CPS 731 - Software Engineering I

Project - Designing an Online Registration System

Phase 3

For: Dr. Tirandazian

From: James Diamond 500657801

Marcel-Pierre Samuels 500617694

Due: Sunday, November 12, 2017

Phase 1 - Requirements gathering

Gathering Process:

Given the scope of this project, the following was done to aid in the requirements gathering stage.

Firstly, the current system at Ryerson was observed. We went through RAMSS and made a note of what we liked and didn't like, as well as what basic operations we liked and would use for our own system. What we found from observing RAMSS was that overall it is a fine system but it can be very slow. We have made a requirement for our system to have a maximum of 10 seconds per transaction. As the system scales we want to avoid a slow user experience, which can lead to frustration and avoidance of the system.

Some of the features in RAMSS we did include in our own system, which can be seen from the use cases but have been organized differently. An example of this is the student user "View/Edit Account Details" where in here the student is able to easily see anything that pertains to specifically them and their account in our system and have the options, where applicable, to make changes. We felt that the student experience should be a very simple and painless one and this was made a high priority.

RAMSS does have a simple user interface, which we liked, and gave a decent idea of the minimal functional requirements we would need. To aid us farther in the requirements gathering we spoke with some students and an office member. They were asked a few simple questions about the current system, such as what they liked and didn't like, which we then took and wrote what we thought would make for a simple and efficient system of our own. A few TA's were also consulted to get more of a faculty perspective. Not specifically with RAMSS but with D2L as well. It was important to get information from a back-end user to make sure we did not miss any of the functional requirements for the three main account types.

The rest of the requirements gathering were very straight forward. Once we had an idea of what functional requirements each user type needed to get a functional system operating, the non-functional requirements seemed rather obvious. From the people we spoke to, as well as our own experience in RAMSS, we ranked these non-functional requirements to make an effective system. Some features such as security and cost are very important, but something like server up-time we wanted to improve upon RAMSS and aim for the gold standard at 99.999% uptime. With a system such as this, with the

many users it has across many different schedules we did not want to limit people from accessing the system if we did not have to.

Lastly, we needed a system that was scalable and modifiable. Starting from a few users to up to 4000 simultaneous users and possibly supporting tens of thousands of accounts, the system needs to be able to grow as its user database does. This is very important and can have an impact of the cost, user experience, up-time, and transaction time. A scalable system is key here; this system should only be designed once and not have to re-do anything as the system needs to grow.

Functional:

General

- List available courses
- Store system content on server
- public/private listing of courses

3 different users with separate requirements

Registrar

- Check if class can be enrolled in
- Enroll students
- Unenroll students
- Add classes
- Remove classes
- Edit class visibility
- Release Grades
- Create student accounts
    - Login
    - Student ids

Faculty

- Create course
- Remove course
- Set class size limit
- Edit class information
    - I.e. course description, cost of course, number of students
- Post grades
- Login/Logout

Students

- Enroll
- Drop
- Swap
- Shopping cart
- View enrolled courses (current schedule)
- View account details
    - See student number, change password, see fees due, see enrolled courses list view, see course grades
- login/out

The functional requirements were based on what we deemed necessary to get software working. We focused on the functions of the system that are most important and crucial for the running of an online registration system such as this. It came down to having 3 distinct user groups each with some unique and shared functions in the system.

Non-Functional:

- Support 4000 users simultaneously
- GUI login screen with password protection
- Scaleable for many students
- Reasonable Costs for the project
- Web accessibility
- Ease of use (Easy to navigate and enroll)
    - Mobile devices
- Security
    - Each user group has distinct permissions
    - Ensuring course content is kept private to each user
    - User ID is kept personal
    - Cost cost
- Performance
    - Able to handle the load of multiple process
    - 10s should be a max time for transactions
        - I.e. enrolling, dropping, logging in, etc
- Availability (Access should be 24/7)
    - Aiming for 99.999% uptime
    - Maintenance time, middle of the night during not peak hours
- Scalable for many more than 4000 students in the system
    - Just 4000 simultaneous
- Modifiability
    - Increasing the number of users able to access

The non-functional requirements were chosen on how the system was to perform. It was important to separate the three user groups for the functional requirements but here in the non-functional requirements the actual system specifications were designed and obtained for system behaviour.

Priority Table

Functional

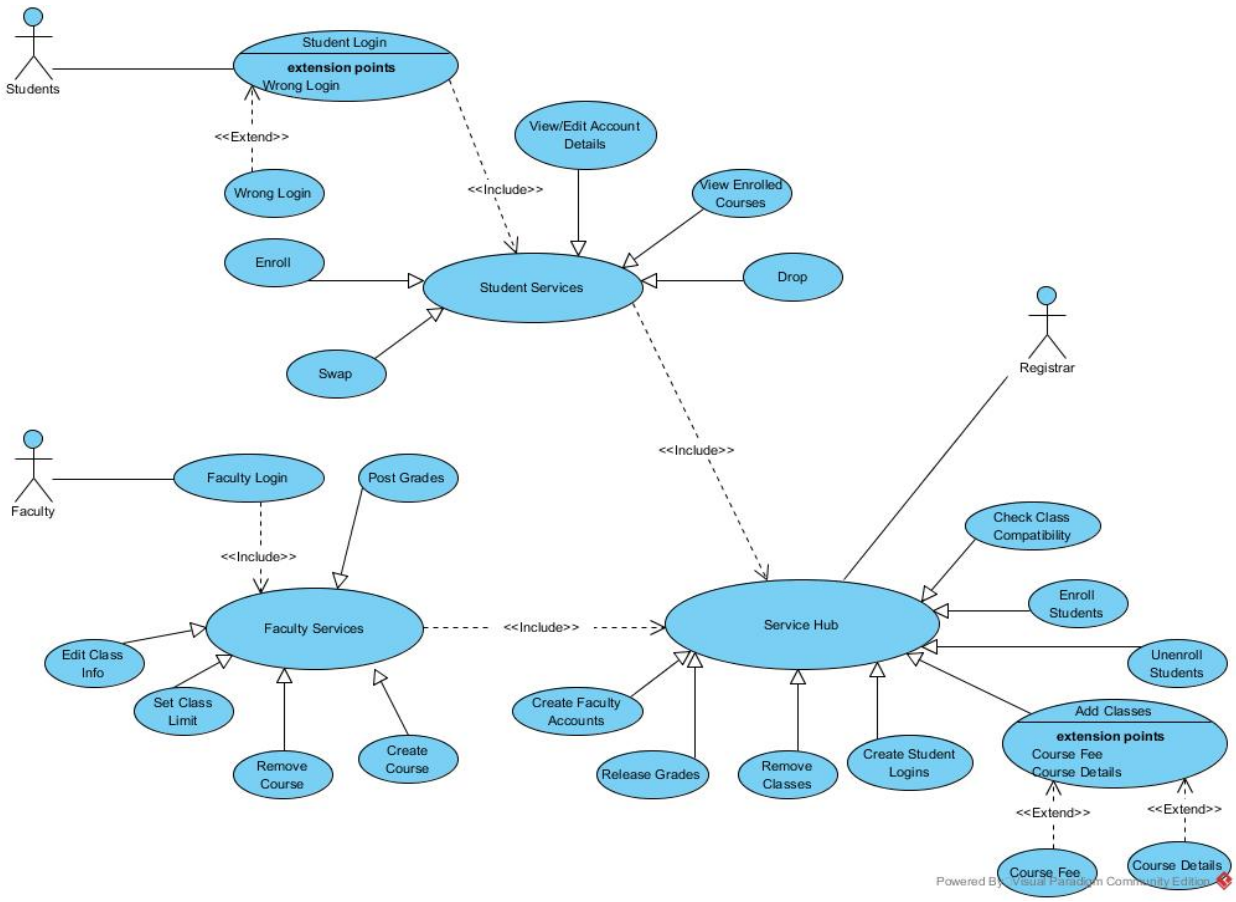
High Priority	Medium Priority	Low Priority
List available courses	public/private listing of courses	Edit class visibility
Create Student accounts	View enrolled courses	Edit class specifics
Store System content on server	View account details	Post grades
Enroll, Drop, Swap	Set class size limit	Release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

Non-Functional

High Priority	Medium Priority	Low Priority
GUI login screen with password protection	10s should be a max time for	

	transactions	
Support 4000 users simultaneously	Increasing the number of users able to access	
Scaleable for many students		
Ease of use		
Ensuring course content is kept private to each user		
User ID is kept personal		
Each user group has distinct permissions		
Able to handle the load of multiple process		
Enrolment Costs		
Availability (Access should be 24/7)		
Project Cost		
Web Accessibility		

UC Diagram



The use case diagram was designed to show an overview of the actions the different actors in the system can take and how they will impact the system (and other actors/roles) so that the system can achieve its goals.

Use Cases

Use Case Name: Student Login

Brief Description:

- To initiate the access to one's student services account.

Primary Actors

- Students

Secondary Actors

- None

Preconditions:

1. Student must have a Ryerson Student Login

Main flow:

1. Prompt to enter student username and password
2. If Student enters the wrong login
   2.1. Extend (Wrong Login)
3. Student successfully enters login to Online Registration System
   3.1. include(Student Services)

Postconditions:

1. Login succesfully entered

Alternative flows:

1. None

The student login use case was one of the first designed. As this system is an online registration system for a university it is important and necessary to have student users able to log into the system so that they can use it.

Extension Use Case: Wrong Login

Brief Description:

- Student provides an invalid Login when attempting to verify their identity.

Primary Actors:

- Student

Secondary Actors:

- None

Segment 1 Preconditions:

1. The user tried their student login.
2. The user provided incorrect login information.

Segment 1 Main flow:

1. The system requests the user to attempt to correctly verify their identity once again.
2. The system checks the new login information provided by the student.
3. If the login information provided is correct
   3.1 The Online Registration System proceeds to the Student Services Use Case
4. Else
   4.1 The Registration System restarts this use case, requesting the user to enter a valid login once again.

Segment 1 Postconditions:

1. The new login information provided by the student is correct.

Alternative flows:

- None

An extension use case, the wrong login is very important for security. You can not have anyone logging into the system and behaviour was needed to make sure only those who have accounts are allowed to login.

Include Use Case Name: Student Services

Brief Description:

- Student Service hub for all Student services accessible by use case Student

Primary Actors

- Student

Secondary Actors

- None

Preconditions:

1. Student Login successfully entered.

Main flow:

1. Prompt screen so the student can view all the Student Service options
2. When student services are accessed all information is updated and acquired through Service Hub.
   2.1. Include (Service Hub)

Postconditions:

1. Update Student information
2. Update Service Hub

Alternative flows:

- None

This use case describes how the student actors interact with the system once they are logged in. It is here, from the service hub, that the actions of the students will take place. When designing the system it was important to have a central hub for all actions so that individual users will share one experience and it will be easy to create the UI. Furthermore, it allows for the easy creation of other hubs when necessary. All information in one place, secured from login, with easily presented information.

Use Case Name: Enroll

Brief Description:

- All the student to enroll into courses of their choice

Primary Actors

- Students

Secondary Actors

- None

Preconditions:

1. Student successfully logged in and Student Services promoted all options.

Main flow:

1. Student Clicks the enroll option
2. The system prompt a search where the student can input and find the courses of their liking and add it to their schedule

Postconditions:

1. Course is found
2. Course is successfully added to student schedule.

Alternative flows:

- None

Students need to be able to enroll in courses. Without enrollment why have an online registration system. This is one of the main goals of the system for the students.

Use Case Name: View/Edit Account Details

Brief Description:

- Student can view and edit all details pertaining to their account

Primary Actors

- Student

Secondary Actors

- None

Preconditions:

1. Student successfully logged in and Student Services promoted all options.

Main flow:

1. Student selects View/Edit Account Details options
2. The Registration System shows details such as Student username, Password change, Student email, Student number, Student Fees

Postconditions:

1. The requested details are showed on the screen or edited.

Alternative flows:

- **None**

Account balances, address, grades, etc. The students should be able to view this important account information from an online registration system. Although not 100% needed, this use case was included as we felt this information is very important for this kind of software. The more a student can do online, the easier it is for the school and the students to keep track of this personal account information.

Use Case Name: Swap

Brief Description:

- Allows the Student to swap one currently enrolled course with an non enrolled course

Primary Actors

- Student

Secondary Actors

- None

Preconditions:

1. Student successfully logged in and Student Services promoted all options.
2. Has already enrolled into a course

Main flow:

1. Student selects the swap option
2. Student then enters the new course of their choice and the currently course they wish to change
3. IF time constraints are fine
   3.1. Swap the course
4. ELSE prompt to choose a different course

Postconditions:

1. Changes are successfully made and schedule is updated.

Alternative flows:

- None

Similar to the add classes use case, the students need to be able to swap courses if desired. Again, this is important for an online registration system. You need to be able to control the courses you are registered in.

Use Case Name: View Enrolled Courses

Brief Description:

- View courses that the Student has enrolled in

Primary Actors

- Student

Secondary Actors

- None

Preconditions:

1. Student successfully logged in and Student Services promoted all options.
2. Has already enrolled into a course

Main flow:

1. Student chooses the View Enrolled Courses option
2. All courses and their details are shown
3. IF there isn't any courses added
   3.1. Prompt student to enroll into courses

Postconditions:

1. All course details are shown

Alternative flows:

- None

Viewing your registered courses. From here you can see your schedule and see if you have failed to enroll in a course you wanted or any course at all. This use case is important for the practical usability of the system. A student needs to be able to view this information.

Use Case Name: Drop

Brief Description:

- Student can unenroll from courses

Primary Actors

- Students

Secondary Actors

- None

Preconditions:

1. Student successfully logged in and Student Services promoted all options.
2. Has already enrolled into a course.

Main flow:

1. Student choose the Drop option
2. All courses that are currently enrolled are shown and you can drop any choice
3. IF no courses have been enrolled enrolled
   3.1. Prompt student to enroll into courses before dropping a course

Postconditions:

1. Students courses are successfully updated

Alternative flows:

- None

Dropping a course is another important aspect of an online registration system. Whatever the reason may be, if the student wishes to drop a course they should be able to do so and this use case makes that possible.

Use Case Name: Faculty Login

Brief Description:

- To initiate the access to one's faculty services account.

Primary Actors

- Faculty

Secondary Actors

- None

Preconditions:

1. Faculty must have a Ryerson Faculty Login

Main flow:

1. Prompt to enter faculty username and password
3. If faculty enters the wrong login
   2.1. Extend (Wrong Login)
3. Faculty successfully enters login to Online Registration System
   3.1. include(Faculty Services)

Postconditions:

2. Login succesfully entered

Alternative flows:

2. None

Secure login for the faculty group of actors. Security is very important when there is so much sensitive and personal information made available on the other side. Having a faculty login allows the system to ensure only valid faculty members can get into faculty services.

Include Use Case Name: Faculty Services

Brief Description:

- Faculty Service hub for all Faculty services accessible by use case Faculty

Primary Actors

- Faculty

Secondary Actors

- None

Preconditions:

1. Faculty Login successfully entered.

Main flow:

1. Prompt screen so the faculty can view all the Faculty Service options

When student services are accessed all information is updated and acquired through Service Hub.

2.1. Include (Service Hub)

Postconditions:

1. Update Faculty information
2. Update Service Hub

Alternative flows:

- None

Once login is successful, the faculty members need to be able to add and edit certain information pertaining to their faculty account. The faculty services allows this to happen. An online registration system needs specific information from a faculty member in order to list course specifics and have changes to the course characteristics if the faculty member deems it necessary.

Use Case Name: Post Grades

Brief Description:

- Post grades for the courses in which the Faculty teaches

Primary Actors

- Faculty

Secondary Actors

- None

Preconditions:

1. Successfully logged in Faculty account
2. Must have a course created

Main flow:

1. Faculty selects the Post Grades option
2. Faculty can post grades of each student for the class that they teach
3. IF class is not added
   3.1. Prompt faculty to Create a course

Postconditions:

1. Grade successfully posted

Alternative flows:

- None

It was important to allow grades to be posted online. This adds nice functionality to the registration system, making it more of an all around hub as opposed to strictly registering for courses.

Use Case Name: Edit Course Info

Brief Description:

- Edit course information that the Faculty teaches

Primary Actors

- Faculty

Secondary Actors

- None

Preconditions:

1. Successfully logged in Faculty Account
2. Must have a Course created

Main flow:

1. Select the Edit Course Info option
2. Edit the description of the course and its outline
3. IF no course exists
   3.1. Prompt Faculty to Create Course

Postconditions:

1. Course information successfully updated

Alternative flows:

- None

If there have been changes made to a course or the faculty member simply wished to update their course description they should be able to make those changes within the system. This also allows students to see those updates when they are looking to register for specific courses.

Use Case Name: Insert Class Limit

Brief Description:

- Allows the setting of a class size to limit the enrollment

Primary Actors

- Faculty

Secondary Actors

- None

Preconditions:

1. Successfully logged in Faculty Account
2. Must have a Course created

Main flow:

1. Select the Insert Class Limit option
2. Edit the Class size limit of each Course
3. IF no course exists
   3.1. Prompt Faculty to Create Course

Postconditions:

1. Class size successfully updated

Alternative flows:

- None

We felt that it was important for a faculty member to be able to set a class limit if they so desired. Sometimes that number will be large and sometimes it will be small. The choice is up to the faculty member, which will limit the number of spots in the class they are teaching.

Use Case Name: Remove Course

Brief Description:

- Remove a course that the Faculty is teaching

Primary Actors

- Faculty

Secondary Actors

- None

Preconditions:

1. Successfully logged in Faculty Account
2. Must have a Course created

Main flow:

1. Select the Remove course option
2. Remove any course that is listed in which the Faculty teaches
3. IF no course exists
   3.1. Prompt Faculty to Create Course

Postconditions:

1. Courses updated

Alternative flows:

- None

If a faculty member leaves for sabbatical or isn't teaching a course this semester they needed to be able to remove it from the registration system. Remove course allows them to do just that.

Use Case Name: Create Course

Brief Description:

- Create a course that the Faculty will teach

Primary Actors

- Faculty

Secondary Actors

- None

Preconditions:

1. Successfully logged in Faculty Account

Main flow:

1. Faculty creates a course they would like to teach

Postconditions:

1. Courses updated

Alternative flows:

- None

Recreating an old course or adding a brand new course to the available course options the school is offering is another feature of the faculty hub in the registration system. The faculty member needs to be able to add a course so that it is available for registration.

Include Use Case Name: Service Hub

Brief Description:

- Service hub for all information that Student Services and Faculty Services rely on which is accessible by the Registrar

Primary Actors

- Registrar

Secondary Actors

- None

Preconditions:

1. Registrar accesses Service Hub

Main flow:

1. Prompt screen so the registrar can view all the Service Hub options
2. When Service Hub is accessed all the Student Services and Faculty Services are regulated through hre

Postconditions:

1. Update Service information

Alternative flows:

- None

This use case allows the registrar to have access and view all of the information in the system. This was important so that someone from the registrar's office can overwrite some information if needed or make exceptions for a student or faculty, which is why they have access to both of those services as well as their own service hub.

Use Case Name: Add Classes

Brief Description:

- Add courses into the database so Faculty and Students can find classes to enroll and teach

Primary Actors

- Registrar

Secondary Actors

- None

Preconditions:

1. Registrar accesses service hub

Main flow:

1. Select the Add Classes option
2. Add Classes as well as the Course Fee and Details
   2.1. Extends(Course Fee)
   2.2. Extends(Course Details)

Postconditions:

1. Classes Successfully added

Alternative flows:

- None

Just like the faculty members, the registrar is also able to add classes to the database. Having this in the system allows for the registrar to take control of class additions so that the faculty members do not need to do it themselves if so desired. This also allows the setting of course fees and details.

Extension Use Case: Course Fee

Brief Description:

- Provides price of the Course

Primary Actors:

- Registrar

Secondary Actors:

- None

Segment 1 Preconditions:

1. Registrar accesses service hub
2. A Class is Created

Segment 1 Main flow:

1. Registrar adds the cost of the Course

Segment 1 Postconditions:

Course Fee successfully updated

Alternative flows:

- None

The registrar, not the faculty member, is in charge of setting the course fees. This ensures a faculty member cannot set course fees for their own benefit (or personal gain), instead giving this responsibility to the registrar office to ensure consistency throughout the system.

Extension Use Case: Course Details

Brief Description:

- Update details about a course

Primary Actors:

- Registrar

Secondary Actors:

- None

Segment 1 Preconditions:

1. Registrar accesses service hub
2. A Class is Created

Segment 1 Main flow:

1. Registrar adds details to the courses description

Segment 1 Postconditions:

1. Course Details successfully updated

Alternative flows:

- None

Similar to the faculty members, the registrar is also able to edit the course details. This was important for the registration system for the same reasons that they registrar is able to do everything the other actors are. They can help out or make the changes on their own without needed to bother other actors in the system.

Use Case Name: Create Faculty Accounts

Brief Description:

- Create accounts for Faculty users

Primary Actors

- Registrar

Secondary Actors

- None

Preconditions:

1. Registrar accesses service hub

Main flow:

1. Registrar creates Faculty accounts so the Faculty can login and access information

Postconditions:

1. Account successfully created

Alternative flows:

- None

The system needed a way of making sure not just anyone can login. The registrar is responsible for creating the faculty accounts and their permissions so that they can access the information in the system specific to them.

Use Case Name: Release Grades

Brief Description:

- Release grades so students can view their marks

Primary Actors

- Registrar

Secondary Actors

- None

Preconditions:

1. Registrar accesses service hub
2. Grades must already be posted

Main flow:

1. Registrar releases the grades that have been posted from the Faculty

Postconditions:

1. Grades are posted for Student viewing

Alternative flows:

- None

Although the faculty members post the grades, the registrar is the one who releases the grades. The difference here is that when grades are posted to the system they student is unable to see them. Only once the registrar has released the grades are they visible to the students in the system.

Use Case Name: Remove Courses

Brief Description:

- Remove courses that were once created

Primary Actors

- Registrar

Secondary Actors

- None

Preconditions:

1. Registrar accesses service hub
2. A Class is Created

Main flow:

1. Remove courses that were once created
2. IF no course is created
   2.1. Prompt Registrar to create courses

Postconditions:

1. Courses have been updated

Alternative flows:

- None

Just like the faculty members, the registrar needed to be able to remove courses from the system. No matter the reason, if a course needs to be removed the registrar needs to be able to do that as well.

Use Case Name: Create Student Logins

Brief Description:

- Create logins so Student users can access information

Primary Actors

- Registrar

Secondary Actors

- None

Preconditions:

1. Registrar accesses service hub

Main flow:

1. Create an account for student users to enable them to login and access information

Postconditions:

1. Student Login has been created

Alternative flows:

- None

New students need to be added to the system. This is how the registrar does that. A new student and unique information is generated and the registrar adds them to the system. This allows new students to access the registration system with unique logins and access only their information.

Use Case Name: Enroll Students

Brief Description:

- Enroll Students into the course that they added

Primary Actors

- Registrar

Secondary Actors

- None

Preconditions:

1. Registrar accesses service hub
2. A Course is Created

Main flow:

1. Enroll student into the course of their choosing

Postconditions:

1. Successfully update the course of the student's choosing

Alternative flows:

- None

If a student was unable to enroll in a class for some reason, we wanted the system to have a feature to override that. The registrar is able to enroll a student in a course. They can make exceptions to the systems enroll requirements if they deem it necessary. This use case is important for any registration system.

Use Case Name: Unenroll Students

Brief Description:

- Remove Student from a course that they are enrolled in

Primary Actors

- Registrar

Secondary Actors

- None

Preconditions:

1. Registrar accesses service hub
2. A Class is Created
3. Students must be previously enrolled

Main flow:

1. Remove student from the course that they are currently enrolled in

Postconditions:

1. Update the course in which the student has been removed from

Alternative flows:

- None

If a student is unable to drop a course, or needs to be forcefully removed from it in the system the registrar is able to do that.

Use Case Name: Check Class Compatibility

Brief Description:

- Check if the class the student request is compatible with their time slot

Primary Actors

- Registrar

Secondary Actors

- None

Preconditions:

1. Registrar accesses service hub
2. A Class is Created

Main flow:

1. Check if the time slot of the students requested course fits
2. IF the class fits
   2.1. Enroll student
   2.2 Else
      2.2.1. Revoke request

Postconditions:

1. Update the compatibility of course

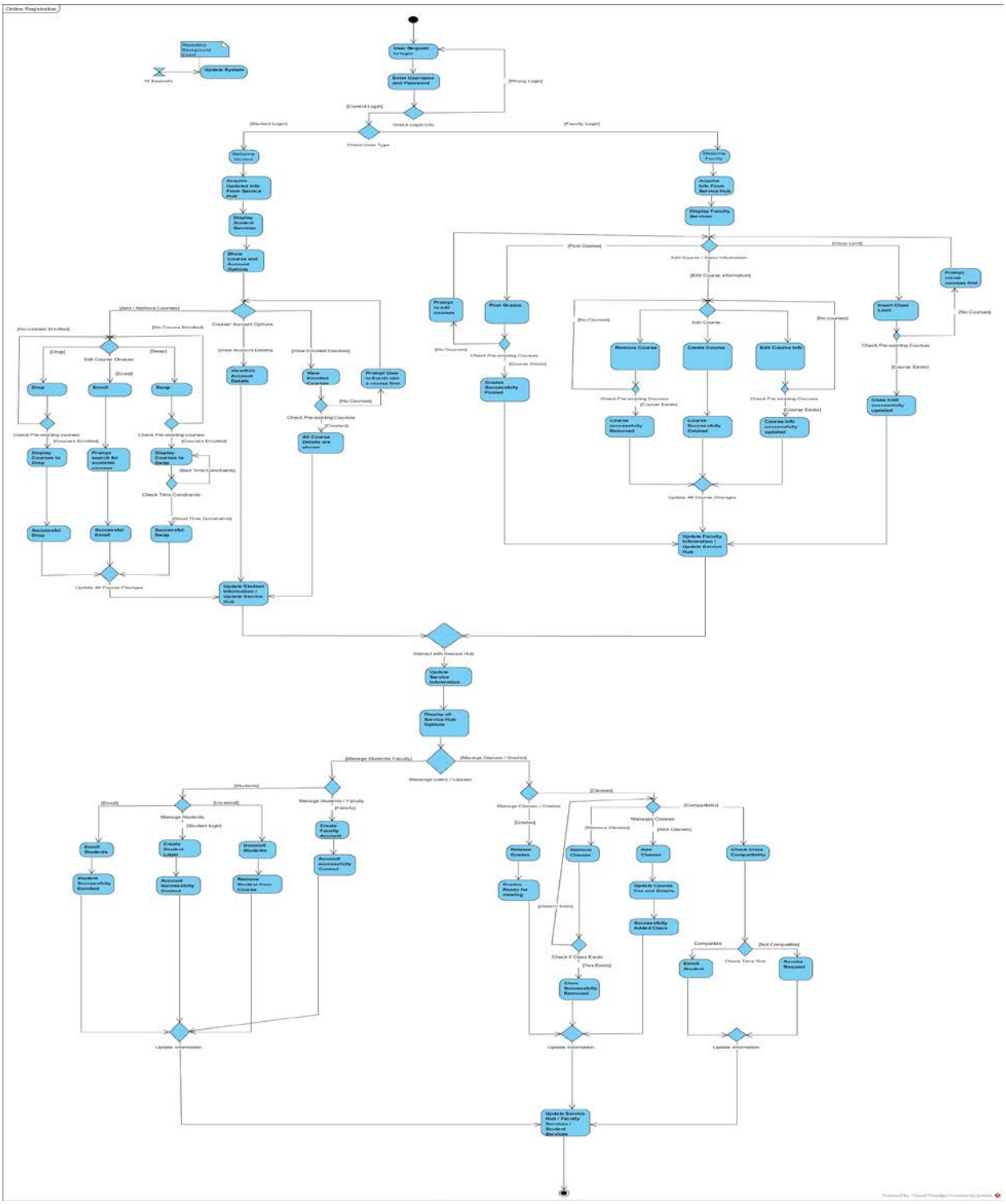
Alternative flows:

- None

When helping a student enroll in a class the registrar can’t just add them. They needed to be able to check that that student can actually attend the class and that it will fit into their schedule without any course conflicts.

Phase 2 - Activity Diagram and High Level Architecture

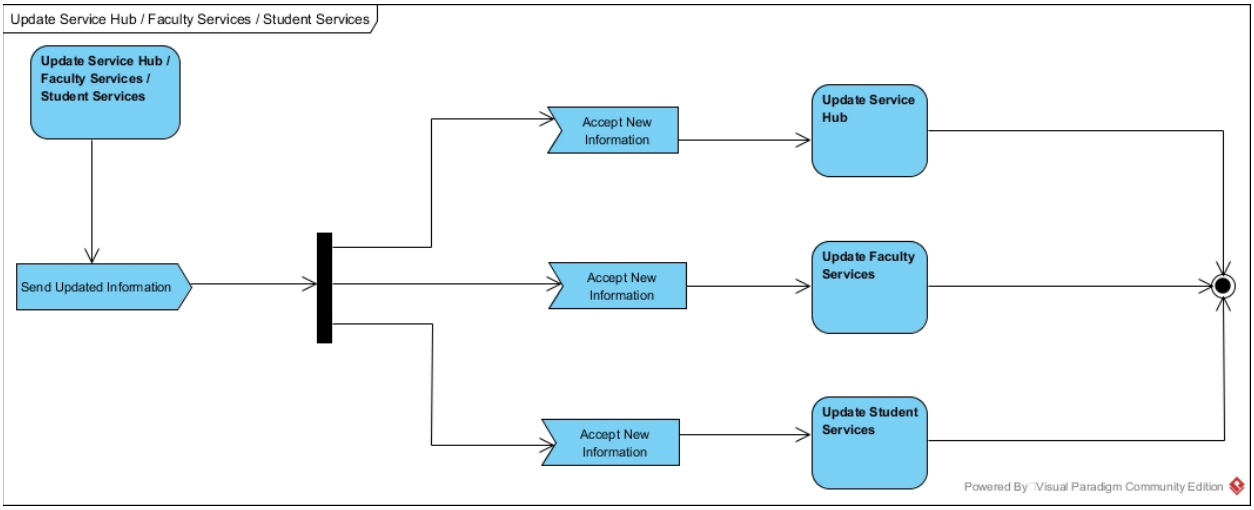
Activity Diagram



The activity diagram was designed off of the requirements gathering. It was decided from these how they system components would interact. This is a graphical representation of the components in the system and the paths of how the different goals are achieved. For a few of the systems sub diagrams were made to avoid over clutter and give more detail for specific activities.

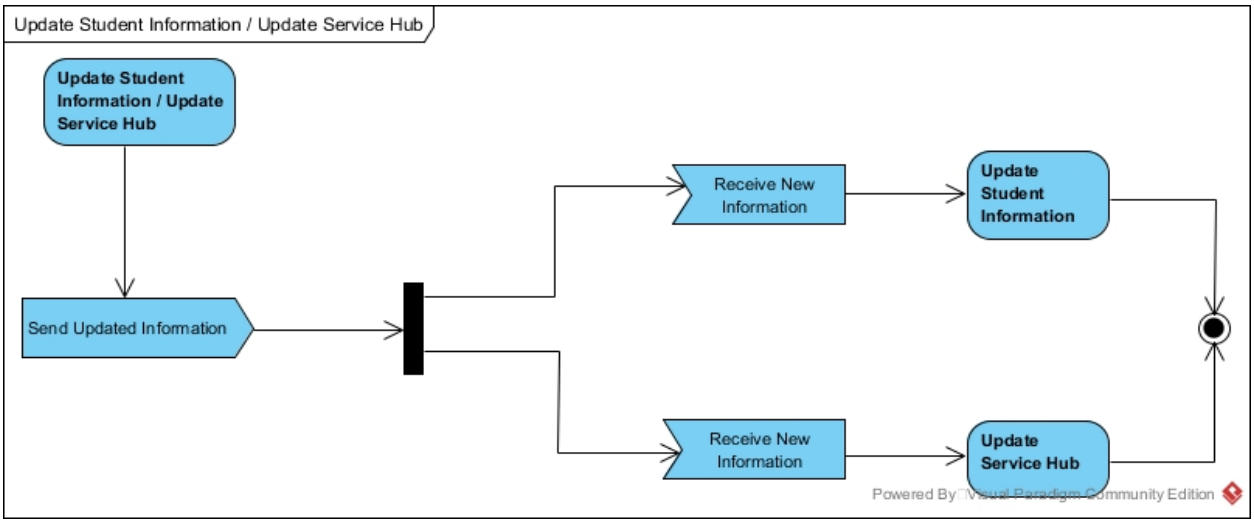
Sub Diagrams

Update Service Hub

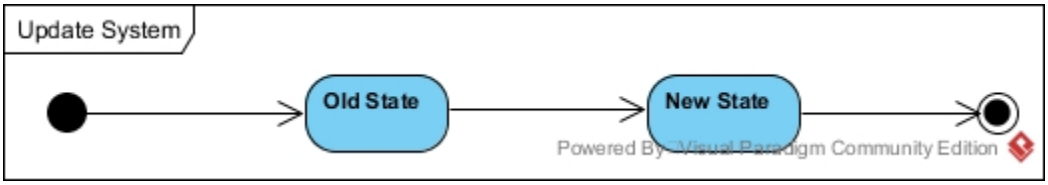


The update service hub, faculty services, and student services activity diagram. This simple activity diagram describes how the update goals are achieved for the system.

Update Student Information

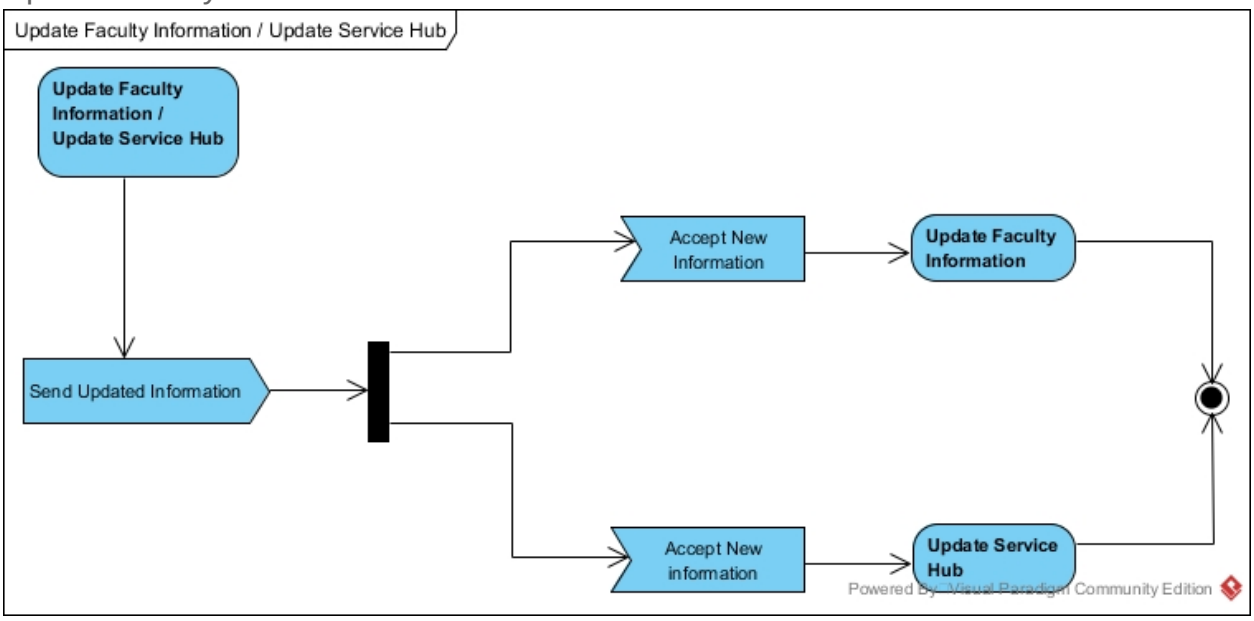


This activity diagram shows how student information gets updated in the system as well as updating the hub information and reaches its goal.



The goal of updating the system is shown here. This is a repeating action of keeping track of what was changed and making a new state of the system as its goal.

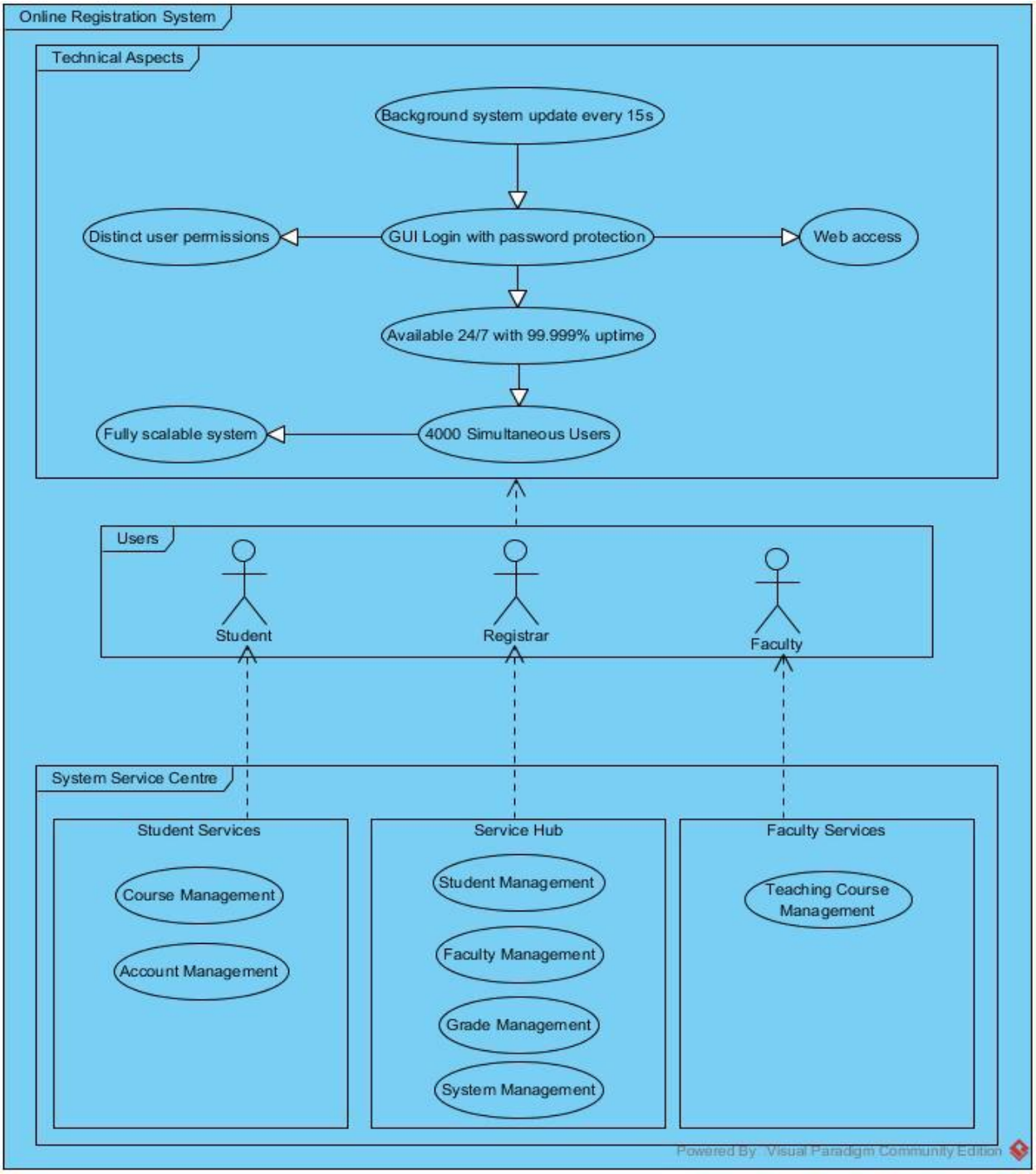
Update Faculty Information



This subdiagram shows the process of reaching the goal for updating faculty information and in the service hub.

Marketecture Document

Diagram



An overview of the system. This diagrams shows a very high level view of the system. Which components are related to which actor and the important technical aspects of the system someone should know at a glance.

Description

The online registration system has been designed from the ground up while taking some guidance from the current Ryerson University online registration system. Specific requirements were taken into account to assure a reliable and complete system that students, faculty, and the registrar's office can all use.

The marketecture diagram seen above provides a user overview of the system. It briefly shows how each user can interact with the system and some of the system specifications. Below is a more detailed description of the important system requirements and how they pertain to the users in the diagram above.

The important components of the system are its usability, reliable, security, and scalability. The registration system will be built to have a very easy to navigate user interface. This will ensure a smooth user experience and allow the different users to operate the system efficiently. The system is built around some important requirements, which are being implemented at the start, to allow for the basic operations a system like this would need. These include the student services, faculty services and the registrar services. The students are able to view, add, and swap their courses, a very important component of a system such as this. They can also manage their personal information including financial services. The faculty members are able to manage the courses that they are teaching. They are able to manage the courses they are already teaching by setting enrollment limits (if applicable), as well as the course description. On top of these capabilities the faculty members are able to add new courses they are going to teach or remove courses they are no longer teaching. Finally the faculty members are able to upload and edit the individual grades of students in their specific classes. The registrar is able to do a lot more than either of the other two users. They are able to get into both the student and faculty accounts and do changes on their behalf. Not only are they able to make changes for existing users but it is their responsibility to add new students and faculty to the system as needed. They maintain the system, ensuring accurate information, as well as release final grades once the semesters are completed.

The system itself has a secure login. Each student, faculty member, and registrar has a unique login and can only view/manage data related to their account type and the hub that they login to. This is very important for our system as we do not want students to be able to manage their own grades or manipulate sensitive information of other students and specific course information, which is only accessible for faculty and registrar.

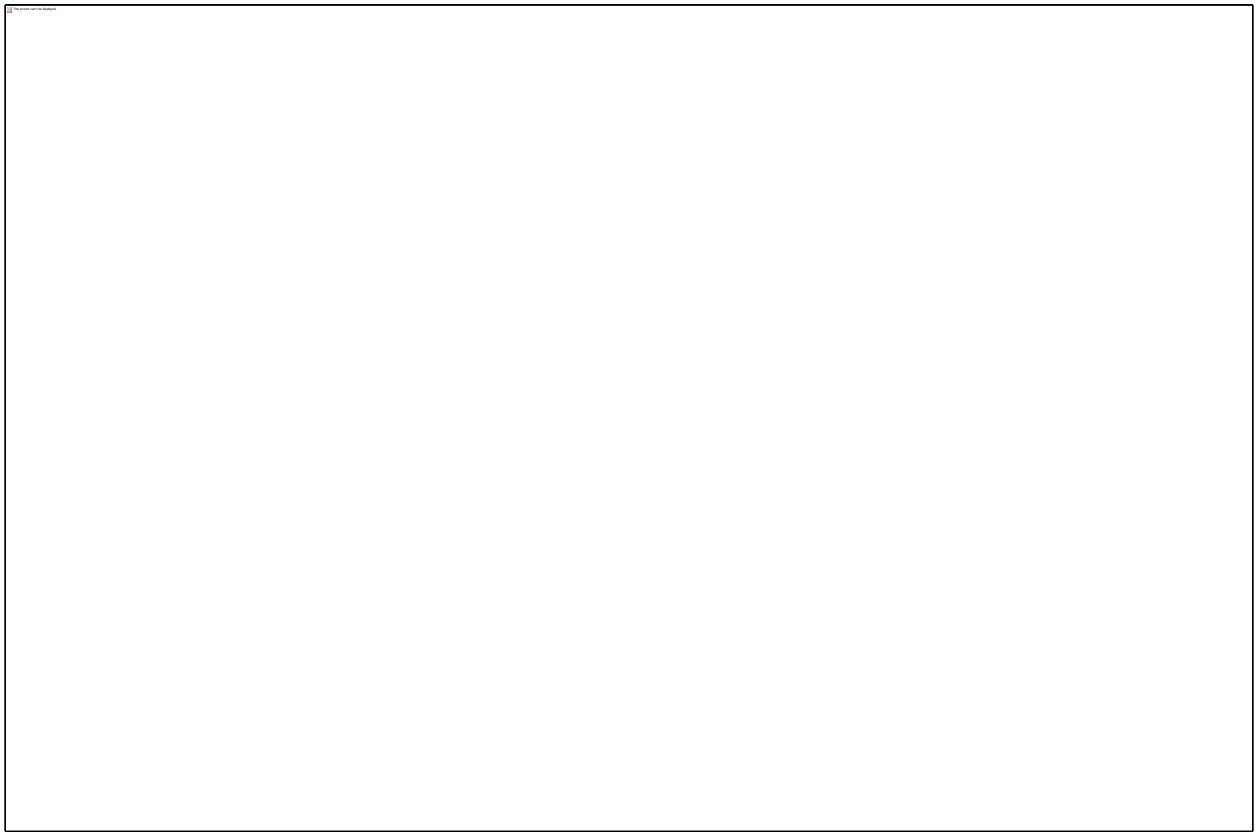
This system is also a scalable one. It is currently set-up to handle 4000 simultaneous users but has the capability to be scaled larger if needed. This is feasible with the

hardware requirements and just scaling up the systems servers, which strive for a 99.999% uptime to ensure constant access for the users through various web applications.

Phase 3 - More Architecture

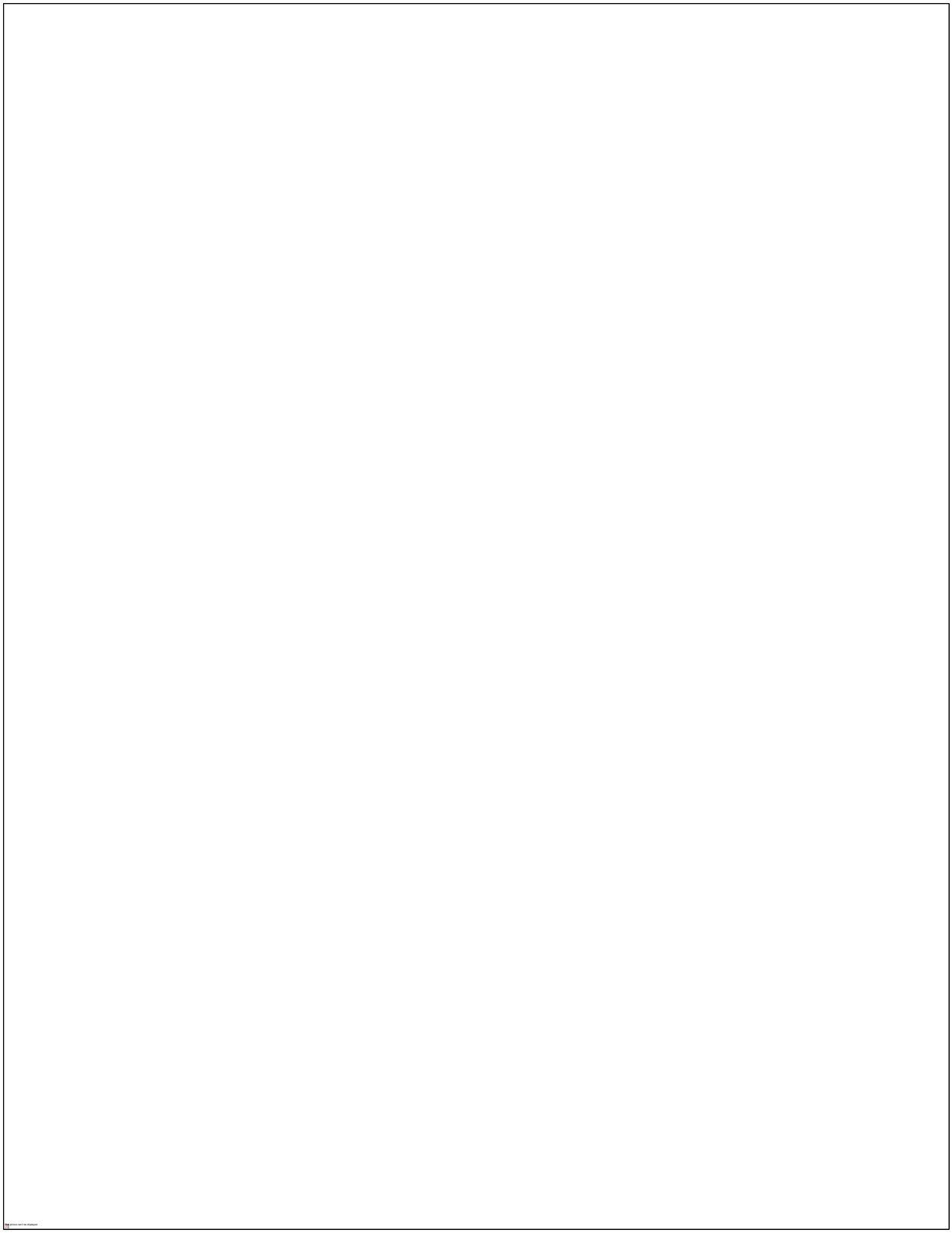
Component Diagrams

High Level Architecture



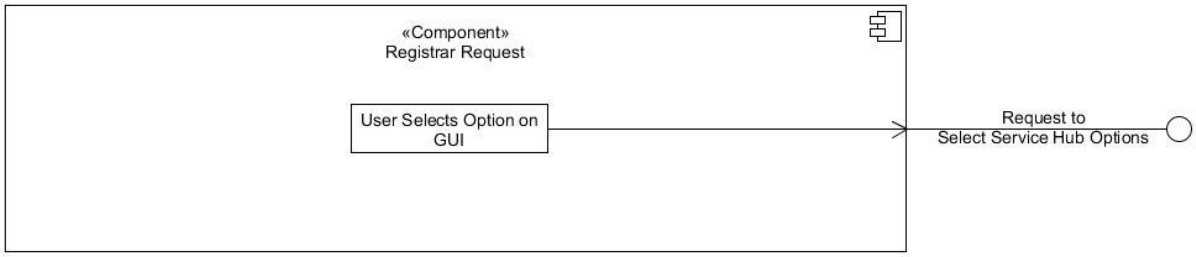
A high-level view of the entire system. There are three components which act upon a central service hub database. The three components are the three actors in the system, faculty, students, and registrar. Each of these actor groups are able to interact with the system in a way unique to the group they are apart of and have specific components within each of their systems explaining exactly how. However, each are able to take information from the system, view it, and update it if required at which point the updated information is propagated through the entire system so all users and actor groups have up to date information.

Registrar - Mid Level Architecture



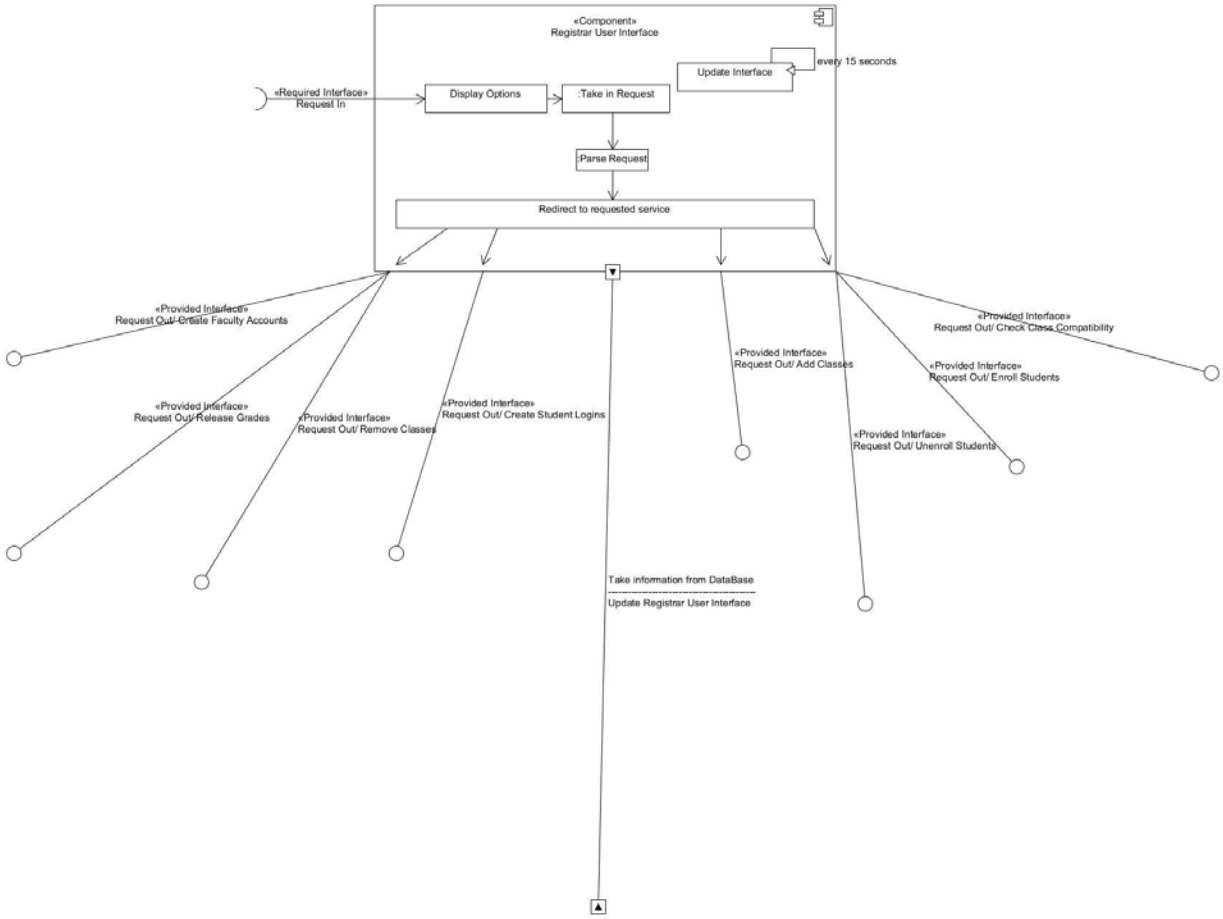
Registrar - Sub Diagrams

Request



The Registrar Request component is one of the most modular parts of this system. The system depends on Registrar Requests in order to to process functions such as the Registrar User Interface, which is further linked to Create Faculty Accounts, Release Grades, Remove Classes, Create Student Logins, Add Classes, Unenroll Students, Enroll Student, and Check Class Compatibility. The Request sent out by this component is processed within the interface and sent down so more specific components can handle the services. Each request sends out information that changes the contents within the Service Hub DataBase, that is later accessed by other components such as the Student Service Hub and Faculty Service Hub.
These Requests demonstrate the Web accessibility. The interactions between the Registrar Request and the rest of the system is expressed through the “Request To” interface, which sends out information and packets to the corresponding Interface that the user has permissions for, expressing the privacy and security within our system. A User, Keyboard, and a simple internet connection is enough to fulfill the hardware expectations of this component.

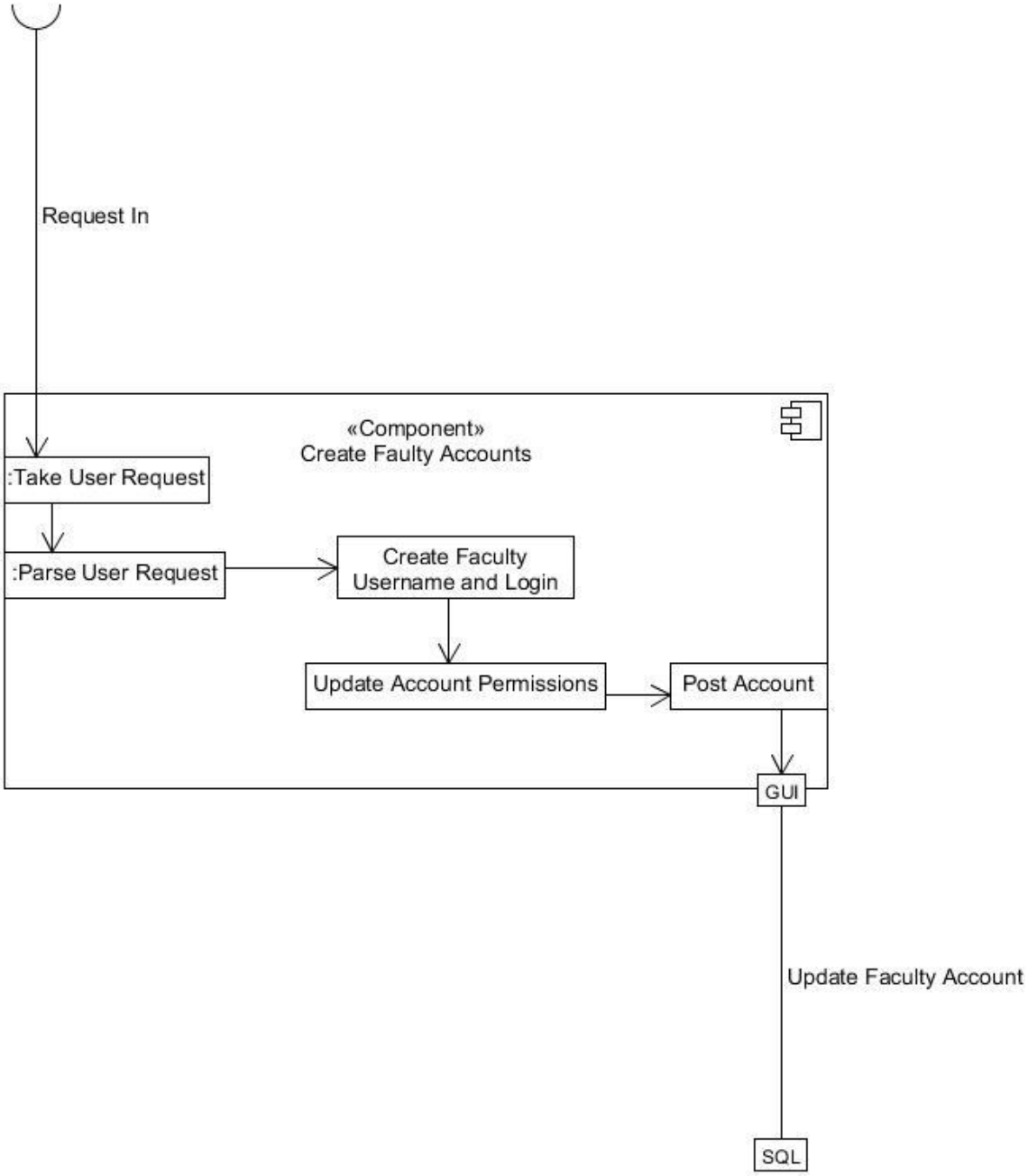
User Interface



The Registrar User Interface Component is the main graphical User Interface that the user will be viewing and interacting with. Here all the options that the Registrar needs to make decisions and regulate online enrolment will be possible. This component depends on the “Registrar Request” component so options can be selected and decision can be made. The availability of the Registrar user interface component is a 24/7 availability time because students always request assistance. This component is simple with all options being visible right on the screen, no need for hamburger menus.

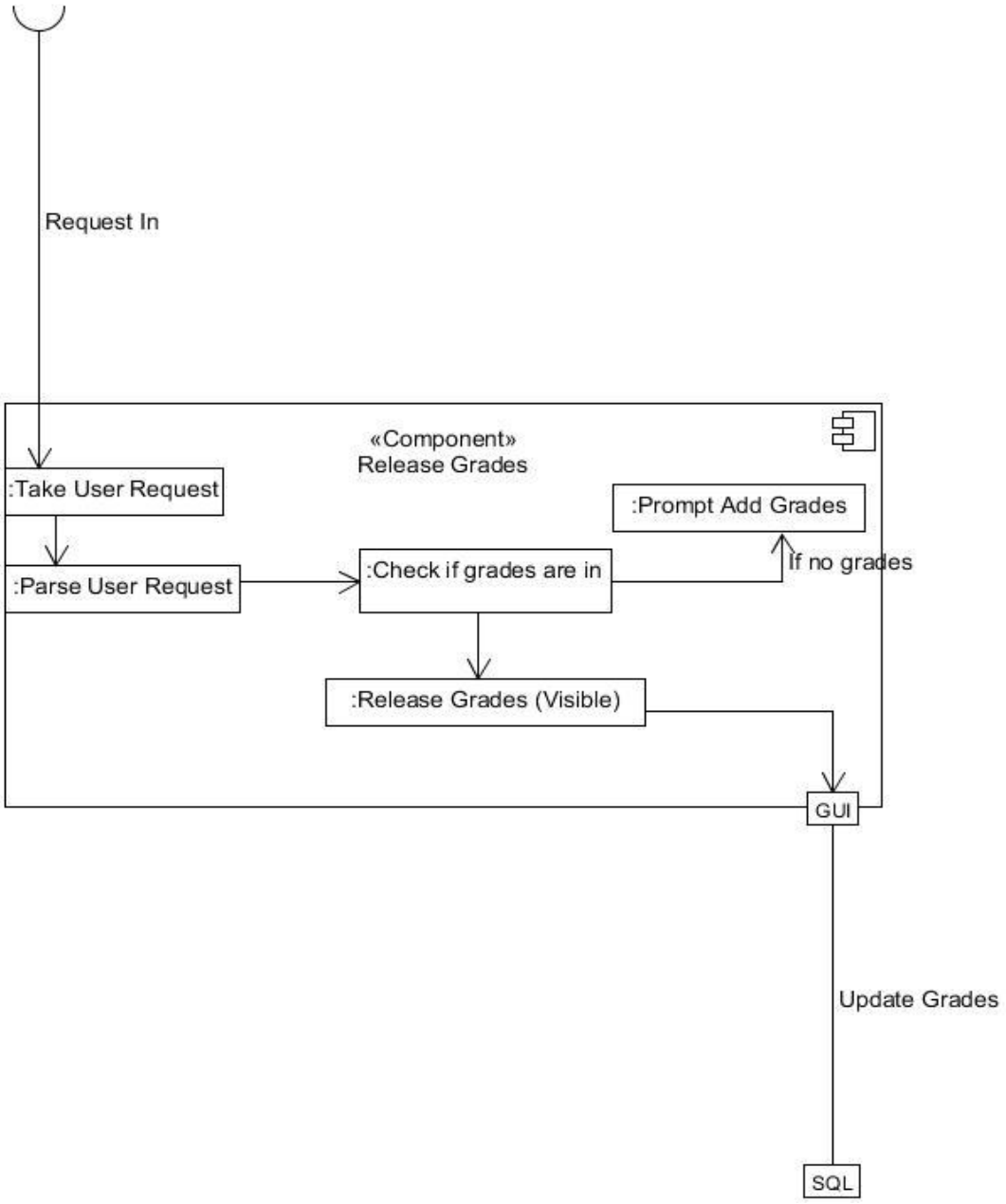
Being the main component of the Registrar Service Hub, multiple “Request out” interfaces are sent in order to manage the Create Faculty Accounts, Release Grades, Remove Classes, Create Student Logins, Add Classes, Unenroll Students, Enroll Student, and Check Class Compatibility components.

Create Faculty Accounts



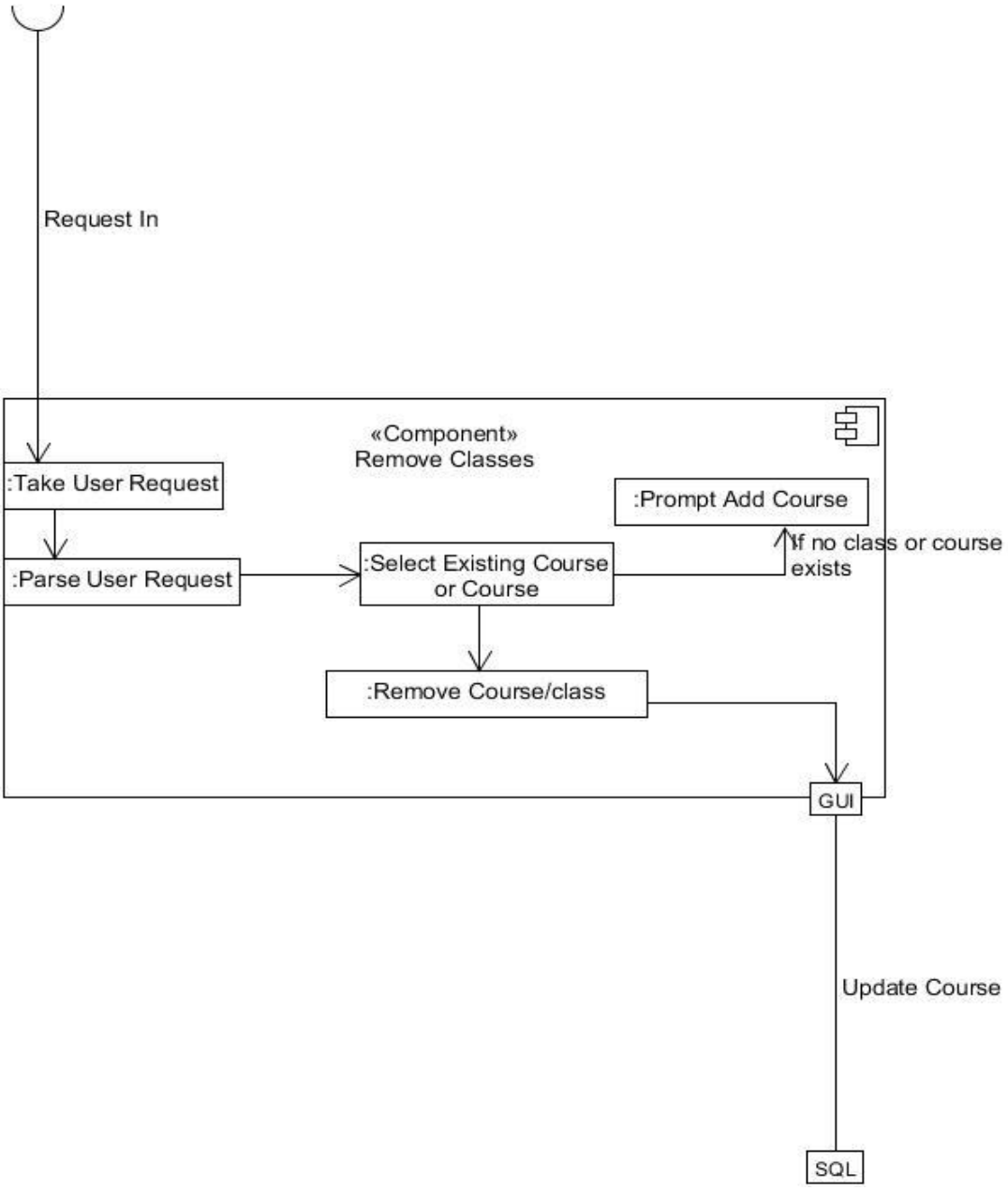
The component “Create Faculty Accounts” is very quick and responsive with transaction times between the Service Hub Database and the component occurring easily under 10 seconds where faculty accounts are made. Here, there is a dependency on the “Registrar User Interface” component, which is apparent based on the “Request in” interface. This component is depicted as an optional window within the “Registrar User Interface”, GUI.

Release Grades



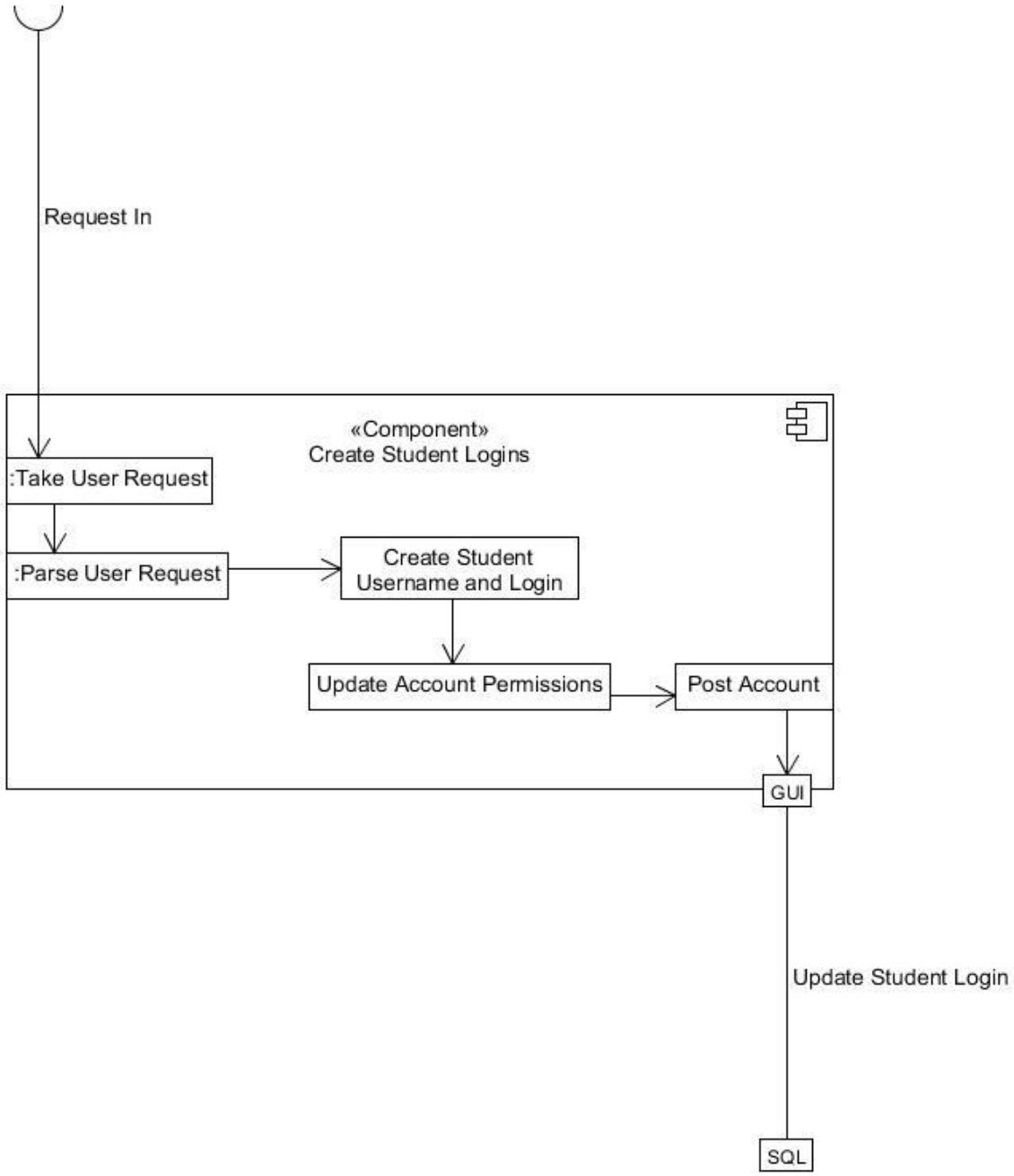
The “Release Grades” component is very quick and responsive with transaction times between the Service Hub Database and the component occurring easily under 10 seconds where grades that have been posted by the faculty have officially been released and updated for the students to see. Here, there is a dependency on the “Registrar User Interface” component, which is apparent based on the “Request in” interface. This component is depicted as an optional window within the “Registrar User Interface”, GUI. This component is specific to the Registrar to ensure security within the system.

Remove Classes



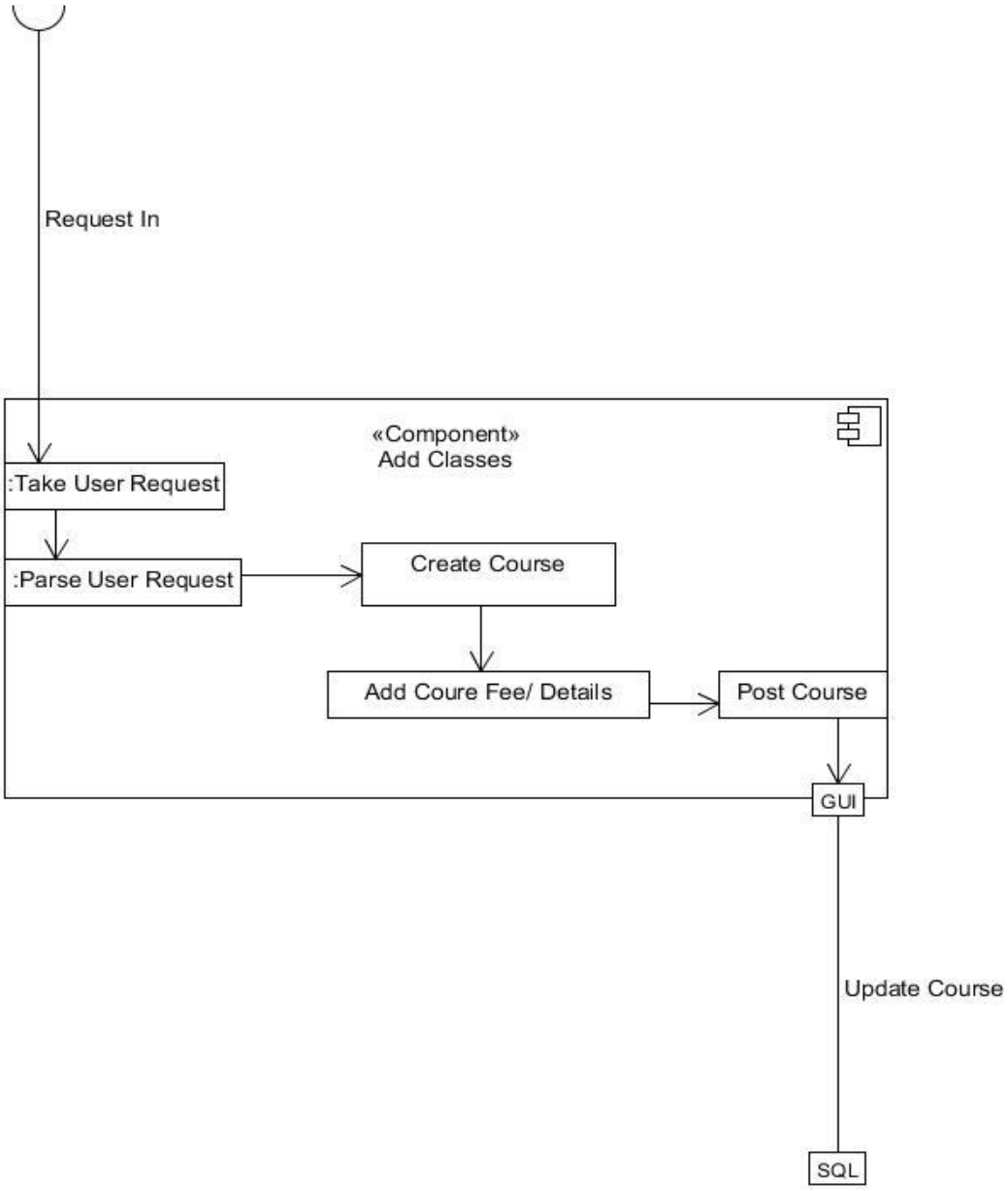
The “Remove classes” component as all components in fact are very quick and responsive where classes are removed from the accessibility of students and faculty. Here, there is a dependency on the “Registrar User Interface” component, which is apparent based on the “Request in” interface. This component is depicted as an optional window within the “Registrar User Interface”, GUI. Only the Registrar can manage the removal of courses and classes from all parts of this system. All finalized actions are updated to the database which it affects the rest of the system.

Create Student Logins



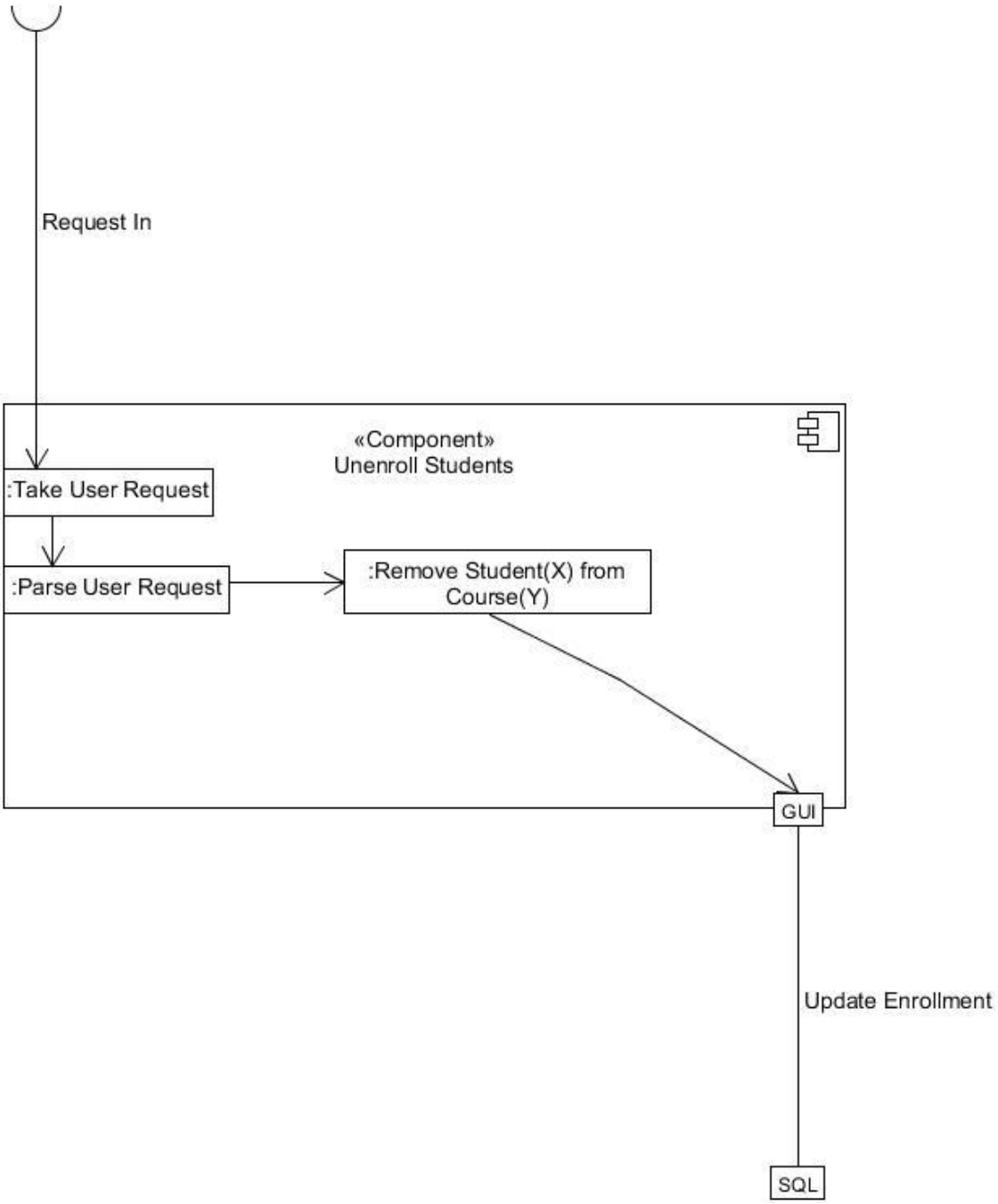
The “Create Students logins” is another exclusive component that is only manageable by the Registrar. Again, all components are very quick and responsive where student accounts and permissions are instantly updated for immediate use. Here, there is a dependency on the “Registrar User Interface” component, which is apparent based on the “Request in” interface. This component is depicted as an optional window within the “Registrar User Interface”, GUI. All finalized actions are updated to the database which it affects the rest of the system.

Add Classes



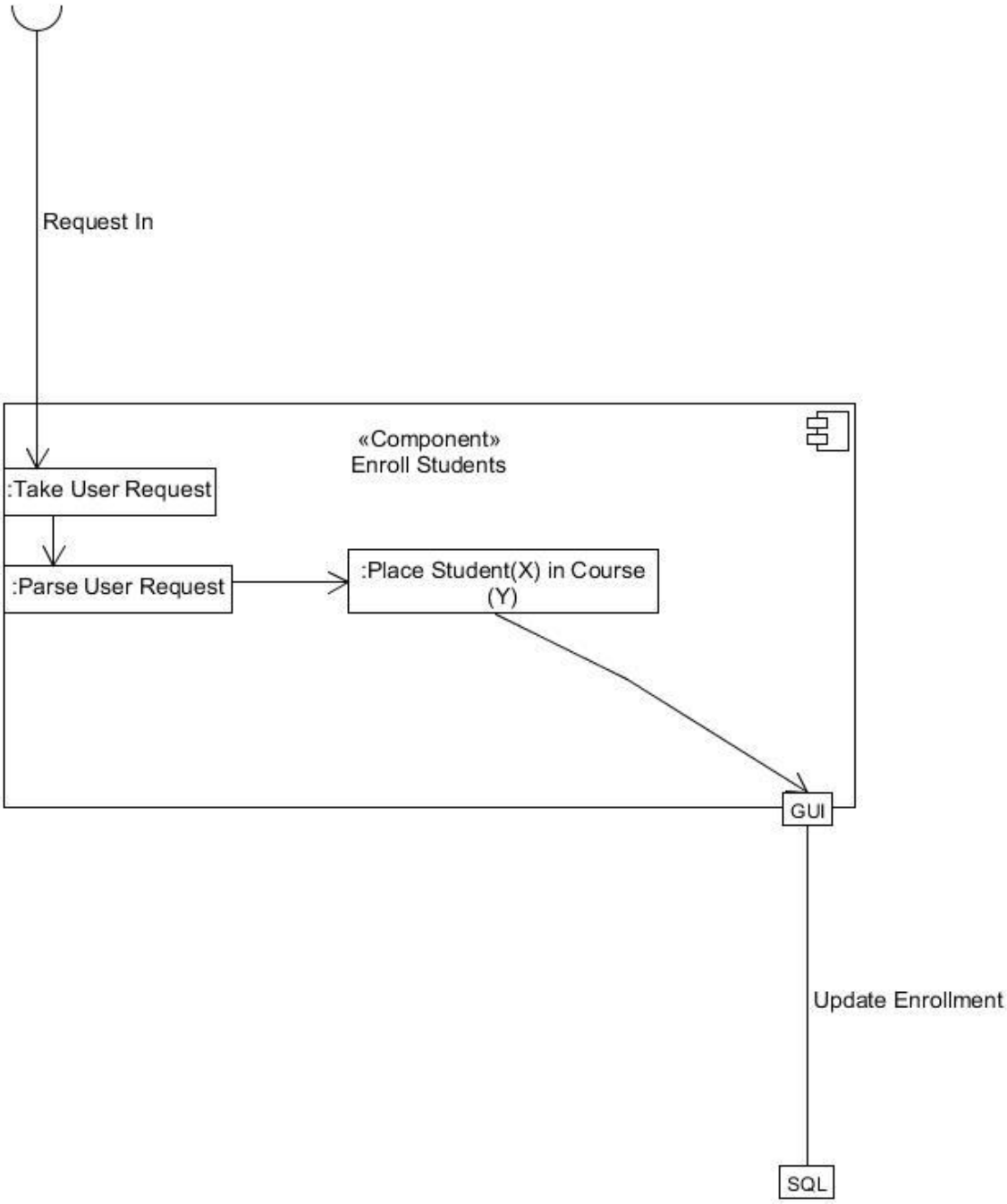
The “Add Classes” is another exclusive component that is only manageable by the Registrar. Again, all components are very quick and responsive where class and course information are instantly updated for immediate use . Here, there is a dependency on the “Registrar User Interface” component, which is apparent based on the “Request in” interface. Within this component the Registrar has the ability to create courses as well as add fee’s and other details to allow for the use of faculty and students. This component is depicted as an optional window within the “Registrar User Interface”, GUI. All finalized actions are updated to the database which it affects the rest of the system.

Unenroll Students



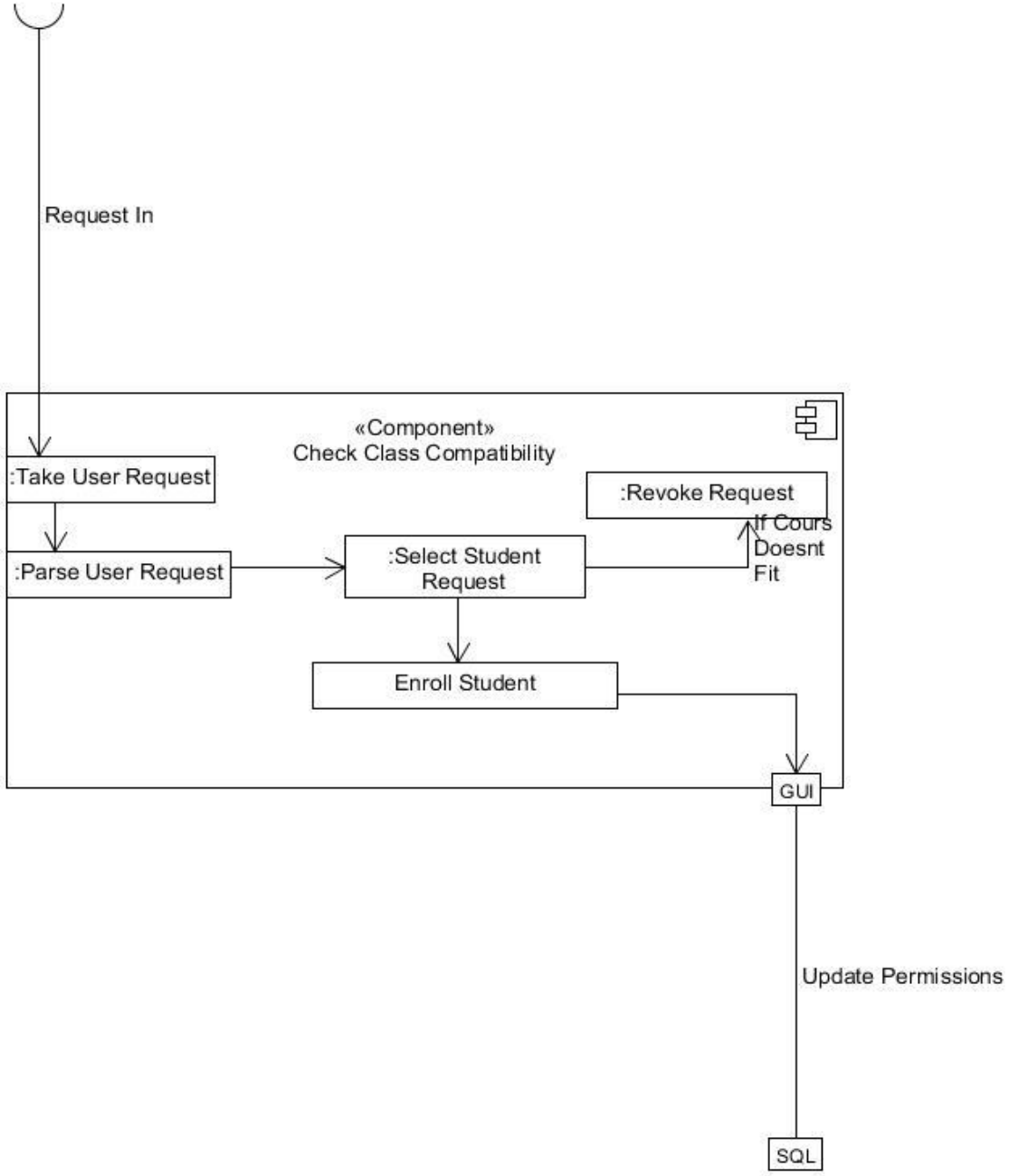
The “Unenroll Students” is another exclusive component that is only manageable by the Registrar. Here, there is a dependency on the “Registrar User Interface” component, which is apparent based on the “Request in” interface. Within this component the Registrar has the ability to remove students from courses that they are enrolled in. This component is depicted as an optional window within the “Registrar User Interface”, GUI. All finalized actions are updated to the database which it affects the rest of the system.

Enroll Students



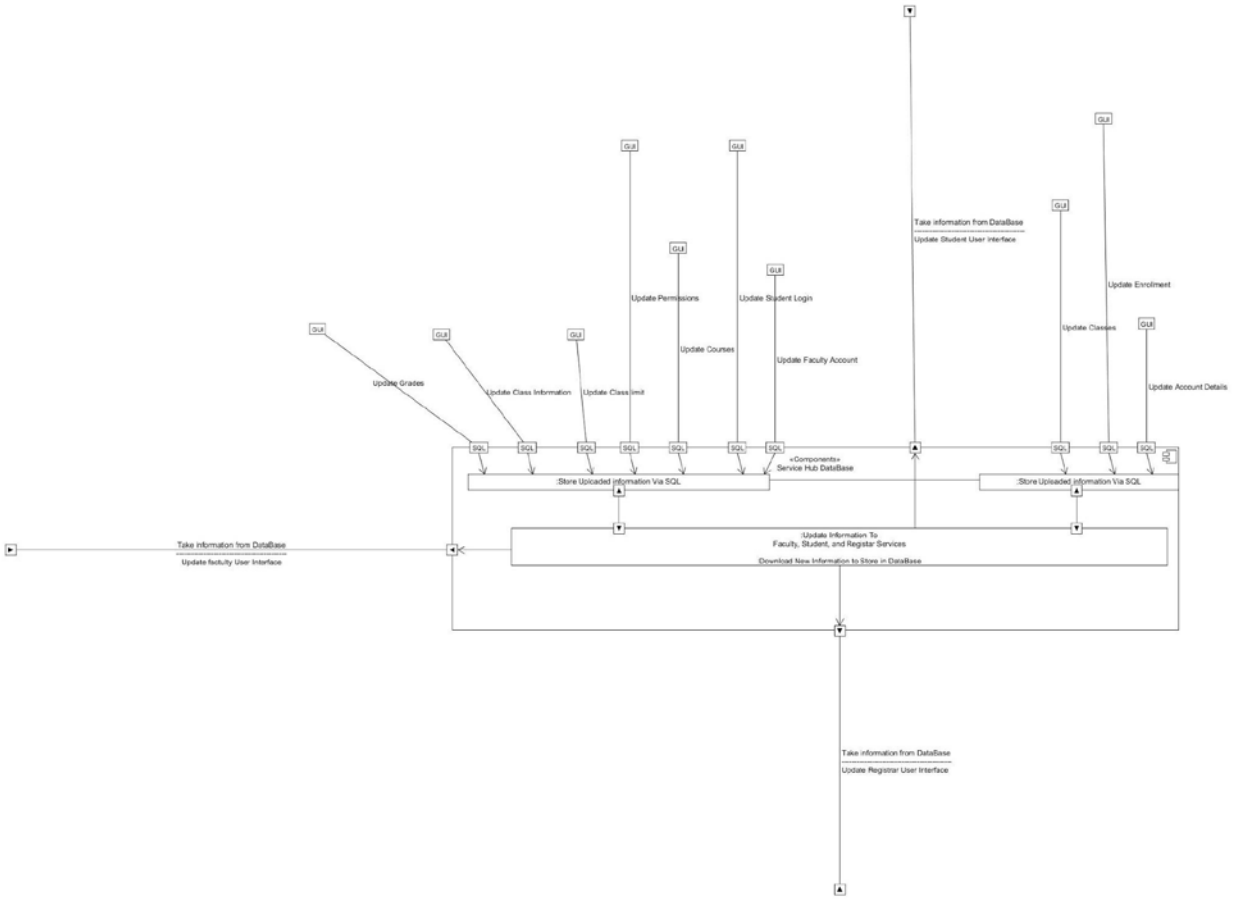
The “Enroll Students” is another exclusive component that is only manageable by the Registrar. Here, there is a dependency on the “Registrar User Interface” component, which is apparent based on the “Request in” interface. Within this component the Registrar has the ability to add students to courses that they request to be enrolled in. This component is depicted as an optional window within the “Registrar User Interface”, GUI. All finalized actions are updated to the database which it affects the rest of the system.

Check Class Compatibility



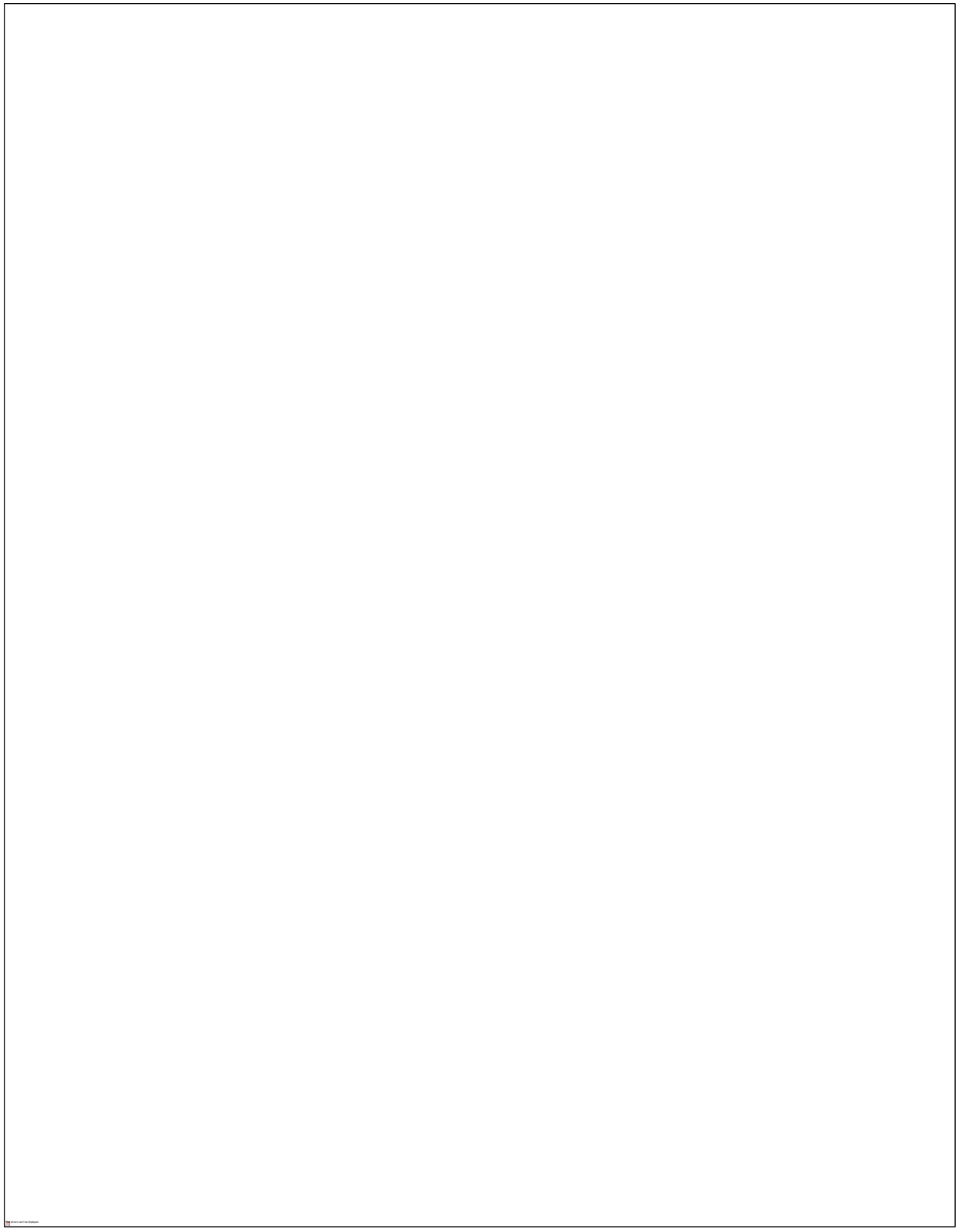
The “Check Class Compatibility” is another exclusive component that is only manageable by the Registrar. Here, there is a dependency on the “Registrar User Interface” component, which is apparent based on the “Request in” interface. Within this component the Registrar has the ability to check the request of students and either approve or revoke them. It is ensured that students are allowed into the appropriate classes and courses. This component is depicted as an optional window within the “Registrar User Interface”, GUI. All finalized actions are updated to the database which it affects the rest of the system.

Service Hub DataBase



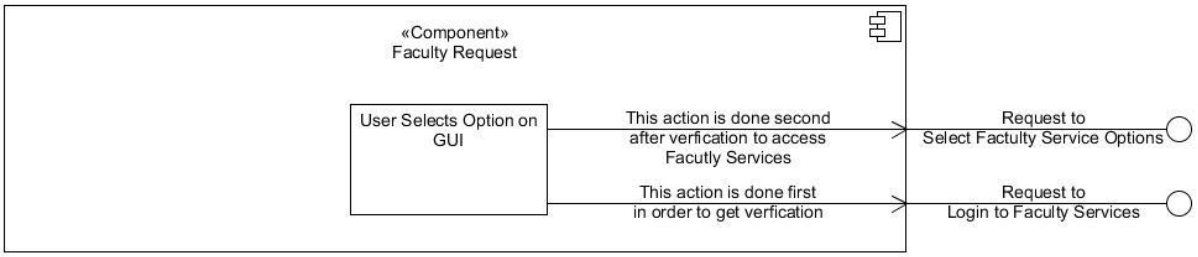
The “Service Hub DataBase” Component depends on all components to function, as do the components depend on the Database in order to get the correct information. Here there are no “Request in” interfaces, but “GUI to SQL” interfaces showing that all the formation stated within the components are sent and stored in the database through SQL. Arrows also depict that constant uploading and downloading of information between the Student Services, Faculty Services, and Registrar Services ensures the most recent and correct content. All users have indirect access to this file through tightly managed and specific actions that are unique to each user of the system. Each user has their own permissions that enable them to do certain actions to ensure the safety and privacy of the information downloaded and stored on the database. The database can handle over 400,000 upload and download processes at any given moment, which allows all resources to be accessible at any time.

Faculty - Mid Level Architecture



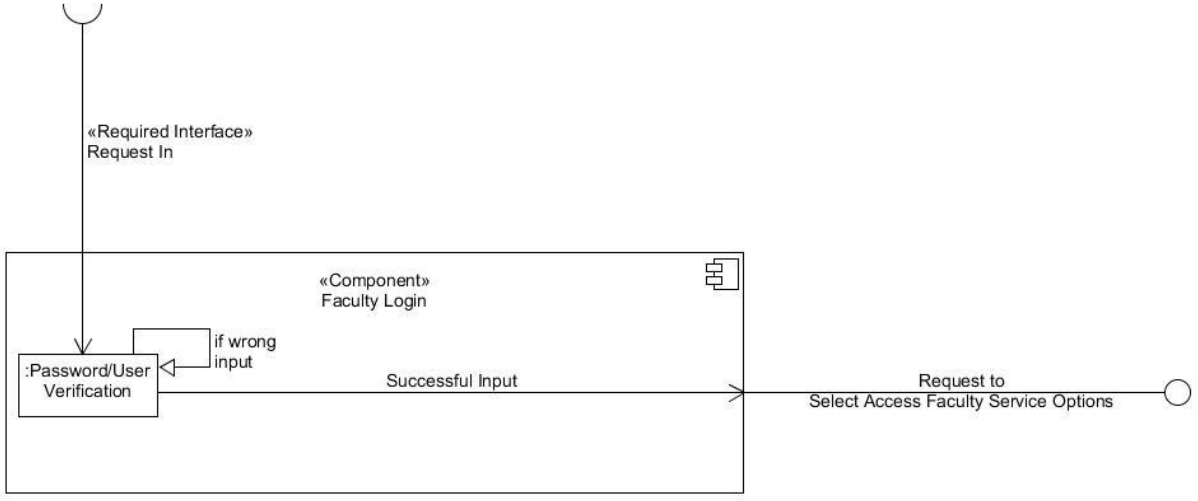
Faculty- Sub Diagrams

Request



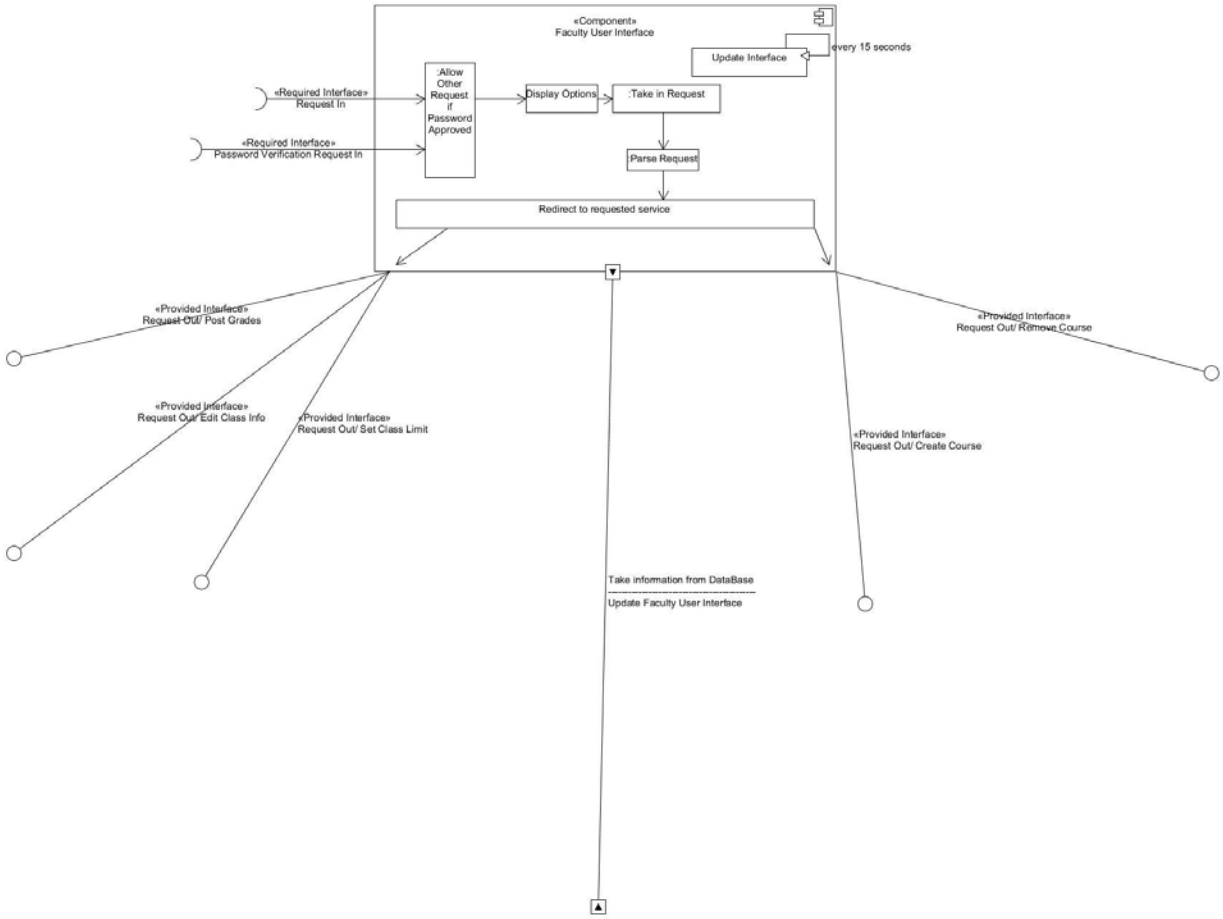
“Faculty Request” is a component only accessible by the user Faculty. Here Requests that are identified as Faculty permission requests are sent out throughout the system giving the user access to specific rights unique to other users on the system. At this component the user is allowed to firstly, sign in, because all the service cannot be accessed without a correct user login and then the second action of sending out service request to the Faculty User Interface can be made. The components “Faculty User interface”, Faculty Login, Post Grades, Edit Class Info, Set Class Limit, Create Course and Remove Course components, depend on the Faculty Request to function. All the “Request in” interfaces are fulfilled by the “Request to” initially send by the user when a GUI option is selected.

Login



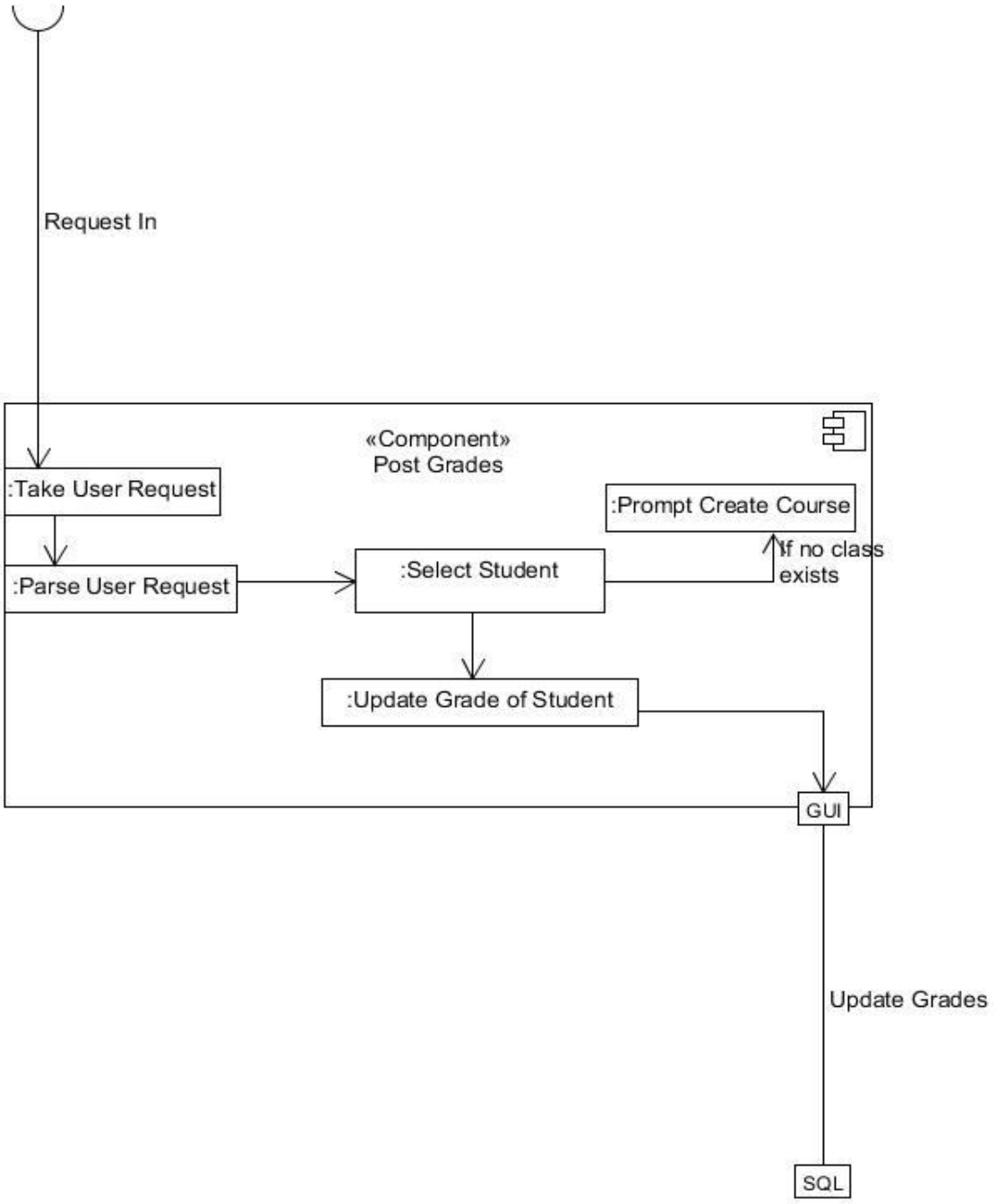
“Faculty Login” is a component that depends on the “Faculty Request” component, this is demonstrated through the “Request In” interface. Within this component, the user password and login is verified for the use of security. This ensures that the faculty services and permissions are used by the correct user. After this component successfully verifies the user, it sends a “Request to” interface to the “Faculty User Interface ” to display the services needed by the user.

User Interface



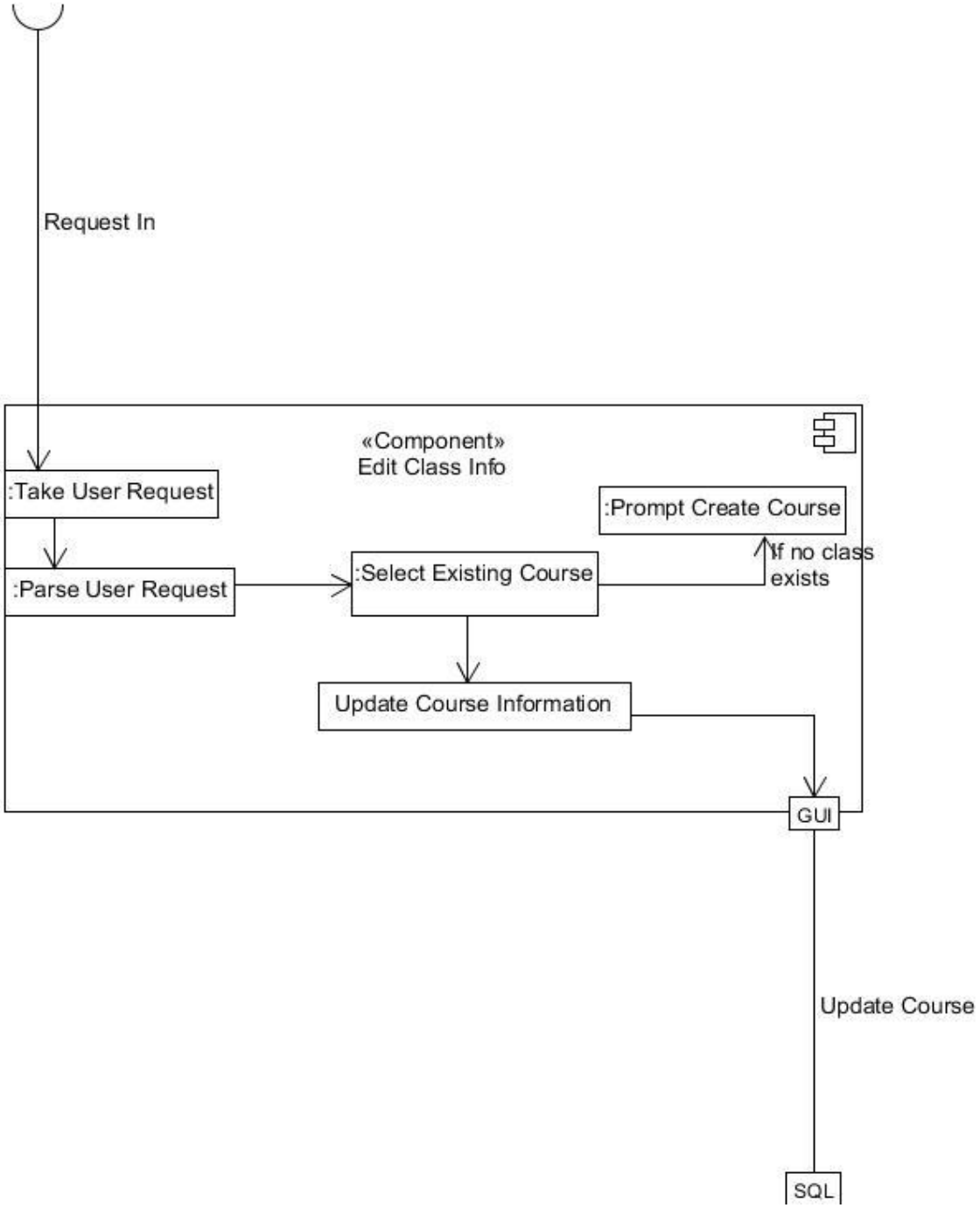
Similar to the “Registrar User interface”, the “Faculty User interface” acts as the main hub of requests on services. All behaviors and communications to other components are handled here in the Faculty side of the system. The interface “Request In” handles the request from the “Faculty Request” component and displays a graphical user Interface where the user can see and selected exclusive components available to only users with Faculty accounts and permissions. Information is also downloaded and uploaded every 15 seconds in the background from the “Service Hub Database” component to ensure the user is getting the most recent content. This component is available 24/7 and can handle the load of over 4000 faculty members.

Post Grades



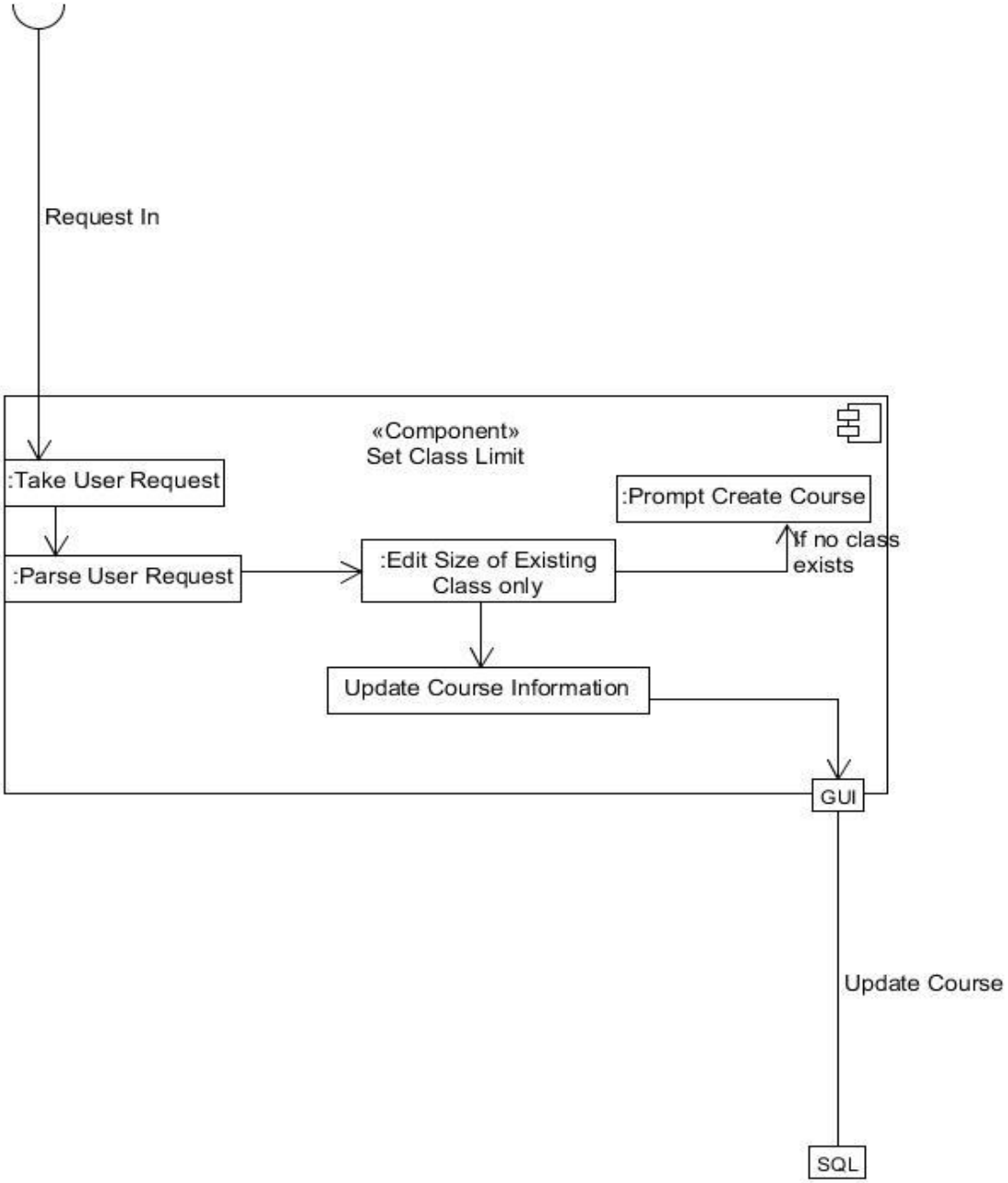
Exclusive to Faculty users, the “Post Grades” component allows Faculty to post the grades of students and uploading it on the database for the exclusive access of the Registrar until the Registrar makes it visible to students via “Release Grades”. This exists within an optional window for Faculty to select on the GUI of “Faculty User Interface”. “Post Grades” relies on the “Faculty User Interface” component and is evident through the interface “Request in” on the diagram.

Edit Class Information



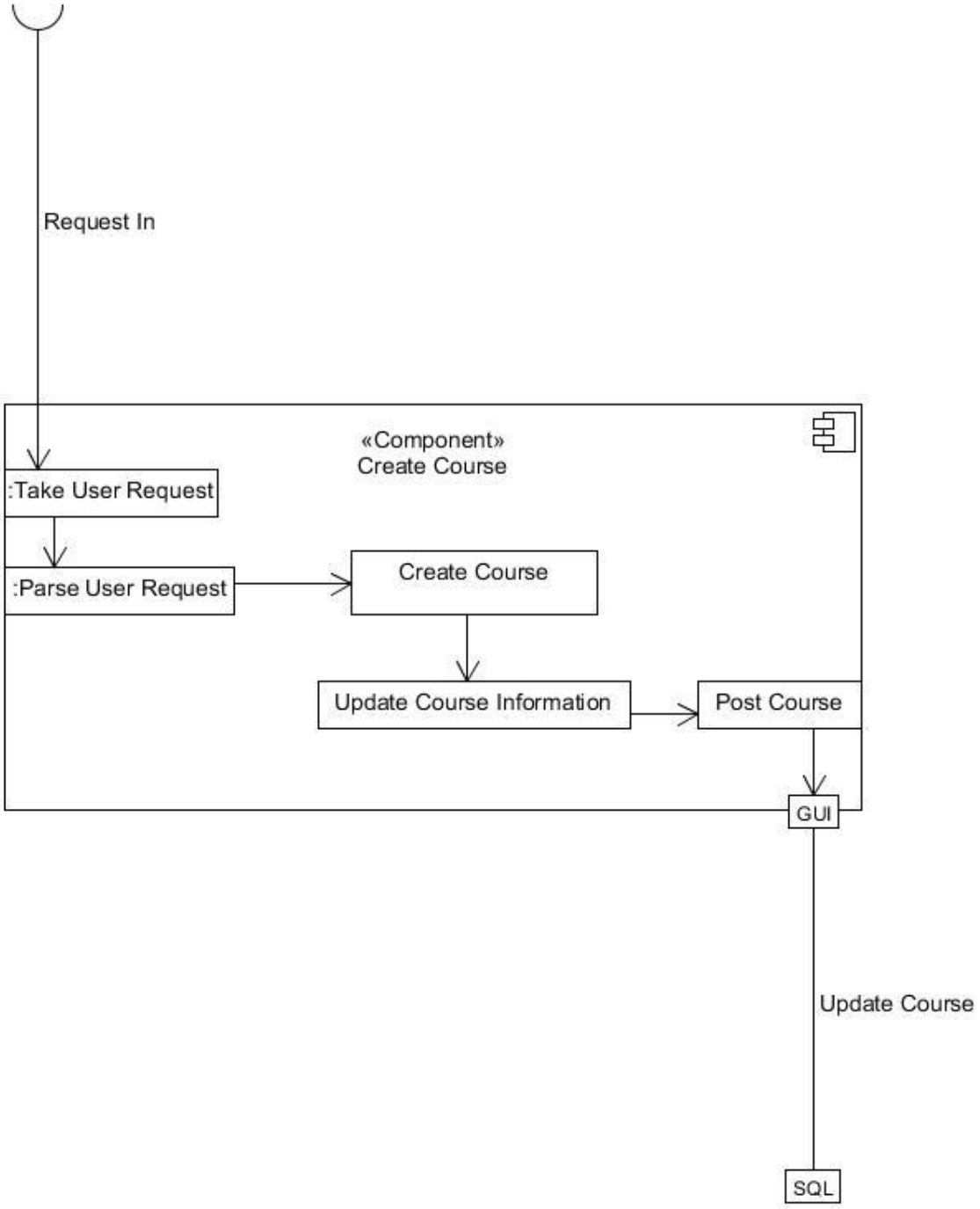
Exclusive to Faculty users, the “Edit Class Info” component allows Faculty to edit course information. This exists within an optional window for Faculty to select on the GUI of “Faculty User Interface”. Here Class information is edited according to the liking of the faculty. Once completed the information is uploaded to the “Service Hub Database” so information can be accessed by the Students and Registrar.

Set Class Limit



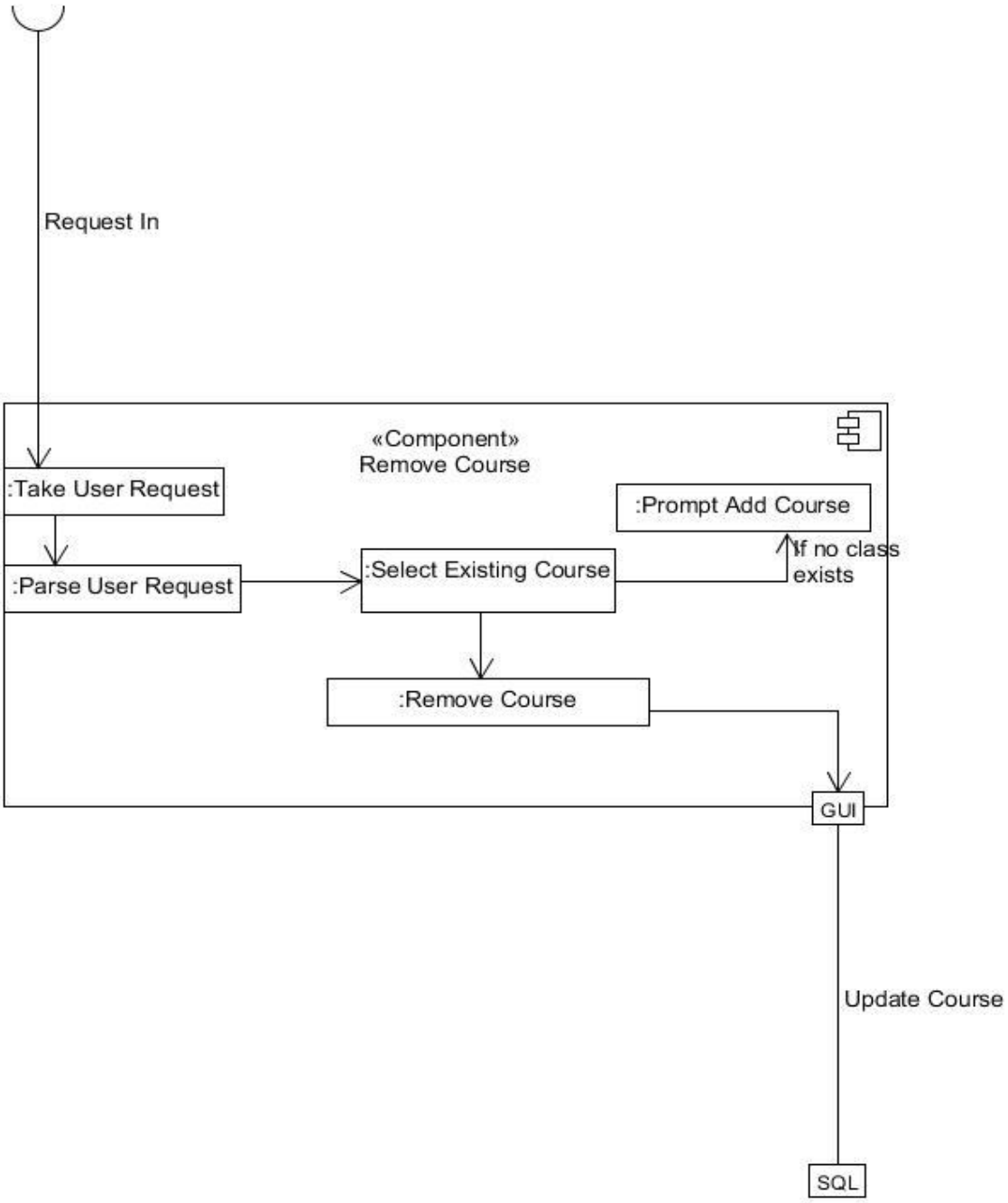
“Set Class Limit” is an exclusive component for Faculty users. Here faculty have the ability to choose how many students they want in their classroom. Once the faculty is fine with their decision the information will be updated to the “Service hub Database” which will restrict the amount of student users from enrolling into the class. This component is dependent on the “Faculty User Interface” component and is evident through the “Request in” interface within the diagram. The component is expressed as a selectable optional window within the “Faculty User interface”.

Create Course



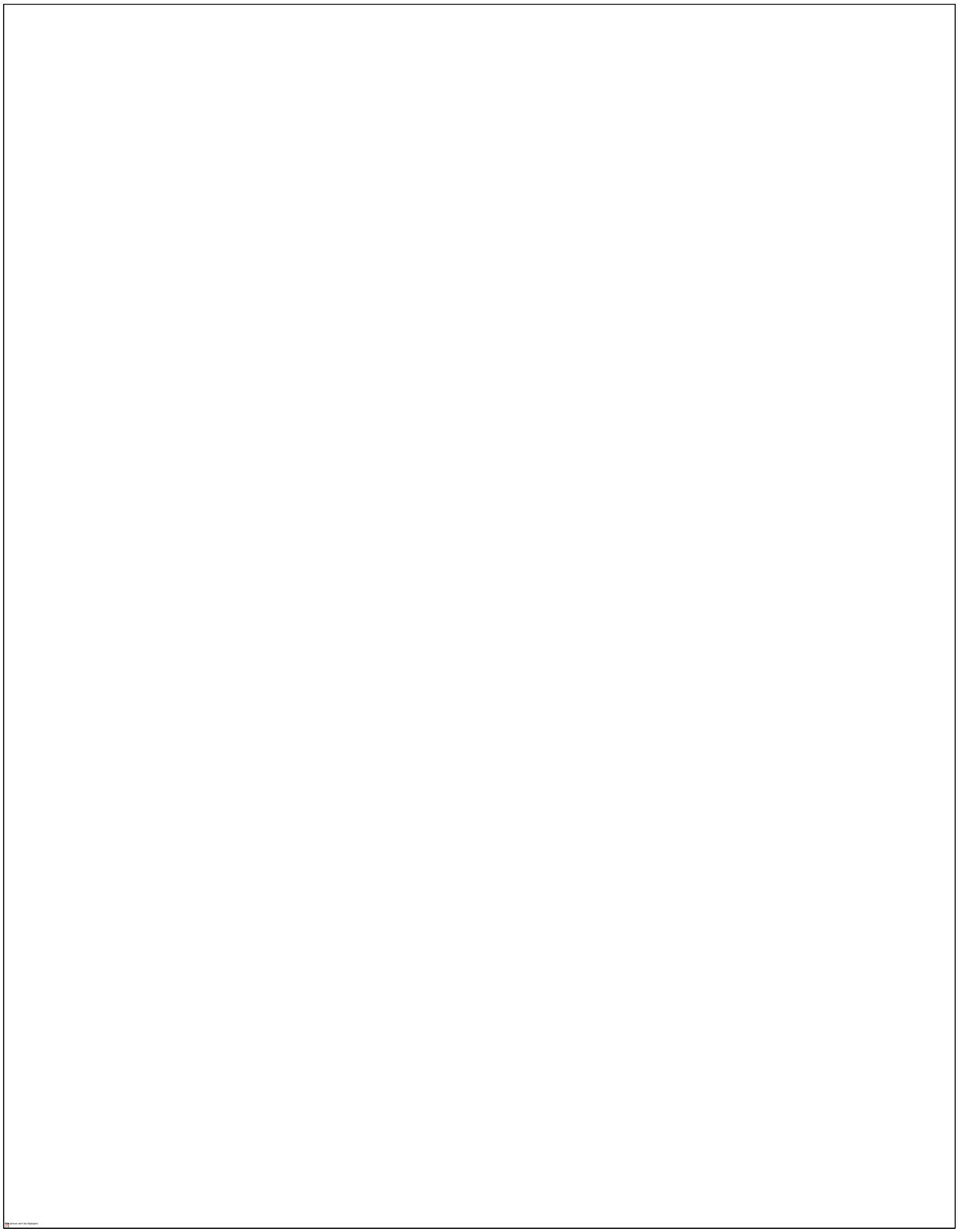
“Create Course” is an component available for Faculty user which allows them to choose what course they want to instruct. When the user has finalized the course it is then updated to the “Service Hub Database” to be see by the Registrar and Student Services. This is an optional, selectable window within the “Faculty User Interface” GUI. The “Create Course” component also relies on the “Faculty User Interface” component in order to function.

Remove Course



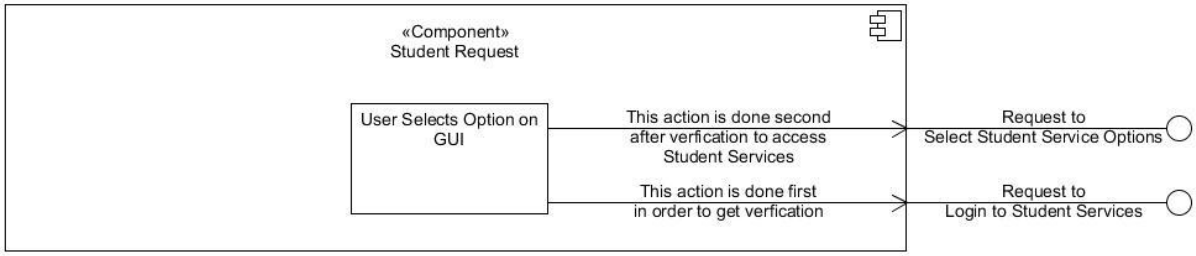
The “Remove Course” component is exclusive to the user of Faculty permission and allows the faculty to stop teaching a specific course that they don’t want. This is then updated via SQL to the “Service Hub Database” which will then affect both Student and Registrar services as a course specific the a faculty user will be removed. This component relies on the “Faculty Service Hub” component in order to function and is visible through the “Request In” interface on the diagram.

Student - Mid Level Architecture



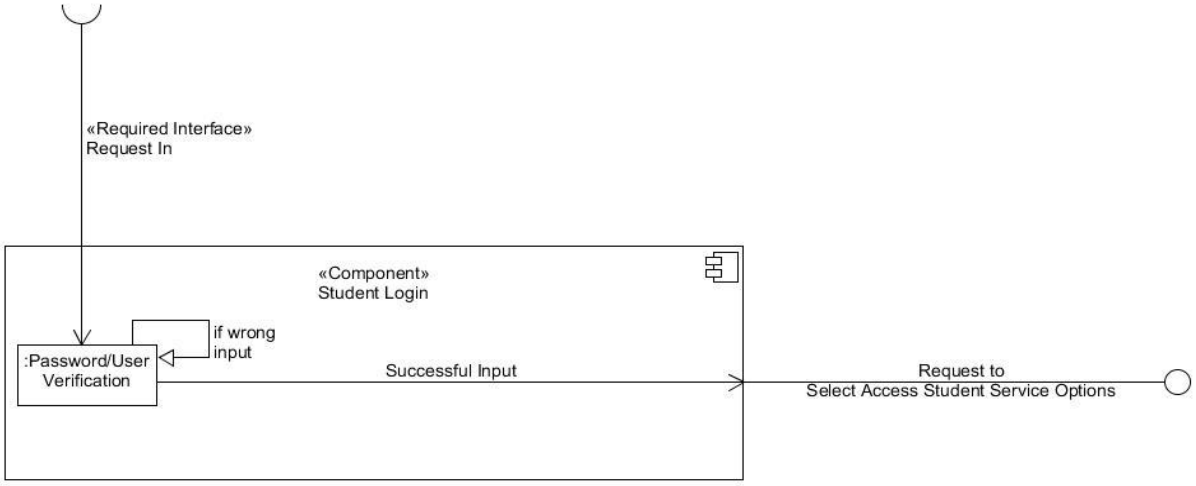
Student - Sub Diagrams

Request



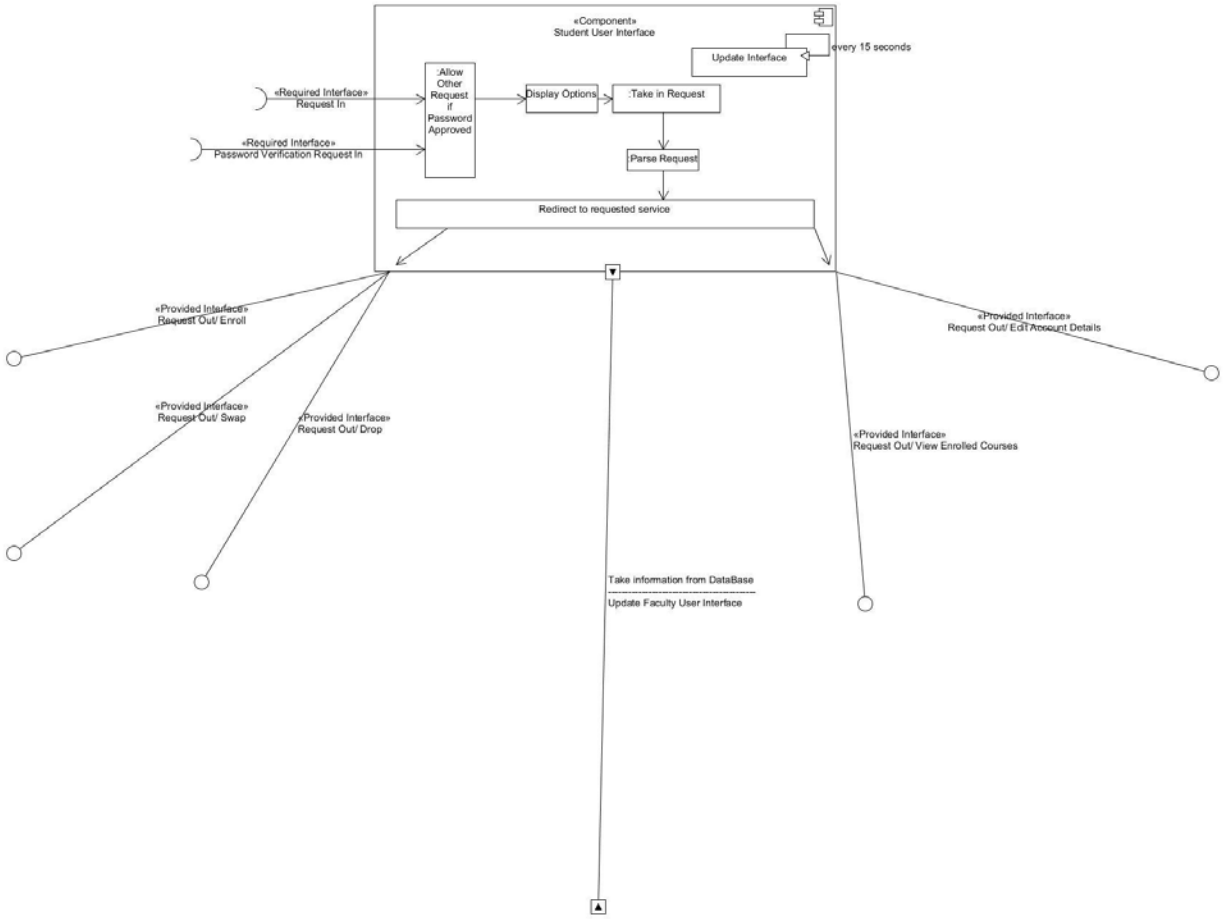
“Student Request” is a component only accessible by the user Student. Here Requests that are identified as Student permission requests are sent out throughout the system giving the user access to specific rights unique to other users on the system. At this component the user is allowed to firstly, sign in, because all the service cannot be accessed without a correct user login and then the second action of sending out service request to the Student User Interface can be made. The components “Student User interface”, Student Login, Enroll, Swap, Drop, View Enrolled Courses, View Account Details, depend on the Student Request to function. All the “Request in” interfaces are fulfilled by the “Request to” initially send by the user when a GUI option is selected.

Login

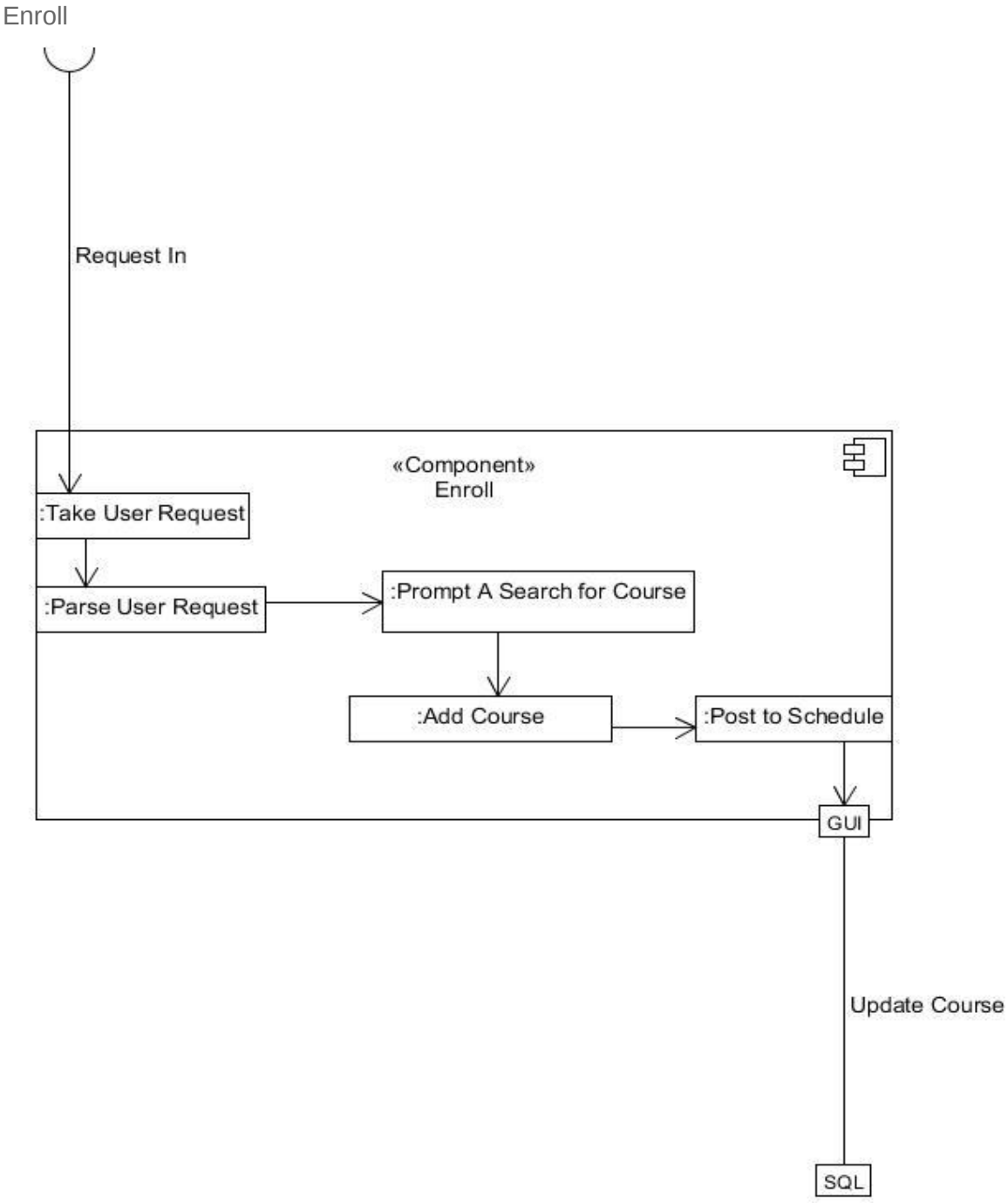


“Student Login” is a component that depends on the “Student Request” component, this is demonstrated through the “Request In” interface. Within this component, the user password and login is verified for the use of security. This ensures that the student services and permissions are used by the correct user. After this component successfully verifies the user, it sends a “Request to” interface to the “Student User Interface ” to display the services needed by the user.

User Interface

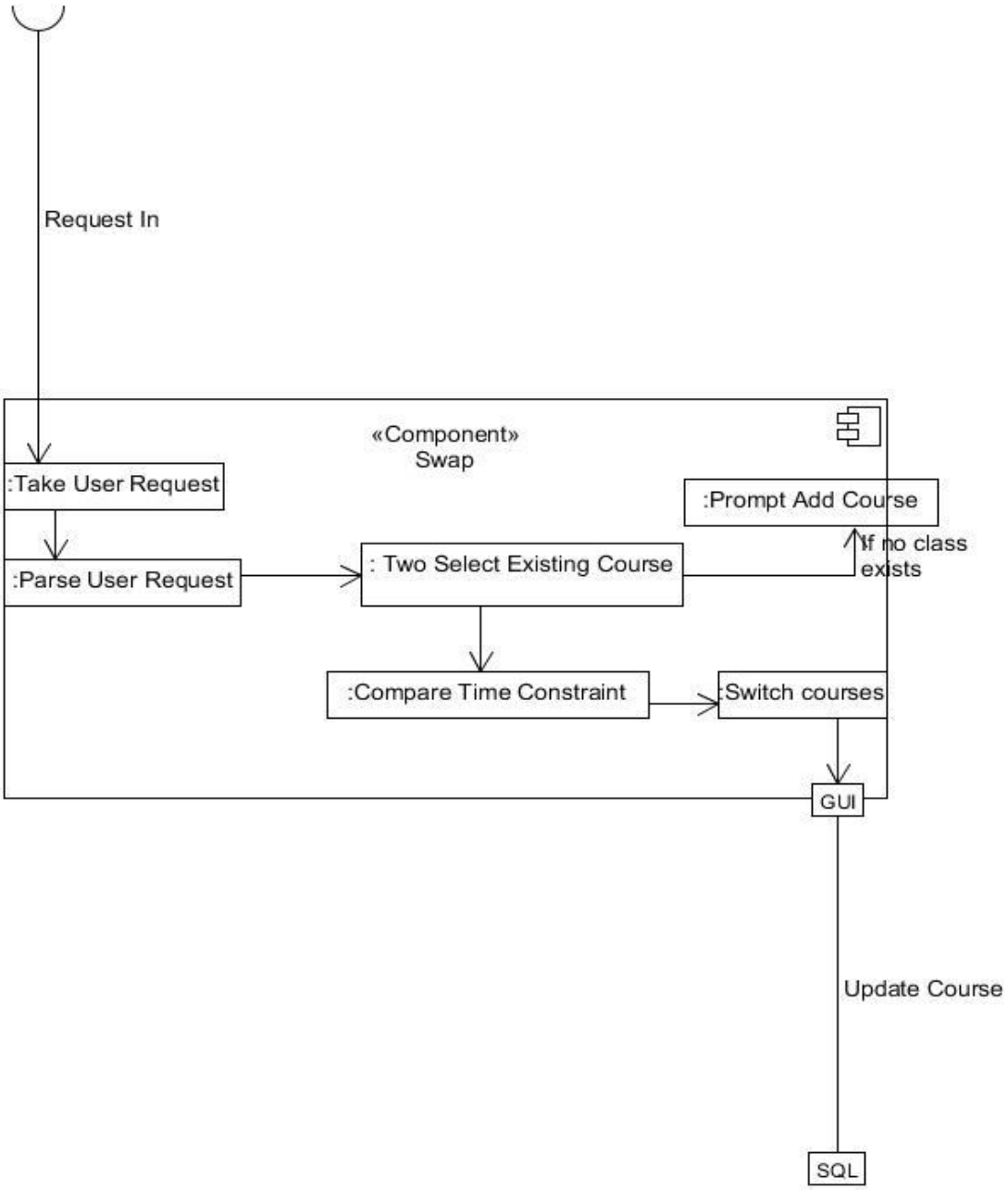


Similar to the “Faculty User interface”, the “Student User interface” acts as the main hub of requests on services for the Student users. All behaviors and communications to other components are handled here in the Student side of the system. The interface “Request In” handles the request from the “Student Request” component and displays a graphical user Interface where the user can see and selected exclusive components available to only users with Student accounts and permissions. Information is also downloaded and uploaded every 15 seconds in the background from the “Service Hub Database” component to ensure the user is getting the most recent content. This component is available 24/7 and can handle the load of over 4000 students.This component is simple with all options being visible right on the screen, no need for hamburger menus. Being the main component of the Student Service Hub, multiple “Request out” interfaces are sent in order to manage Enroll, Swap, Drop, View Enrolled Courses, and Edit Account Information.



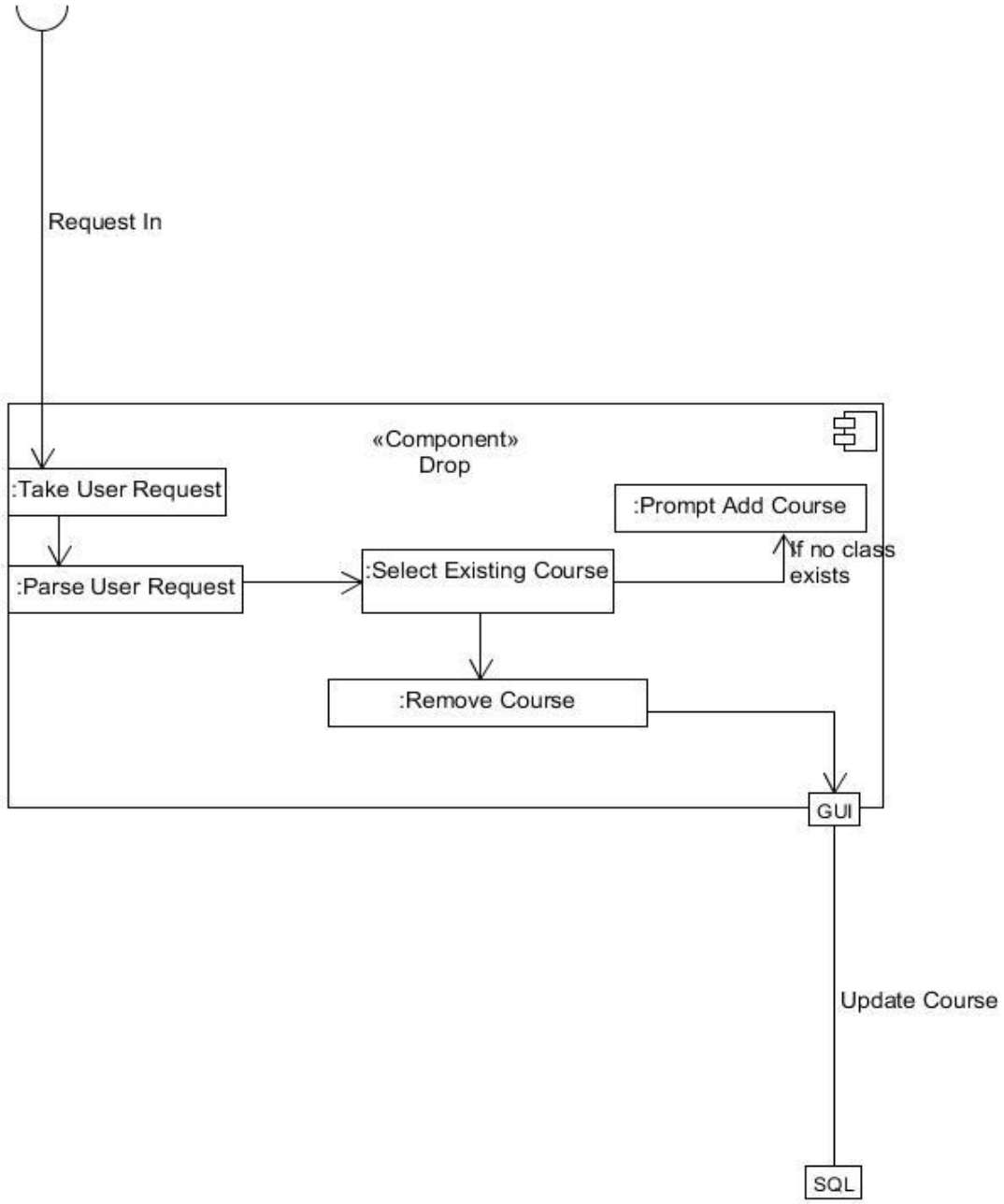
The component “Enroll” prompts the user to search for courses. Once the desired course is found they student is able to add the course to their schedule after which they are brought to view their updated schedule. Once complete the Database gets updated and the student is sent back to the main GUI. There is a dependency on the “Student User Interface” component, which is apparent based on the “Request in” interface. This action is as fast as possible provided the system needs to search through the database of courses being offered.

Swap



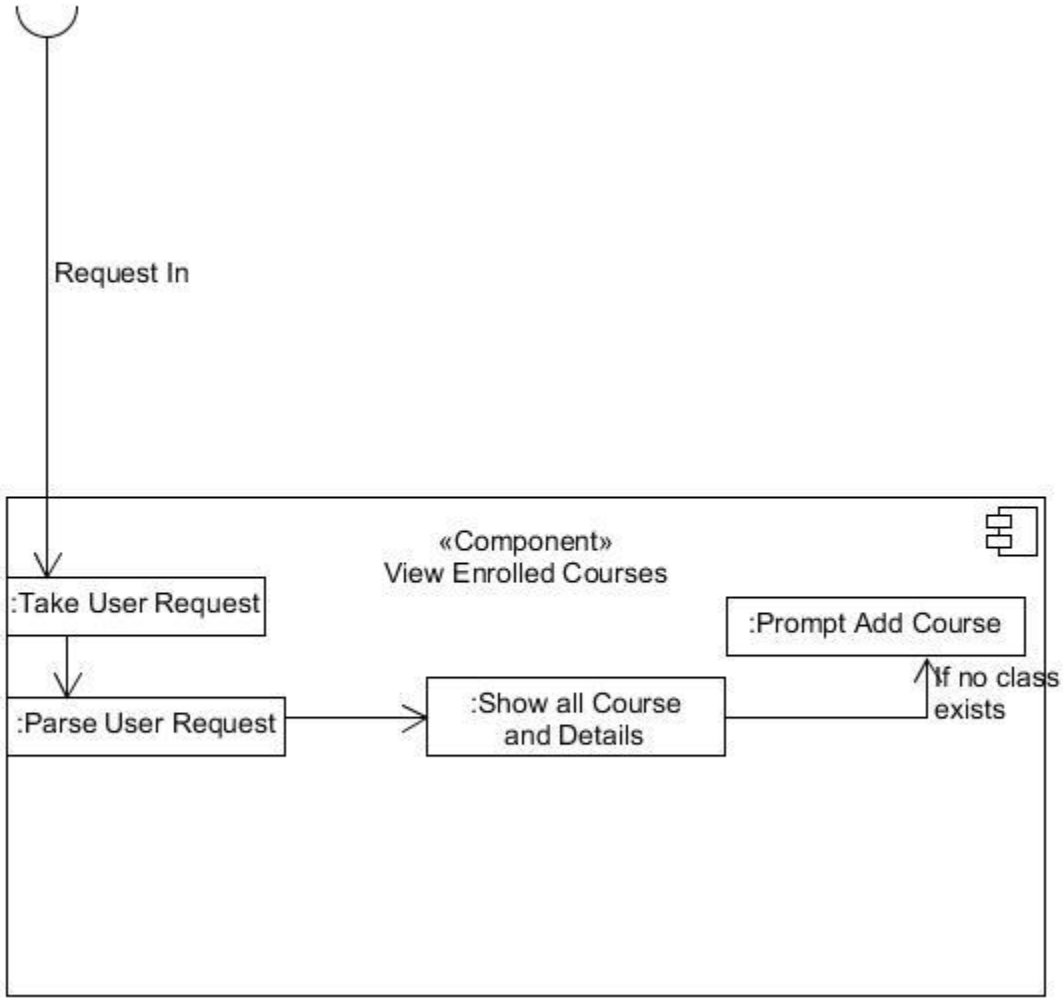
The component “swap” allows the user to change one course for another. Within the student GUI they are prompted to select an existing course with another course they are not enrolled in. Before the change can be made the system validates that the new course fits within the time constraints of the student's schedule. Once completed the Database gets updated. There is a dependency on the “Student User Interface” component, which is apparent based on the “Request in” interface. If a student tries to swap a course without being enrolled in any they are instead prompted to add a course they are taken to the “Add Course” component of the system.

Drop



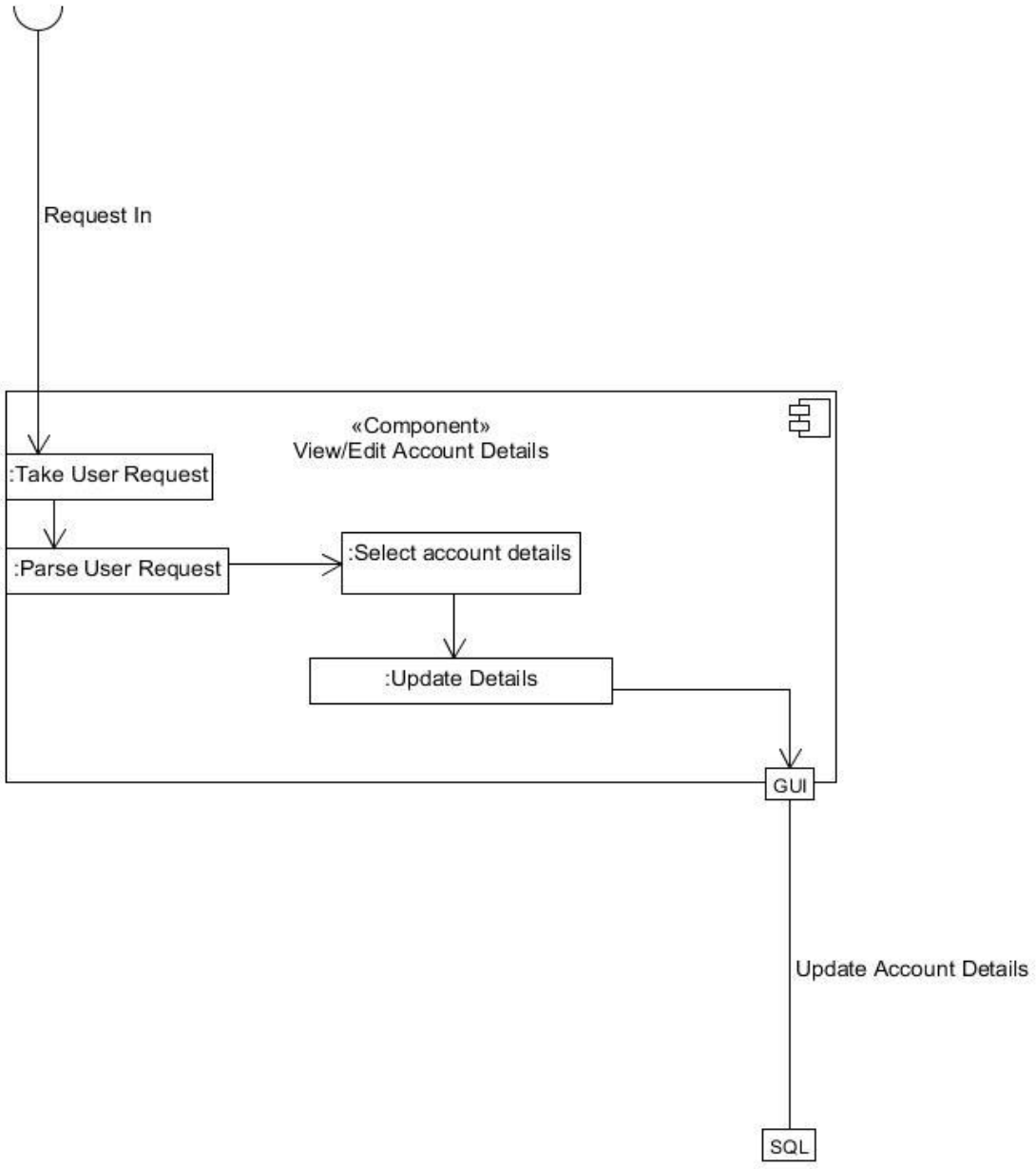
The component “Drop” will allow the user to drop a specific course. They navigate a GUI which asks them to select the course they wish to drop at which point the Database gets updated upon confirmation. If this location in the system is reached and the student has no classes to drop they instead get prompted to enroll in courses now from an option within the GUI. There is a dependency on the “Student User Interface” component, which is apparent based on the “Request in” interface. This action is a very fast transaction between the database and the student GUI.

View Enrolled Courses



The component “View Enrolled Courses” will show the user all of the courses they are currently enrolled in and if no classes are enrolled they get prompted to enroll in courses now from an option within the GUI. There is a dependency on the “Student User Interface” component, which is apparent based on the “Request in” interface. This action is a very fast transaction between the Database and the student GUI. This is just a viewing platform and no new information needs to be updated for the rest of the system. If they are prompted to add a course they are taken to the “Add Course” component of the system.

View/Edit Account Details



The component “View/Edit Account Details” is a quick transaction between the Service Hub DataBase and the GUI the student is viewing. Here they are taken through the “Student User Interface” GUI to a page which displays their account information and allows them to make changes if necessary. There is a dependency on the “Student User Interface” component, which is apparent based on the “Request in” interface. If any changes are made their account information is updated within the Database and they are sent back to the main account details page.

Scenarios

Quality Attribute	Stimulus	Response
Availability	24/7 Server Uptime	Fluid server response, Students, Faculty, and Registrar having access to all information as well as user support at all hours of the day.
Scalability	Heavy Usage (I.e. over 4000 simultaneous users)	Delayed server responses and updated information to the service hub. Possibility for the servers to crash and any changes made during this time will be lost and all active users will be disconnected from the servers.
Security	Incorrect Login	The user is displayed an error message and prompted to try logging in again. Only matching user-passwords are allowed to login to the system to ensure a secure database.
Modifiability	Remodeling GUI	The change in the GUI of the “User Interface” component of the Faculty, Student and Registrar will not affect the other components in terms of flowing information and communication but the Ease of Use might get affected, users may have to re-teach themselves how to get use to the new user interface.
Privacy	User trying to view Another User's’ information	With the restricted amount of actions that our system provides as well as the strict placement of permission, it makes it safer to ensure that a user can only view what is allocated to their own account and what they have permissions to avoid tampering with sensitive information.