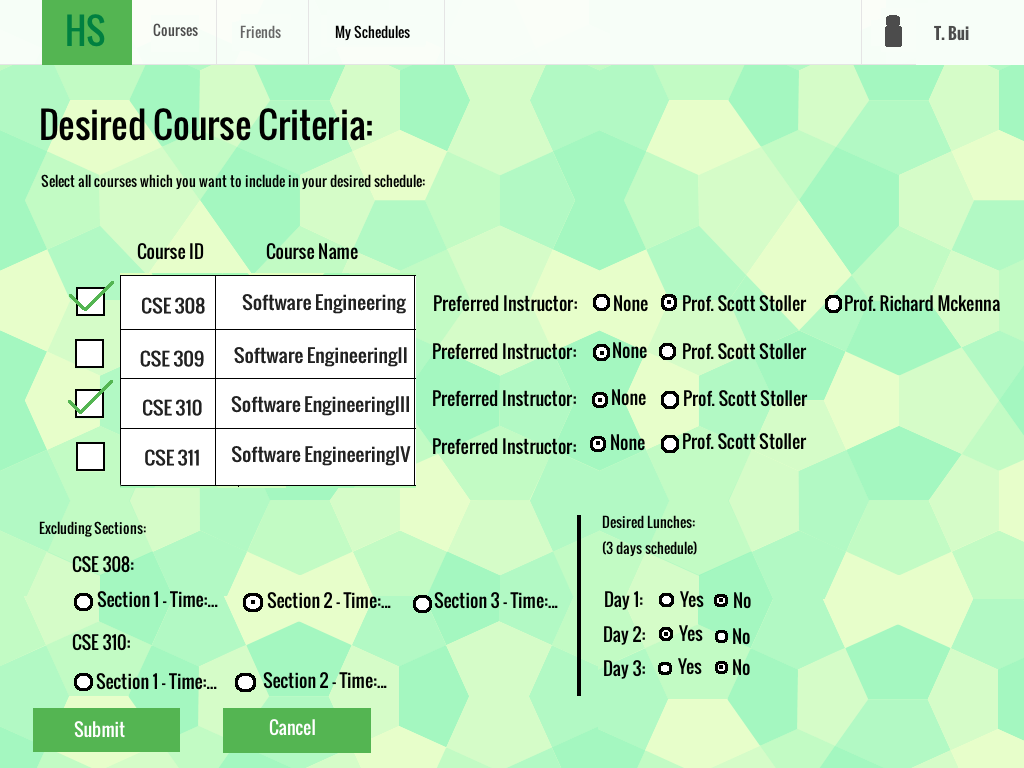
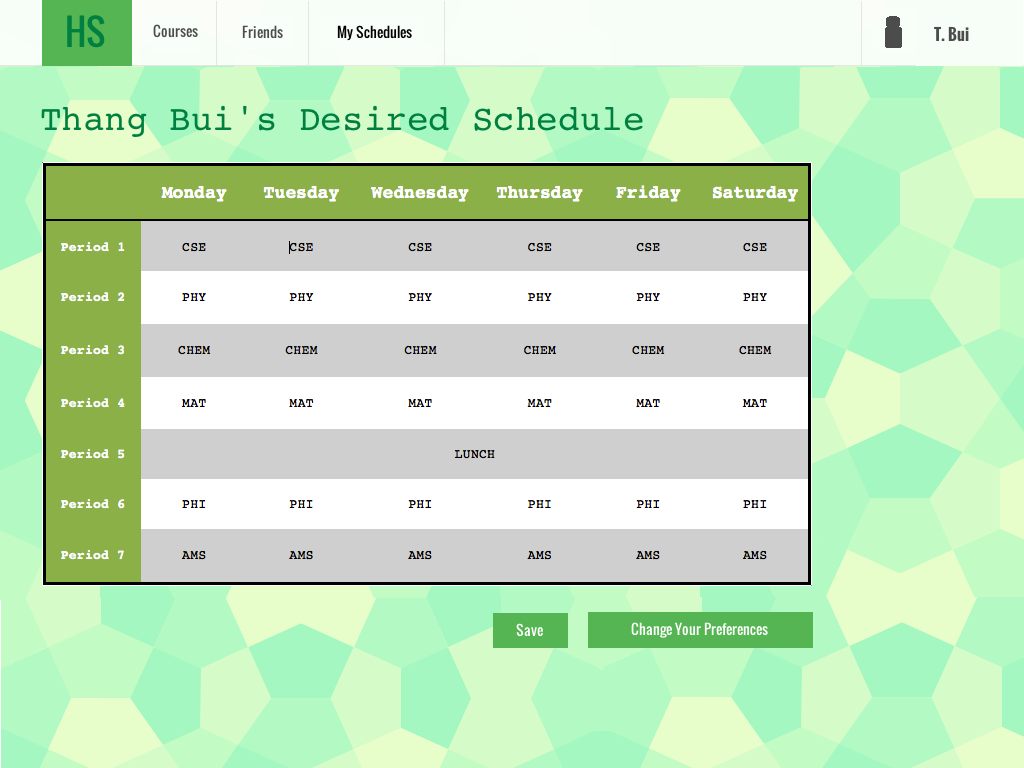


Figure 8.18 Student’s desired schedule criteria page

The student will input the desired schedule criteria from this page. List of all courses offered from the current semester will be shown, and the student choose the courses to include in the desired schedule, each optionally with a set of sections to exclude, and optionally with a preferred instructor. The student can also decide whether lunch is desired for each day in the schedule. Then student click ‘Submit’ to generate a new desired schedule (Figure 8.19)

Figure 8.19: Student’s desired schedule

After entering and submitting all the criteria, a new desired schedule will be generated as shown in figure 8.18 .Beneath the desired schedule is the ‘Change Your Preferences’ button that allows the student to change their criteria for a desired schedule to generate another desired schedule and ‘Save’ to save this schedule as the desired schedule.

**9. Dynamic Model:**

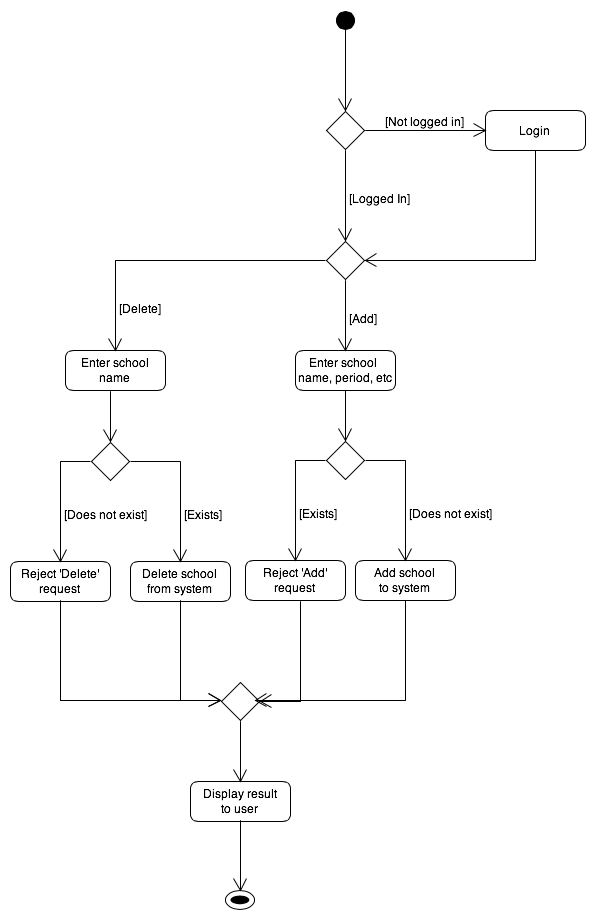


Figure 9.1: Activity diagram for ‘Add or delete school’

Add or delete school

First the admin must be logged in to proceed. Once logged in, she chooses to add or delete a school. To delete a school, she only needs to enter the school name. To add a school, she needs to enter the school name, number of semesters in an academic year, the number of days in the schedule, and the number of periods in each day. If the school exists in the system, then the add/delete request is executed. If not, then the request is rejected. Finally, a notification of success or failure is displayed on the screen.

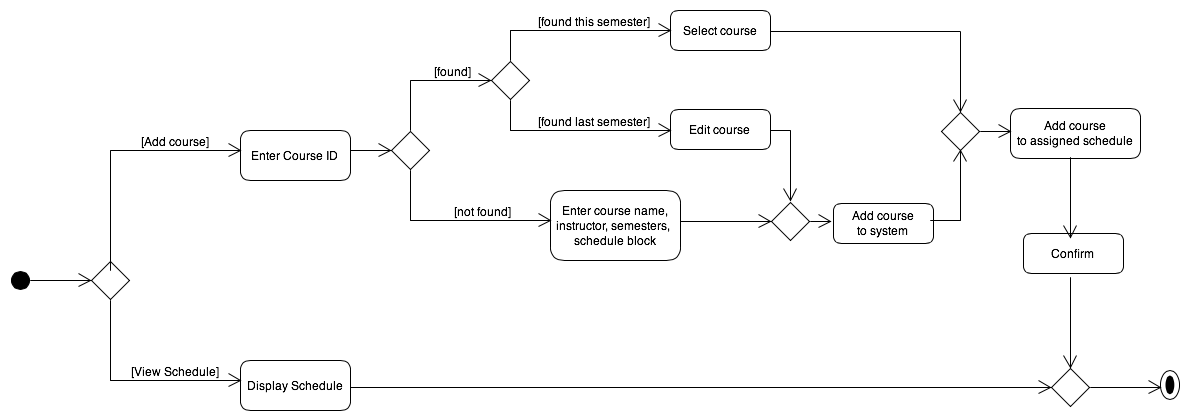


Figure 9.2: Activity diagram for ‘Enter and view assigned schedule’

Enter and view assigned schedule

First the student must be logged in. That step can be seen in diagram 3.1 above. The student can then choose to add an assigned course or view his assigned course schedule. To add a course, she first enters the course identifier. The course either is found in the system or not found.

If found, the course is either from the current semester or last semester. If from last semester, the student can edit the course if needed. If this semester, the student only needs to select it.

If not found, the student enters the relevant course info.

After entering the course info, she proceeds to add the course to the system.

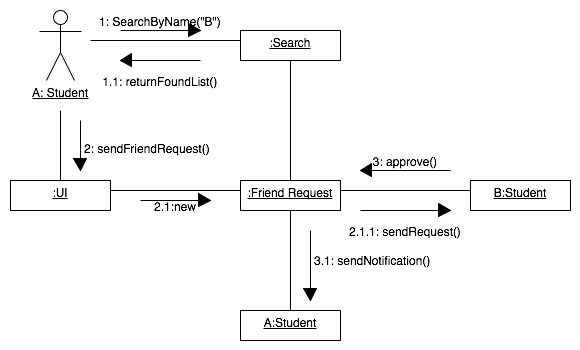


Figure 9.3: Communication diagram for adding friend use case

Add friend

The diagram describes the main flow of adding a new friend. A student can search friend by exact name and the list of found friends will be shown, the student can send friend request by clicking on the friend’s name from the list. Then the requestee can approve the request and a notification will be sent to the requester.

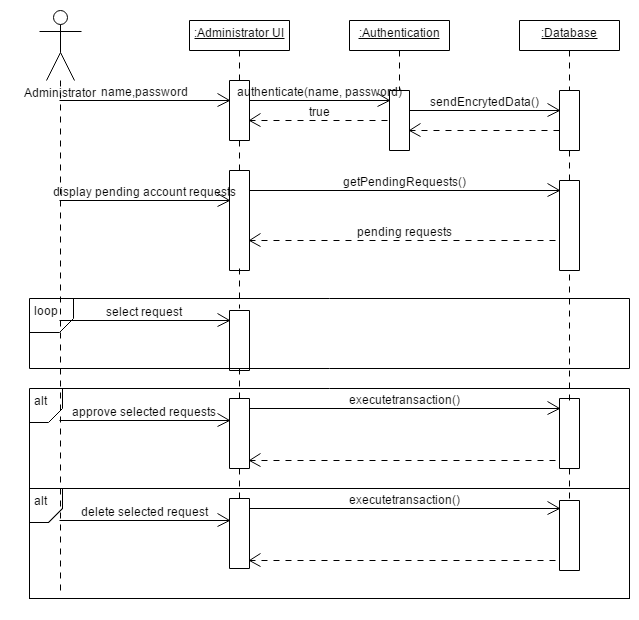
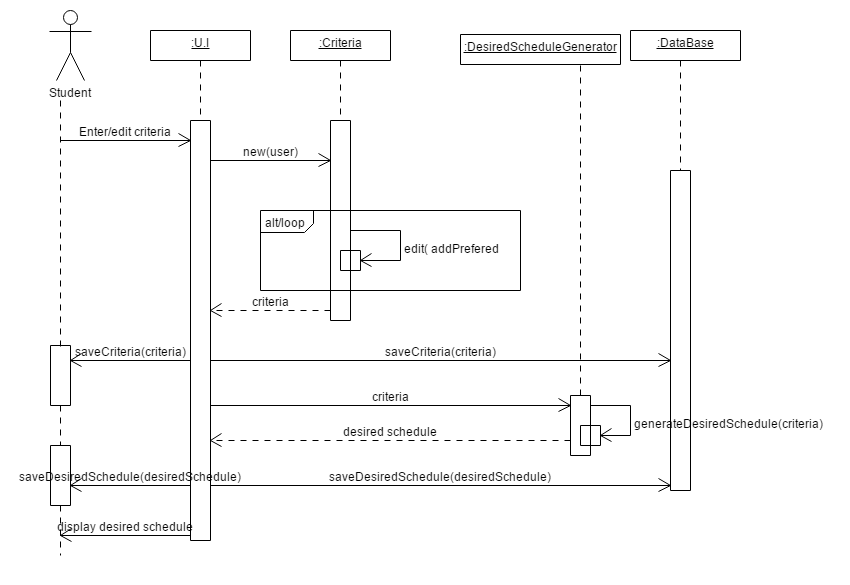


Figure 9.4: Sequence diagram for student account request

Manager student account requests

* First the administrator needs to log in. The administrator will enter the username and password, go to the Authentication object to get the data encrypted and then send to database.
* Once logged in, in order to manage student account requests the admin have to choose to display all the pending requests.
* Then the admin can manually choose some of the requests to approve or delete. The action of approve/delete is optional. The admin has the option to approve all requests, which is simply choose all requests then approve.

Figure 9.5: Sequence diagram for ‘Generate desire course schedule’

Generate desire course schedule

* Once the student presses the button “Generate Desired Schedule”, he will be taken to the Edit Criteria page.
* The information entered by the student will be used to create a new Criteria object ( create the new object, then add the needed attributes into it) associated with a student. We do it rather than edit the existing criteria since the attributes after change can vary greatly. When student presses submit, the criteria got saved into the database and the current student object, the old criteria in the database get deleted.
* Once the criteria is made, the student is taken to the display desired schedule page. The U.I uses the DesiredScheduleGenerator helper class’ static generateDesiredSchedule() method, which takes a Criteria object as a parameter and interacts with the database to generate the student’s desired schedule. Upon success, the system will return a Schedule object to the U.I which will then got saved into database and student object, and displayed in the U.I

**10. Languages, Technologies, and Tools**

UML Diagram Tool : UMLet.

Our team decided to use the MEAN development stack for our website. MEAN is made up of: MangoDB, ExpressJS, AngularJS, and NodeJS. We will not be using a relational database for this project since MangoDB is an object-oriented database. This is suited for this project since the description wants the system to store data in objects.

Additional tools we will be using are: Mongoose(middleware between our database MangoDB and our server NodeJS) and Twitter Bootstrap(GUI).

We decided to use the MEAN framework because we wanted to learn how to use these new languages and we read that MEAN is similar to the LAMP-based framework.