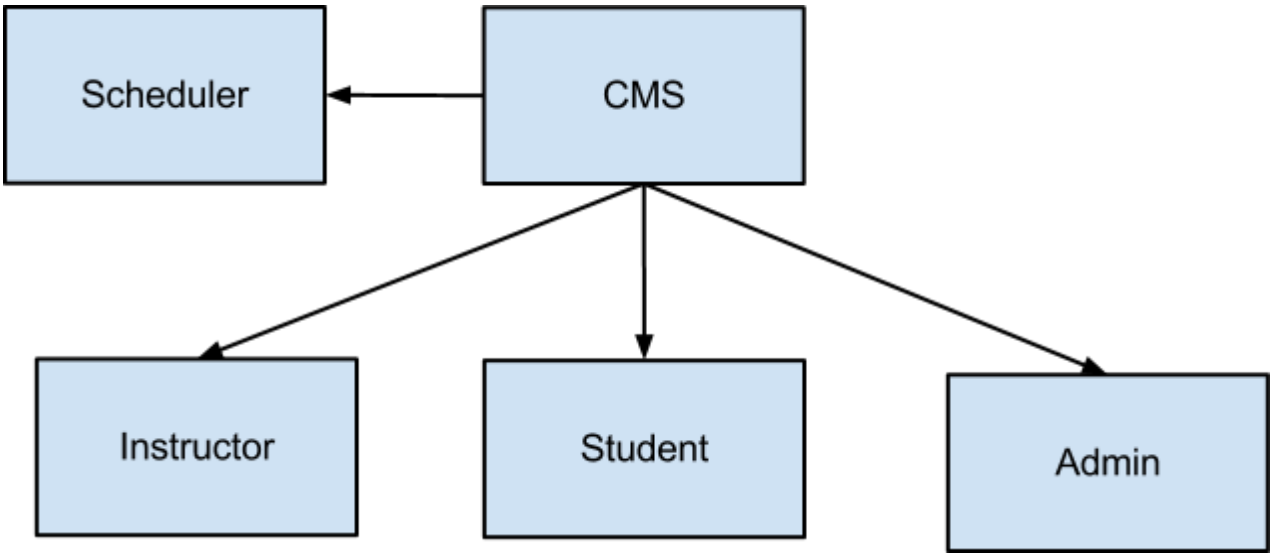
Inception Paragraph:
A small school faculty would like to develop a system that will combine the students' interest in taking electives with the instructors schedule for given semester. The Administrative Assistant must be able to log on/off of the system, add classes, and restrict certain times for those classes (as per the teacher's schedule, some may only be able to teach the first half of the day, for instance). The user must also be able to add certain students, and their top 3 classes (ordered by rank) that they desire to take. Each student would have a default class list of what is required for their grade level (ex: sophomores are required to take Algebra II), but the user would be able to add and subtract classes from that list (some people may have taken Algebra II freshmen year instead, so sophomore year wouldn't contain Algebra II... so the ability to delete it from the list would be required). The program will develop more than one schedule for the user to print/save.

List of Nouns
Faculty <-Actor
System
Student's interest
Electives list
Instructors schedule/ Time
Semester
Administrative Assistant <- Actor
Classes
Students (profile) <- Actor
Class list
Grade level
User
Program
Master Schedule

Conceptional Classes:
Student's interest
Electives list
Instructors schedule/ Time
Semester
Classes
Class list
Grade level
Master Schedule

Domain Model:



List of Use Case Names:

Admin

- 1-Login/Logoff
- 1- AddStudent/RemoveStudent
- 1- AddTopElectives
- 1- RemoveTopElectives
- 2- importUsers
- 2- GenerateSchedules (RISKY!)
- 1-AddInstructor/RemoveInstructor/AddInstructorClassTime
- 1-AddClasses/RemoveClasses
- 1-AddClassesInstructorCanTeach
- 2-displayMasterSchedule
- 2- displayStudentSchedule
- 2-displayTeacherSchedule

Students

- 3-ModifyTopElectives
- 3-displayStudentSchedule

Teachers

- 3-AddClassesInstructorCanTeach
- 3-displayTeacherSchedule

Iteration Plan for Three Iterations :

Iteration 1
Admin
1 - Login/Logoff
1 - AddStudent/RemoveStudent
1 - ModifyTopElectives
1 - AddInstructor/RemoveInstructor/AddInstructorClassTime
1 - AddClasses/RemoveClasses/editClass
1 - AddClassesInstructorCanTeach

Iteration 2
Admin
2 - GenerateSchedules (RISKY!)
2 - displayMasterSchedule
2 - displayStudentSchedule
2 - displayTeacherSchedule
2 - importUsers

Iteration 3
Students
3-ModifyTopElectives
3-displayStudentSchedule

Teachers
3-AddClassesInstructorCanTeach
3-displayTeacherSchedule

Iteration 1

Complete Use Cases:

Name: Login

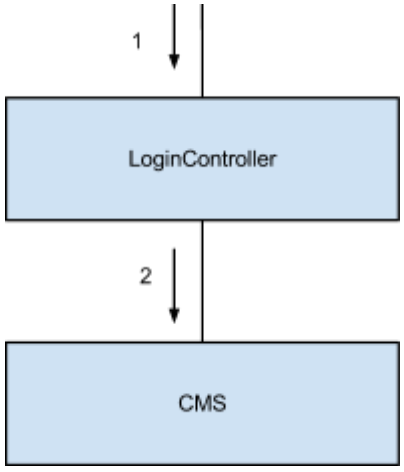
Actor: Students/Teachers/Admin

Precondition: None

MSS: 1. User enters credentials
2. System will give the user access to the correct information

Information Experts: Data knows the users

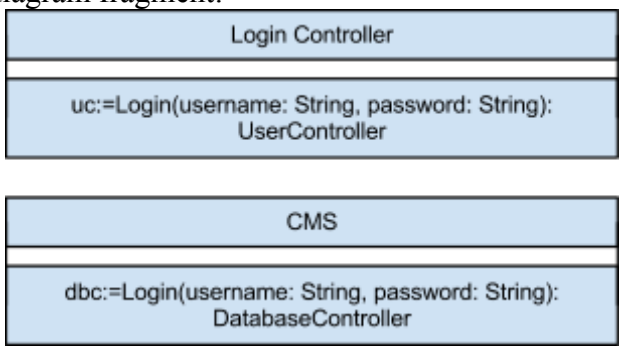
Interaction Diagram fragment:



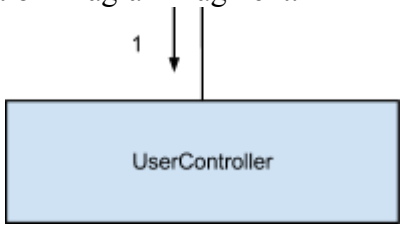
1: uc:=Login(username: String, password: String): UserController

2: dbc:=Login(username: String, password: String): DatabaseController

Class diagram fragment:

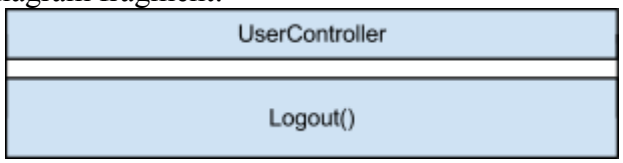


Name: LogOut
Actor: Students/Teachers/Admin
Precondition: None
MSS: 1. Computer will verify the user wants to discard all unsaved changes
2. User logs out
3. Logon screen is shown
Information Experts: NA
Interaction Diagram fragment:



1: LogOut(): void

Class diagram fragment:



Name: AddStudent

Actor: Admin

Precondition: NA

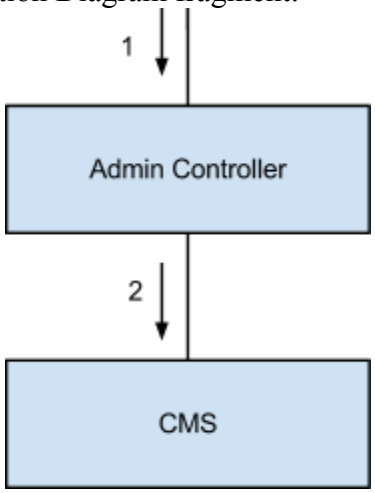
MSS: 1. Admin adds the student

2. New student is created with correct ID and name. All else is empty

Information Experts: Admin Controller knows Students and their password

CMS knows the Students and their password

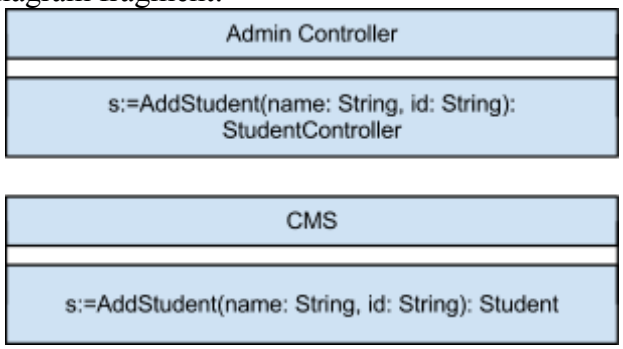
Interaction Diagram fragment:



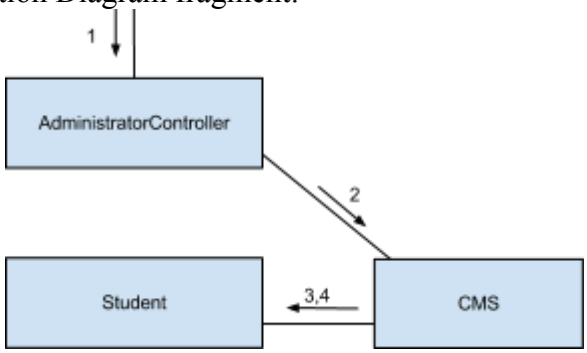
1: sc:=AddStudent(username: String, password: String, name:String): StudentController

2: s:=AddStudent(username: String, password: String, name:String): Student

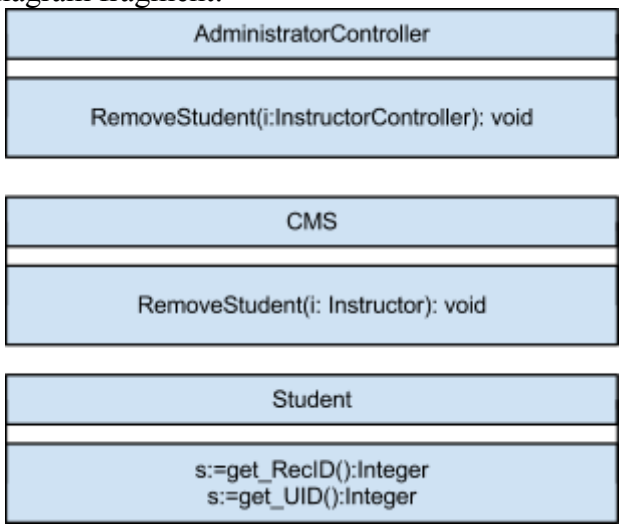
Class diagram fragment:



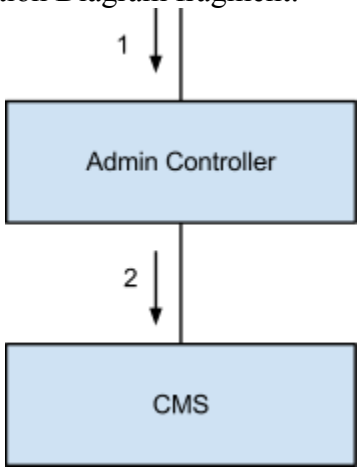
Name: RemoveStudent
Actor: Admin
Precondition: None
MSS: 1. User removes the student
2. Student information is removed, but ID cannot be taken again
Information Experts: Admin Controller knows Students
Data knows the Students
Interaction Diagram fragment:



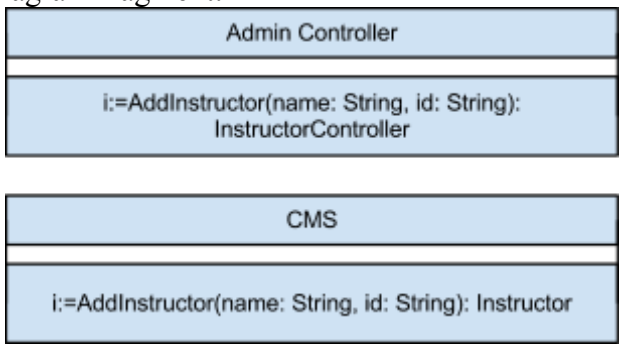
1: RemoveStudent(s: StudentController): void
2: RemoveStudent(s: Student): void
3: s:=get_RecID():Integer
4: s:=getUID():Integer
Class diagram fragment:



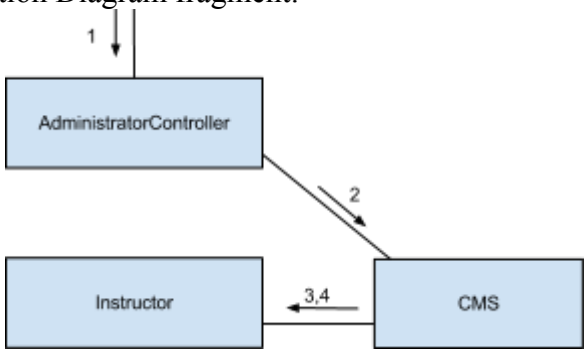
Name: AddInstructor
Actor: Admin
Precondition: NA
MSS: 1. Actor adds instructor
2. Instructor is added with everything blank
3. System populates the Employee ID and password
Information Experts: Admin Controller knows Instructor
Data knows Instructor
Interaction Diagram fragment:



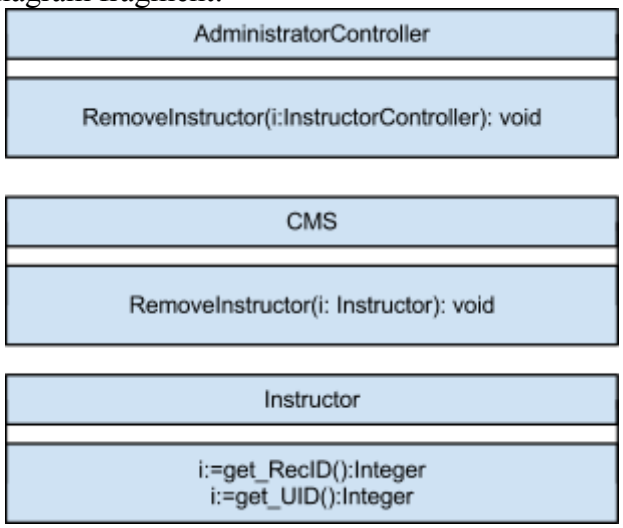
1: ic:=AddInstructor(username: String, password: String, name: String): InstructorController
2: i:=AddInstructor(username: String, password: String, name: String): Instructor
Class diagram fragment:



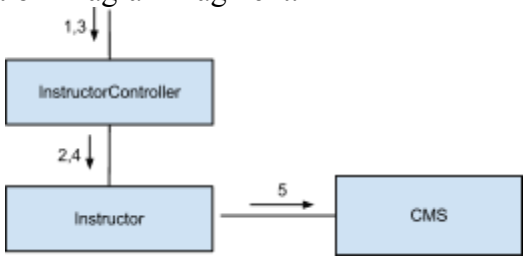
Name: RemoveInstructor
Actor: Admin
Precondition: None
MSS: 1. Instructor is removed via UI
2. Instructor's classes' status become "NeedsATeacher"
Information Experts: Admin Controller knows Instructor
Data knows Instructor
Interaction Diagram fragment:



1: RemoveInstructor(i: InstructorController): void
2: RemoveInstructor(i:Instructor): void
3: i:=get_RecID():Integer
4: i:=getUID():Integer
Class diagram fragment:

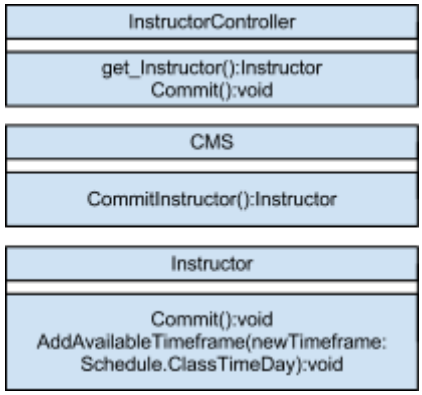


Name: AddTimeInstructorCanTeach
Actor: Teacher/Admin
Precondition: None
MSS: 1. Actor adds a time to schedule
2. Schedule reflects the changes
Information Experts: Admin Controller knows Instructor and their teaching time
Data knows Instructor and their teaching time
Instructor knows their teaching time
Interaction Diagram fragment:

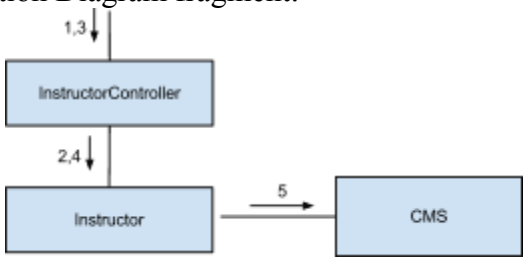


1: i:get_InstructorObject():Instructor
2: AddTimeframeAvailable(newTimeframe: Schedule.ClassTimeDay): void
3: Commit():void
4: Commit():void
5: CommitInstructor(i:Instructor):void

Class diagram fragment:

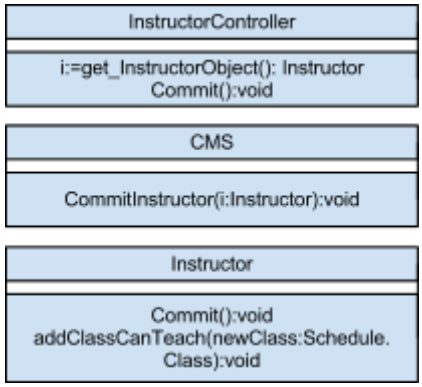


Name: AddClassesInstructorCanTeach
Actor: Admin
Precondition: None
MSS: 1. The classes added will be saved on the teachers teaching list
2. The classes will then be available for the students to add
Information Experts: Admin Controller knows Instructor and Class
Data knows Instructor and Class
Instructor knows Class
Interaction Diagram fragment:

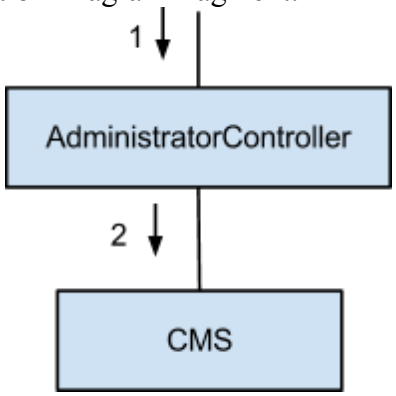


1: i:get_InstructorObject():Instructor
2: AddClassCanTeach(newClass: Schedule.Class): void
3: Commit():void
4: Commit():void
5: CommitInstructor(i:Instructor):void

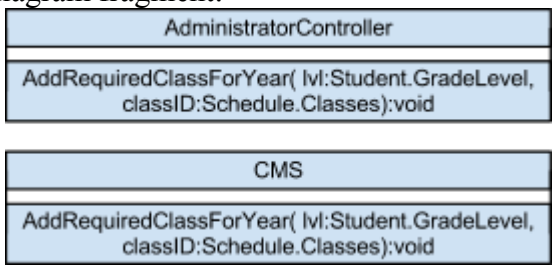
Class diagram fragment:



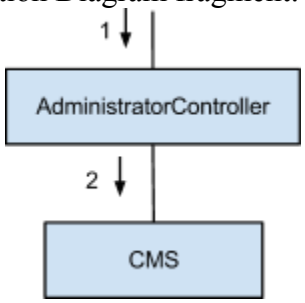
Name: AddClassesForStudents
Actor: Admin
Precondition: None
MSS: 1. Classes will be added to the student's schedule
2. Students will be able to see all classes they have scheduled
Information Experts: Admin Controller knows Student and Class
Data knows Student and Class
Student knows Class
Interaction Diagram fragment:



1: AddRequiredClassForYear(lvl:Student.GradeLevel, classID:Schedule.Classes):void
2: AddRequiredClassForYear(lvl:Student.GradeLevel, classID:Schedule.Classes):void
Class diagram fragment:

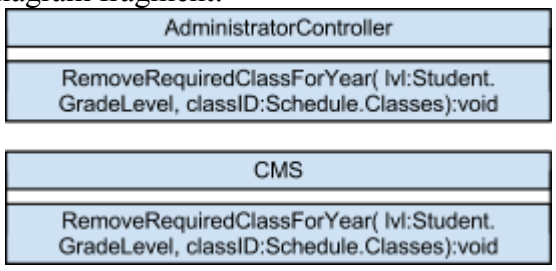


Name: RemoveClassesForStudents
Actor: Admin
Precondition: None
MSS: 1. The classes get removed from student's schedule
2. A StudyHall is put in place of that class until another class is added in it's place.
Information Experts: Admin Controller knows Student and Class
Data knows Student and Class
Student knows Class
Interaction Diagram fragment:

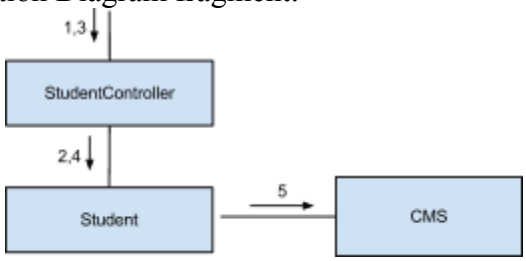


1: RemoveRequiredClassForYear(lvl:Student.GradeLevel, classID:Schedule.Classes):void
2: RemoveRequiredClassForYear(lvl:Student.GradeLevel, classID:Schedule.Classes):void

Class diagram fragment:

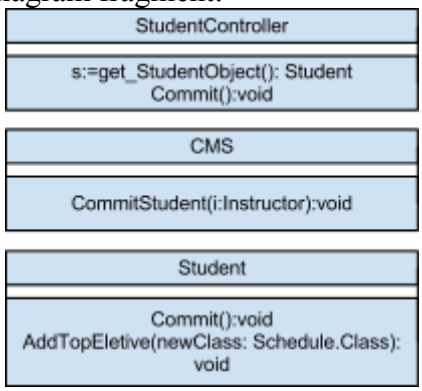


Name: AddTopElectives
Actor: Admin/Students
Precondition: None
MSS: 1. User adds an elective
2. System acknowledges
Information Experts: Admin Controller knows Student and Class
Data knows Student and Class
Student knows Class
Interaction Diagram fragment:

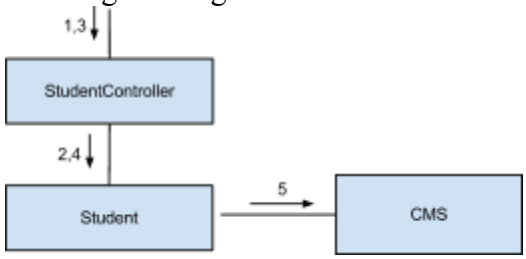


1: s:get_StudentObject():Student
2: AddTopEletive(newClass: Schedule.Class): void
3: Commit():void
4: Commit():void
5: CommitStudent(s:Student):void

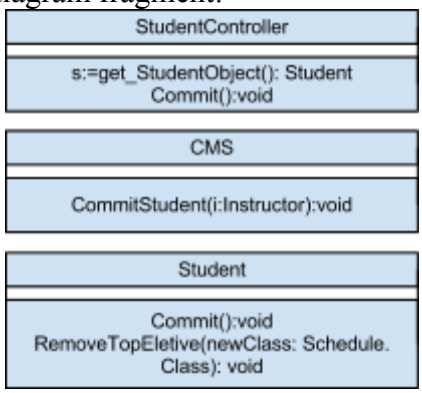
Class diagram fragment:



Name: RemoveTopElectives
Actor: Admin/Students
Precondition: None
MSS: 1. Changes to the student's electives will be saved
2. When the student adds an elective to their list, that elective will be removed from the master list
Information Experts: Admin Controller knows Student and Class
Data knows Student and Class
Student knows Class
Interaction Diagram fragment:



1: s:get_StudentObject():Student
2: RemoveTopEletive(newClass: Schedule.Class): void
3: Commit():void
4: Commit():void
5: CommitStudent(s:Student):void
Class diagram fragment:



Approved changes made to original iterations:

- Removed editClass (You can just delete and read since there isn't anything to edit but the name)
- Moved importUsers to next iteration (we need to write the code for adding users before we can figure out how to import them)
- Split ModifyTopElectives to AddTopElectives and RemoveTopElectives

Iteration 2

Complete Use Cases:

Name: GenerateSchedules
Actor: Admin
Precondition: NA
MSS:
1. Admin requests a schedule to be created
2. Schedules are created

Information Experts:

- Admin controller knows all schedules
- Admin knows all schedules

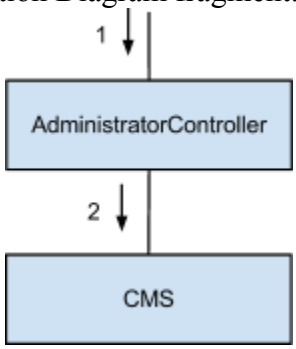
Extensions:
1a. No instructors exist

1. System provides admin with an error saying there needs to be at least one instructor to generate a schedule

1b. Instructor has no classes assigned

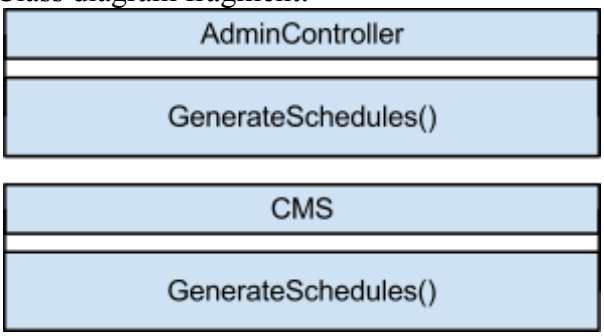
1. System provides admin with an error saying all instructors need to have at least on schedule.

Interaction Diagram fragment:



1: GenerateSchedules()
2: GenerateSchedules()

Class diagram fragment:



Name: displayMasterSchedule
Actor: Admin
Precondition: NA
MSS:

1. Admin requests a schedule to be shown
2. System displays schedule

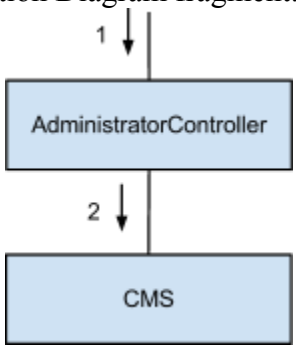
Information Experts:

- Admin controller knows all schedules
- Admin knows all schedules

Extensions
1a. No schedules have been generated

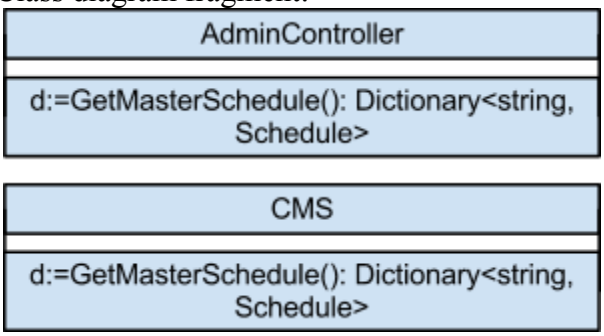
1. System provides admin with an error saying the schedules need to be generated before it can be displayed

Interaction Diagram fragment:



1: dictionary:GetMasterSchedule(): Dictionary<string,Schedule>
2: dictionary:GetMasterSchedule(): Dictionary<string,Schedule>

Class diagram fragment:



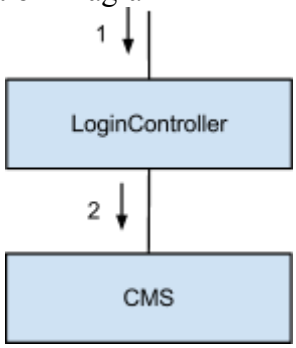
Name: displayStudentSchedule
Actor: Admin/Student
Precondition: NA
MSS:
1. Admin/Student requests student's schedule to be shown
2. System displays student's schedule
Information Experts:

- Admin controller knows Student and its schedule
- Student knows its schedule

Extensions
1a. Student's schedule has not been generated

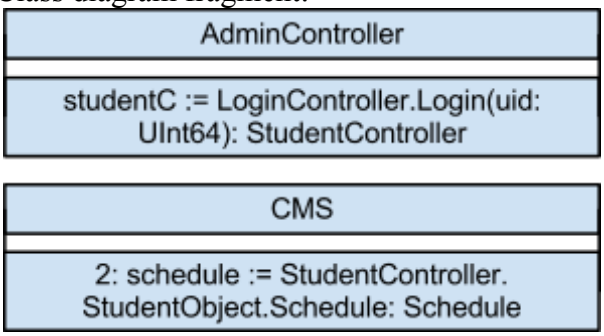
1. System provides admin/student with an error saying the student's schedule needs to be generated before it can be displayed

Interaction Diagram



1: studentC := LoginController.Login(uid:UInt64): StudentController
2: schedule := StudentController.StudentObject.Schedule: Schedule

Class diagram fragment:



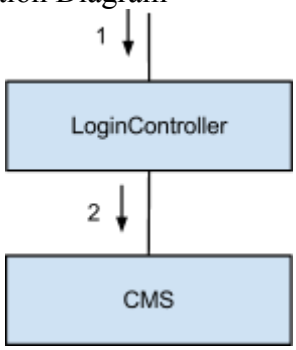
Name: displayTeacherSchedule
Actor: Admin/Instructor
Precondition: NA
MSS:
1. Admin/Instructor requests instructor's schedule to be shown
2. System displays instructor's schedule
Information Experts:

- Admin controller knows Instructor and its schedule
- Instructor knows its schedule

Extensions
1a. Instructor's schedule has not been generated

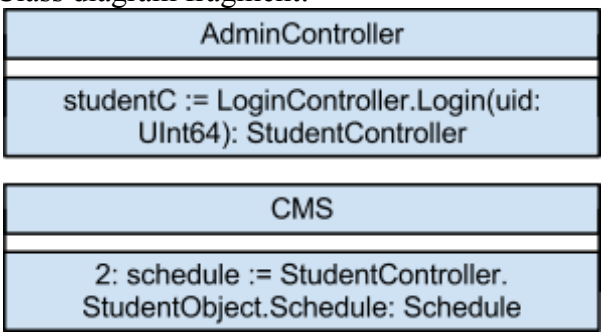
1. System provides admin/instructor with an error saying the instructor's schedule needs to be generated before it can be displayed

Interaction Diagram



1: InstructorC := LoginController.Login(uid:UInt64): InstructorController
2: schedule := InstructorController.InstructorObject.Schedule: Schedule

Class diagram fragment:



Iteration 3

NOTE: Login/Logoff for teachers and instructors was finished in Iteration 1, since they are pretty much identical. Also due to circumstances shown in our report, we are not implementing the “Add/Remove work time” use cases.

Name: ModifyTopElectives

Actor: Students and Admin

MSS: 1. User adds/removes an elective

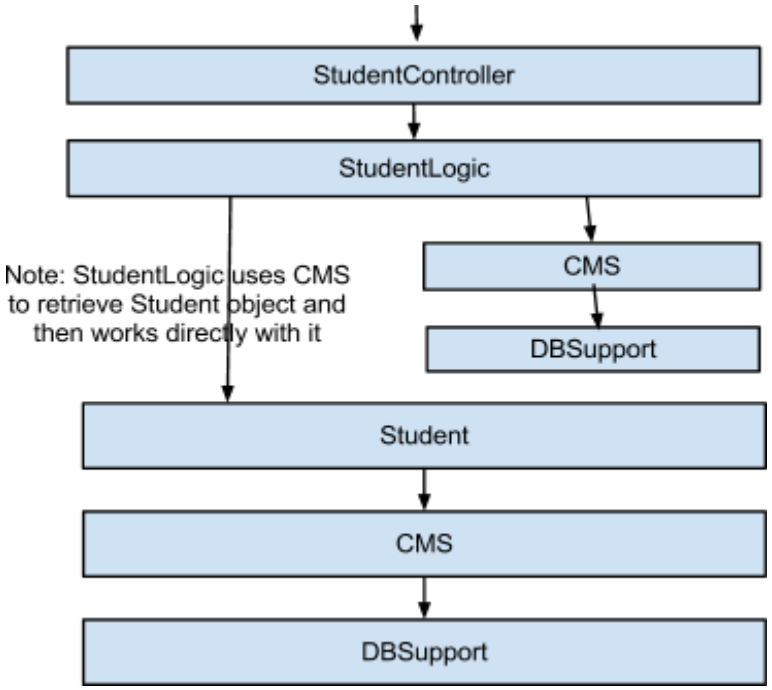
2. Changes to the student's top electives will be saved

Information Experts: Admin Controller knows Student and Class
Data knows Student and Class
Student knows Class

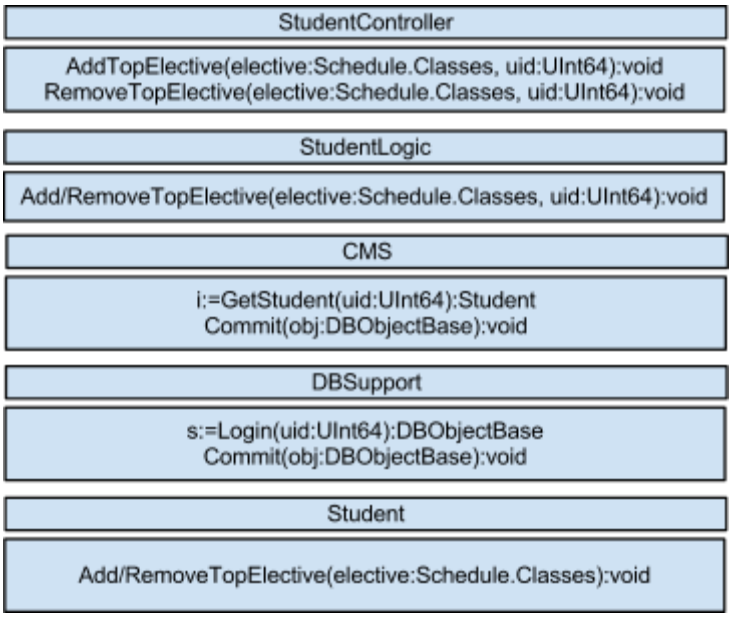
Extensions

1a. User removes an elective that is required for their grade level

1. System provides admin and student with an error saying the student's top elective must contain the required class for their grade level



Class diagram fragment:


StudentController
AddTopElective(elective:Schedule.Classes, uid:UInt64):void
RemoveTopElective(elective:Schedule.Classes, uid:UInt64):void
StudentLogic
Add/RemoveTopElective(elective:Schedule.Classes, uid:UInt64):void
CMS
i:=GetStudent(uid:UInt64):Student
Commit(obj:DBObjectBase):void
DBSupport
s:=Login(uid:UInt64):DBObjectBase
Commit(obj:DBObjectBase):void
Student
Add/RemoveTopElective(elective:Schedule.Classes):void

Name: displayStudentSchedule

Actor: Students

Precondition: NA

MSS:

1. Student requests their schedule to be shown
2. System displays student's schedule

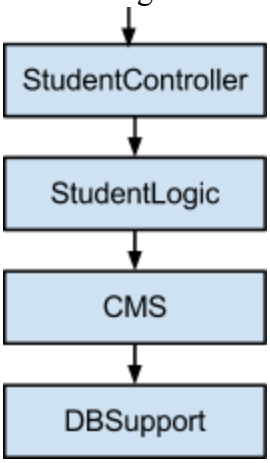
Information Experts:

- Student controller knows Student and its schedule
- Student knows its schedule

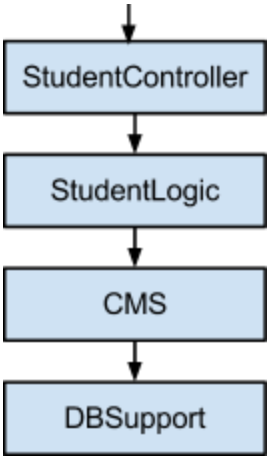
Extensions

1a. Student's schedule has not been generated

1. System provides admin and student with an error saying the student's schedule needs to be generated before it can be displayed

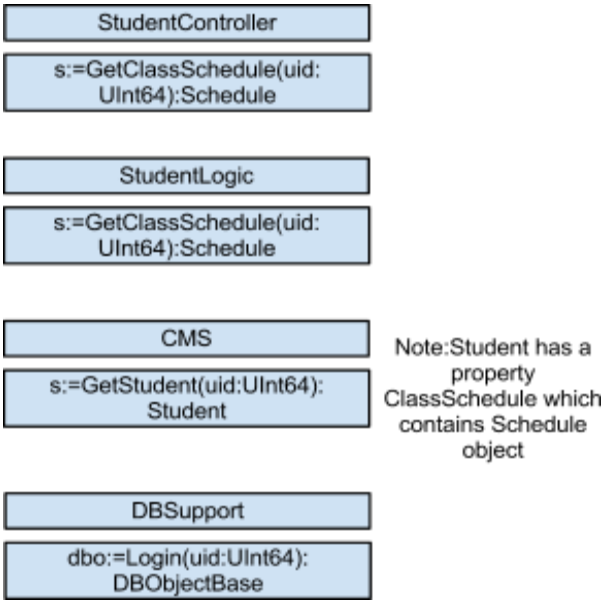


Extension 1a diagram:



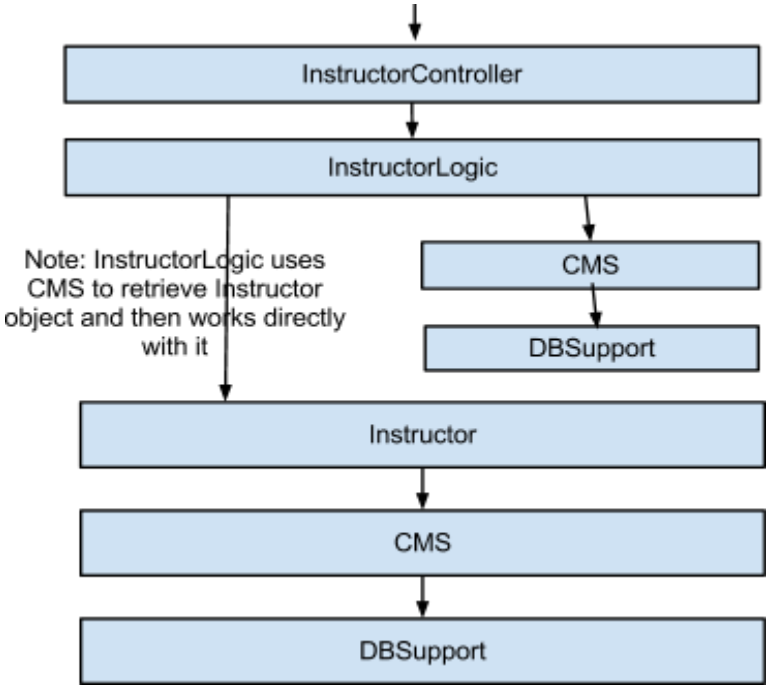
Note: DBSupport will construct a Student object with ClassSchedule property set to null, then UI will process that as an error.

Class diagram fragment:

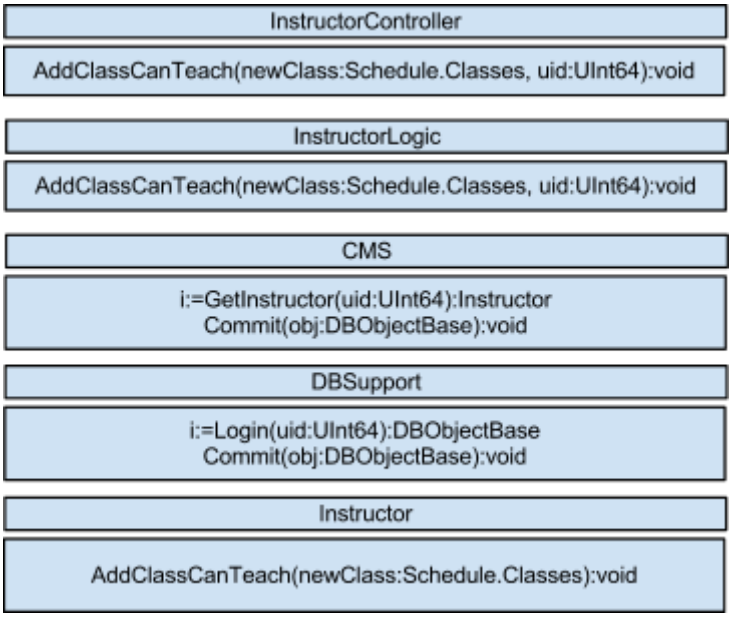


Name: AddClassesInstructorCanTeach
Actor: Admin
Precondition: User must be logged in
MSS:
1. User provides the instructor ID and the class to add
2. System acknowledges successful addition
Information Experts:

- Admin controller knows the schedule and the instructor



Class diagram fragments:

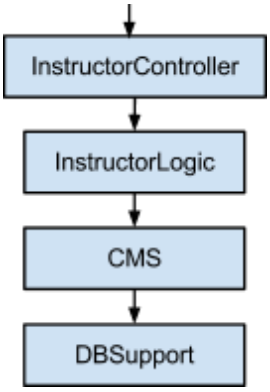


Name: displayTeacherSchedule
Actor: Instructor
Precondition: NA
MSS:
1. Instructor requests their schedule to be shown
2. System displays instructor's schedule
Information Experts:

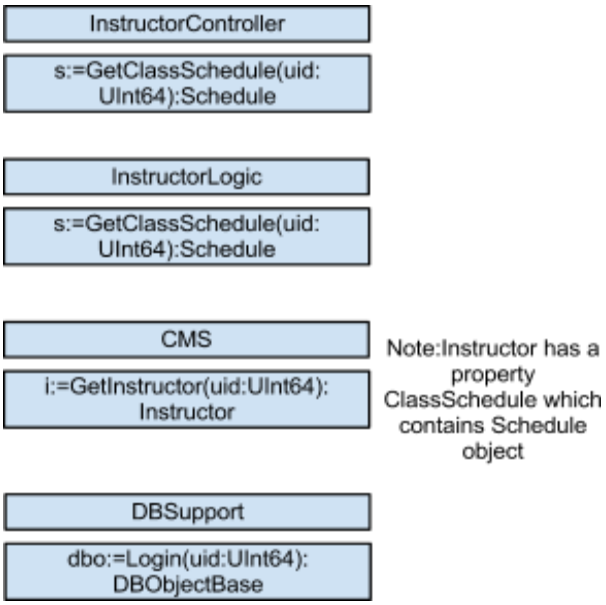
- Instructor controller knows Instructor and its schedule
- Instructor knows its schedule

Extensions
1a. Instructor's schedule has not been generated

1. System provides admin and instructor with an error saying the instructor's schedule needs to be generated before it can be displayed



Class diagram fragment:


InstructorController
s:=GetClassSchedule(uid: UInt64):Schedule
InstructorLogic
s:=GetClassSchedule(uid: UInt64):Schedule
CMS
i:=GetInstructor(uid:UInt64): Instructor
Note:Instructor has a property ClassSchedule which contains Schedule object
DBSupport
dbo:=Login(uid:UInt64): DBObjectBase