CPS 731 - Software Engineering I

Project - Designing an Online Registration System

Phase 4

Analysis and Design

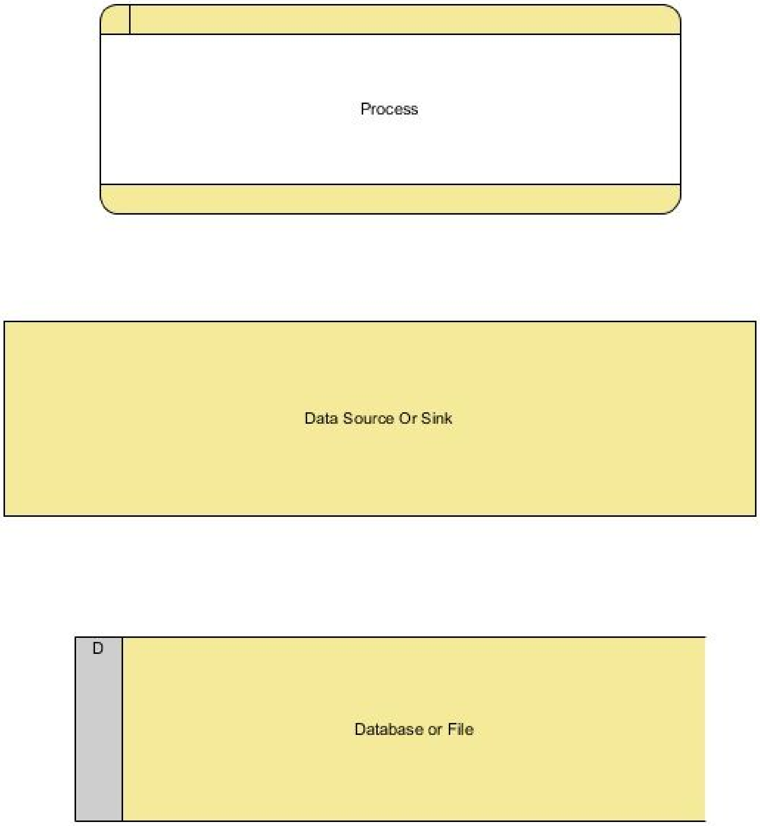
For: Dr. Tirandazian

From: James Diamond 500657801

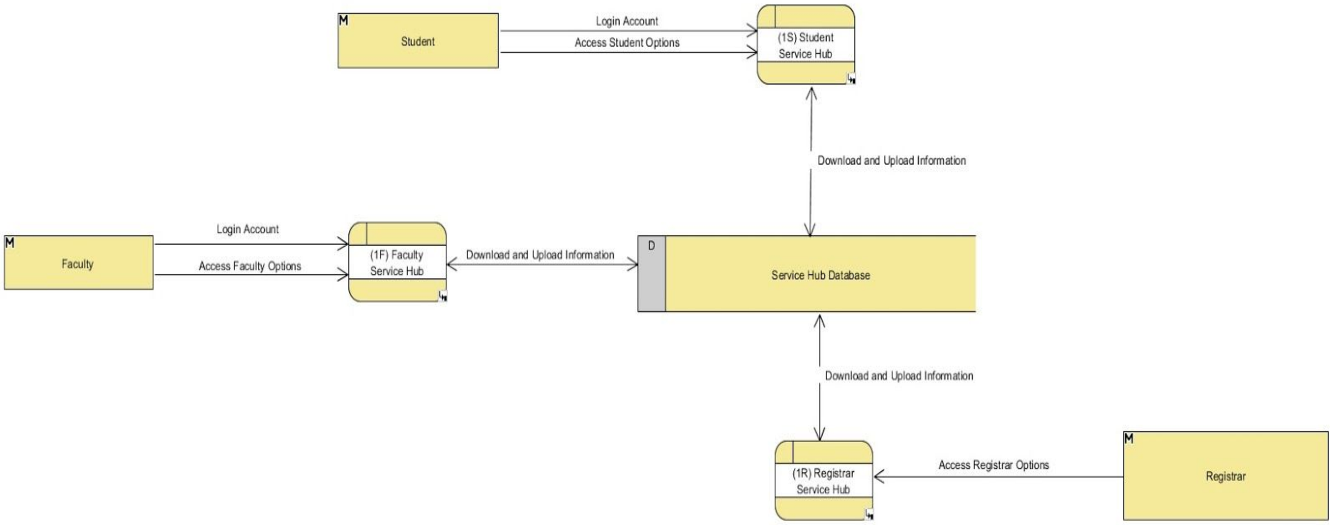
Marcel-Pierre Samuels 500617694

Due: Sunday, November 26, 2017

Data Flow Legend:

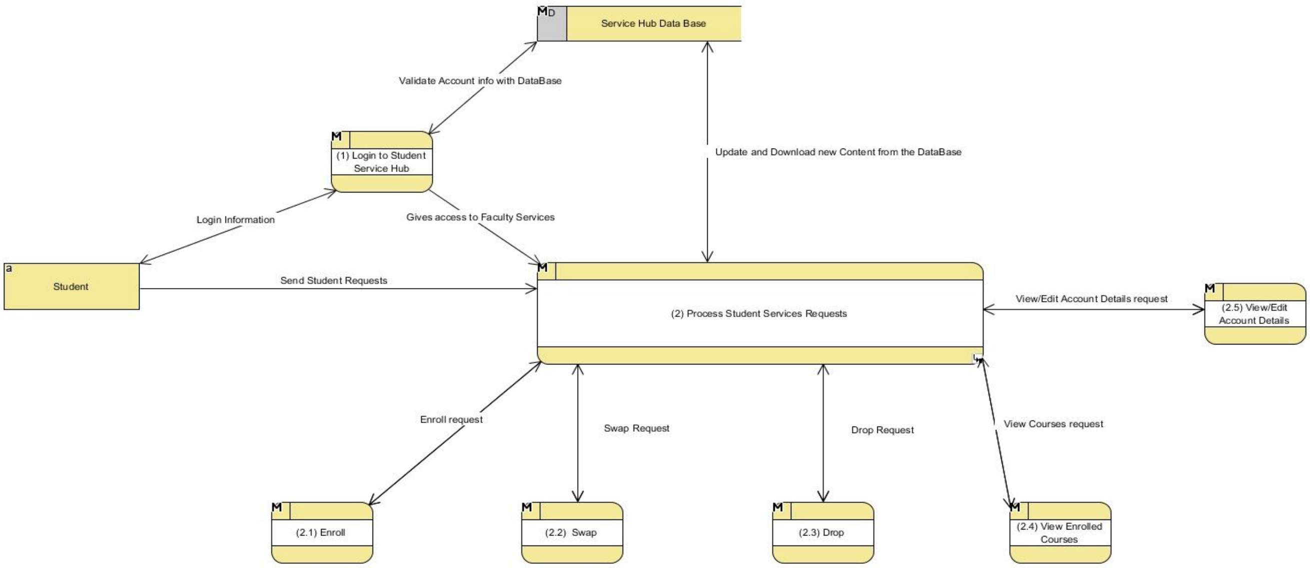


Level 0 Data Flow Diagram

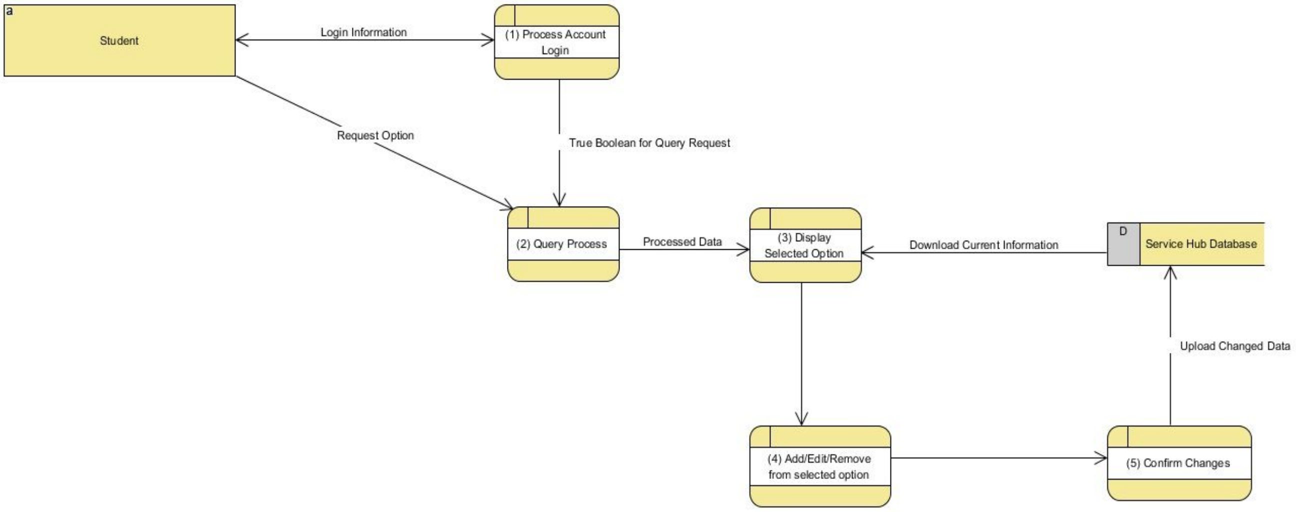


Student View

Level 1 Student Service Hub Decomposition

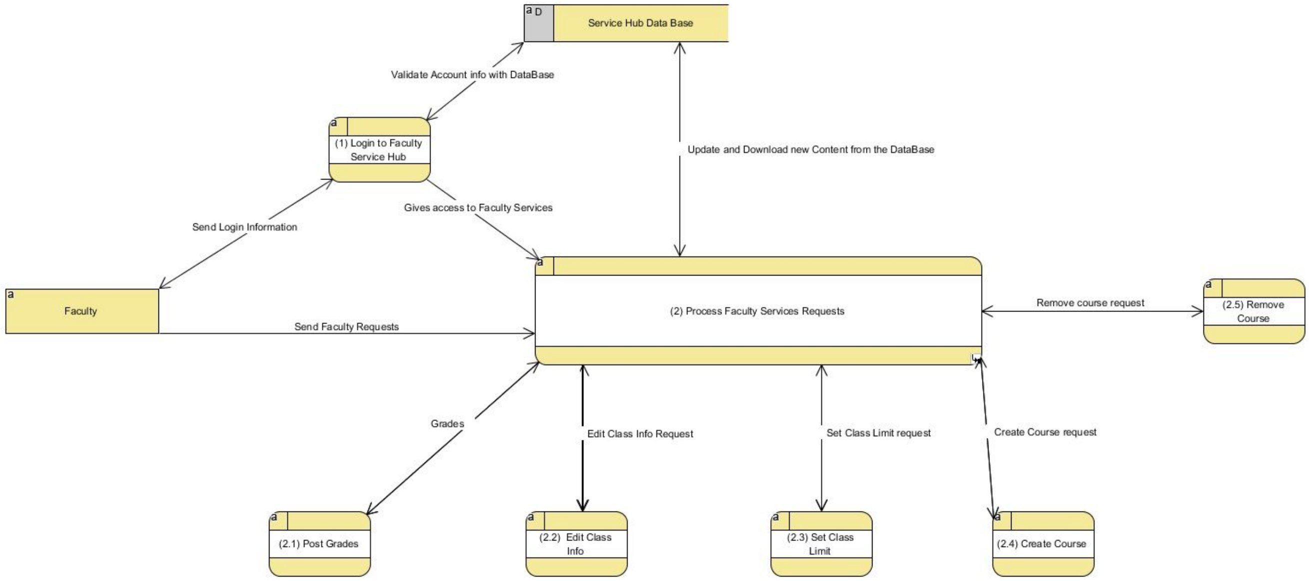

Service Hub Data Base
Validate Account info with DataBase
(1) Login to Student Service Hub
Update and Download new Content from the DataBase
Login Information
Gives access to Faculty Services
Student
Send Student Requests
(2) Process Student Services Requests
View/Edit Account Details request
(2.5) View/Edit Account Details
Enroll request
Swap Request
Drop Request
View Courses request
(2.1) Enroll
(2.2) Swap
(2.3) Drop
(2.4) View Enrolled Courses

Level 2 Student Services Decomposition

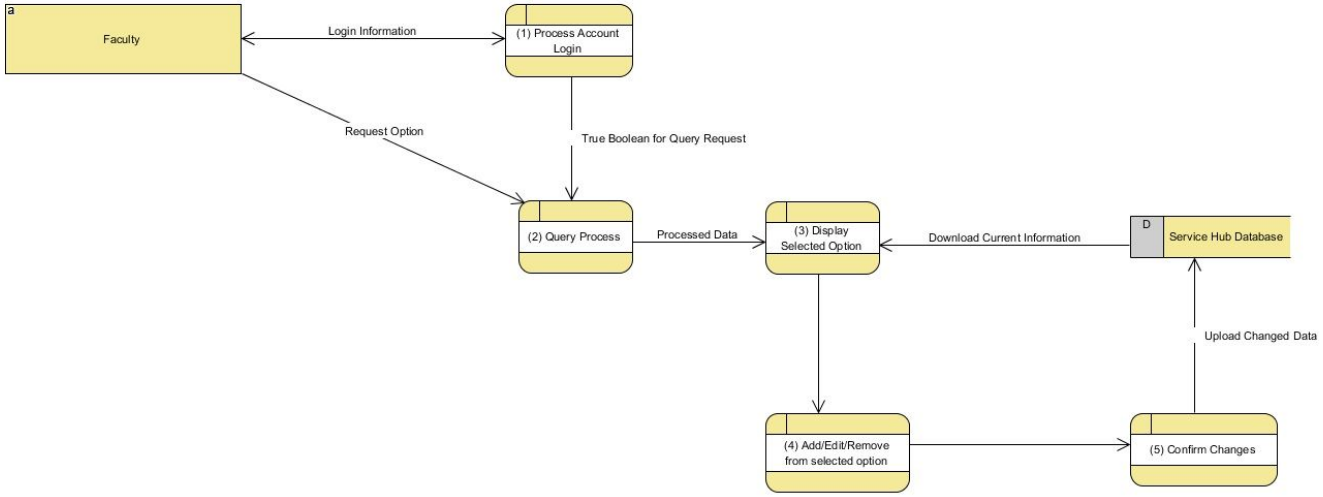

Student
Login Information
(1) Process Account Login
Request Option
True Boolean for Query Request
(2) Query Process
Processed Data
(3) Display Selected Option
Download Current Information
Service Hub Database
Upload Changed Data
(4) Add/Edit/Remove from selected option
(5) Confirm Changes

Faculty View

Level 1 Faculty Service Hub Decomposition

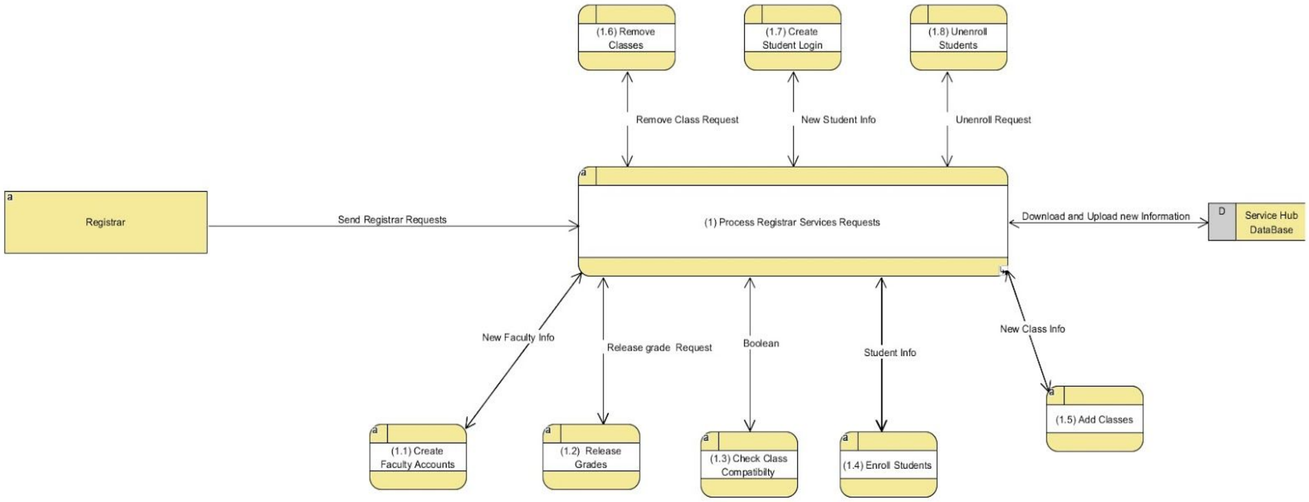


Level 2 Faculty Services Decomposition

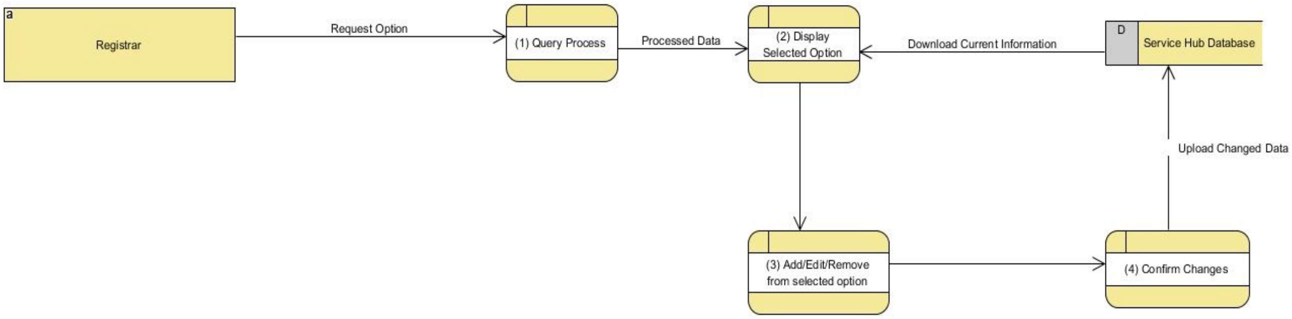

a
Faculty
Login Information
(1) Process Account Login
Request Option
True Boolean for Query Request
(2) Query Process
Processed Data
(3) Display Selected Option
Download Current Information
D
Service Hub Database
Upload Changed Data
(4) Add/Edit/Remove from selected option
(5) Confirm Changes

Registrar View

Level 1 Registrar Service Hub Decomposition



Level 2 Registrar Services Decomposition



Data Dictionary

Level 0 Diagram

1S. Login-Account= 1{User-name+Password}
1S. Download-and-Upload= Student-Acc-login+Class-Size+Acc-Details+Avail-Courses

1F. Login-Account= 1{User-name+Password}
1F. Download-and-Upload=Class-size+Faculty-Acc+Grades+Active-Classes

1R. Download-and-Upload=Students-Acc+Faculty-Acc+Grades+All-Courses

Level 1 Student Service Hub

1. Login-Account= 1{User-name+Password}
1. Give-Access= [Yes | No]

2. Send-Student-Req= 0{ [Enroll | Swap | Drop | View-Enrolled-Course | V/E-Acc-Details] }
2. Update-and-Download= Student-Acc-login+Class-Size+Acc-Details+Avail-Courses+ Enrolled-Courses

2.1. Enroll-req= Avail-Course-req
2.2. Swap-req= Enrolled-Courses
2.3. Drop-req= Enrolled-Course
2.4. View-Course-req= Enrolled-Courses + Course-Details
2.5. V/E-Acc-Details= Student-num+Student-email+Inquiry+Academic-Standing

Level 2 Student Service Hub

1. Login-Account= 1{User-name+Password}
1.Give-Access= [Yes | No]

2. Req-Option= 0{ [Enroll | Swap | Drop | View-Enrolled-Course | V/E-Acc-Details] }
2. Processed-Data= 0{ [Enroll | Swap | Drop | View-Enrolled-Course | V/E-Acc-Details] }

3. Download-Curr= Student-Acc-login+Class-Size+Acc-Details+Avail-Courses +Enrolled-Course

5. Upload-Changed= Class-Size+Avail-Courses+Enrolled-Course

Level 1 Faculty Service Hub

1. Login-Account= 1{User-name+Password}

2. Give-Access= [Yes | No]
2. Send-Faculty-Req= 0{ [Post-Grades | Edit-Class-Info| Set-Limit | Create-Course | Remove-Course] }
2. Update-and-Download= Faculty-Acc-login+Faculty-Info + Class-details+Grades

2.1. Grades-Req = Class-number+ Student-number+Grades

2.2. Edit-Class-Info-req = Class-description
2.3. Class-Limit-Req = Class-Size
2.4. Create-Course-req = Course-Description
2.5. Remove-Course-req = Class-Info

Level 2 Faculty Service Hub

1. Login-Account= 1{User-name+Password}
1.Give-Access= [Yes | No]

2. Req-Option= 0{ [Post-Grades | Edit-Class-Info| Set-Limit | Create-Course | Remove-Course] }
2. Processed-Data= 0{ [Post-Grades | Edit-Class-Info| Set-Limit | Create-Course | Remove-Course] }

3. Download-Curr= Faculty-Acc-login+Faculty-Info + Class-details+Grades

5. Upload-Changed= Class-Size+Class-description+Grades

Level 1 Registrar Service Hub

1. Send-Registrar-Req =0 {[Create Faculty Accounts | Release Grades | Check Class Compatibility| Enroll Students | Add Classes | Remove Classes | Create Student Login | Unenroll Students]}
1. Download-and-Upload = Faculty-number+ Faculty-Name+ Faculty-email + Student-Info +Student-name+ Student-Number+Student-Email+ Faculty-Acc+Student-Acc+ Class-Info+Grades

1.1. New-Faculty-Req = Create-Faculty-Accounts + Faculty-Name+ Faculty-Number+ Faculty-Email
1.2. Release-Grade-Req = 0{[Yes | No]}
1.3. Class-Compatibility-Check = Class-Sched+ Student-number
1.4. Enroll-Students = Class-Info + Student-number
1.5. Add-Class-Req = Class-details+Class-Sched+Class-Size
1.6. Remove-Class-Req = Class-details+Class-Sched+Class-Size
1.7. New-Student-Req = Create-Student-Account + Student-Name+ Student-Number+ Student-Email
1.8. Unenroll-Req = Class-Info + Student-Info

Level 2 Registrar Service Hub

1. Req-Option= 0{ [Post-Grades | Edit-Class-Info| Set-Limit | Create-Course | Remove-Course] }

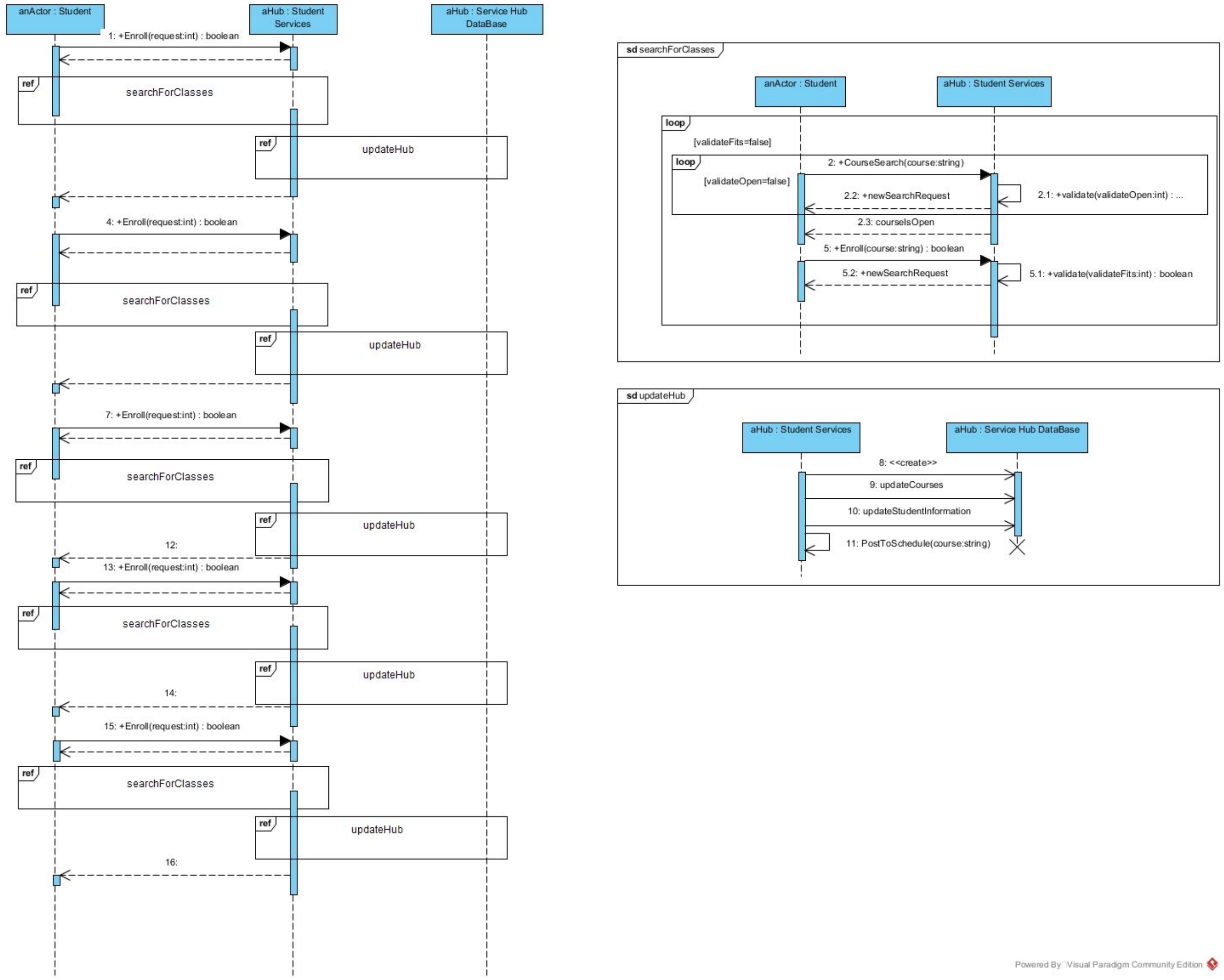
2. Processed-Data= 0{ [Post-Grades | Edit-Class-Info| Set-Limit | Create-Course | Remove-Course] }

3. Download-Curr= Faculty-number+ Faculty-Name+ Faculty-email + Student-Info +Student-name+ Student-Number+Student-Email+ Faculty-Acc+Student-Acc+ Class-Info+Grades

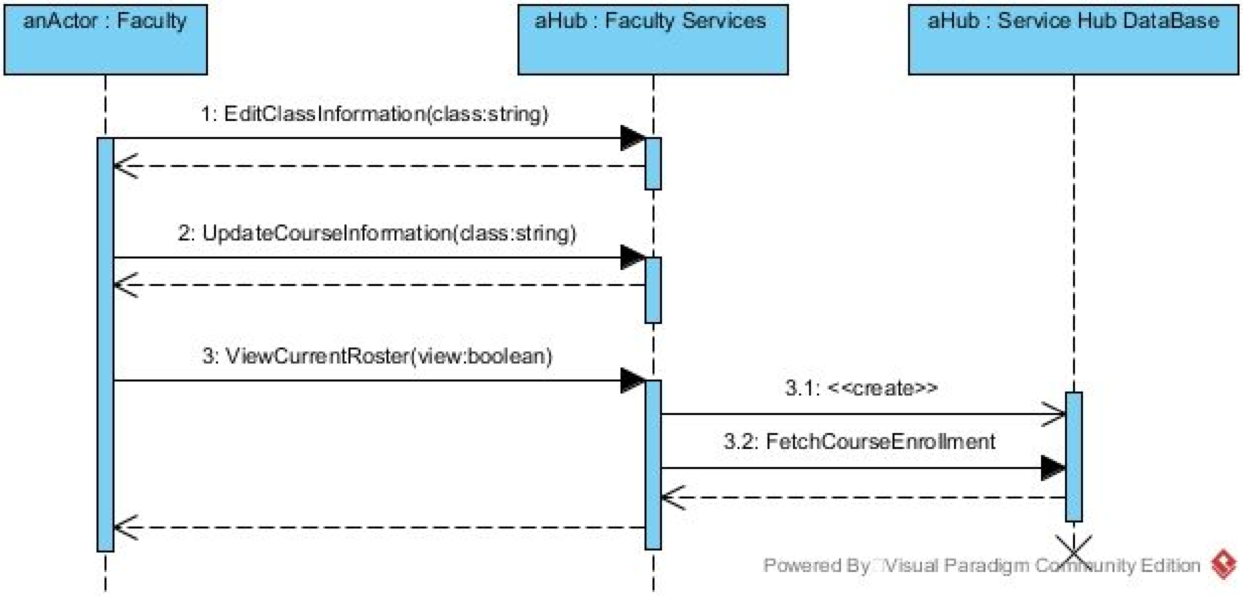
4. Upload-Changed= Class-Information+ Enrolled-Courses + Release-Grade-Boolean + FacultyAcc+Student-acc

Sequence Diagrams:

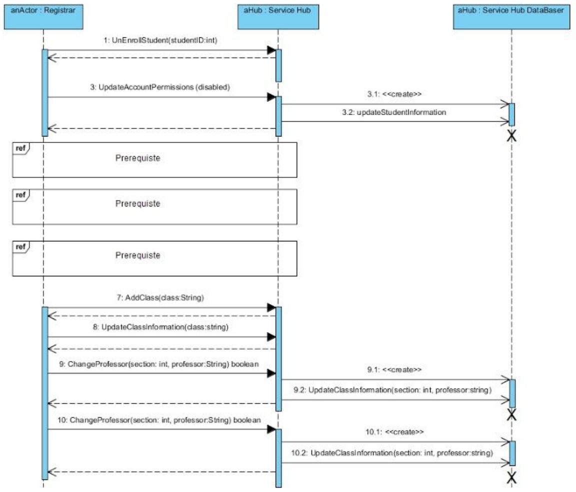
Student:

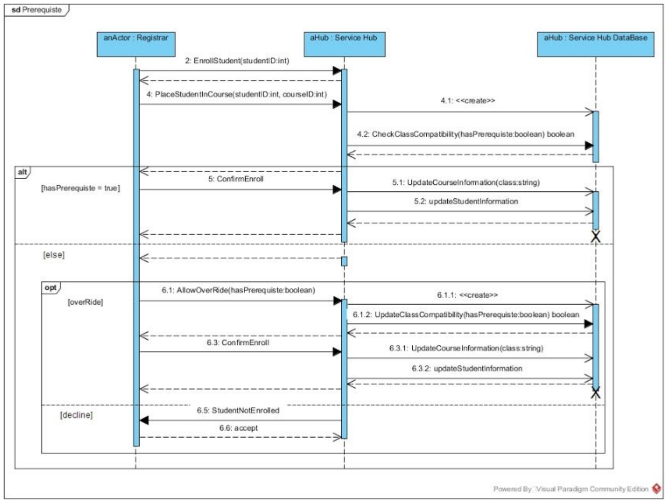

anActor : Student
aHub : Student Services
aHub : Service Hub DataBase
1: +Enroll(request:int) : boolean
ref
searchForClasses
ref
updateHub
4: +Enroll(request:int) : boolean
ref
searchForClasses
ref
updateHub
7: +Enroll(request:int) : boolean
ref
searchForClasses
ref
updateHub
12:
13: +Enroll(request:int) : boolean
ref
searchForClasses
ref
updateHub
14:
15: +Enroll(request:int) : boolean
ref
searchForClasses
ref
updateHub
16:
sd searchForClasses
anActor : Student
aHub : Student Services
loop
[validateFits=false]
loop
[validateOpen=false]
2: +CourseSearch(course:string)
2.1: +validate(validateOpen:int) : ...
2.2: +newSearchRequest
2.3: courseIsOpen
5: +Enroll(course:string) : boolean
5.1: +validate(validateFits:int) : boolean
5.2: +newSearchRequest
sd updateHub
aHub : Student Services
aHub : Service Hub DataBase
8: <<create>>
9: updateCourses
10: updateStudentInformation
11: PostToSchedule(course:string)
Powered By Visual Paradigm Community Edition

Faculty:


anActor : Faculty
aHub : Faculty Services
aHub : Service Hub DataBase
1: EditClassInformation(class:string)
2: UpdateCourseInformation(class:string)
3: ViewCurrentRoster(view:boolean)
3.1: <<create>>
3.2: FetchCourseEnrollment
Powered By Visual Paradigm Community Edition

Registrar:


anActor : Registrar
aHub : Service Hub
aHub : Service Hub DataBaser
1: UnEnrollStudent(studentID:int)
3: UpdateAccountPermissions (disabled)
3.1: <<create>>
3.2: updateStudentInformation
ref
Prerequiste
ref
Prerequiste
ref
Prerequiste
7: AddClass(class:String)
8: UpdateClassInformation(class:string)
9: ChangeProfessor(section: int, professor:String) boolean
9.1: <<create>>
9.2: UpdateClassInformation(section: int, professor:string)
10: ChangeProfessor(section: int, professor:String) boolean
10.1: <<create>>
10.2: UpdateClassInformation(section: int, professor:string)


sd Prerequiste
anActor : Registrar
aHub : Service Hub
aHub : Service Hub DataBase
2: EnrollStudent(studentID:int)
4: PlaceStudentInCourse(studentID:int, courseID:int)
4.1: <<create>>
4.2: CheckClassCompatibility(hasPrerequiste:boolean) boolean
alt
[hasPrerequiste = true]
5: ConfirmEnroll
5.1: UpdateCourseInformation(class:string)
5.2: updateStudentInformation
[else]
opt
[overRide]
6.1: AllowOverRide(hasPrerequiste:boolean)
6.1.1: <<create>>
6.1.2: UpdateClassCompatibility(hasPrerequiste:boolean) boolean
6.3: ConfirmEnroll
6.3.1: UpdateCourseInformation(class:string)
6.3.2: updateStudentInformation
[decline]
6.5: StudentNotEnrolled
6.6: accept
Powered By Visual Paradigm Community Edition

Updated Priority Table

Based on the iterative development approach, the priority table has been redesigned. The goals of the updated priorities are to allow for the implementation of a rudimentary system whose initial goals are to allow students to register for a semester of courses as well as view their schedule. The reasoning behind using this approach is to get a simple version of the system up and running where feedback is easy to gather for future design. Due to this, the high ranked requirements are the minimum needed for the system to work, as opposed to those that will help with general functionality, ease of use, etc.

Functional

High Priority	Medium Priority	Low Priority
Create Student accounts	public/private listing of courses	Edit class visibility
Store System content on server	View enrolled courses	Edit class specifics
Enroll, Drop	View account details	Post grades
Login/Logout	Set class size limit	Release Grades
Enroll students	Create Course	
Unenroll students	Remove Course	
Add classes	List available courses	
Remove classes	Swap course	
Check if class can be enrolled in	Shopping Cart	

The following were removed from high priority and moved to medium priority. List available courses: To enroll you do not need to view all available courses but if you know the course code you can enroll. Limited functionality for version 1. The ability to swap a course and keep a shopping cart are also not needed in the first version. The rest of the priorities have been made the same. This should allow for the system to operate as needed in version one.

Non-Functional

High Priority	Medium Priority	Low Priority
GUI login screen with password protection	10s should be a max time for transactions	Ease of use
Support 4000 users simultaneously	Increasing the number of users able to access	Enrolment Costs
Ensuring course content is kept private to each user	Scalable for many students	
User ID is kept personal	Able to handle the load of multiple process	
Each user group has distinct permissions	Project Cost	
Web Accessibility	Availability (Access should be 24/7)	

Scalability is not a high requirement for the first version. The system needs to be able to support 4000 users but it does not need to be scalable in the first version. The ability of the server to handle the load of students is important but it is not of the utmost importance. The system can be slow and work. Continuing this trend, the availability of the service does not need to be 24/7 in the first version but should strive for that sometime soon. Finally, viewing enrollment costs and the actual project cost are not high priorities for this first version. The system needs to be made and the cost cannot get in the way. As well, students do not need to be able to do finances on the system just yet.