
Requirements Analysis for Scheduled *G08*

Charles Lewis, Quintin Hill, Diego Segundo, Austin Loza, Nick Vitale

---

Project Description

The Scheduled application will take a user's input of classes of interest and output the most optimal schedule(s) based on the ratings of professors from www.ratemyprofessors.com

---

Vision

Problem statement

Determining an optimal schedule is difficult and time consuming, and most students do not take the time to check the ratings of professors. This ultimately results in having poorly selected schedules and undesired professors. This causes students to become less interested in their courses. Students also become stressed without an easy solution to this issue.

Key high-level goals and user goals

- Determine optimal algorithm for selecting the best classes
- User is able to quickly determine which schedule works best for them, based solely on their input of classes of interest

System features

The application allows the user to input:

- A desired list of classes they would like to take
- A desired number of classes they would like to take
- Classes of highest priority that *will* be included in the schedule

The application returns the most optimal schedule based on the ratings of professors from www.ratemyprofessors.com

Other requirements and constraints

- Determine how schedules with conflicting sections will be optimized

Features that will be included with additional time available:

- Inform the user of the probability they will successfully get into a class
- Real time automatic update of class status, open or full

Use Cases

User inputs desired classes

- Primary actor: User
  - The user selects their school from the first drop-down menu.
  - The user selects a subject area from the second drop-down menu.
  - The user selects a class from the third drop-down menu.
  - The user is presented with a confirmation box to add the class to their list of desired classes. An "Add" button, a "Priority" check box, and an "X" to cancel confirmation are included within this confirmation box.
  - Once the user successfully adds one class, the second and third drop-down menus will be reset. The added class will be shown just below the menus in a table displaying the user's previously added classes

and each class' priority status. The user will have the ability to remove a class or edit a class' priority within this table. The user will repeat the process by selecting a subject area from the 2nd drop-down menu.
        - The user must add at least 3 classes before they can proceed.
    - Main success scenario: The users classes of interest (3+) are sent to system
    - Extension: Table of Classes of Interest
        - A table of the classes of interest the user has added will appear below the three drop-down menus. The user will have the ability to view previously added classes, delete previously added classes, and edit the priority status of previously added classes.
    - Extension: Class Prioritization
        - A "Priority" check box will be included in the confirmation box to add a class to the list of desired classes. If a user clicks on the check box, the class will be given priority status and *will* appear on all of the returned optimized schedules. If the user does not click on the check box, the class will not be given priority status.
    - Extension: Error Message: You must add at least 3 classes
        - The user will receive an error message stating "You must add at least 3 classes." if the user attempts to proceed with less than 3 classes.
    - Extension: Error Message: You cannot add more than 10 classes
        - The user will receive an error message stating "You cannot add more than 10 clases" if the user attempts to add an 11th class.

User inputs number of classes they will take

- Primary actor: User
    - The user enters a number greater than or equal to 3 in a text box that appears just below the table of previously added classes of interest.
- Main success scenario: Number of classes the user will take is sent to system
- Extension: Error Message: You must input a value of 3 or greater
    - The user will receive an error message stating "You must input a value of 3 or greater" if the user attempts to enter a value less than 3.

System prints most optimal schedule

- Primary actor: System
    - Once the user has added 3 or more classes to their classes of interest and specified how many classes they will take, an "Optimize" button will appear. If the user clicks on this button, the most optimal schedule will appear. The schedule will include details of every class.
    - If the user is not satisfied with this schedule, the user can click on a "Next" button just below the schedule which will generate the next most optimal schedule for viewing.
    - The user can only click on the "Next" button twice. The user can only view the three most optimal schedules of their classes of interest.
- Main success scenario: Most optimal schedule is returned
- Extension: Next Most Optimal Schedule
    - If the user is not satisfied with the schedule, the user can click on a "Next" button just below the schedule which will generate the next most optimal schedule for viewing.

        - The user can only click on the “Next” button twice. The user can only view the three most optimal schedules of their classes of interest.

Schedule

1st iteration

- We will ensure all use cases of the application are functional with terminal.
    - Time allotted: 3 weeks

2nd iteration

- We will convert the application to GUI or Android.
    - Time allotted: 2 weeks

3rd iteration

- We will include extra functionality.
    - Time allotted: 1 week

Glossary

Priority

Must be included, must not be left out

Iteration

A version or prototype of the final project

Optimal/Ideal

Highest rating average of schedule’s professors

Functionality

Other possible use cases that this application could include

Terminal

User interface on a computer for entering and displaying data (ie: unix)

(Android) App

User interface on a phone for entering and displaying data

Class and Method Glossary

Scheduled Terminal: Class containing main method of program

- Main(): method used to run program

Schedule: Class containing data for schedule

- There are five private variable Course objects that are the five possible classes in a schedule
- courseCount is an integer holding the number of courses in the schedule
- totalRating is an integer holding the total rating of the schedule that is calculated by adding all ratings of each class in schedule
- totalRating(): method that returns totalRating
- add(Course): method that adds Course object to schedule
- toString(): method that converts Schedule to string

Menu: Class logging user activity, handling optimization, displaying schedule
- courses is a List containing the courses the user has added to be optimized
- desired is an integer holding how many courses the user wants to take
- optimizations is an integer holding how many schedules have been outputted
- schedules is an array of Schedule objects used to display schedules to user
- schedulesRequested is an integer that tracks schedules displayed
- priorities is an integer holding value of courses prioritized
- Menu(): constructor
- addCourse(): method that adds course to courses List
- sortedAdd(): method used to add course to courses List in a sorted pattern
- deleteCourse(): method that deletes course form added courses list
- setDesired(): method that sets desired value
- changePriority(): method that changes priority of an added course
- optimize(): method that optimizes added courses
- printOptimized() method that prints optimized courses
- optimize3(): method that optimizes 3 courses
- optimize4(): method that optimizes 4 courses
- optimize5(): method that optimizes 5 courses
- sortOptimizedSchedule(): method that sorts optimized schedule
- val3(): method that validates whether 3 courses are compatible
- val4(): method that validates whether 4 courses are compatible
- val5(): method that validates whether 5 courses are compatible
- printCourses(): method that prints courses added to courses List

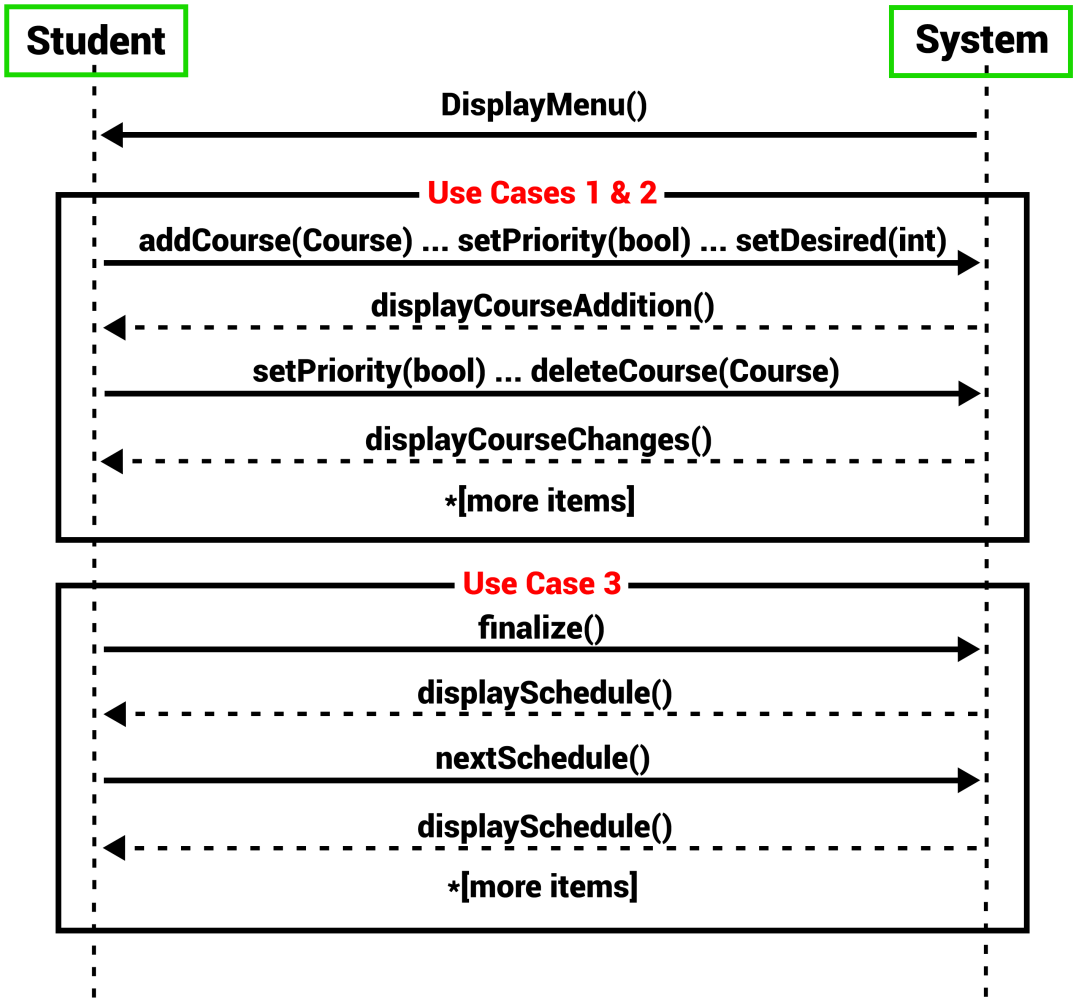
Professor: Class containing data for professor
- last is a String holding the last name of the professor
- other is a String holding the first and middle initial of professor
- rating is a double holding the rating of the professor
- Professor(): constructor
- getLast(): method that returns last
- getOther(): method that returns other
- getRating(): method that returns rating
- toString(): method that converts Professor to string

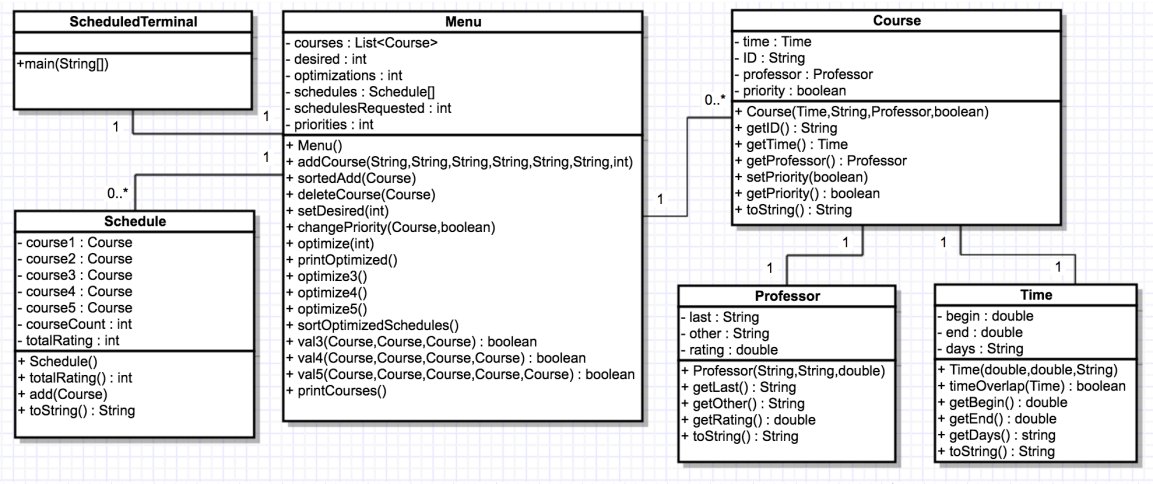
Time: Class containing data for time
- begin is a double holding the beginning time value
- end is a double holding the end time value
- days is a String holding the days of the course
- Time(): constructor
- timeOverlap(): method that checks if time objects overlap
- getBegin(): method that returns begin
- getEnd(): method that returns end
- getDays(): method that returns days
- toString(): method that converts Time to string

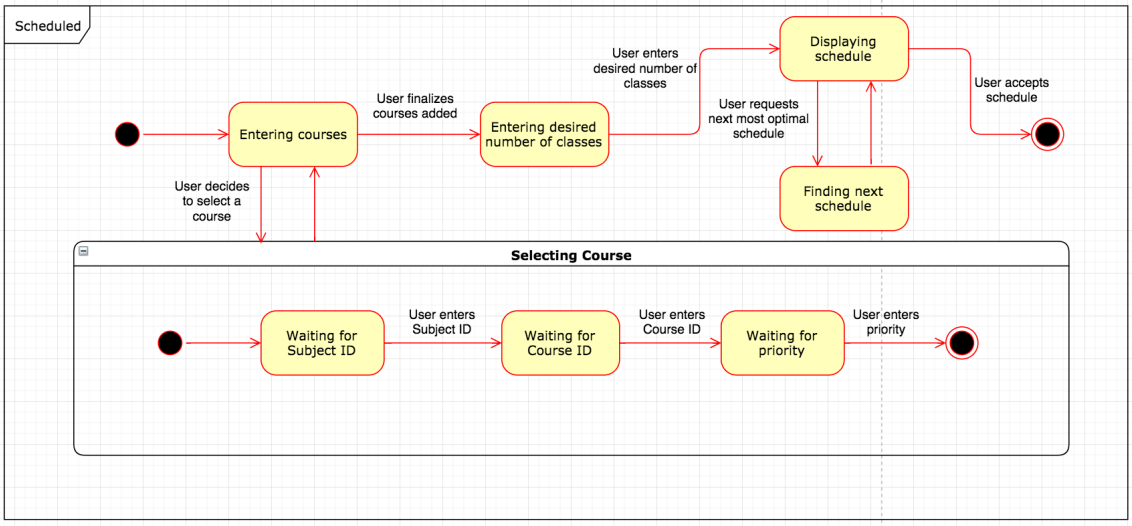
Course: Class containing data for course
- time is a Time object holding the time values of the course
- ID is a String holding the course name
- professor is a Professor object holding the professor values of the course

- priority is a boolean holding the priority value of a course
- Course(): constructor
- getID(): method that returns ID
- getTime(): method that returns Time object
- getProfessor(): method that returns Professor object
- setPriority(): method that sets course priority
- getPriority(): method that returns course priority
- toString(): method that converts Course to string

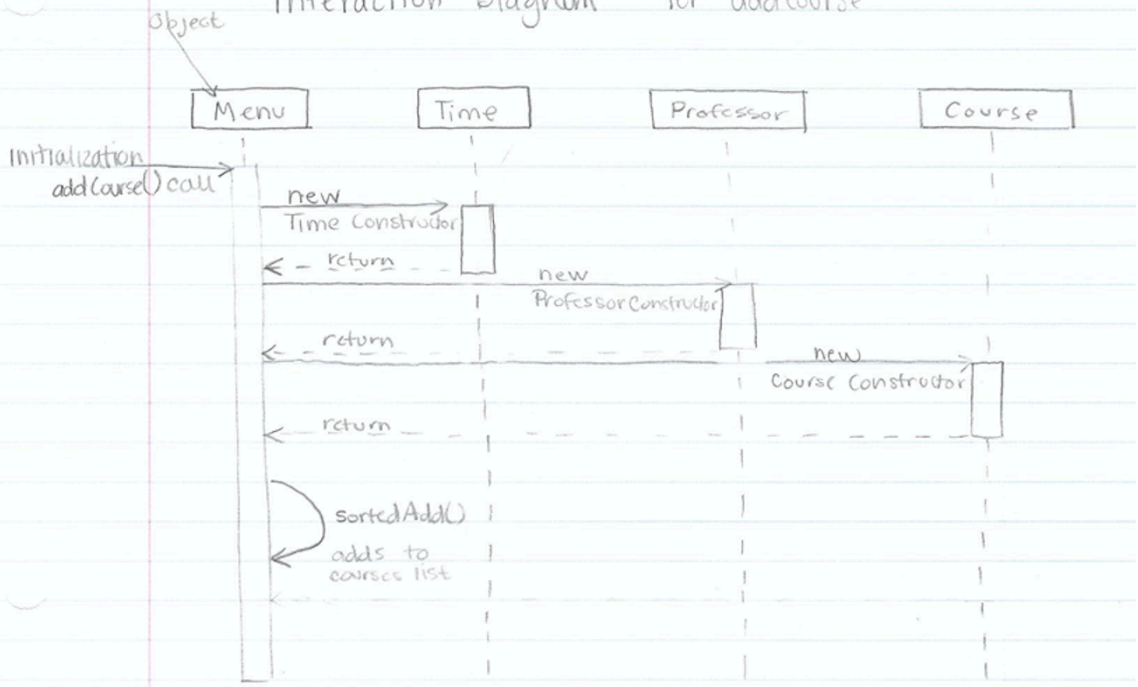
Scheduled G08 System Sequence Diagram: 3-in-1 diagram of existing use cases (5/9/2017)




ScheduledTerminal
+main(String[])
Menu
- courses : List<Course>
- desired : int
- optimizations : int
- schedules : Schedule[]
- schedulesRequested : int
- priorities : int
+ Menu()
+ addCourse(String,String,String,String,String,String,int)
+ sortedAdd(Course)
+ deleteCourse(Course)
+ setDesired(int)
+ changePriority(Course,boolean)
+ optimize(int)
+ printOptimized()
+ optimize3()
+ optimize4()
+ optimize5()
+ sortOptimizedSchedules()
+ val3(Course,Course,Course) : boolean
+ val4(Course,Course,Course,Course) : boolean
+ val5(Course,Course,Course,Course,Course) : boolean
+ printCourses()
Course
- time : Time
- ID : String
- professor : Professor
- priority : boolean
+ Course(Time,String,Professor,boolean)
+ getID() : String
+ getTime() : Time
+ getProfessor() : Professor
+ setPriority(boolean)
+ getPriority() : boolean
+ toString() : String
Schedule
- course1 : Course
- course2 : Course
- course3 : Course
- course4 : Course
- course5 : Course
- courseCount : int
- totalRating : int
+ Schedule()
+ totalRating() : int
+ add(Course)
+ toString() : String
Professor
- last : String
- other : String
- rating : double
+ Professor(String,String,double)
+ getLast() : String
+ getOther() : String
+ getRating() : double
+ toString() : String
Time
- begin : double
- end : double
- days : String
+ Time(double,double,String)
+ timeOverlap(Time) : boolean
+ getBegin() : double
+ getEnd() : double
+ getDays() : string
+ toString() : String
1
0..*


Scheduled
Entering courses
User finalizes courses added
Entering desired number of classes
User enters desired number of classes
Displaying schedule
User requests next most optimal schedule
User accepts schedule
Finding next schedule
User decides to select a course
Selecting Course
Waiting for Subject ID
User enters Subject ID
Waiting for Course ID
User enters Course ID
Waiting for priority
User enters priority

Interaction Diagram for addCourse



1. Menu class initialized
2. addCourse() function called
   addCourse() does the following
   1. creates time object
   2. creates professor object
   3. creates course object with time/professor objects
      (time/professor objects exist as data in course object)
   4. sortedAdd() function called, adds course object to
      the list of added courses (courses they want optimized)
3. addCourse() function complete