ETHERA

Multifaceted Student Industrial Placement System
Version 1.00

Software Design Specification

Rajarata University of Sri Lanka
Faculty of Applied Sciences
Information & Communication Technology

Table of Content

1. INTRODUCTION 1

1.1. Purpose 1

1.2. Document Conventions 1

2. ARCHITECTURAL DESIGN 1

2.1. high level components and their interactions 1

2.1.1 Component (Sub-System) Design 1

2.1.2 Components 2

2.1.3 Interfaces 3

2.2. architectural styles / patterns 5

2.2.1 Layered Architectural Style 5

2.3. physical arrangements of devices 8

2.4. design decisions 9

2.4.1. MVC (Model-View-Controller) development style is been used 9

2.4.2. Object Oriented Software development method is used 9

2.4.3. AJAX (Asynchronous JavaScript and XML) is used 9

3. COMPONENT AND DETAIL DESIGN 10

3.1. DESIGN PATTERNS 10

3.1.1. Prototype Pattern 10

3.1.2. Observation Pattern 10

3.1.2. Composite Design Pattern 10

3.1.2. Strategy Design Pattern 11

3.2. Class Diagram 12

3.3. sequence diagrams 13

3.3.1 Use case 1 : Login 13

3.2.2 Use case 2 : Change password 14

3.2.3 Use case 3 : Edit System User Profile 15

3.2.4 Use case 4 : Edit Student Profile 16

3.2.5 Use case 5 : Edit Student Personal Information 16

3.2.6 Use case 6 : Edit Student Initial Choices 17

3.2.7 Use case 7 : Student Choices Finalizing 17

3.2.8 Use case 8 : Registration 18

3.2.9 Use case 9 : Share Industry Experience 18

3.2.10 Use case 10 : View others experience 19

3.2.11 Use case 11 : Add feedbacks 20

3.2.12 Use case 12 : Edit feedback 20

3.2.13 Use case 13 : Registration on/off 21
3.1.14 Use case 14 : Feedback management 21
3.2.15 Use case 15 : Industry Grant Opportunity 22
3.2.16 Use case 16 : Category phase selection 22
3.2.17 Use case 17 : Messaging 23
3.2.18 Use case 18 : Messaging to students 24
3.2.19 Use case 19 : Messaging to staff 25
3.2.20 Use case 20 : Messaging to company 26
3.2.21 Use case 21 : Auto calculate student 27
3.2.22 Use case 22 : View auto calculated result 27
3.2.23 Use case 23 : Add marks to extra activities 28
3.2.24 Use case 24 : Notice management 28
3.2.25 Use case 25 : Notice creation 29
3.2.26 Use case 26 : Notice modify 30
3.2.27 Use case 27 : Public view notice 30
3.2.28 Use case 28 : Public view members 31
3.2.29 Use case 29 : Public view articles 31
3.2.30 Use case 30 : Category Management 32
3.2.31 Use case 31 : Create Category 32
3.2.32 Use case 32 : Category Modification 33
3.2.33 Use case 33 : Choice Management 33
3.2.34 Use case 34 : Choice Creation 34
3.2.35 Use case 35 : Choice Modify 35
3.2.36 Use case 36 : Organization Management 35
3.2.37 Use case 37 : Organization Creation 36
3.2.38 Use case 38 : Organization Modify 37
3.2.39 Use case 39 : Opportunity Management 38
3.2.40 Use case 40 : Filter Students 39
3.2.41 Use case 41 : Search 39
3.2.42 Use case 42 : Search by academic information 40
3.2.43 Use case 43 : Search by choice 41
3.2.44 Use case 44 : Article management 41
3.2.45 Use case 45 : Article creation 42
3.2.46 Use case 46 : Article modify 42
3.2.47 Use case 47 : Student profile approval 43
3.2.48 Use case 48 : Approval for Registration 43
3.2.49 Use case 49 : Initial choice and CV approval 43

3.2.50 Use case 50 : Finalized choice approval 43
3.2.51 Use case 51 : Industry ready or not 44
3.2.52 Use case 52 : Category freeze 44
3.2.53 Use case 53 : User Management 45
3.2.54 Use case 54 : Create users 45
3.2.55 Use case 55 : Modify users 46
3.2.56 Use case 56 : View profile 46
3.2.57 Use case 57 : Update student's profile 47
3.2.58 Use case 58 : Report Generation 47
3.2.59 Use case 59 : Print Student Profile 47
3.2.60 Use case 60 : Add Course 48
3.2.61 Use case 61 : Add Marks 48
3.3 Algorithm design 49
3.3.1 Filter Students Algorithm 49
3.3.2 Sort algorithm 49
3.3.3 Search Students by Academic Information 50
3.3.4 Student selection scientific algorithm (Still developing) 50
3.3.5 Algorithm Optimization 52
3.4 Database design 53
3.4.1 Relational Model 53
3.4.2 Normalization/Denormalization 54
3.4.3 Data Dictionary 55
3.4.4 Decisions made to manage transactions concurrent 60
3.4.5 Indexes 60
3.5. User interfaces 64
3.6. Rules and guidelines for interface design 64
3.6.1 User interface design framework 64
3.6.2 User Input validation methods 64
3.6.3 Alert messages decomposition 65
3.7. User interfaces design 66
3.7.1 Use case 1 : Login 66
3.7.2 Use case 2 : Change password 67
3.7.3 Use case 3 : Edit System User Profile 67
3.7.5 Use case 5 : Edit Student Personal Information 68
3.7.6 Use case 6 : Edit Student Initial Choices 69
3.7.7 Use case 7 : Student Choices Finalizing 70
3.7.8 Use case 8 : Registration 71

3.7.9 Use case 9 : Share Industry Experience 72
3.7.10 Use case 10 : View others experience 72
3.7.11 Use case 11 : Add feedbacks 73
3.7.12 Use case 12 : Edit feedback 73
3.7.13 Use case 13 : Registration on/off 74
3.7.14 Use case 14 : Feedback management 74
3.7.15 Use case 15 : Industry Grant Opportunity 75
3.7.16 Use case 16 : Category phase selection 75
3.7.17 Use case 17 : Messaging 76
3.7.18 Use case 18 : Messaging to students 76
3.7.19 Use case 19 : Messaging to staff 77
3.7.20 Use case 20 : Messaging to company 77
3.7.21 Use case 21 : Auto calculate student 78
3.7.22 Use case 22 : View auto calculated result 78
3.7.23 Use case 23 : Add marks to extra activities 79
3.7.24 Use case 24 : Notice management 79
3.7.25 Use case 25 : Notice creation 80
3.7.26 Use case 26 : Notice modify 80
3.7.28 Use case 28 : Public view members 81
3.7.29 Use case 29 : Public view articles 81
3.7.30 Use case 30 : Category Management 82
3.7.31 Use case 31 : Create Category 82
3.7.32 Use case 32 : Category Modification 83
3.7.33 Use case 33 : Choice Management 83
3.7.34 Use case 34 : Choice Creation 84
3.7.35 Use case 35 : Choice Modify 84
3.7.36 Use case 36 : Organization Management 85
3.7.37 Use case 37 : Organization Creation 85
3.7.38 Use case 38 : Organization Modify 86
3.7.39 Use case 39 : Opportunity Management 86
3.7.40 Use case 40 : Filter Students 87
3.7.41 Use case 41 : Search 87
3.7.42 Use case 42 : Search by academic information 88
3.7.43 Use case 43 : Search by choice 88
3.7.44 Use case 44 : Article management 89
3.7.45 Use case 45 : Article creation 89
3.7.46 Use case 46 : Article modify 90

3.7.47 Use case 47 : Student profile approval 90
3.7.48 Use case 48 : Approval for Registration 91
3.7.49 Use case 49 : Initial choice and CV approval 91
3.7.50 Use case 50 : Finalized choice approval 92
3.7.51 Use case 51 : Industry ready or not 92
3.7.52 Use case 52 : Category freeze 93
3.7.54 Use case 54 : Create users 93
3.7.55 Use case 55 : Modify users 94
3.7.56 Use case 56 : View profile 95
3.7.57 Use case 57 : Update student's profile 96
3.7.59 Use case 59 : Print Student Profile 97
3.7.60 Use case 60 : Add Course 98

1. INTRODUCTION

1.1. PURPOSE

This is the Software Design Specification version 1.0 of the third year project, “Multifaceted student industrial placement system” also called the “ETHERA”. This document covers the all the design aspects of the project starting from the high level components of the system to the class level design. It also includes the sequence diagrams and necessary UI mockups of the project.

This document is intended to be referred frequently in the implementation phase, since this document forms the concrete basis for the implementation.

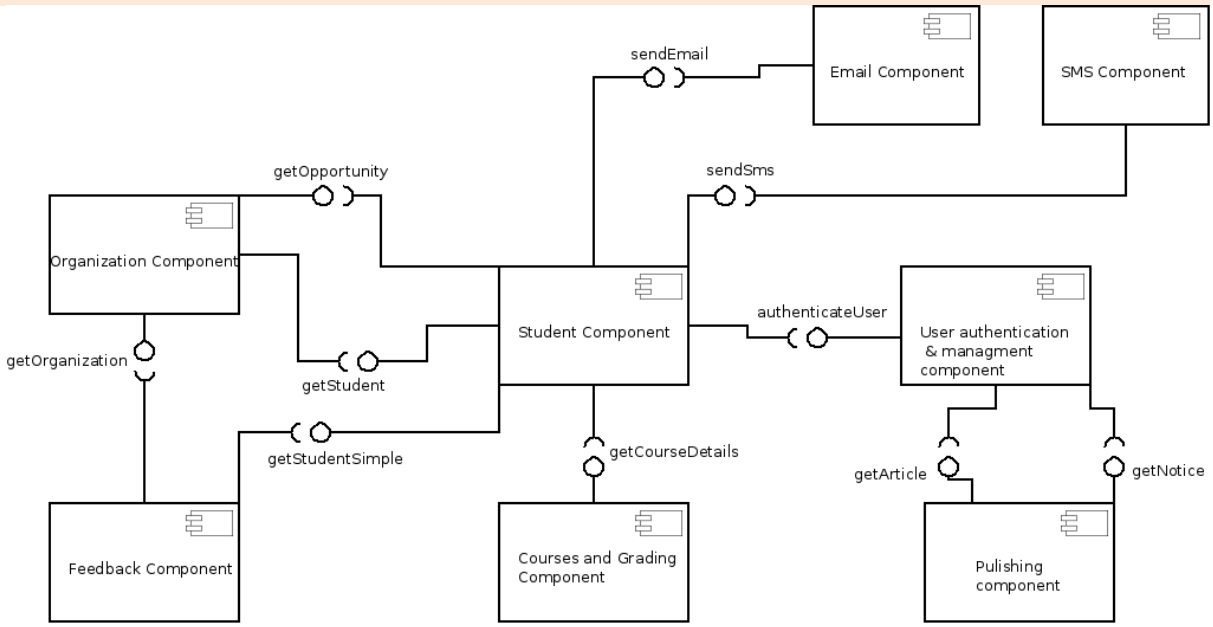
1.2. DOCUMENT CONVENTIONS

Heading Style	Font Size (pt)	Font Type	Font Color
Master Headings	13	Calibri	White
Sub Headings	12	Calibri	Black
Other Headings	11	Calibri	Black
Body	10	Calibri	Black

2. ARCHITECTURAL DESIGN

2.1. HIGH LEVEL COMPONENTS AND THEIR INTERACTIONS

2.1.1 Component (Sub-System) Design



2.1.2 Components

Student component

The main subsystem that has the internal functionality of handling and managing student related data. In this component we'll implementing the scientifically designed selection algorithm of students and that algorithm will get the service from the “Organization Component” and “Courses and Units” component to accomplish optimized selection of students. In other hand this component will maintain student categorization with regarding to the study programs and batches.

Organization component

In this sub-system maintains the interested areas of students and the organizations itself. Also this is the key component of the system that has the power of taking in the granting of opportunities from the organizations in to its data sources. Through the interfaces connected with student component it'll maintain the state of students' industrial placement progress.

Courses and grading component

Basically this component is responsible for the maintenance of the subjects and its related course units. Given that a student passes from a course unit that is related to his/her program, this component will handle the grading for that particular course unit.

As an example :- For the ICT1305 Data Structures course unit will related to the Subject ICT and students who are enrolled in to ICT program can obtain a grading (A+, A, A- . . . .) for that course unit.

User authentication and management component

This component can introduce as one of the key components in the system that bears the security and group based action restriction enforcement of the whole system. For the authentication of the users, this component work as a getaway and If things will go well in the authentication part, this component will route the authenticated user in to the correct place of the application.

For the Access-Control-List part, the component uses a tree like structure which is populated in the database for the access-control-objects (ACO) and access-request-objects (ARO). From this ACL architecture, it opens the administrators the privileged of assigning very flexible permission schemes for controller actions in the MVC.

Feedback Component

Feedback sharing of students regarding organizations is handled by this component. Basically this components is relatively smaller than above explained components and does the job of interconnecting students' ideas regarding various organizations which are already there in the system with those particular organizations.

Publishing Component

This component does its job by publishing “Notices” and “Articles”. In other words this component doing a job that small CMS or blog does. But the special thing is it capable of handling multi user posts. Also the “Notice(s)” will use Google Calendar and Event API to sync the event with very own calendar for Career Guidance Unit.

Email Component

The emailing component is responsible of the sending of single or bulk email as per request. This component services call upon the requirement of sending emails to the users who are currently registered under this system. Since an email is a must for every user, the email database field will not be a blank one for any user including system users and students. Also this component will build to support out-of-box to change between internal MTA (Mail Transport Agent), usually Postfix, and external SMTP (Simple Mail Transfer Protocol).

SMS Component

Same as email component, SMS component will call when there is a need to send a SMS or SMS bulk. For the SMS API, we'll use Dialog IdeaPro hosted SMS API deployed on top of Jetty Server.

2.1.3 Interfaces

Since this system implemented as tight MVC style there will be well defined interfaces between models, controllers and views. But for the sake of modularity, we'll define some interfaces to breakdown the system structure. So below defined interfaces will use all the controllers within particular component as listed under each component.

Organization Component

getOpportunity

Organization Component will serve with the services that related to the opportunities and organizations. Not only the opportunity details but also organizations incorporated with.

getOrganization

As same in the above, the component will serve organization details to the feedback component to index the feedback according to the organization.

Controllers involved:-

- InterestedAreasController
- OrganizationsController

Student Component

getStudent

From this interface all the required services related to the student can be obtained. This is one of the busy interfaces in the system that will serve data to the organization component.

getStudentSimple

This interface seems similar from the name comparing to the above. But this is not a complex interface than the "getStudent" interface. This will just serve the basic details of students while not going in to complex relationships.

Controllers Involved:-

- StudentsController
- BatchesController
- StudyProgramsController

Courses and Grading component

getCourseDetails

As its name explains this interface's soul responsibility is to serve with course details to the "Student Component". Since this sub-system is essential to hold the Grading data for each and every student, it should be implemented with high priority.

Controllers Involved:-

- SubjectsController
- CourseUnitsController
- CvsController
- ExtraActivitiesController

User authentication and management component

authenticateUser

This interface is actually does the system gatekeeper's duty by authenticating the users and routing them with necessary (ACL) access control list privilege enforced.

Controllers Involved:-

- SystemUsersController
- GroupsController

Feedback component

This component doesn't have any serving interfaces, but consuming interfaces from student component and organization component.

Publishing Component

getArticle

Basically this interface is for the article creation of system users. In provide the service for modification of articles and creations of new articles.

getNotice

Same as the "getArticle" interface, this interface does have the facility of modifying existing Notices and creating Notices which will be published on the frontend notice board. One special thing in this interface is that it'll incorporate with "getArticle" interface when extending notices with articles.

Controllers Involved:-

- FeedbacksController
- ArticlesController

Email Component

sendEmail

This is used as the postman interface for the other components. So just consuming the services from this interface, another component can initiate an email sending process.

Controllers involved:-

- EmailsController

SMS Component

sendSms

Just like the "sendEmail" interface, this will also act as a messenger interface and this will complete the SMS sending process through hosted API from Dialog IdeaPro.

Controllers Involved:-

- SmsesController

2.2. ARCHITECTURAL STYLES / PATTERNS

"Ethera" will be developed under two main architectural patterns. As it builds on top of CakePHP framework API, development of project will done in neat MVC architectural style. This MVC pattern can be boiled in to the layered architectural style. Furthermore the implementation of this project will do via the Object-Oriented architectural style. That is because all the Models and Controllers in the layered architectural pattern will write as classes and instantiated as objects in the runtime.

Explanation of used architectural styles

2.2.1 Layered Architectural Style

Layered architecture focuses on the grouping of related functionality within an application into distinct layers that are stacked vertically on top of each other. Functionality within each layer is related by a common role or responsibility. Communication between layers is explicit and loosely coupled. Layering our application appropriately helps to support a strong separation of concerns that, in turn, supports flexibility and maintainability.

The reason for the decision:-

Since layered architectural style is one of the best suitable architectural style for a web application, we tend to use it in our project. As we need to separate our presentation logic from, business logic and our data access logic, MVC will to come across that situation in a very systematical way. The raw mechanism done within the models, controllers and views are explained below.

Model

The Model layer represents the part of our application that implements the business logic. It is responsible for retrieving data and converting it into meaningful concepts for our application. This includes processing, validating, associating or other tasks related to handling data.

View

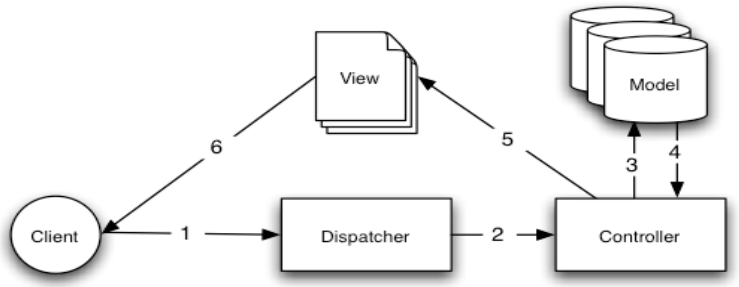
The View renders a presentation of modeled data. Being separated from the Model objects, it is responsible for using the information it has available to produce any presentational interface our application might need. For example, as the Model layer returns a set of data, the view would use it to render a HTML page containing it. Or a XML formatted result for others to consume.

Controller

The Controller layer handles requests from users. It's responsible for rendering back a response with the aid of both the Model and the View Layer. Controllers can be seen as managers taking care that all needed resources for completing a task are delegated to the correct workers. It waits for petitions from clients, checks their validity according to authentication or authorization rules, delegates data fetching or processing to the model, and selects the correct type of Presentational data that the client is accepting, to finally delegate this rendering process to the View layer.

Benefits of MVC architecture cum Layered style

Why use MVC? Because it is a tried and true software design pattern that turns an application into a maintainable, modular, rapidly developed package. Crafting application tasks into separate models, views, and controllers makes our application very light on its feet. New features are easily added, and new faces on old features are a snap. The modular and separate design also allows developers and designers to work simultaneously.



The illustration of how our system MVC architecture works. Numbers represent the point of sequence.

Object-Oriented Architectural Style

Object-oriented architecture is a design paradigm based on the division of responsibilities for an application or system into individual reusable and self-sufficient objects, each containing the data and the behavior relevant to the object. An object-oriented design views a system as a series of cooperating objects, instead of a set of routines or procedural instructions. Objects are discrete, independent, and loosely coupled; they communicate through interfaces, by calling methods or accessing properties in other objects, and by sending and receiving messages.

The reason for the decision :-

Mainly the MVC style we'll be using is build on top of Object-Oriented architecture. The reason for using such an architectural style is the ease of communication between the Models, Views and controllers. As shown in fig a request from the client (a browser in most cases) will come in to a Controller via Dispatcher unit of the CakePHP framework. And then through a method controller will instruct the models to fetch relevant data from the datasources. After manipulating data, the controller again instruct to the view to render the necessary data to client. In this simple scenario the MVC will keep the interconnection through Object-Oriented style and will pass messages to communicate between them.

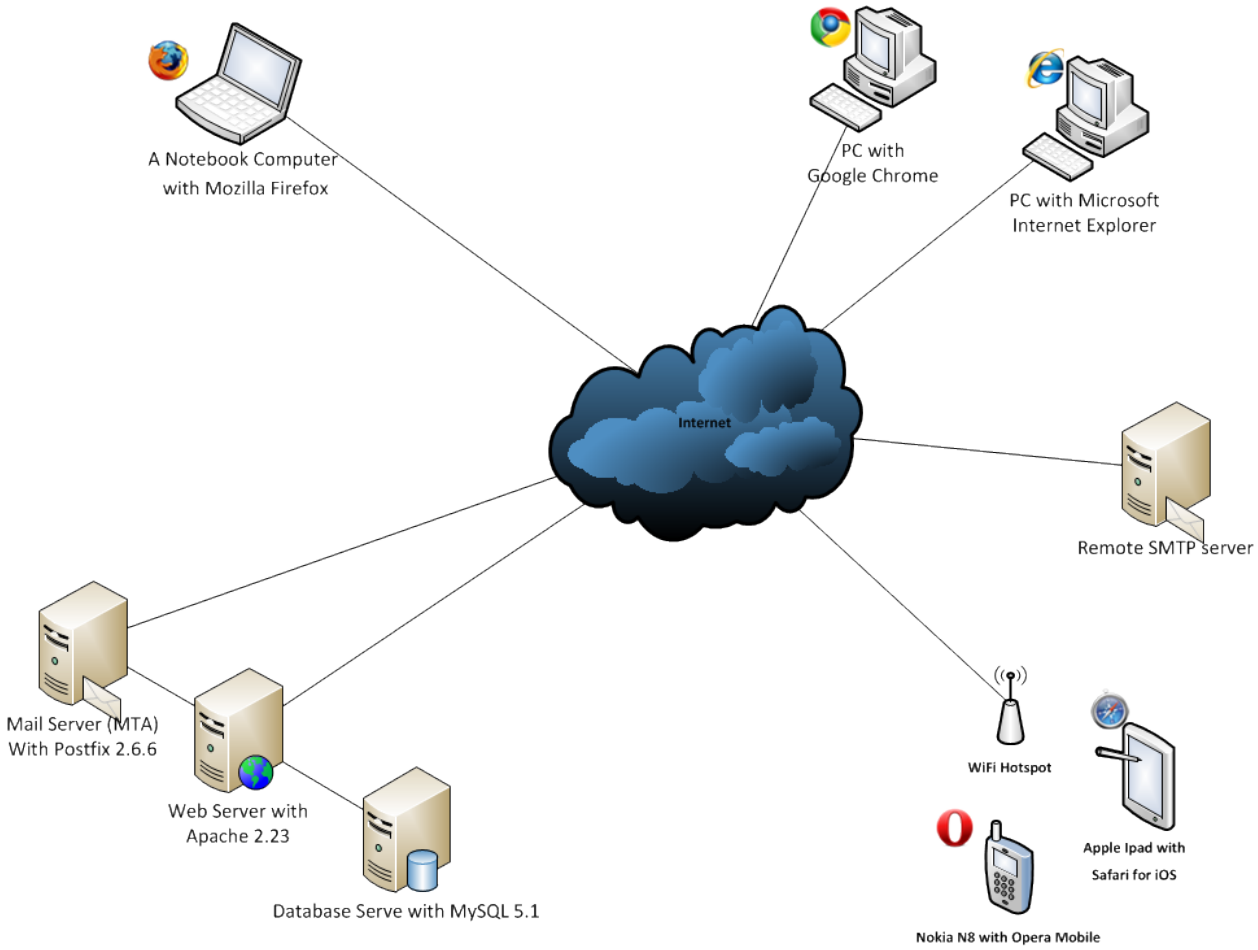
Benefits of using object-oriented architectural style

Understandable. It maps the application more closely to the real world objects, making it more understandable. **Reusable**. It provides for reusability through polymorphism and abstraction.
Testable. It provides for improved testability through encapsulation.
Extensible. Encapsulation, polymorphism, and abstraction ensure that a change in the representation of data does not affect the interfaces that the object exposes, which would limit the capability to communicate and interact with other objects.
Highly Cohesive. By locating only related methods and features in an object, and using different objects for different sets of features, we can achieve a high level of cohesion.

2.3. PHYSICAL ARRANGEMENTS OF DEVICES

Physical arrangement of devices in a typical network

In this diagram, it shows that, the only software a client need is to access this system is a browser.



2.4. DESIGN DECISIONS

2.4.1. MVC (Model-View-Controller) development style is been used

Reason: - MVC can be taken as for a popular and easy to handle web application development style that has the feature of separating the Presentation, Business and Intermediate logics. So using the MVC style, it'll ease the coding and provide well defined interfaces within each logic.

2.4.2. Object Oriented Software development method is used

Reason: - Since this system is used mostly within a specific set of users, we're not expecting millions of hits in to the application within few milliseconds. Rather than we're focusing on the extensibility of the system which will be really helpful for further developments according to requirements. So Object Oriented style is slightly behind from the perspective of performance, it'll satisfy the main target described above.

2.4.3. AJAX (Asynchronous JavaScript and XML) is used

Reason: - AJAX allows web pages to be updated asynchronously by exchanging small amounts of data with the server behind the scenes. So it provides better interactivity to their users. This is due to the fact that implementing AJAX on a website does not require a page to be reloaded for dynamic content on web pages.

3. COMPONENT AND DETAIL DESIGN

3.1. DESIGN PATTERNS

3.1.1. Prototype Pattern

Declare an abstract base class that specifies a pure virtual “clone” method, and, maintains a dictionary of all “cloneable” concrete derived classes. Any class that needs a “polymorphic constructor” capability: derives itself from the abstract base class, registers its prototypical instance, and implements the clone()operation.

Reason for using Prototype pattern:
When applying access control list enforcements to the user, there will be an abstract model class that break in to two concrete model classes name ARO and ACO. These classes can get in to work by making clone() from the abstract model class.

3.1.2. Observation Pattern

Basically this pattern defines an object that is the “keeper” of the data model or business logic (the Subject). Delegate all “view” functionality to decoupled and distinct Observer objects. Observers register themselves with the Subject as they are created. Whenever the Subject changes, it broadcasts to all registered Observers that it has changed, and each Observer queries the Subject for that subset of the Subject’s state that it is responsible for monitoring.

Reason for using Observer pattern:
In our projects' MVC architectural pattern, The separation of Models and Views done through the decoupling method. So that enhance to ability of increasing flexibility and reusing of code in a way that it reduce the redundant coding. In that fact, we identified that it is very efficient to decouple objects so that changes to one can affect any number of others without requiring the changed object to know details of the others, Simply This allows the number and “type” of “view” objects to be configured dynamically, instead of being statically specified.

3.1.2. Composite Design Pattern

The Composite is known as a **structural** pattern, as it's used to form large object structures across many disparate objects. This pattern allows us to set up a tree structure and ask each element in the tree structure to perform a task. After the tree structure is established, we can then ask each element, to perform a common operation.

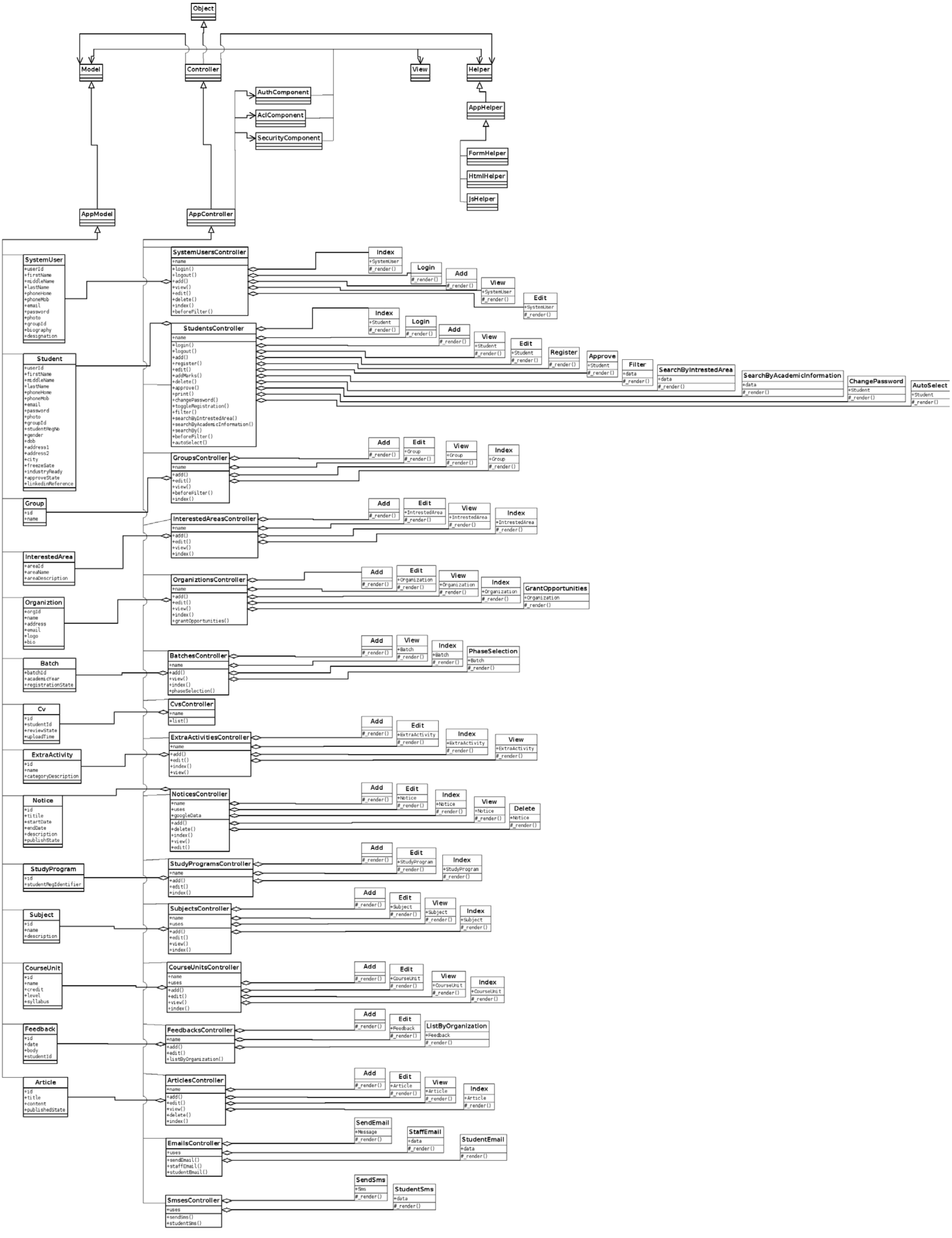
Reason for using Composite pattern:
CakePHP has the native support of view elements that can be added separately in to a view which performs a specific view task. For an example we can have specifically implemented set of buttons under an “index” view, where we need navigation to different pages. It clearly specifies that the view “index” is nesting various types of button elements.

3.1.2. Strategy Design Pattern

This design pattern is very useful when n it comes to define a family of algorithms, encapsulate each one and make them interchangeable. It allows us to change the algorithm independently with out changing the client using it. It converts the generalization of the template method to composition or aggregation.

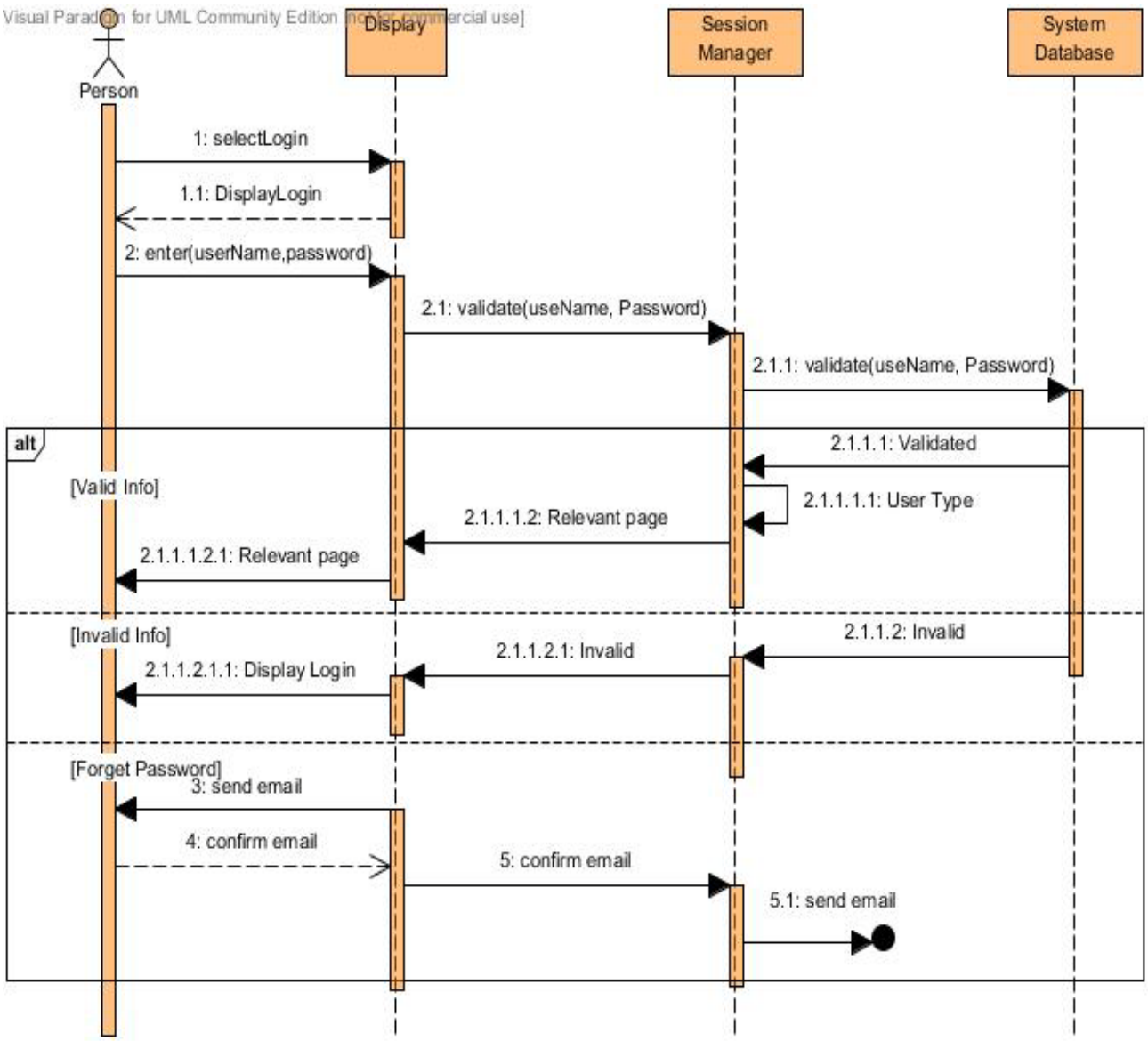
Reason for using Strategy pattern:
In our planned development in MVC pattern, the View-Controller relationship can be maintained by burying the complex algorithms inside the controllers and let the views to render only the content passed by the controllers. It's useful when we want to replace the algorithm either statically or dynamically, when we have a lot of variants of the algorithm for example the filtering scenario of students using diffident algorithmic factors, or when the algorithm has complex data structures that we want to encapsulate.

3.2. CLASS DIAGRAM

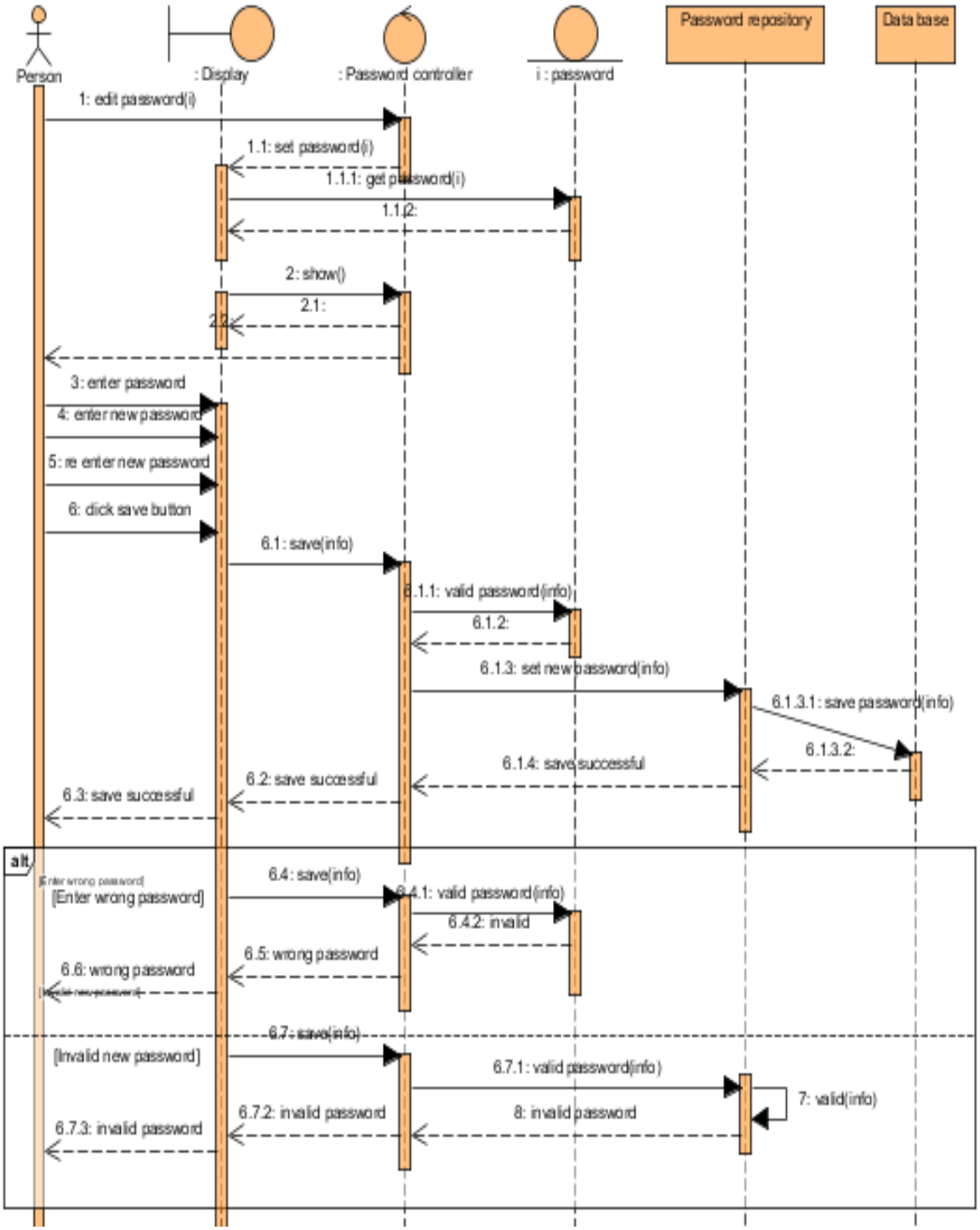


3.3. SEQUENCE DIAGRAMS

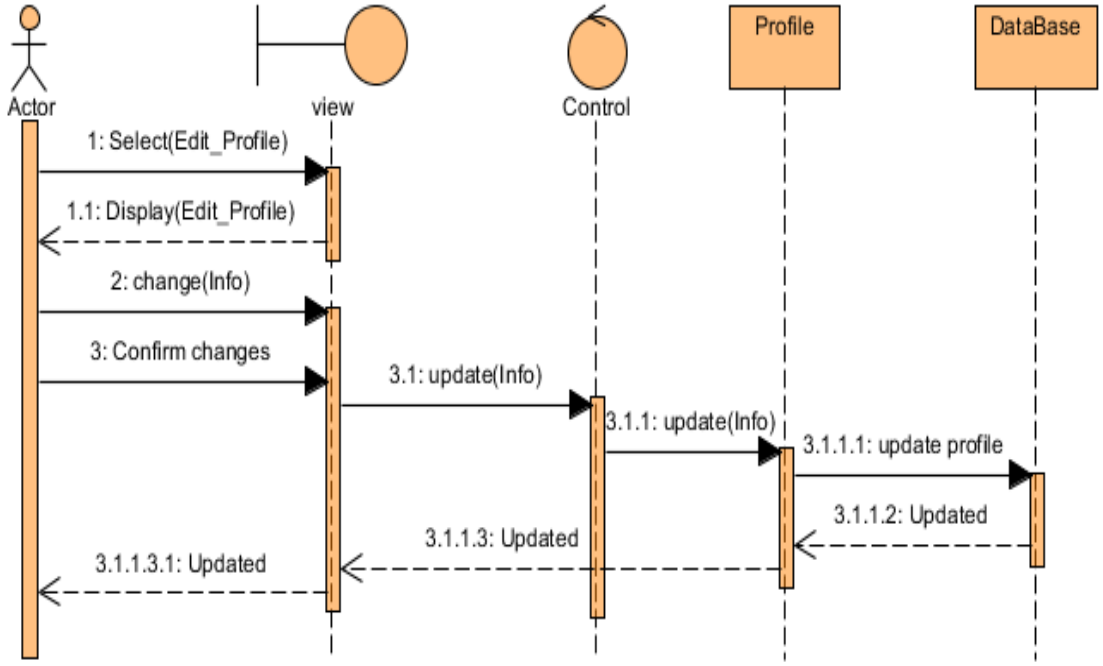
3.3.1 Use case 1: Login



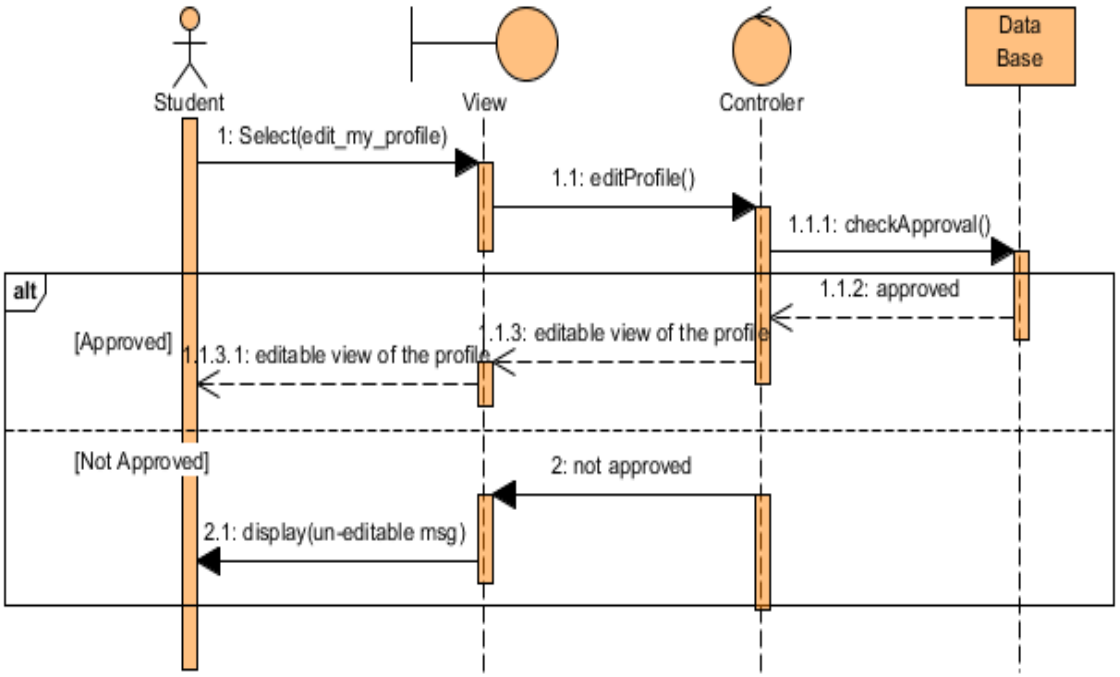
3.2.2 Use case 2: Change password



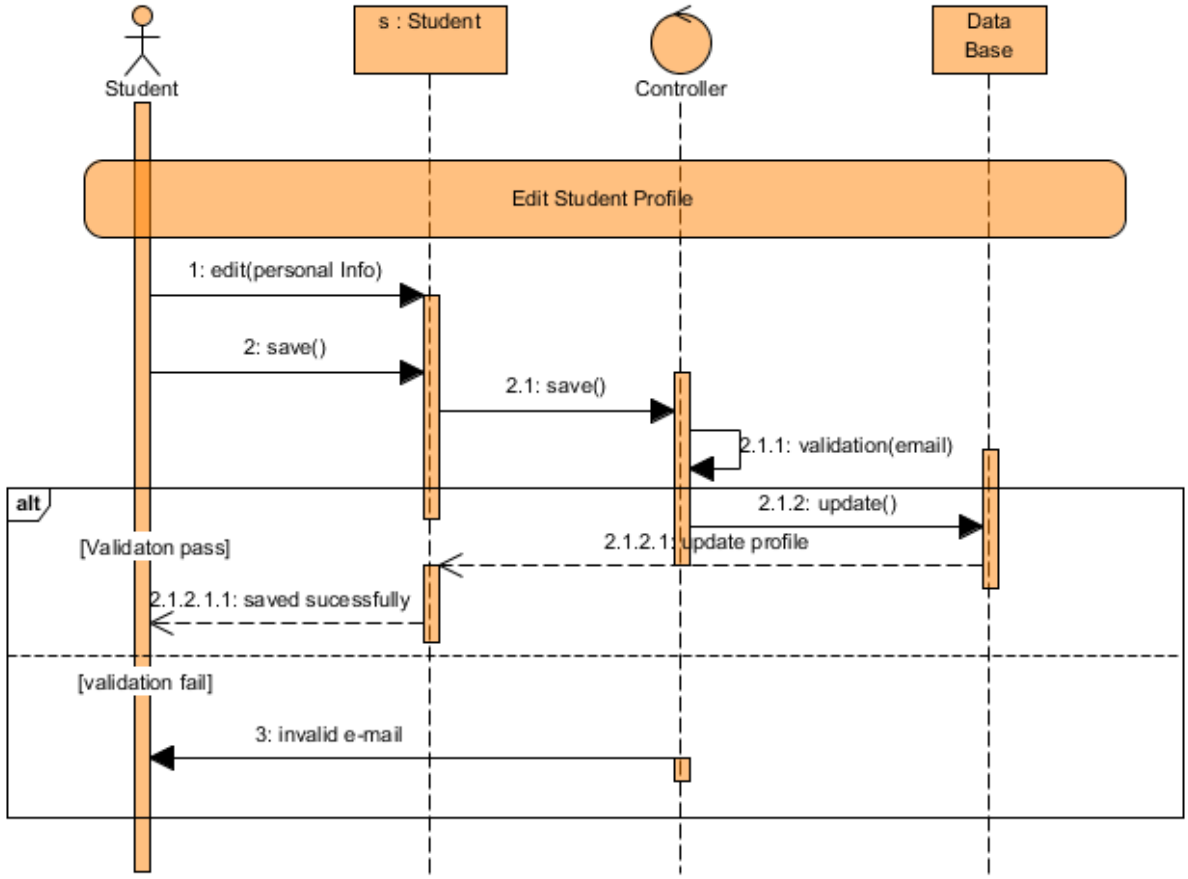
3.2.3 Use case 3: Edit System User Profile



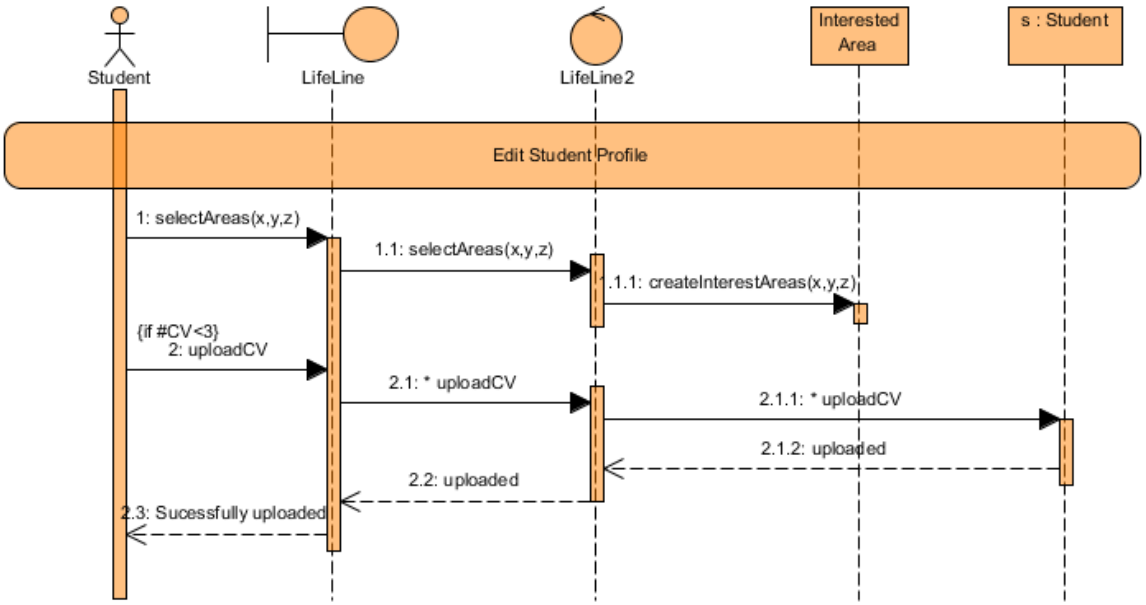
3.2.4 Use case 4: Edit Student Profile



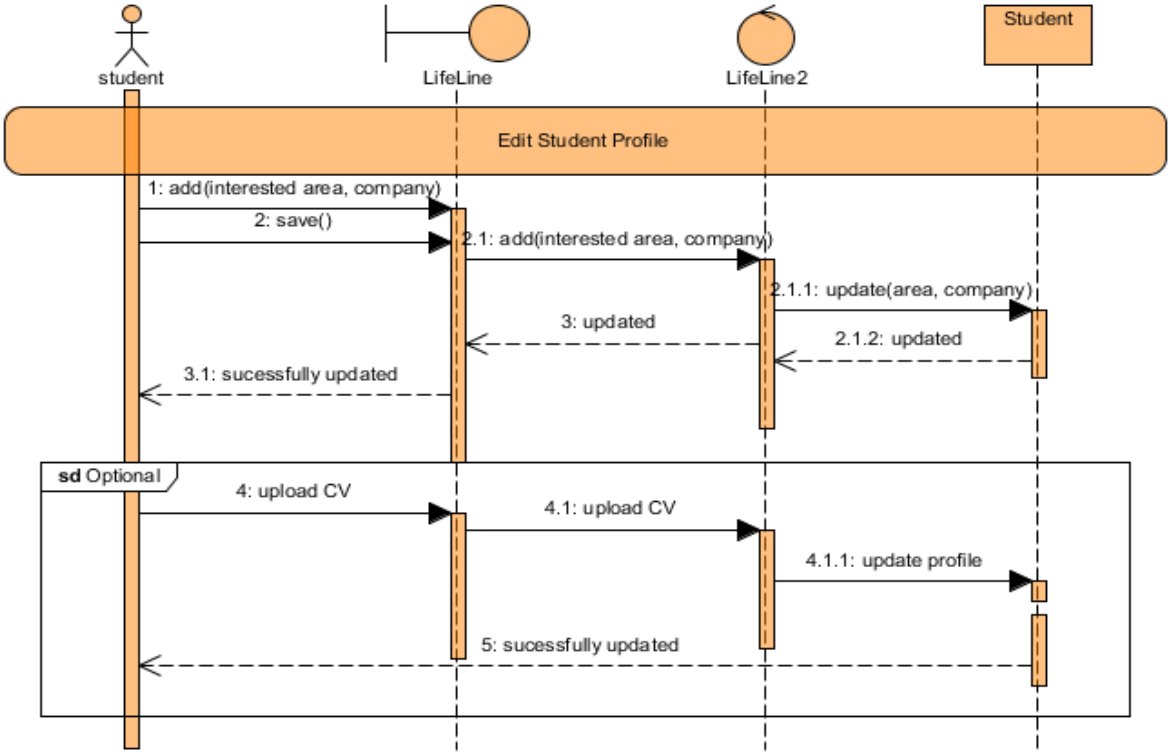
3.2.5 Use case 5: Edit Student Personal Information



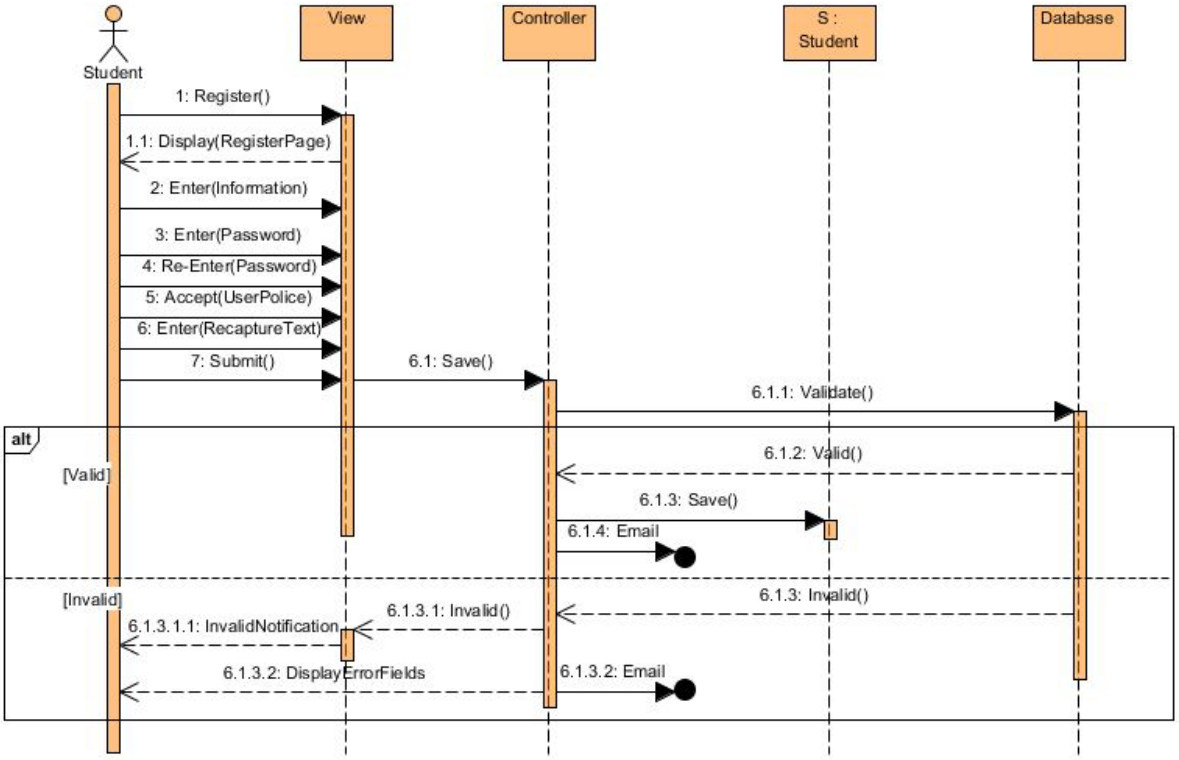
3.2.6 Use case 6: Edit Student Initial Choices



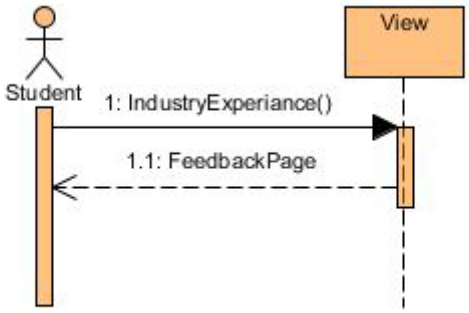
3.2.7 Use case 7: Student Choices Finalizing



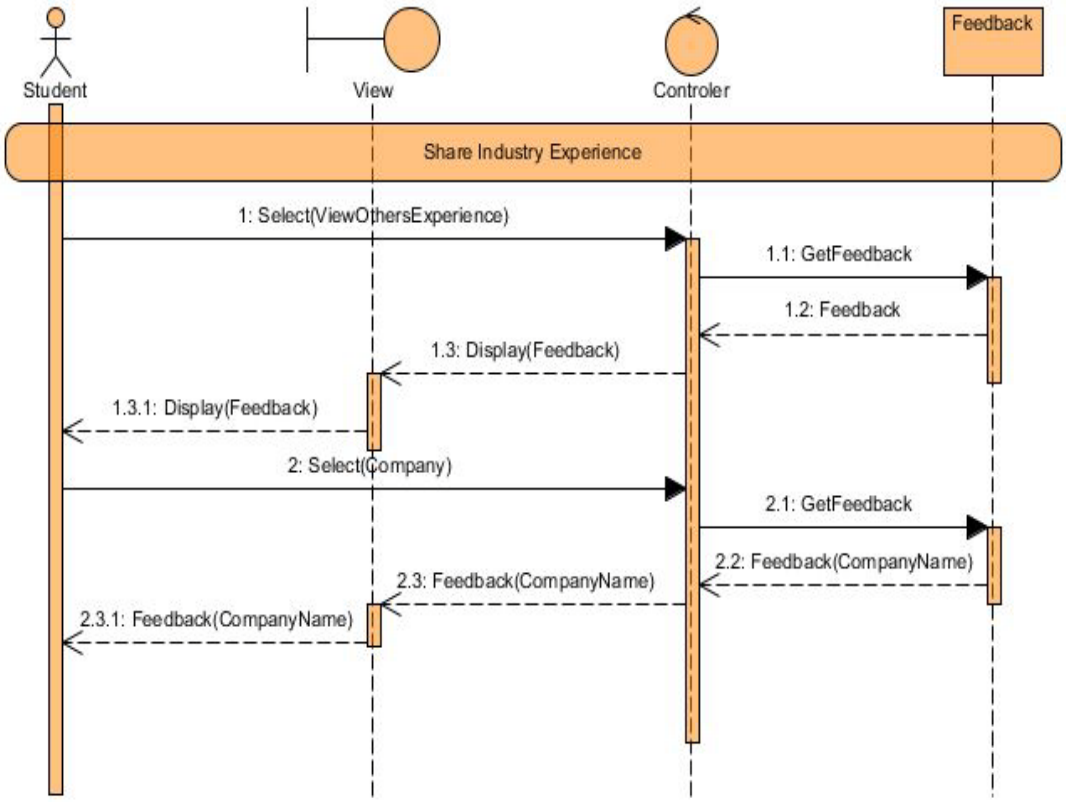
3.2.8 Use case 8: Registration



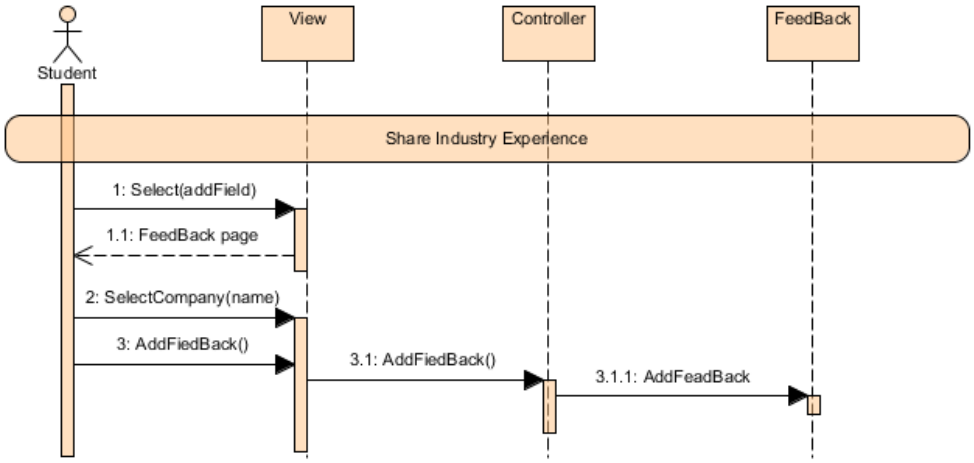
3.2.9 Use case 9: Share Industry Experience



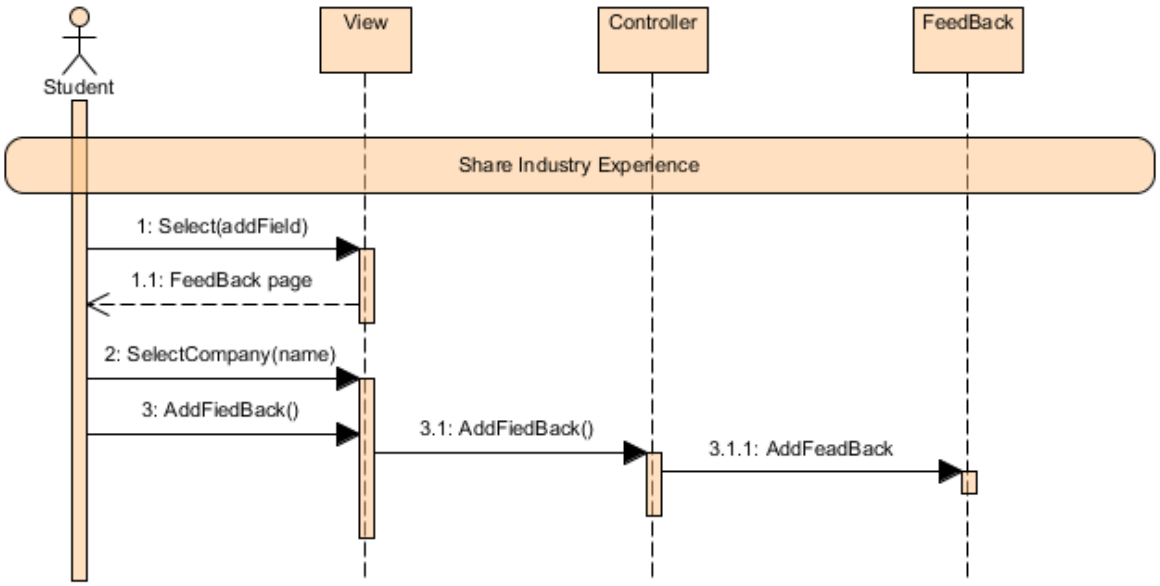
3.2.10 Use case 10 : View others experience



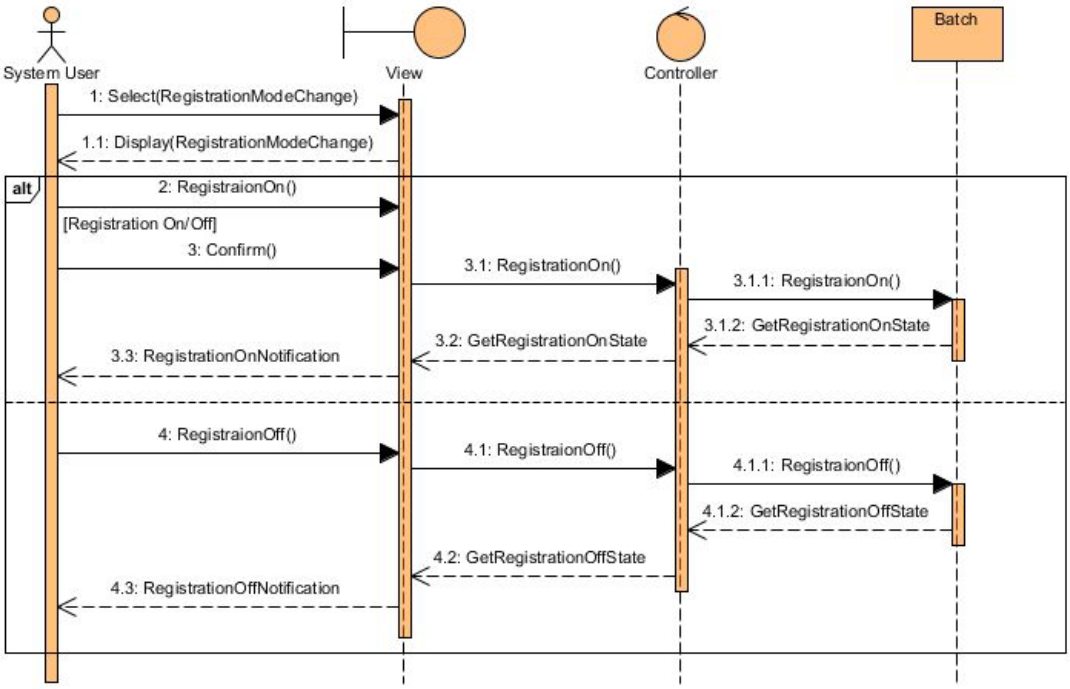
3.2.11 Use case 11 : Add feedbacks



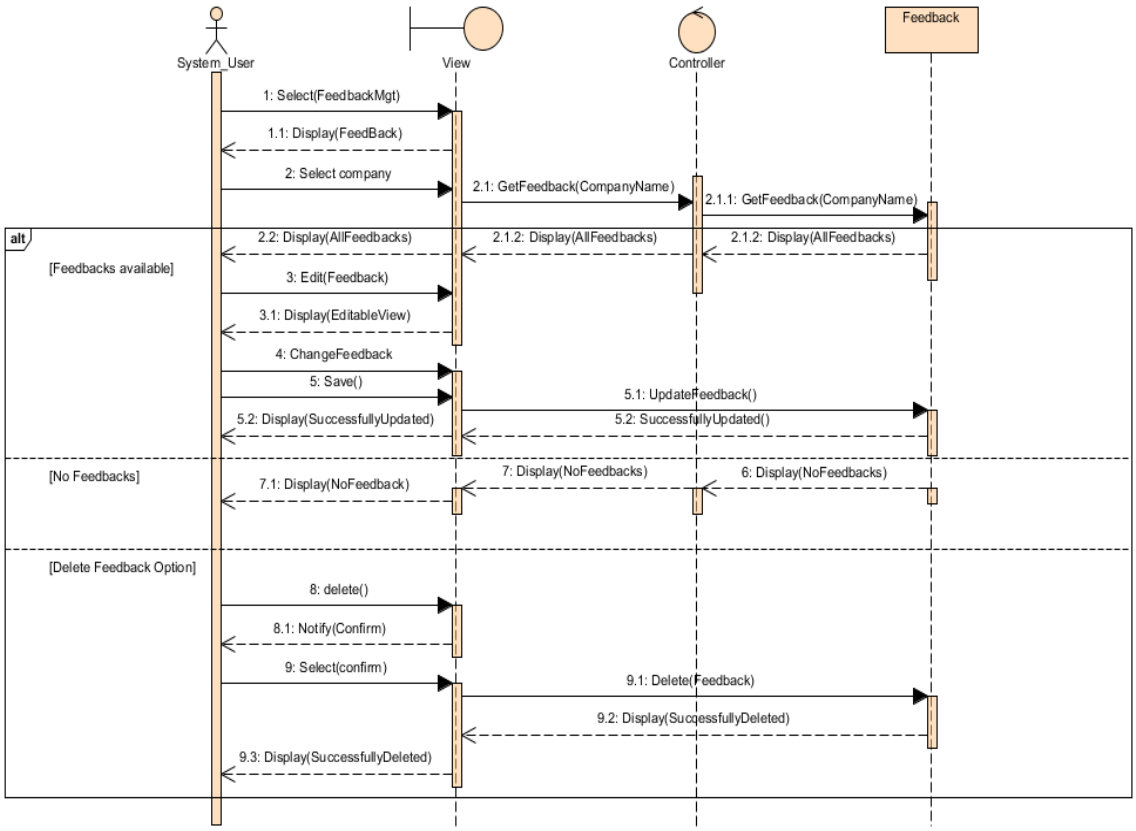
3.2.12 Use case 12 : Edit feedback



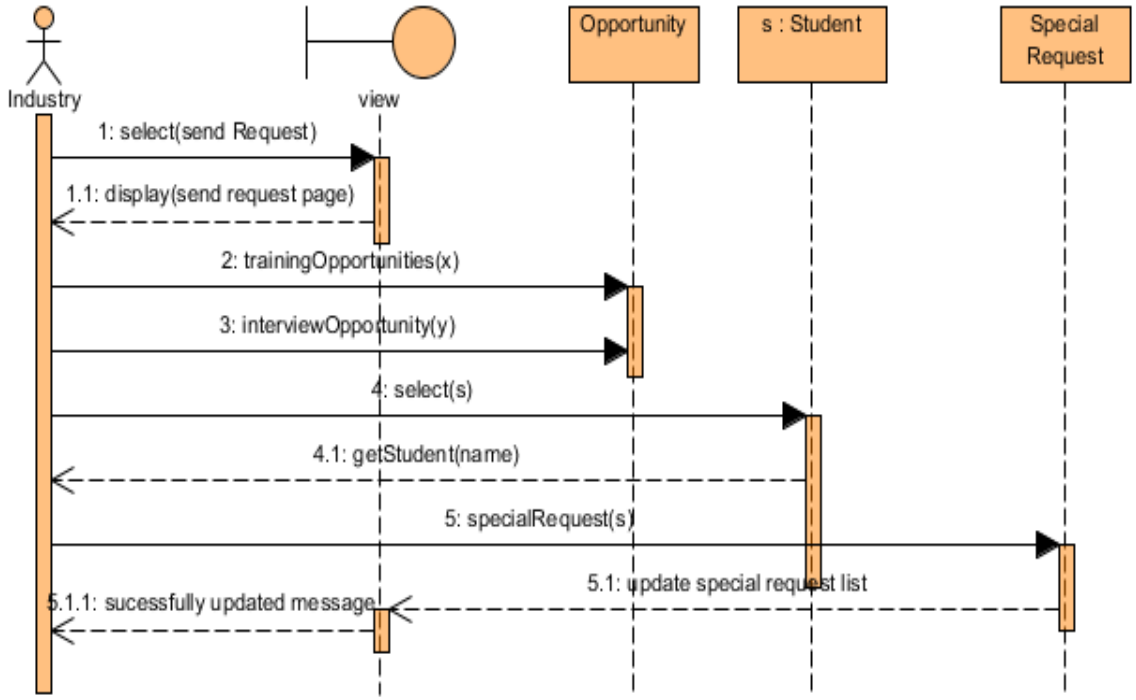
3.2.13 Use case 13 : Registration on/off



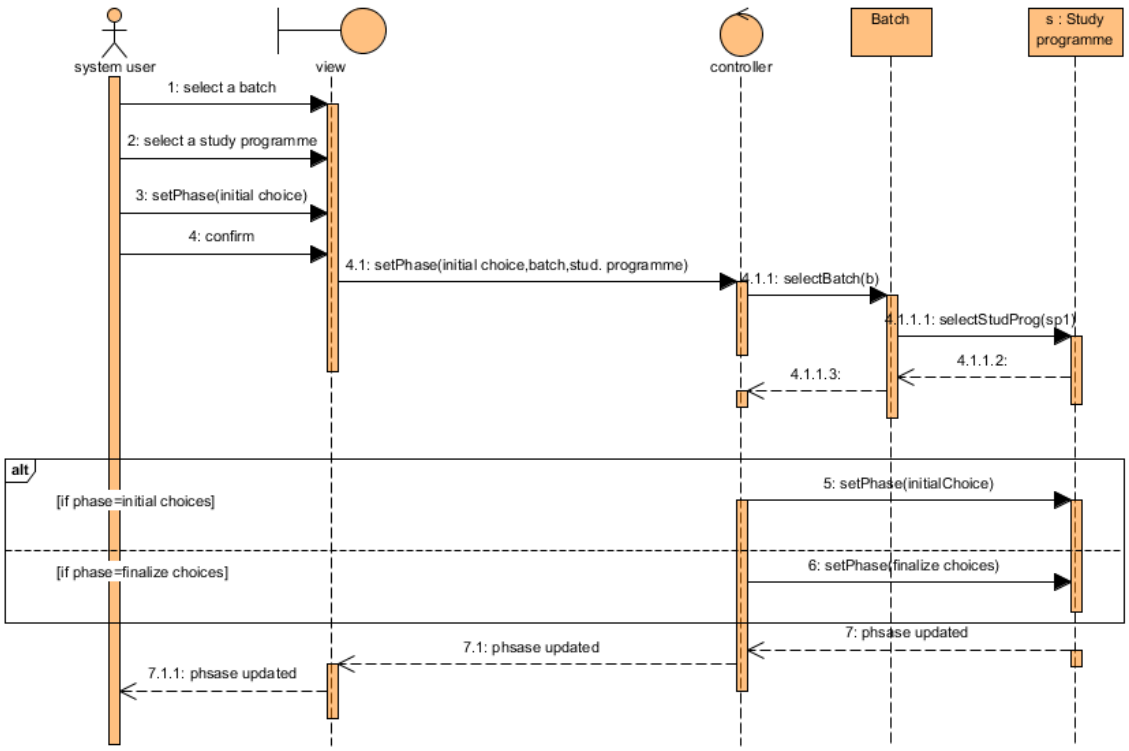
3.1.14 Use case 14 : Feedback management



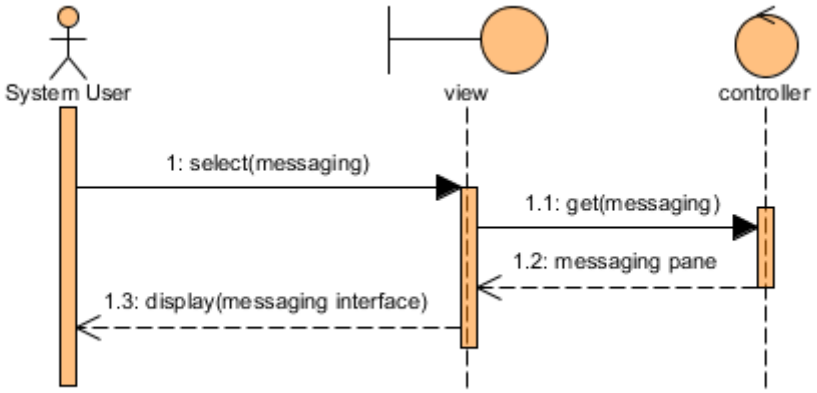
3.2.15 Use case 15 : Industry Grant Opportunity



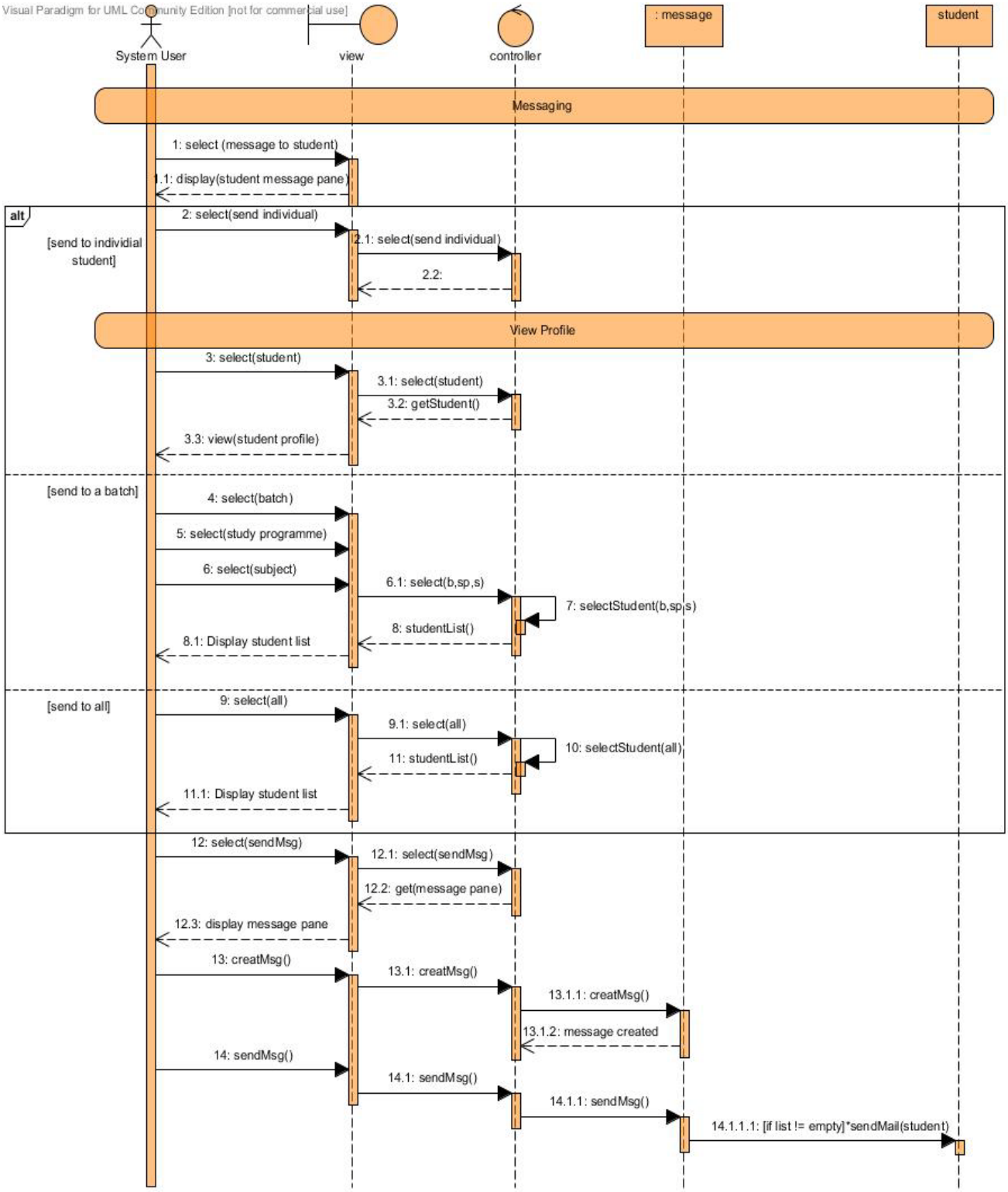
3.2.16 Use case 16 : Category phase selection



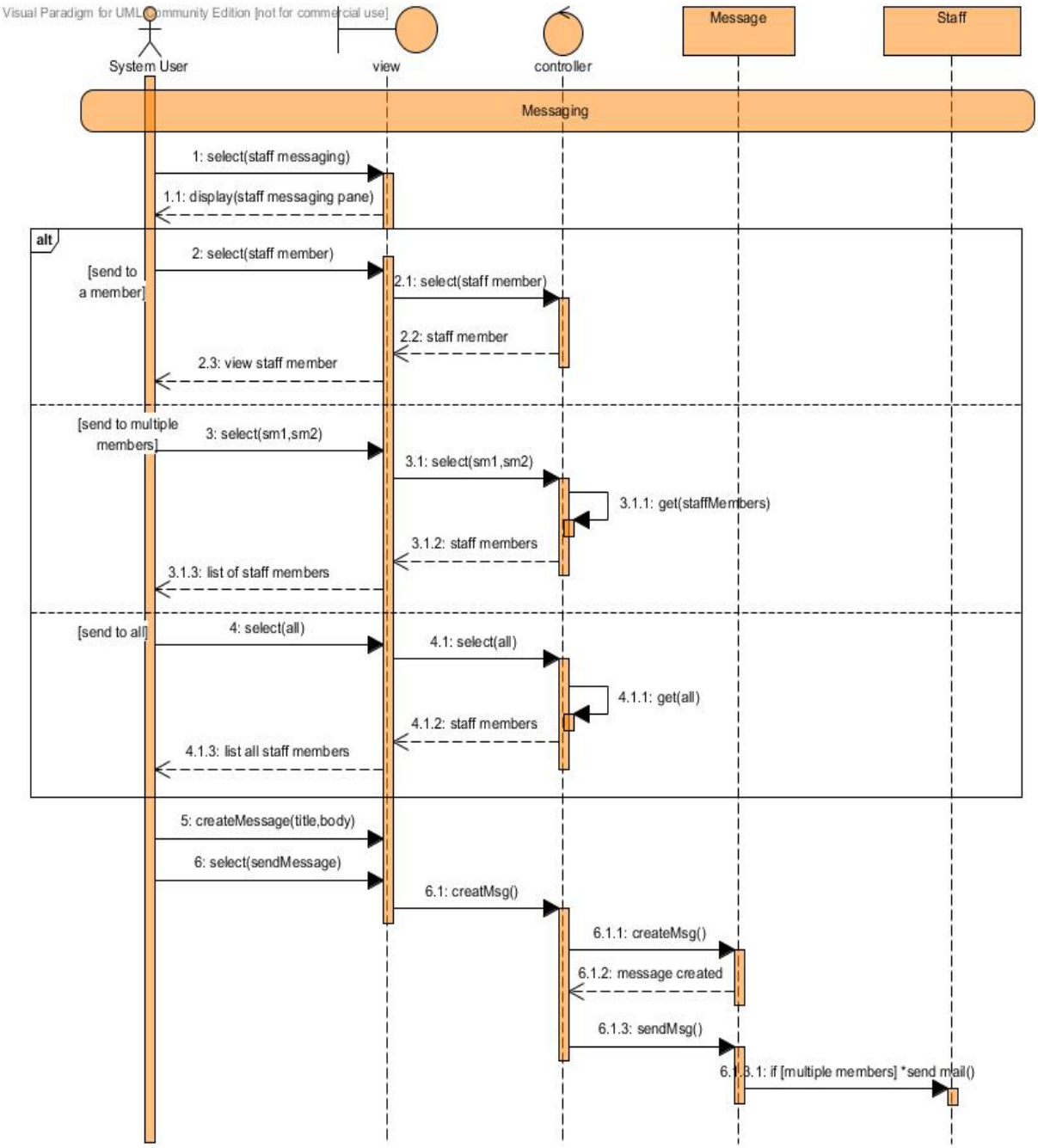
3.2.17 Use case 17 : Messaging



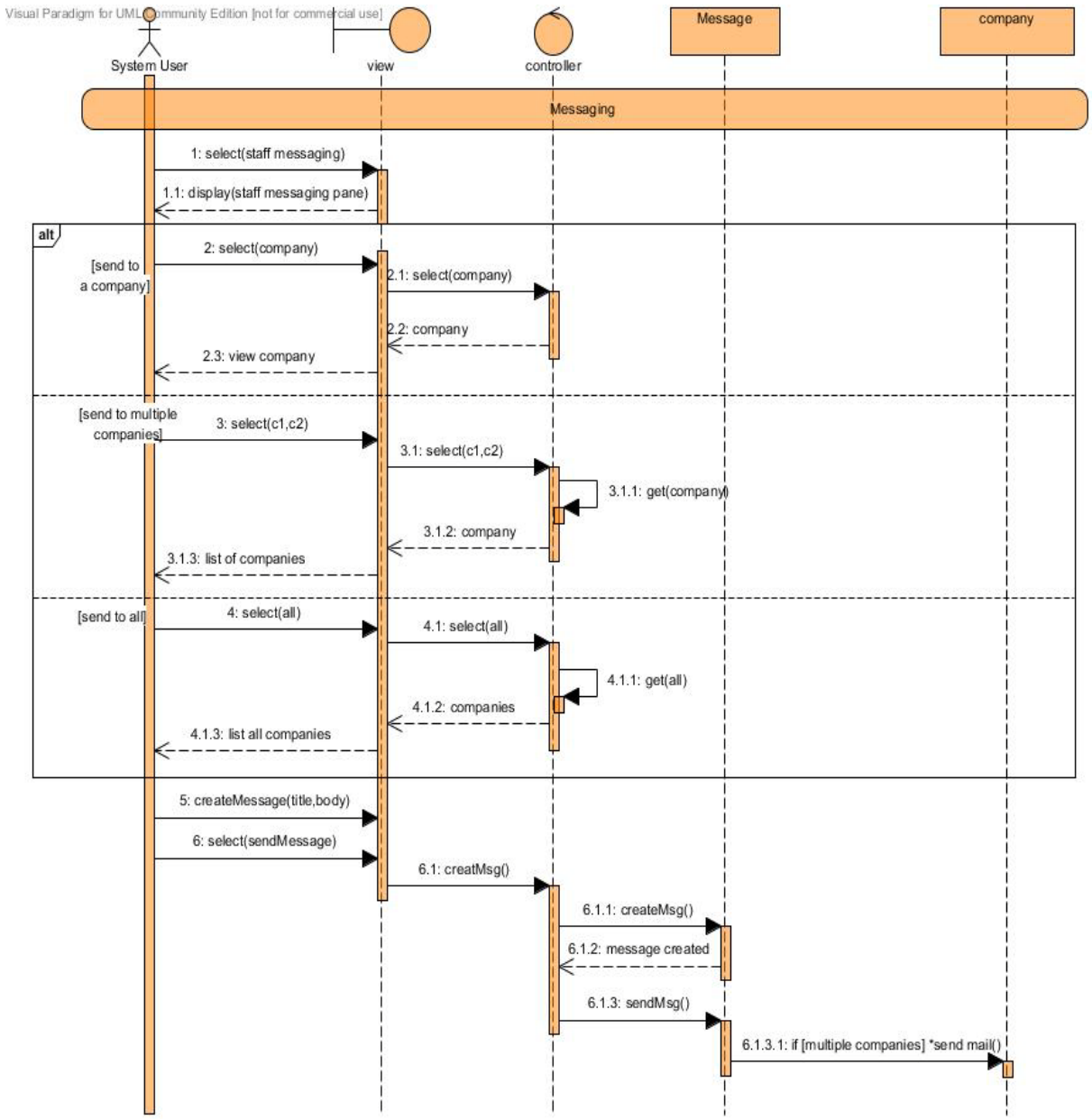
3.2.18 Use case 18 : Messaging to students



3.2.19 Use case 19 : Messaging to staff



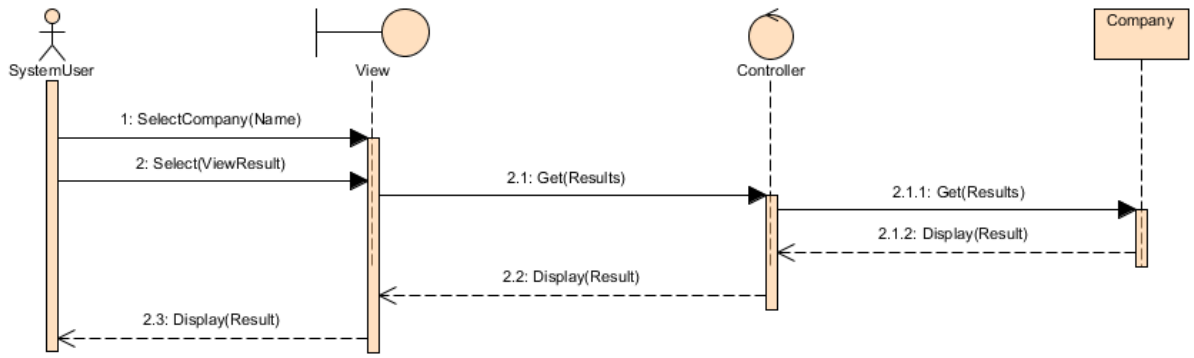
3.2.20 Use case 20 : Messaging to company



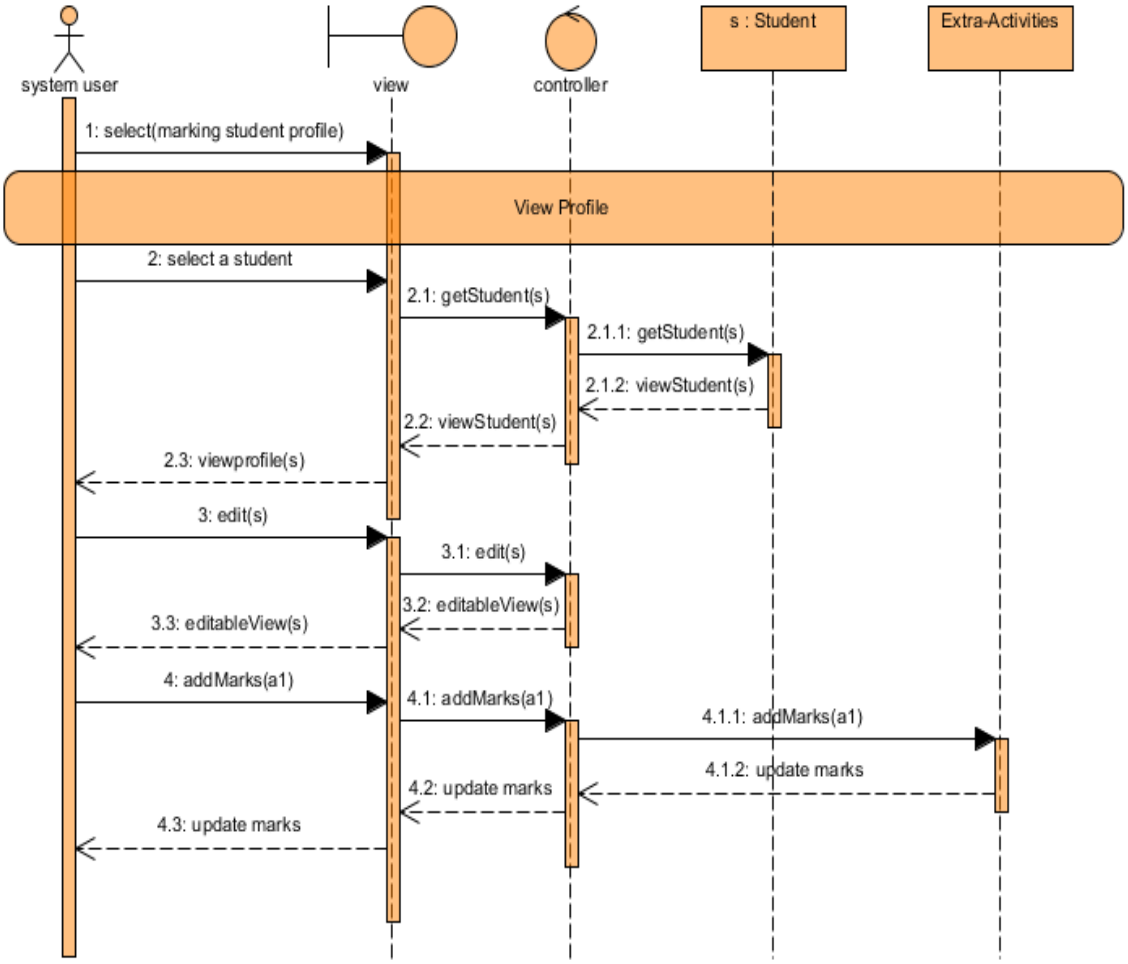
3.2.21 Use case 21 : Auto calculate student

Auto calculation process is depending on an algorithm.

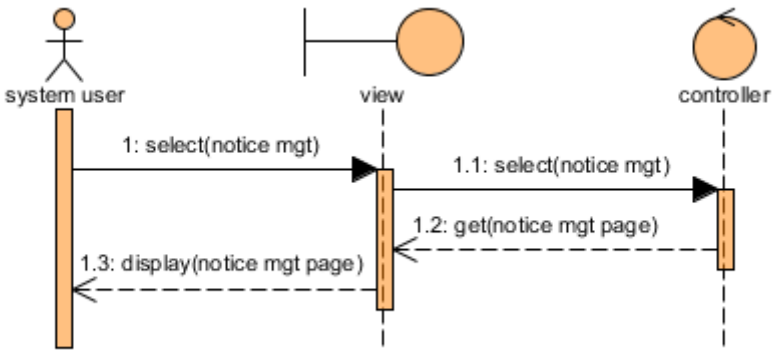
3.2.22 Use case 22 : View auto calculated result



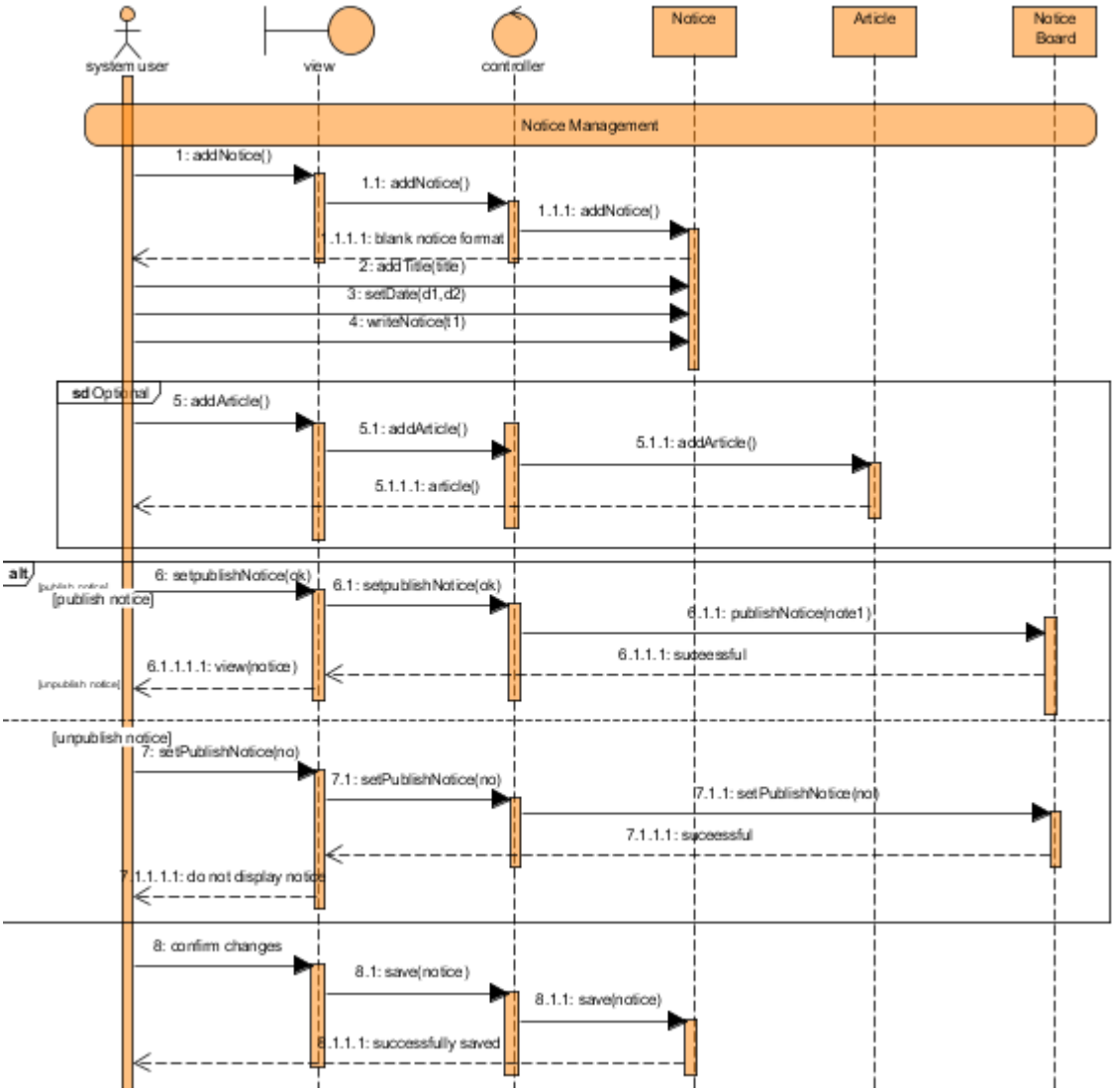
3.2.23 Use case 23 : Add marks to extra activities



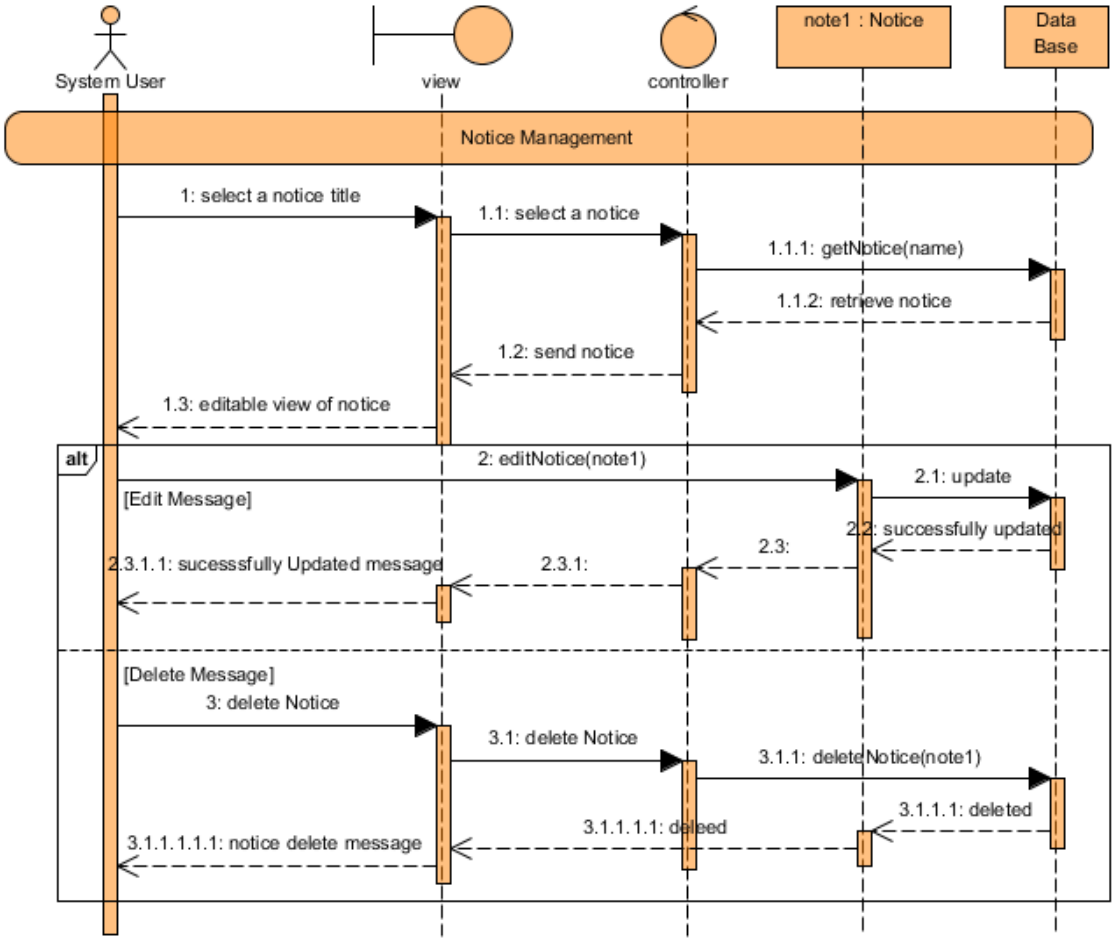
3.2.24 Use case 24 : Notice management



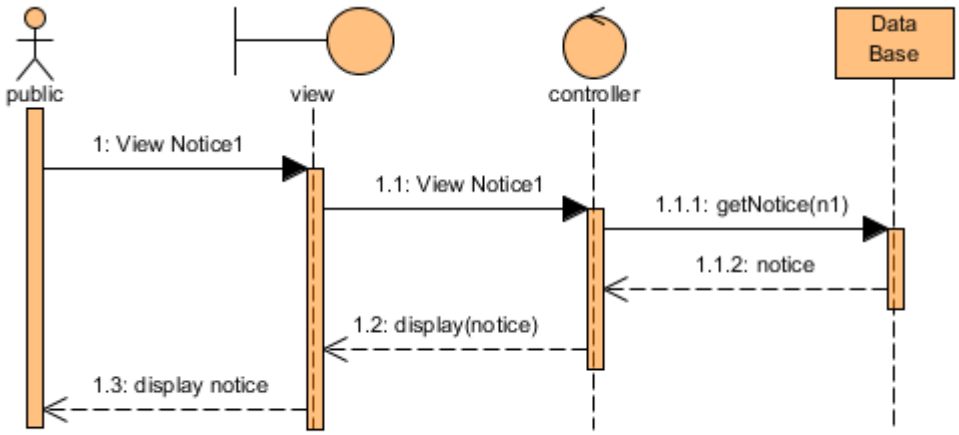
3.2.25 Use case 25 : Notice creation



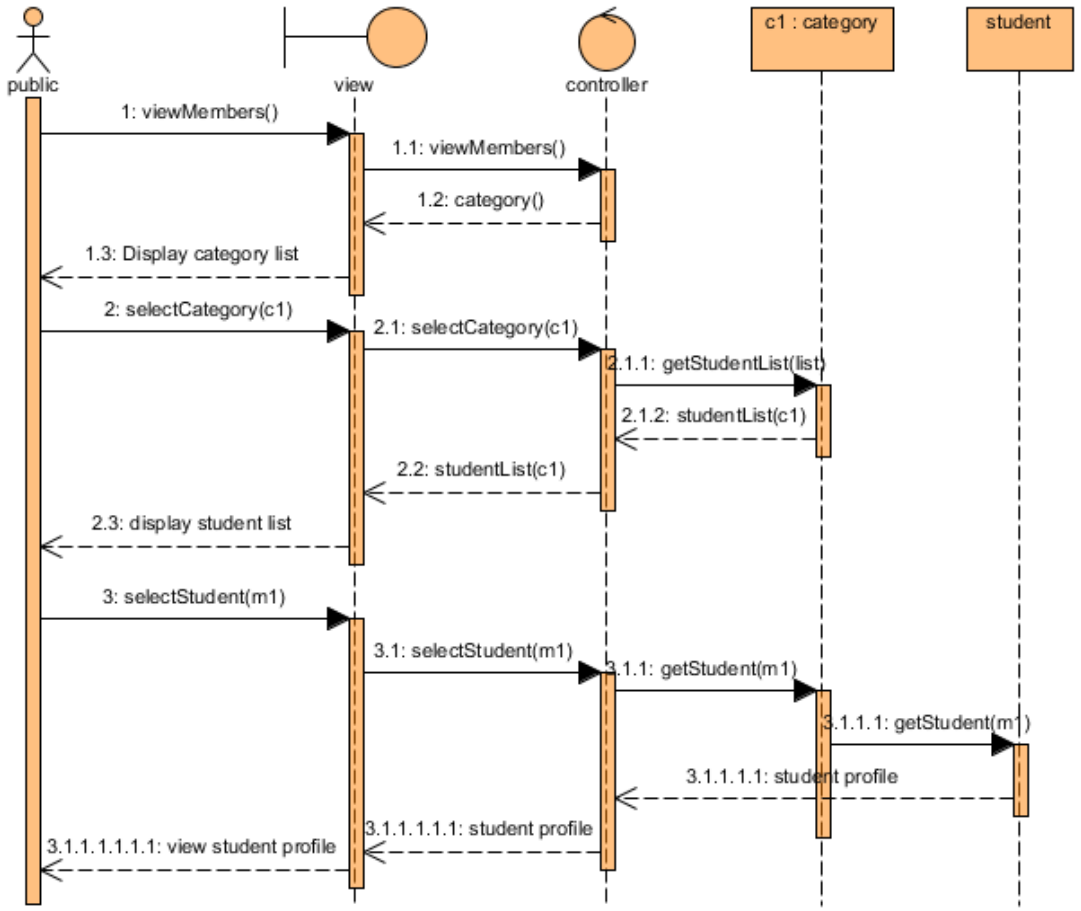
3.2.26 Use case 26 : Notice modify



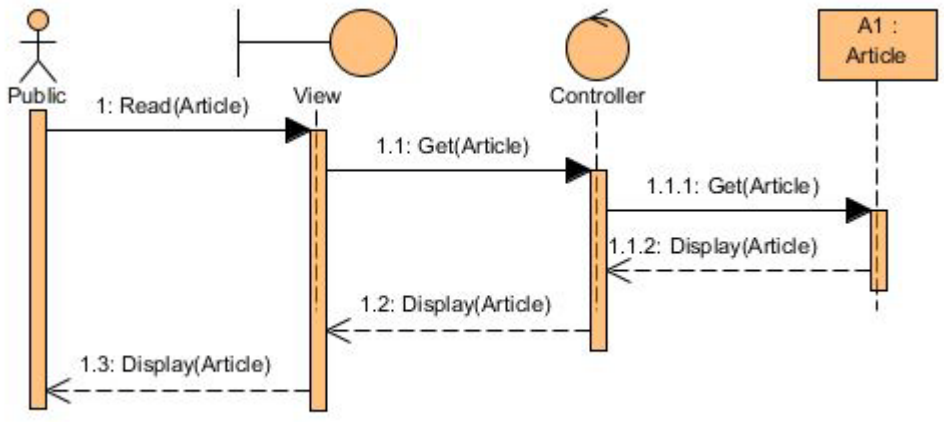
3.2.27 Use case 27 : Public view notice



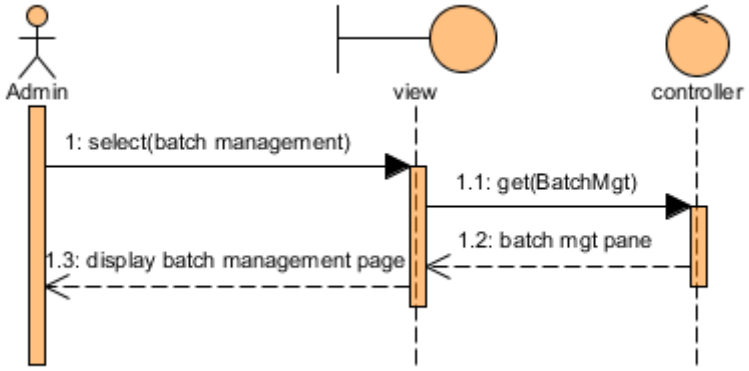
3.2.28 Use case 28 : Public view members



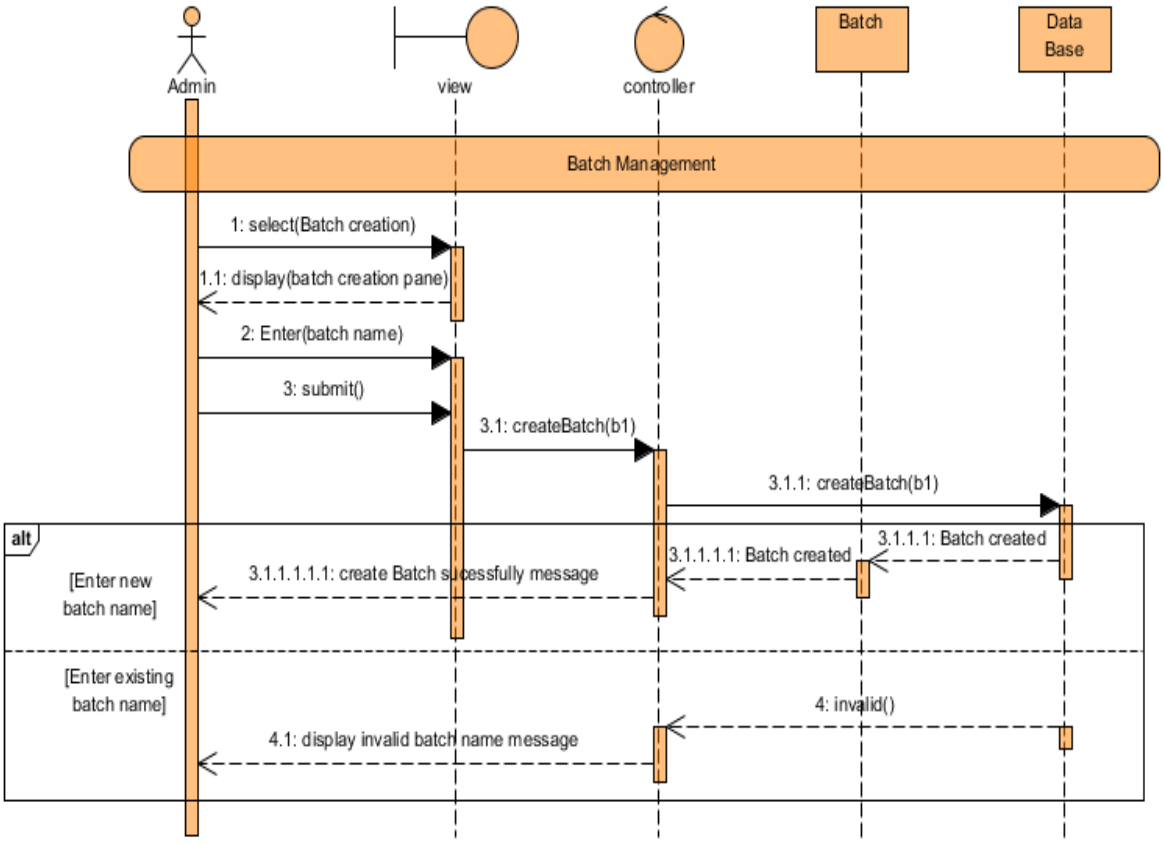
3.2.29 Use case 29 : Public view articles



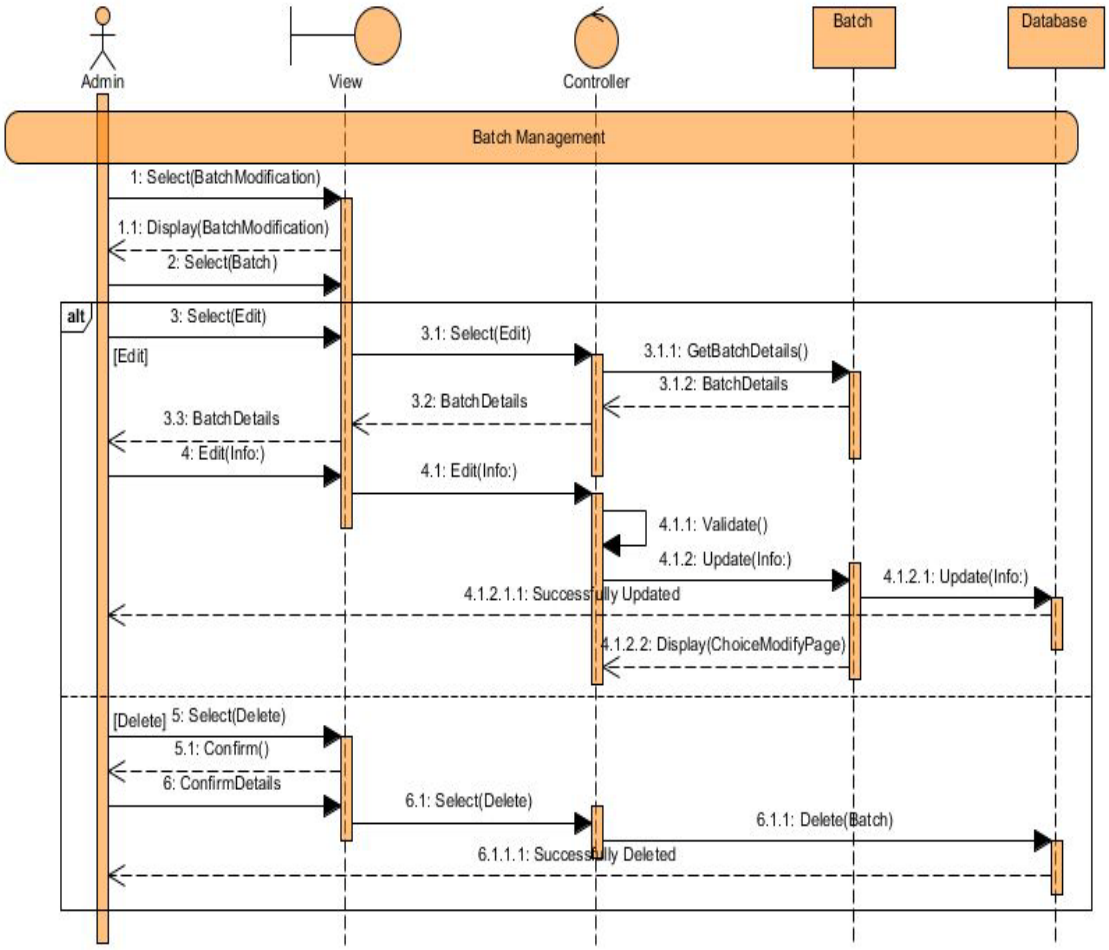
3.2.30 Use case 30 : Category Management



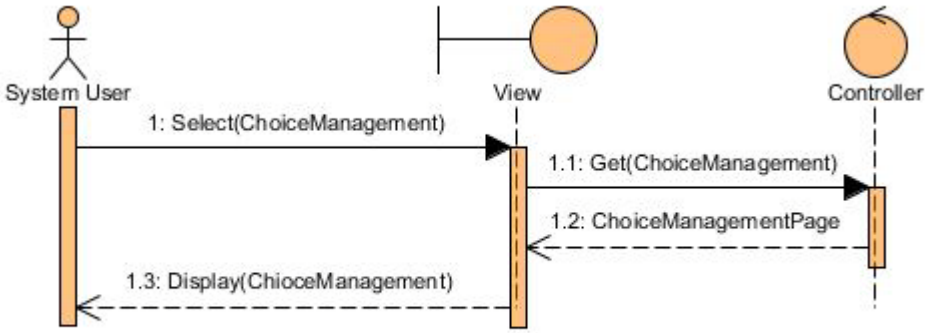
3.2.31 Use case 31 : Create Category



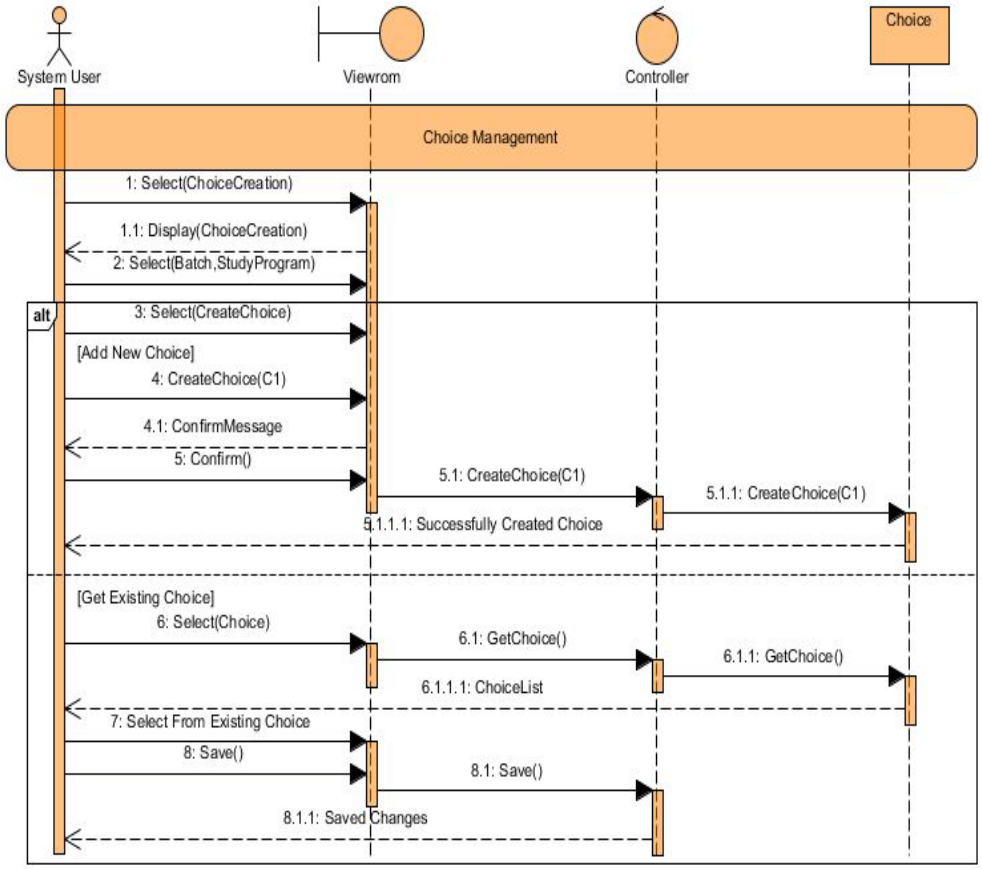
3.2.32 Use case 32 : Category Modification



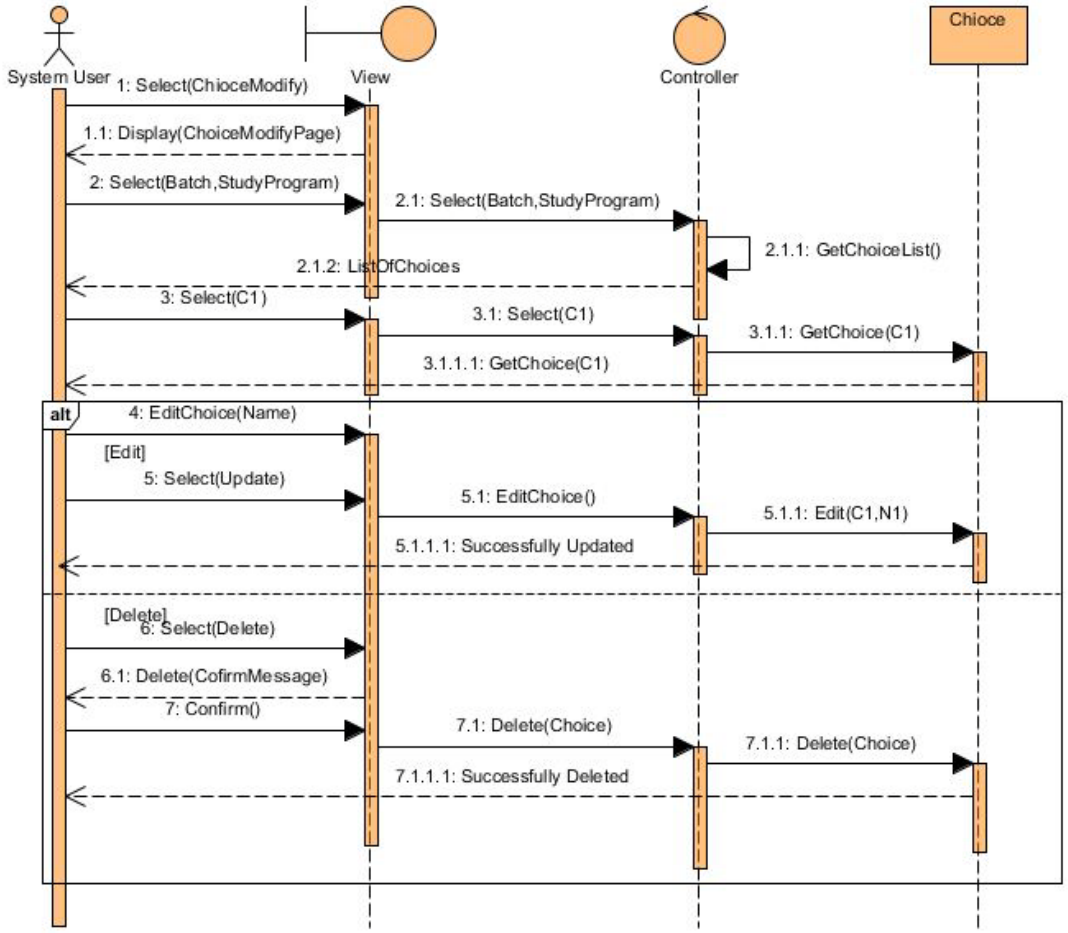
3.2.33 Use case 33 : Choice Management



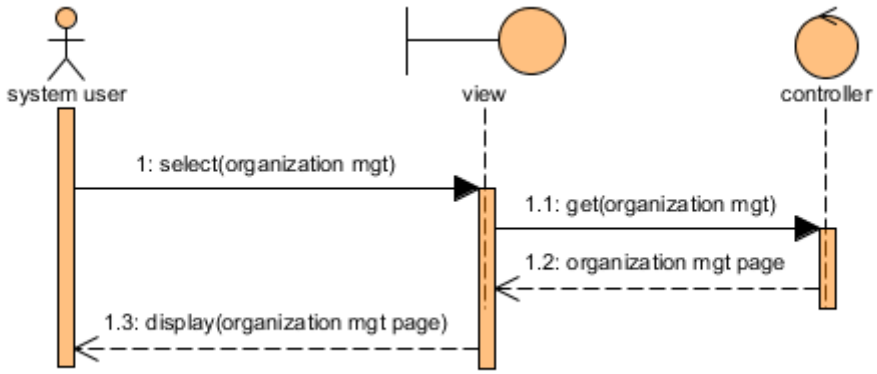
3.2.34 Use case 34 : Choice Creation



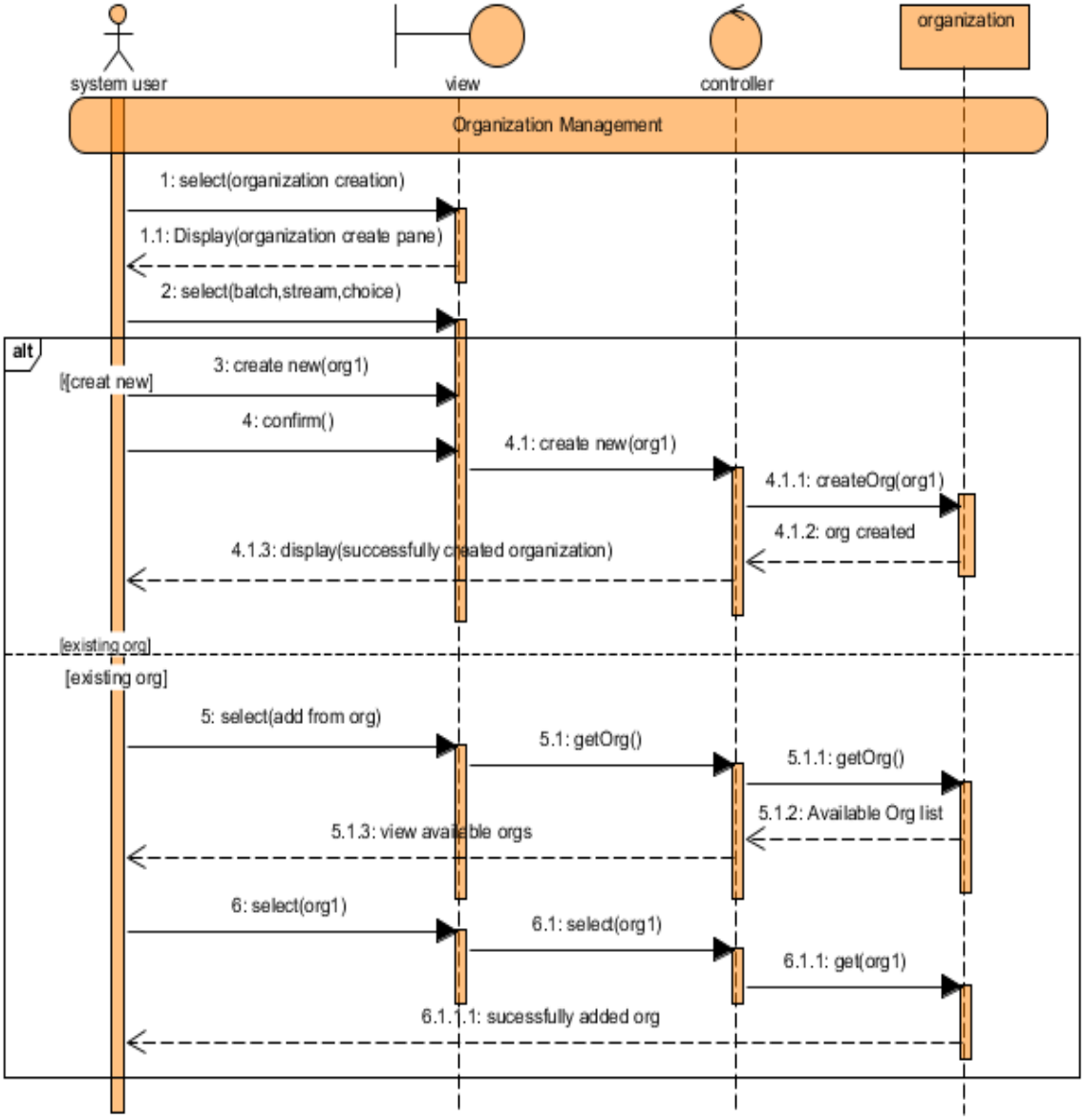
3.2.35 Use case 35 : Choice Modify



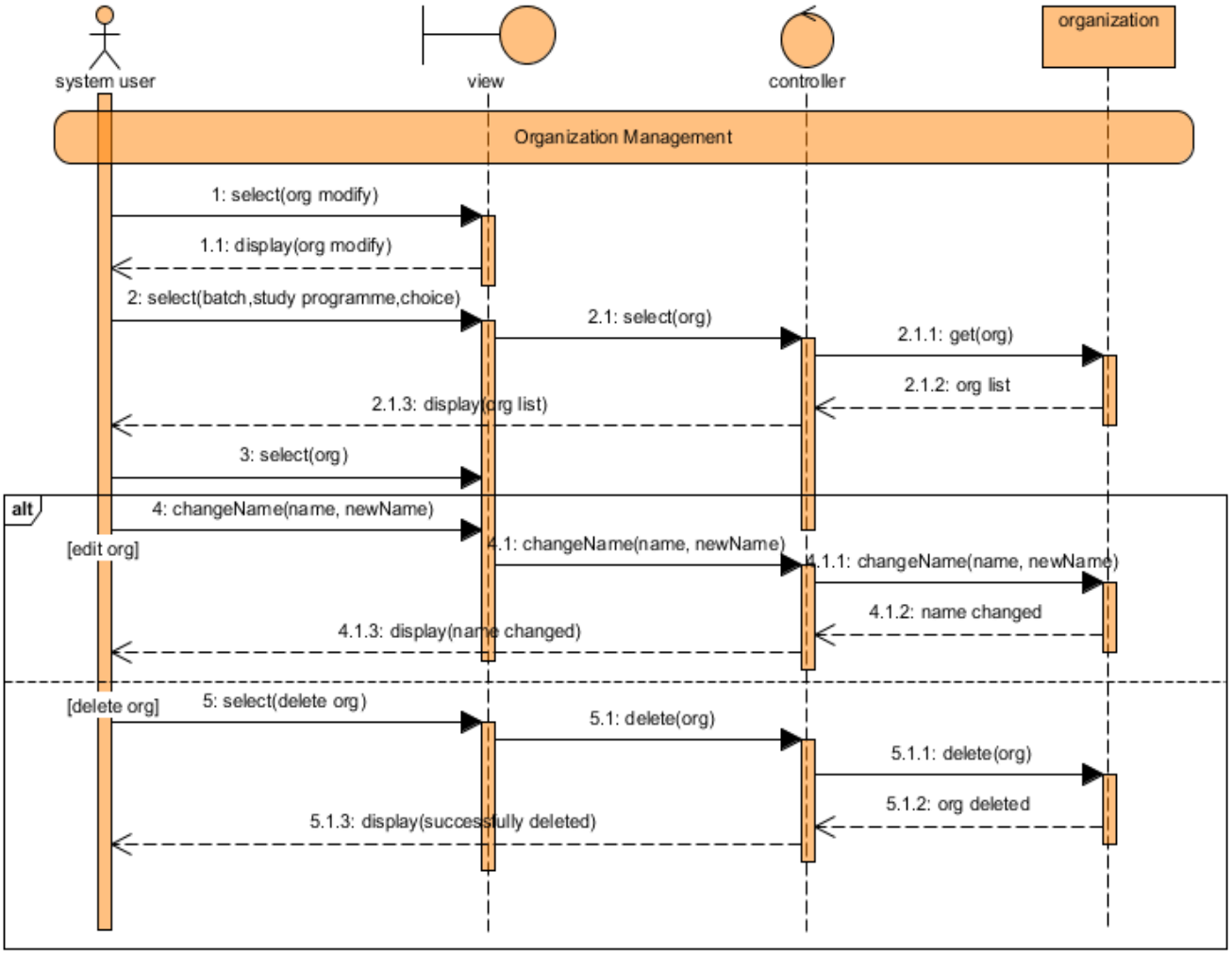
3.2.36 Use case 36 : Organization Management



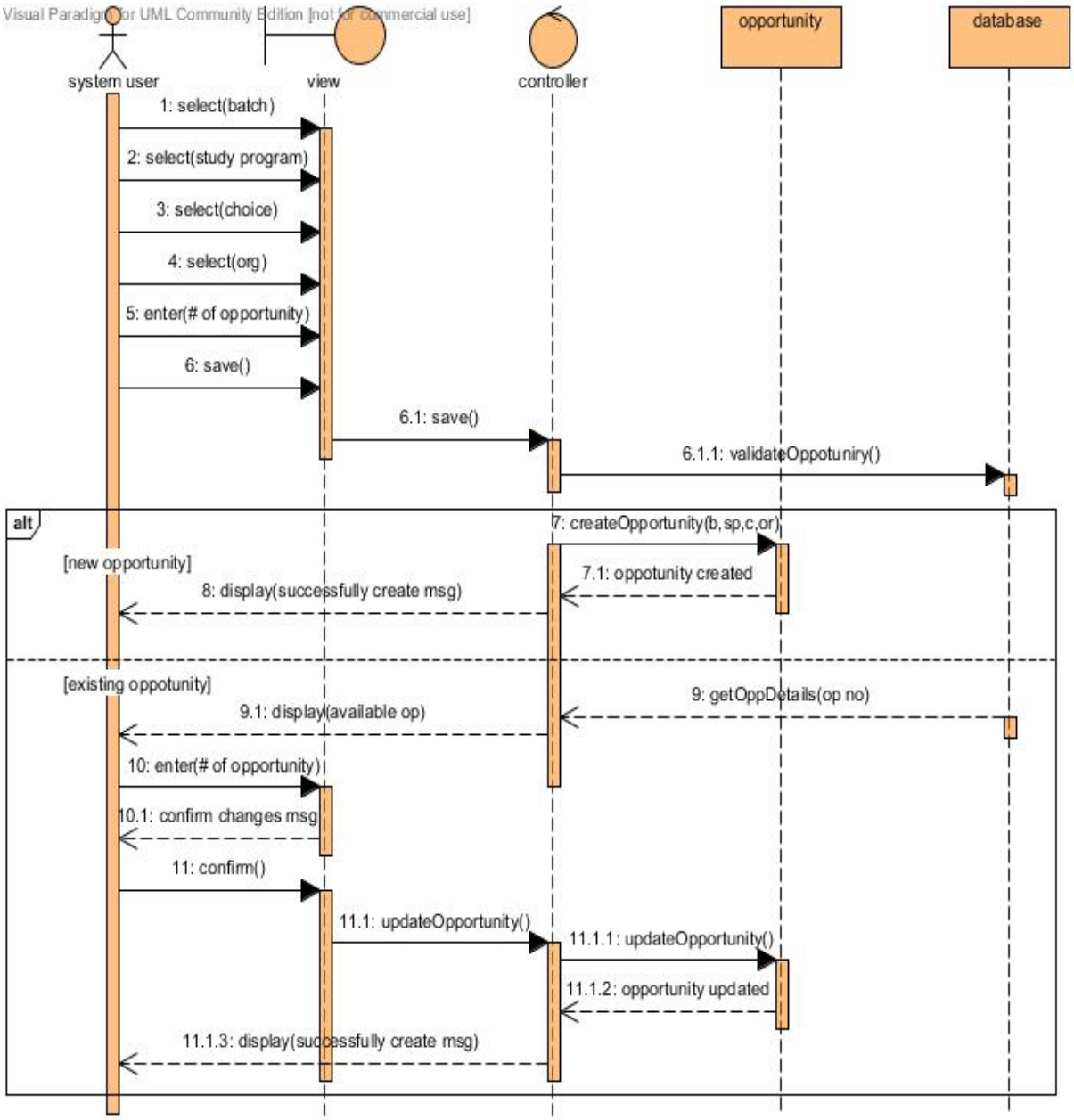
3.2.37 Use case 37 : Organization Creation



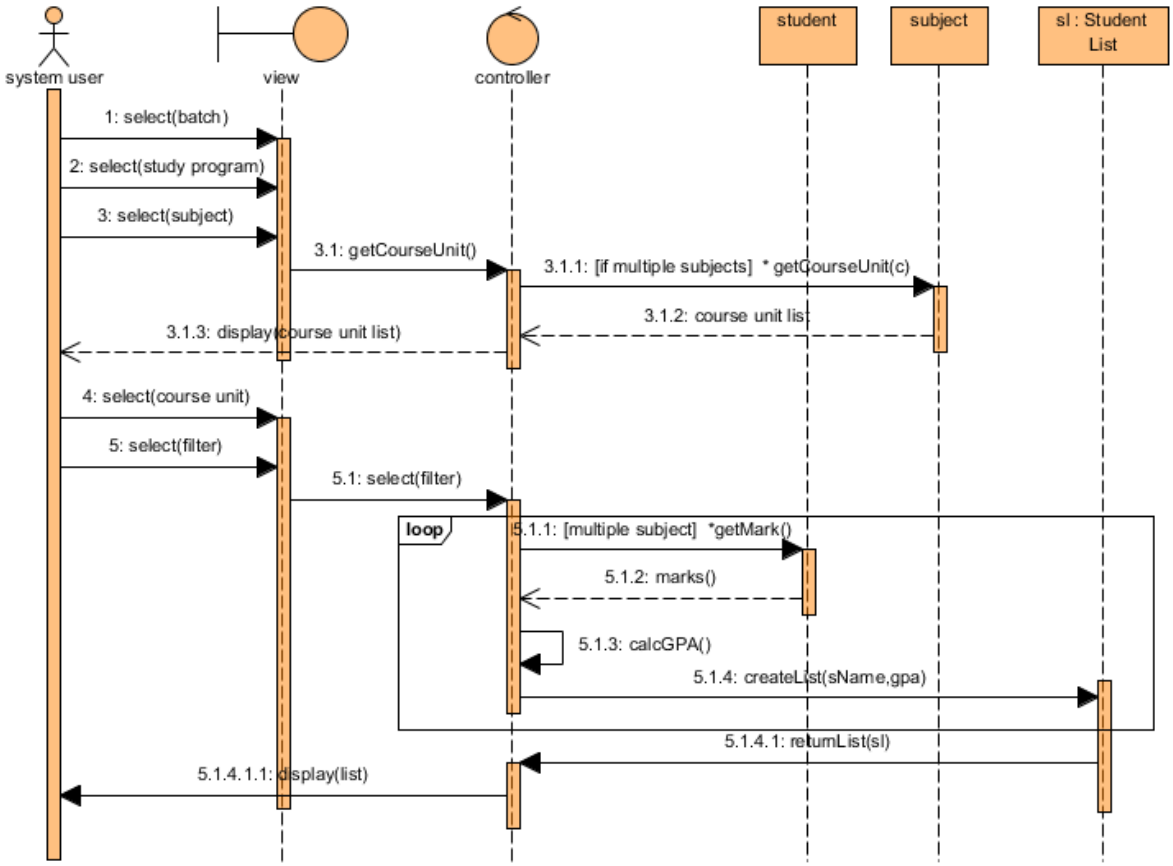
3.2.38 Use case 38 : Organization Modify



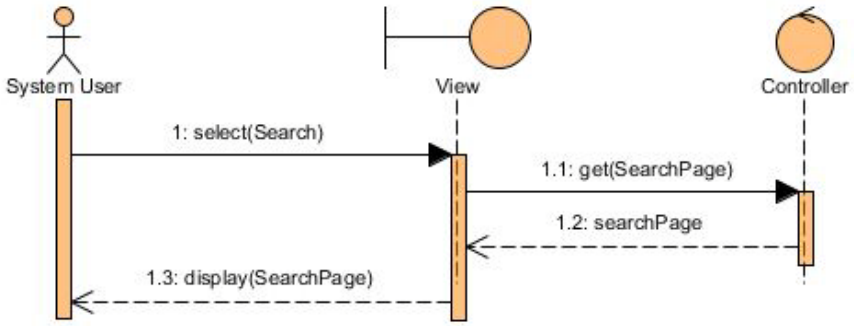
3.2.39 Use case 39 : Opportunity Management



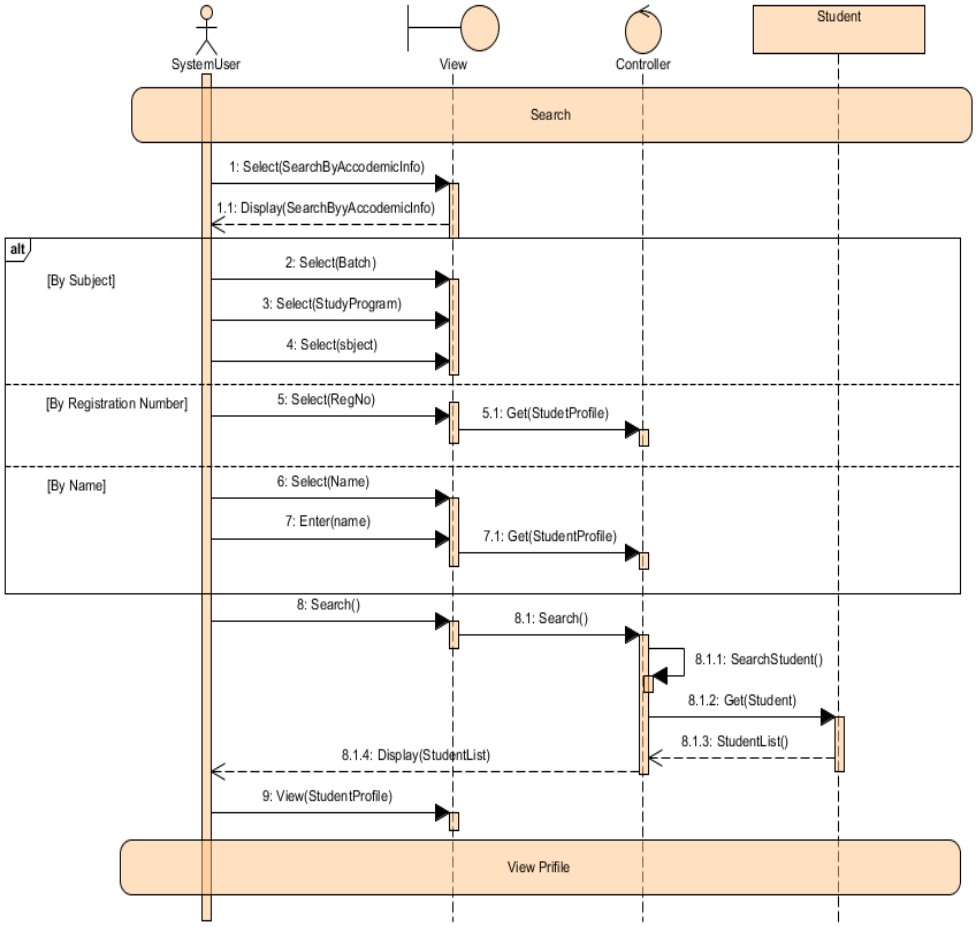
3.2.40 Use case 40 : Filter Students



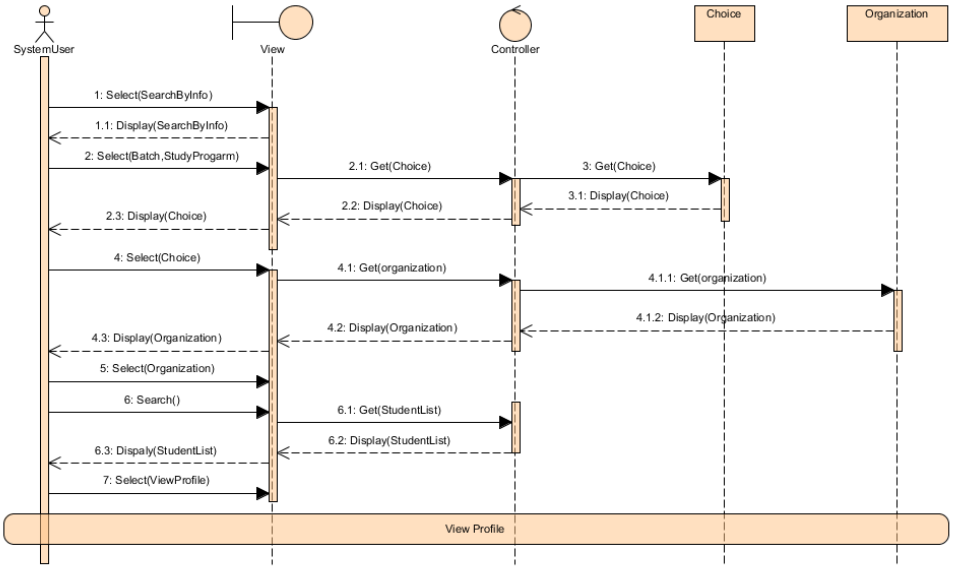
3.2.41 Use case 41 : Search



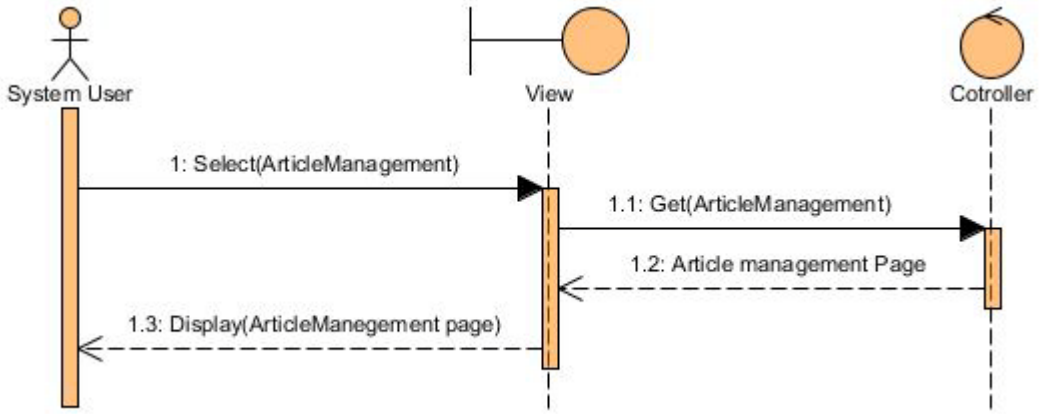
3.2.42 Use case 42 : Search by academic information



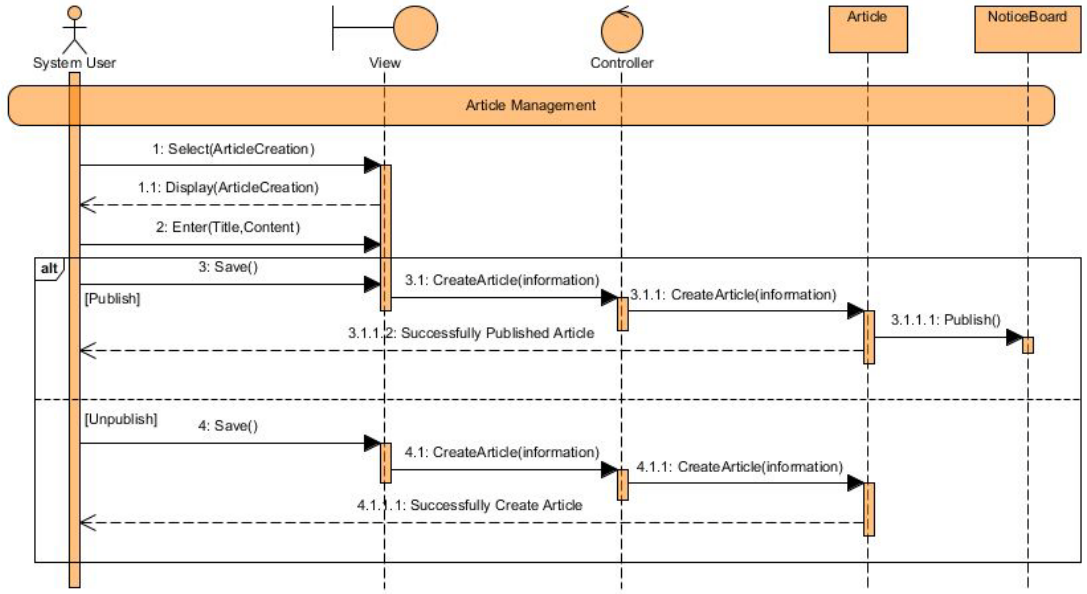
3.2.43 Use case 43 : Search by choice



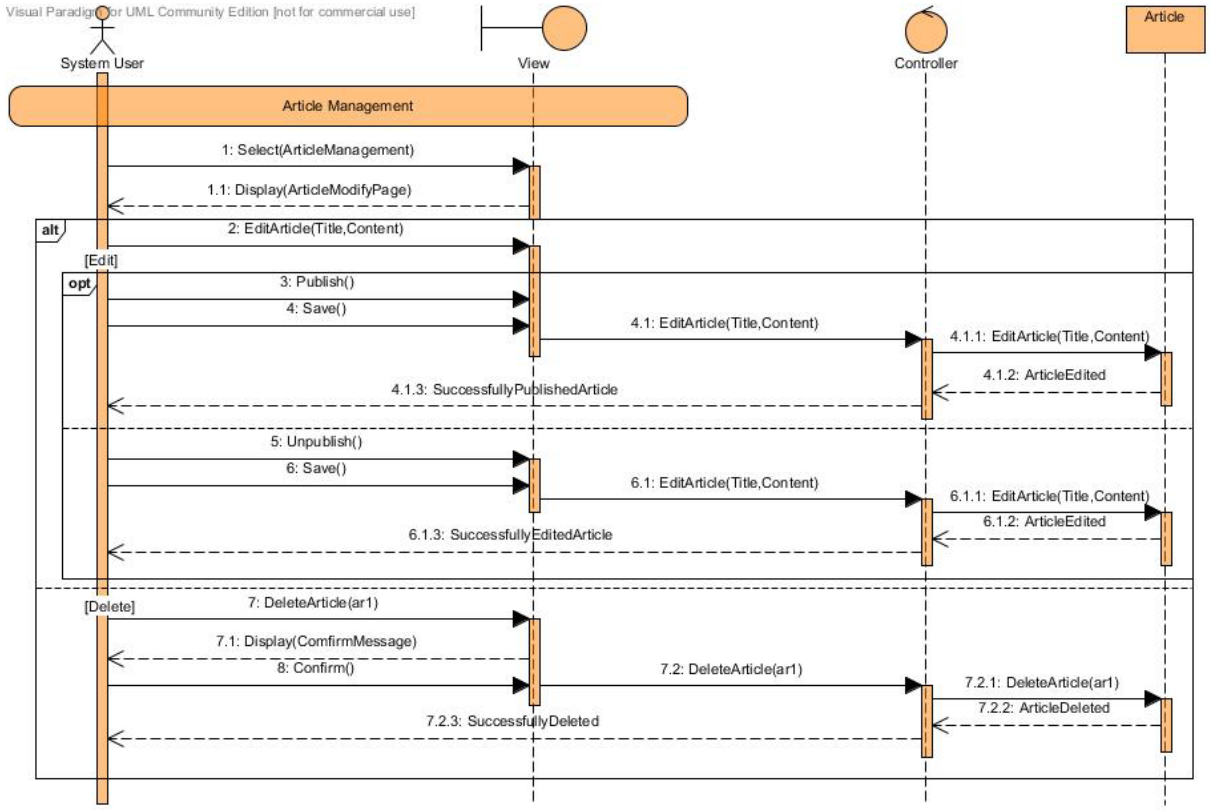
3.2.44 Use case 44 : Article management



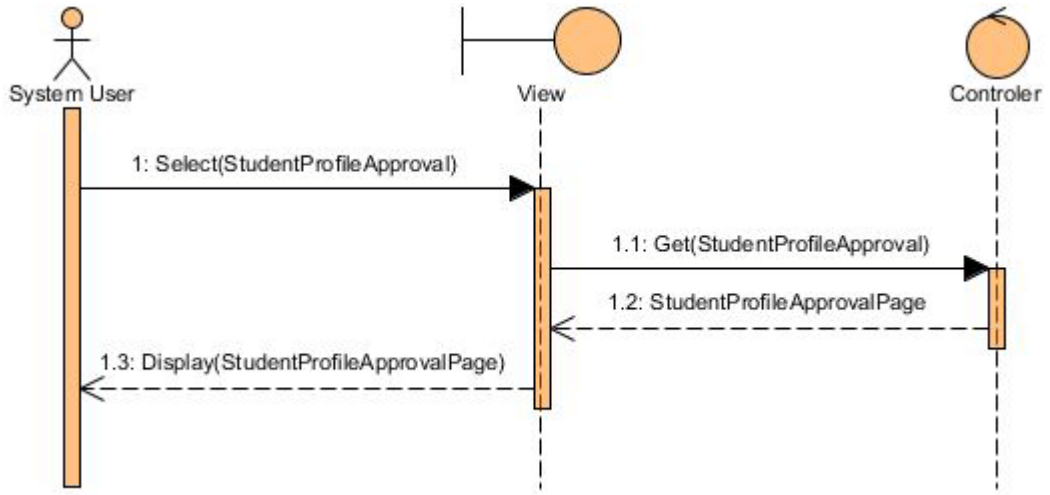
3.2.45 Use case 45 : Article creation



3.2.46 Use case 46 : Article modify



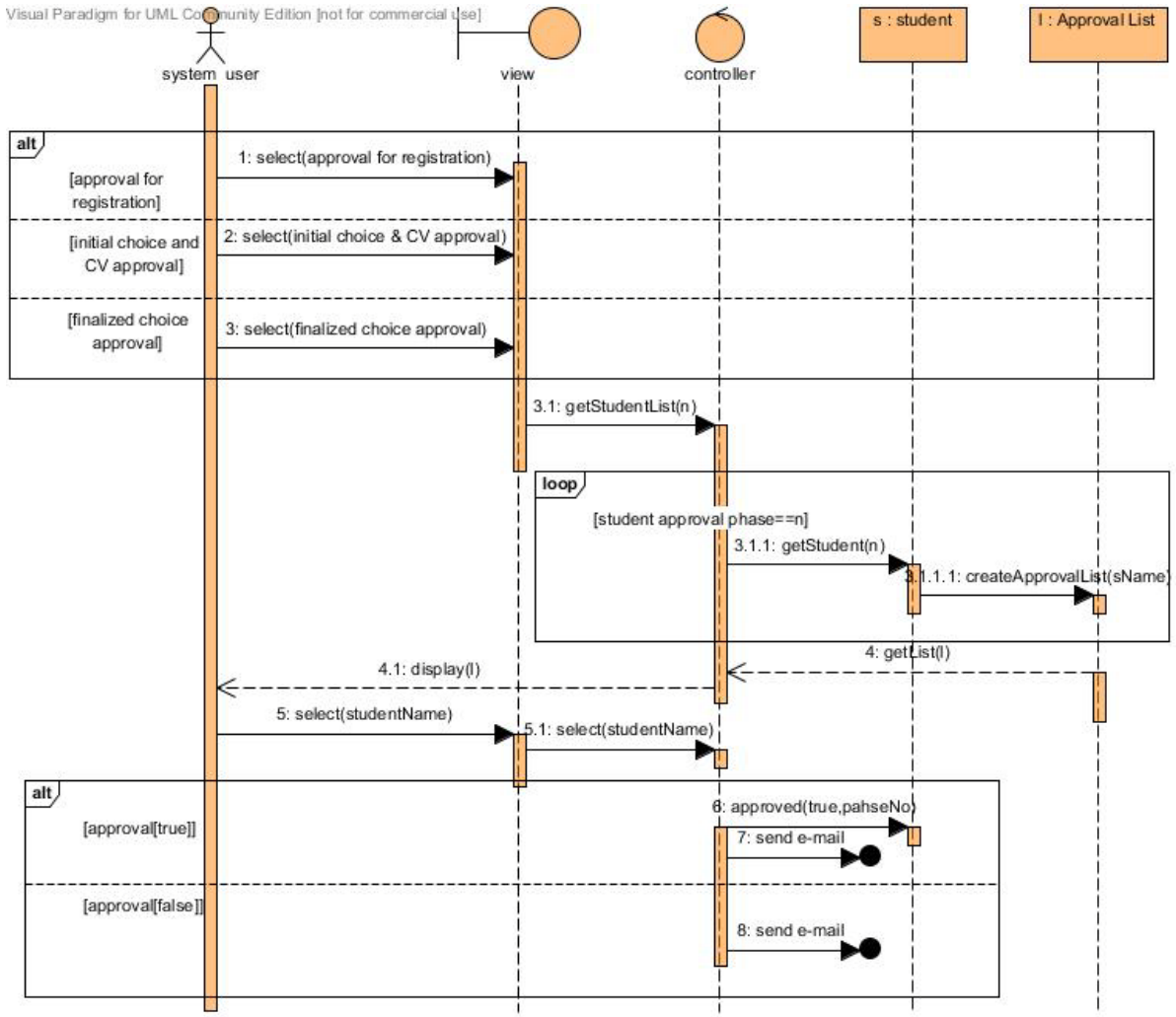
3.2.47 Use case 47 : Student profile approval



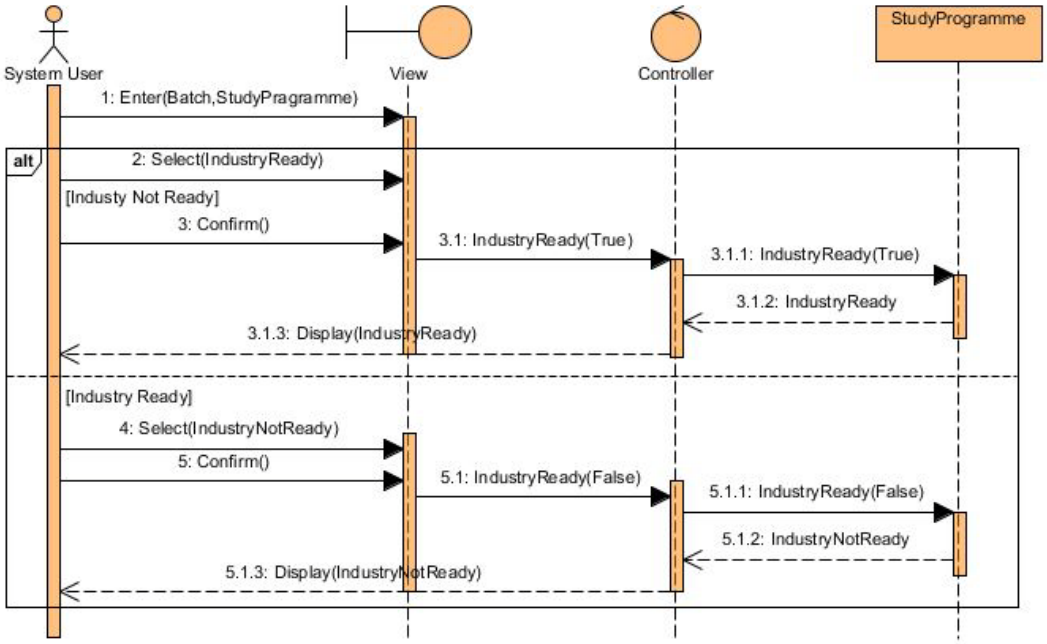
3.2.48 Use case 48 : Approval for Registration

3.2.49 Use case 49 : Initial choice and CV approval

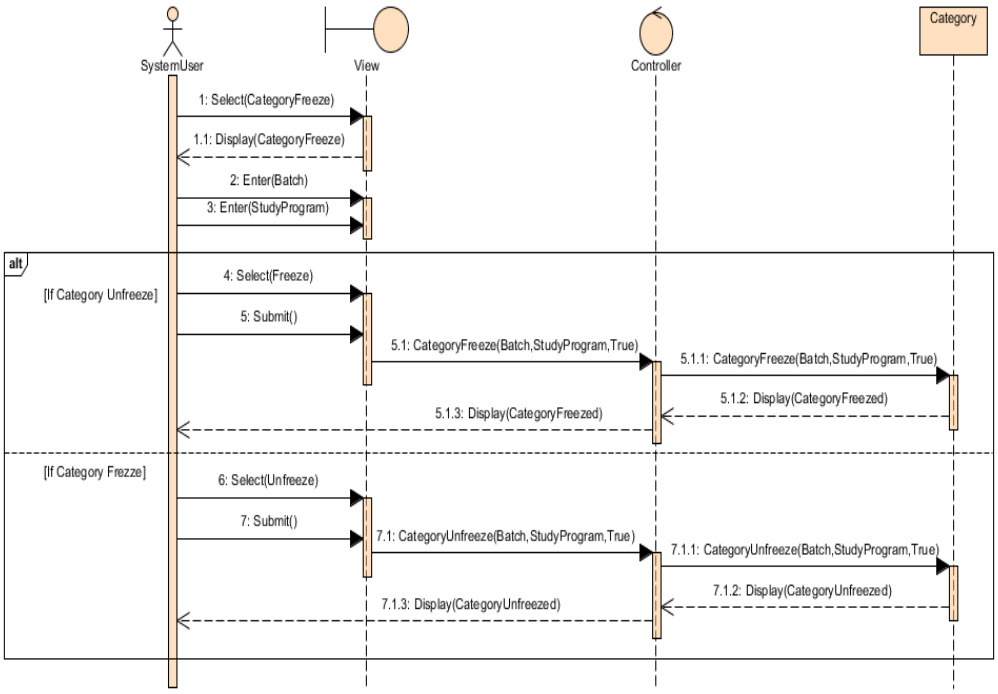
3.2.50 Use case 50 : Finalized choice approval



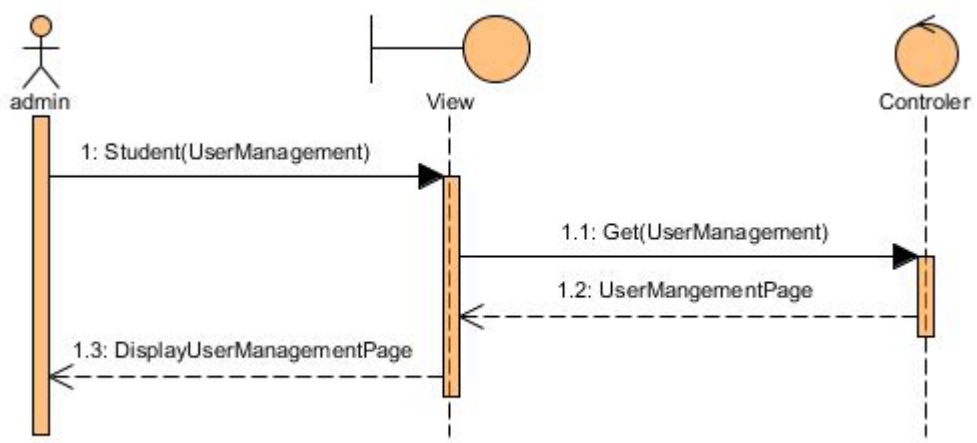
3.2.51 Use case 51 : Industry ready or not



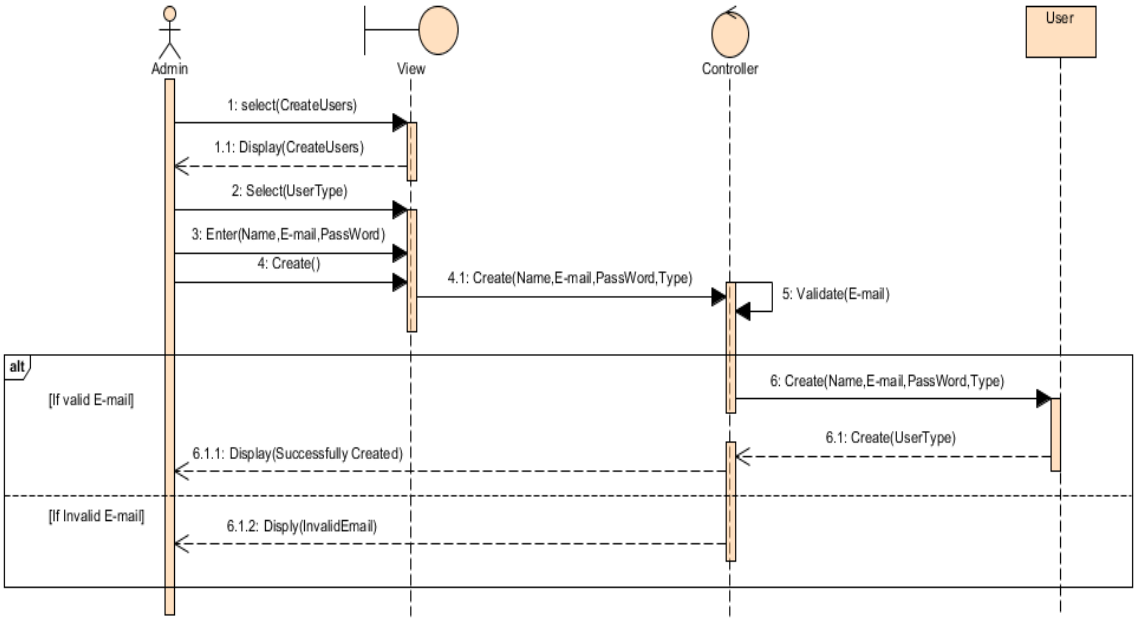
3.2.52 Use case 52 : Category freeze



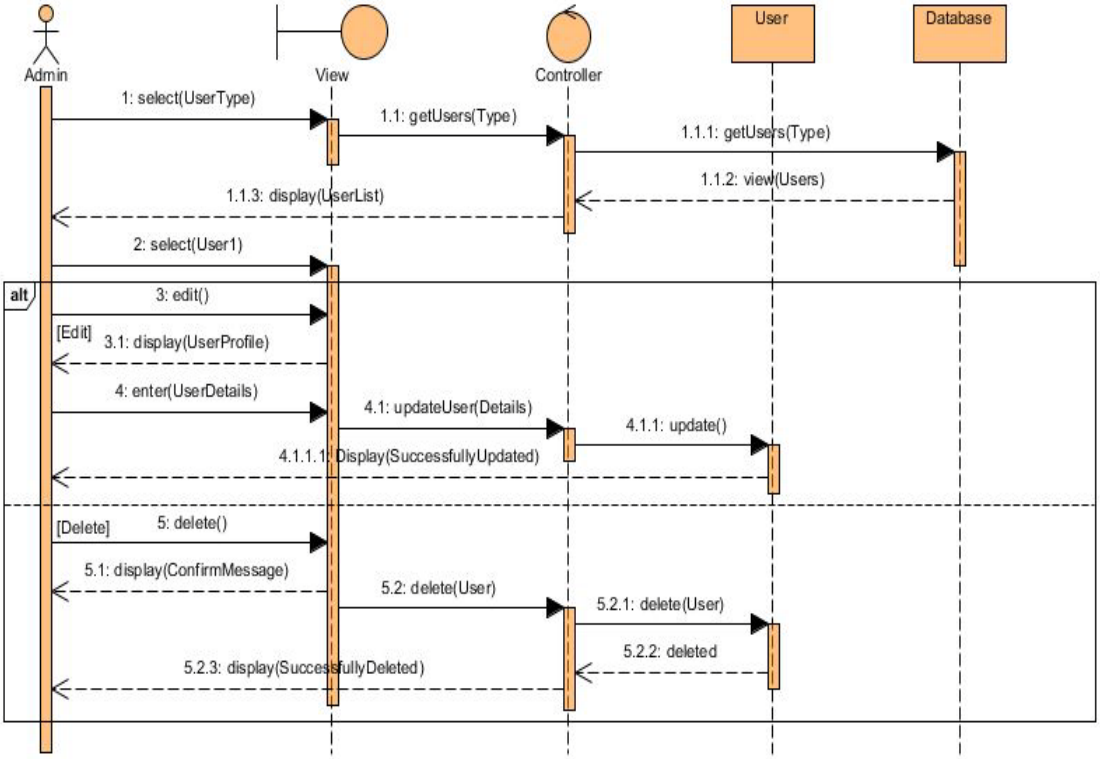
3.2.53 Use case 53 : User Management



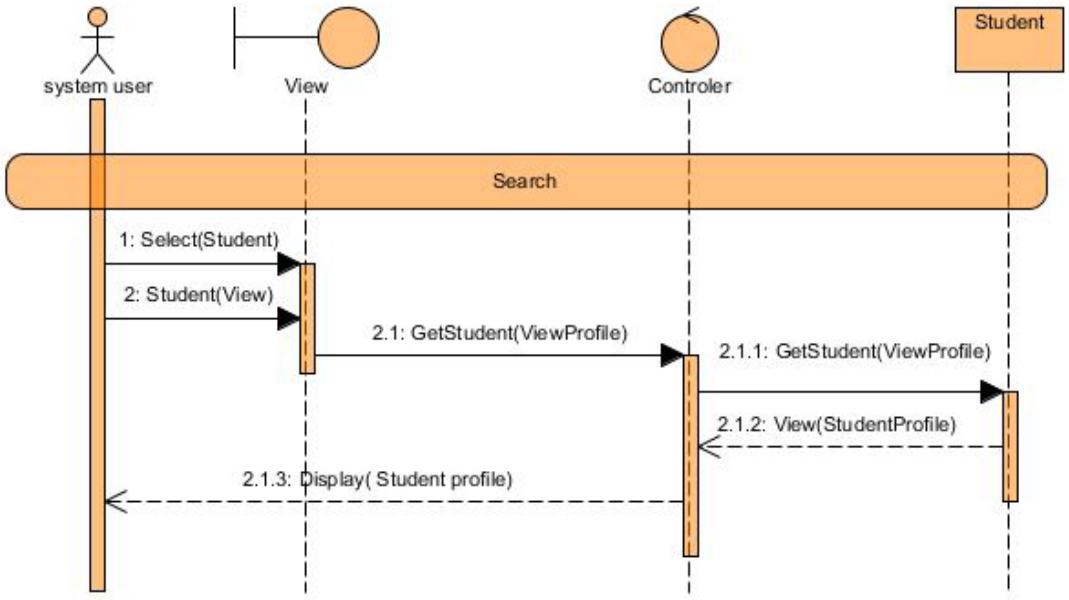
3.2.54 Use case 54 : Create users



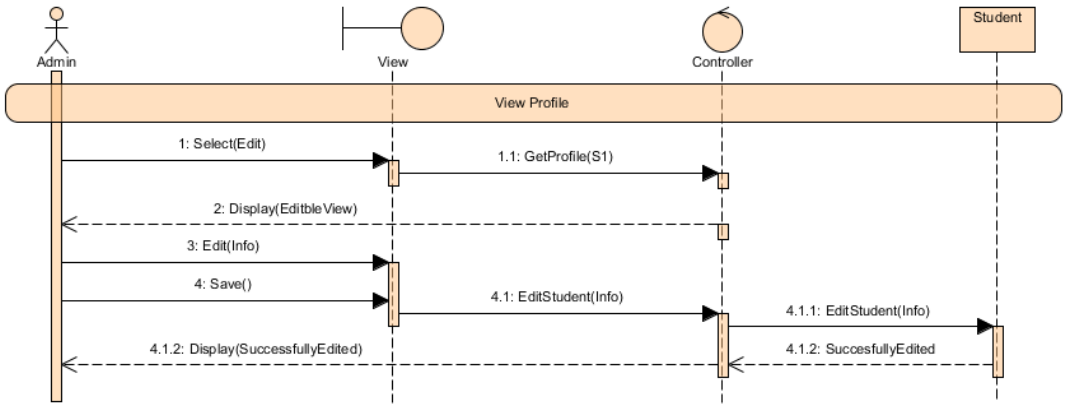
3.2.55 Use case 55 : Modify users



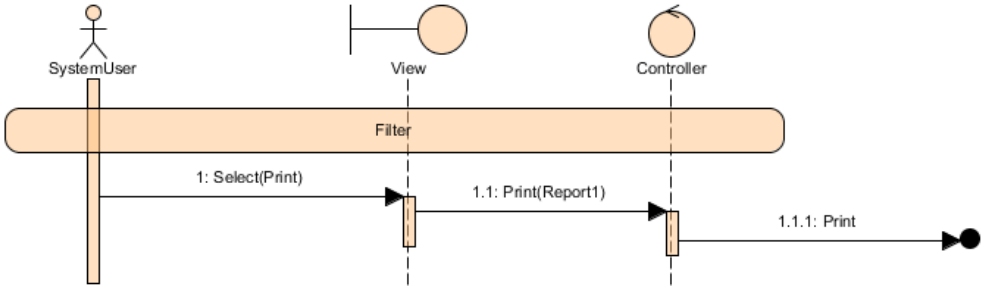
3.2.56 Use case 56 : View profile



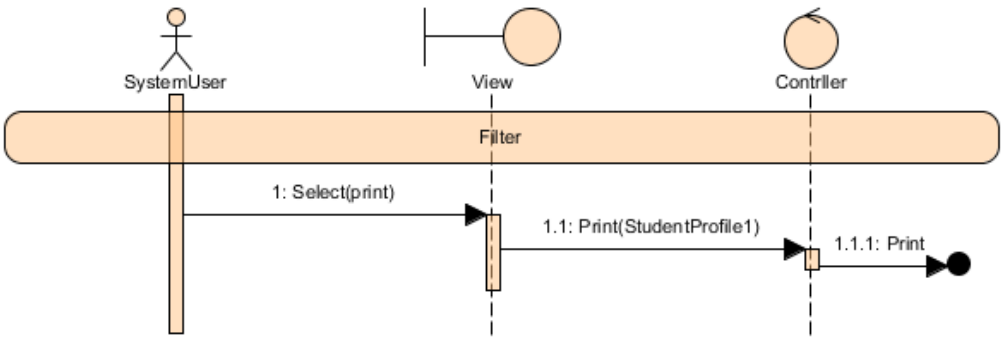
3.2.57 Use case 57 : Update student's profile



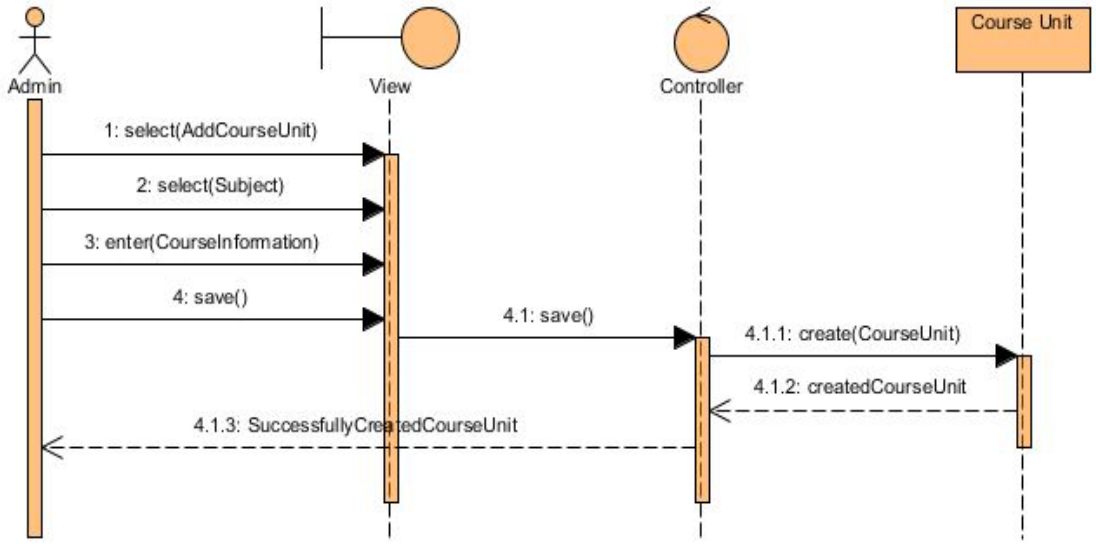
3.2.58 Use case 58 : Report Generation



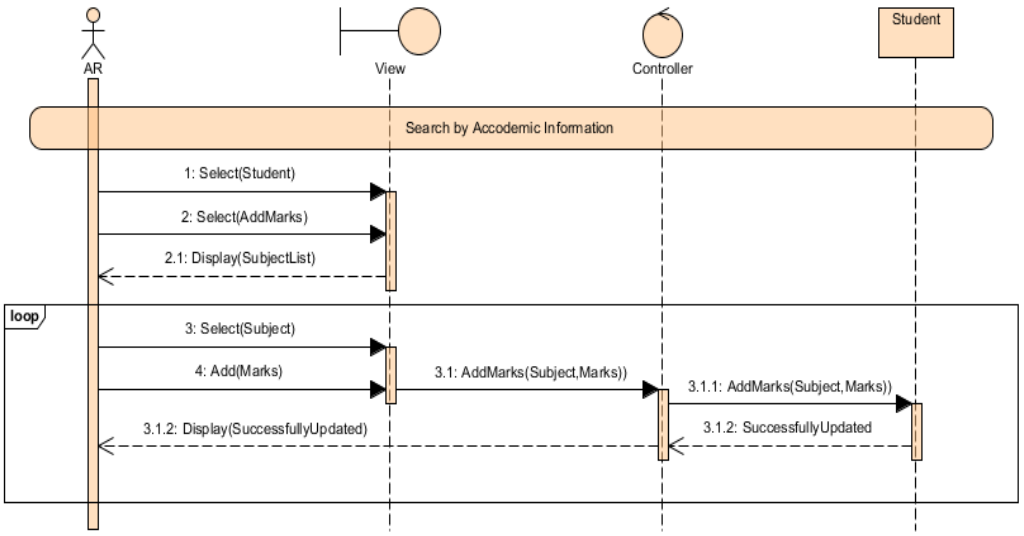
3.2.59 Use case 59 : Print Student Profile



3.2.60 Use case 60 : Add Course



3.2.61 Use case 61 : Add Marks



3.3 ALGORITHM DESIGN

3.3.1 Filter Students Algorithm

1. User loads the filter view
2. User then selects the batch and related academic program.
3. After user selects the relevant subjects for the academic program.
4. From the next list user select any subject or set of subject from any year.
5. After user select the “Filter option”

Algorithm pseudo code

```
if batch_program.industry ready not equals true
      Throw a message
else
      while student equals NULL
            Find students where student done all the course units
selected
            array gpaorder = selected_students
            sorted_gpaorder = sort(gpaorder)
            print sorted_gpaorder
end_if
```

3.3.2 Sort algorithm

```
for I = 1 to N-1
  J = I
  do while (J > 0) and (gpaorder (J) < gpaorder(J - 1)
    Temp = gpaorder(J)
    gpaorder(J) = gpaorder (J - 1)
    gpaorder (J - 1) = Temp
    J = J - 1
  end_do
end_for
```

3.3.3 Search Students by Academic Information

Search Students by Academic Information

1. User loads the search student by academic information option
2. Then user selects the batch.
3. After user selects the academic programs relevant to above batch.
4. User then selects any number of relevant subjects related to the batch and academic program.
5. Finally user selects Search/List option.

Algorithm pseudo code

```
student_array = find students from batch and academic_program
do while student_array = NULL
      if student matches to subject
            return true
      else
            return false
      end_if
end_do
```

3.3.4 Student selection scientific algorithm (Still developing)

Let ***G*** be the GPA for each student, ***E*** be the extra activity mark cumulative, ***A*** be the student interested area and ***O*** be the organization

So we can define S where

$$S = \sum_{m}^{j=1} \sum_{n}^{i=1} G_i, E_i, (A, O)_j$$

Here ***n*** is the number of students and ***m*** is number of organizations combined with interested areas.

Given the constrains

$|A_i| \leq 3$

$\frac{G_i.W_G + E_i.W_E}{2} \leq 4$ Where W_G and W_E are the weights given for the GPA and Extra activities respectively.

The Steps of the algorithm

Group students in to their interested areas and organization couple by the priority of their choice

1. Select students in to groups by the priority of the interested area and organization combination.

   Example.

   Priority(1) => IntrestedArea(Software Engineering) , Organization (WSO2)

Id	$P = \frac{G.W_G + E.W_E}{2}$
ICT/2008/09/012	3.2
ICT/2008/09/052	3.9
…..	…..

   Priority(2) => IntrestedArea(Software Engineering) , Organization (WSO2)

Id	$P = \frac{G.W_G + E.W_E}{2}$
ICT/2008/09/025	3.4
ICT/2008/09/036	2.8
…..	…..

   Priority(3) => IntrestedArea(Software Engineering) , Organization (WSO2)

Id	$P = \frac{G.W_G + E.W_E}{2}$
ICT/2008/09/025	2.1
ICT/2008/09/036	2.7
…..	…..

2. For every table, sort the students according to ***P***
   Example

```
for I = 1 to N-1
        J = I
        do while (J > 0) and (P(J) < P(J - 1)
          Temp = P(J)
          P(J) = P (J - 1)
          P (J - 1) = Temp
          J = J - 1
        end_do
end_for
```

Id	Sort_$ASC\left(P = \frac{G.W_G + E.W_E}{2}\right)$
ICT/2008/09/052	3.9
ICT/2008/09/012	3.2
…..	…..

3. Fill the opportunities granted from organizations combining the interested area with these three tables starting from Priority(1) table of the group.

```
do while (opportunity(Software Engineering + WSO2)>0)
       student.selected = true
       opportunity(Software Engineering + WSO2) = opportunity(Software Engineering + WSO2) – 1
end do
```

4. Iterate this procedure for every priority group.

3.3.5 Algorithm Optimization

To select a maximum amount of students and the most appropriate students, Algorithm can be optimized in the context of W_G and W_E as below

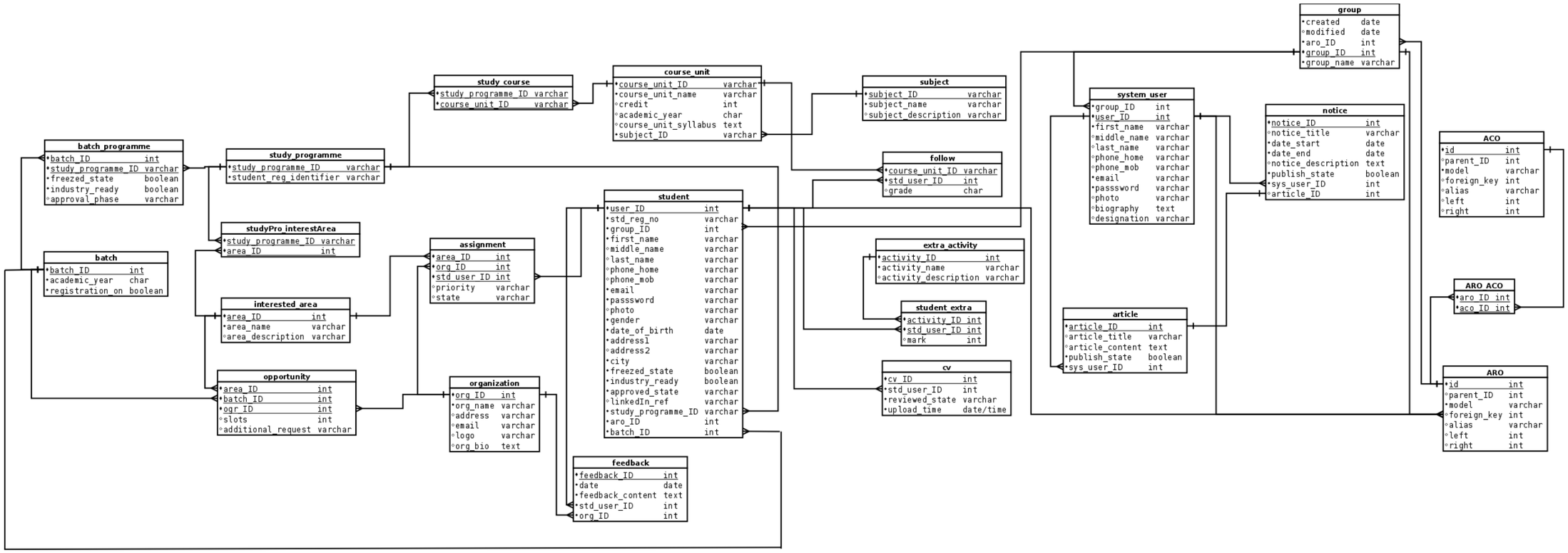
W_G	W_E	S_i
trivial	*trivial*	-
trivial	*trivial*	-
….	….	….
0.6	0.4	S_x
0.7	0.3	S_y
0.8	0.2	S_z
….	….	….
trivial	*trivial*	-
trivial	*trivial*	-

Where S is final selection outcome of the algorithm, $W_G + W_E = 1$ and
Out of S_i the maximum would be

$$\sum S_{max} = W_{G(max)} + W_{E(max)}$$

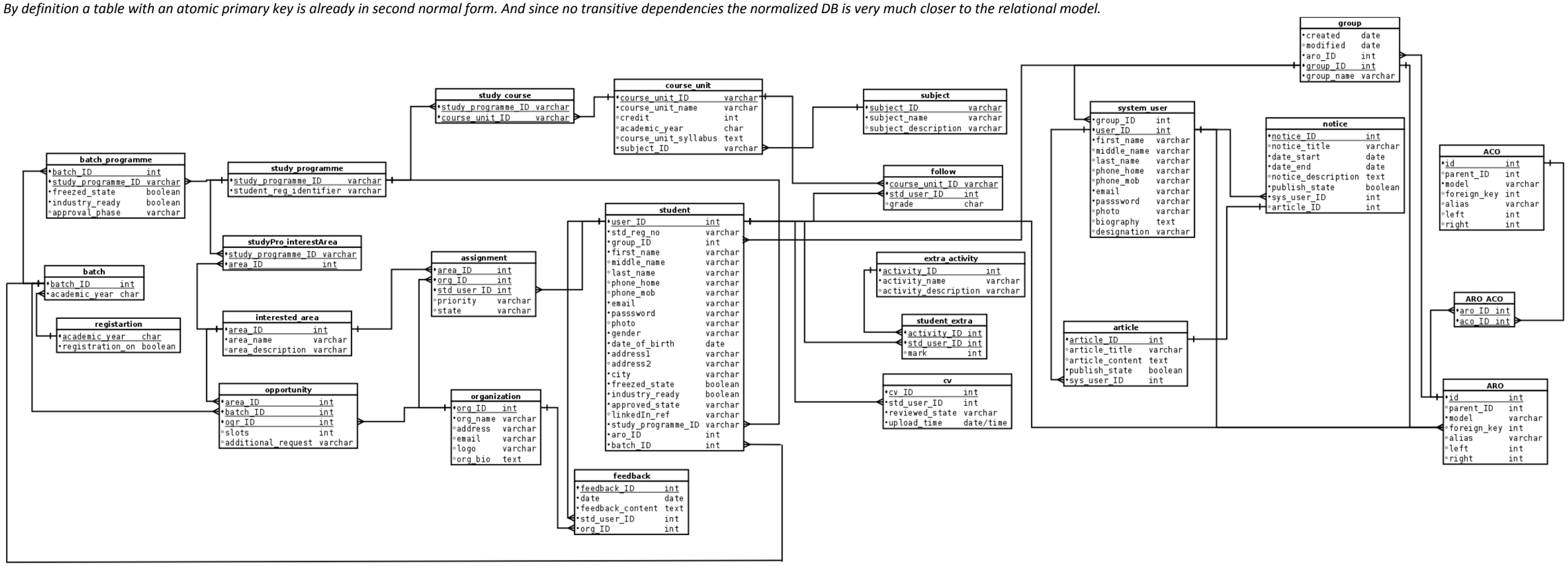
3.4 DATABASE DESIGN

3.4.1 Relational Model



3.4.2 Normalization/Denormalization

By definition a table with an atomic primary key is already in second normal form. And since no transitive dependencies the normalized DB is very much closer to the relational model.



3.4.3 Data Dictionary

system_user			
Field	**Data Type**	**Null**	**Description**
user_ID	int(11)	No	
first_name	Varchar(255)	No	
middle_name	Varchar(255)	Yes	
last_name	Varchar(255)	Yes	
phone_home	Varchar(20)	Yes	
phone_mob	Varchar(20)	Yes	
email	Varchar(255)	No	
password	Varchar(40)	No	
photo	Varchar(255)	Yes	Contains a link to a photo
group_ID	int(11)	No	Specify the user group which differentiate the access levels in the system
Biography	Text	Yes	Small biography on himself/herself
Designation	Varchar(255)	Yes	Title of the job

student			
Field	**Data Type**	**Null**	**Description**
user_ID	int(11)	No	
first_name	Varchar(255)	No	
middle_name	Varchar(255)	Yes	
last_name	Varchar(255)	Yes	
phone_home	Varchar(20)	Yes	
phone_mob	Varchar(20)	Yes	
email	Varchar(255)	No	
password	Varchar(40)	No	
Photo	Varchar(255)	Yes	Contains a link to a photo
group_ID	int(11)	No	Specify the user group which differentiate the access levels in the system
std_reg_no	Varchar(20)	No	Registration no issued from the university
gender	Varchar(10)	No	
date_of_birth	Date	No	
address1	Varchar(255)	No	
address2	Varchar(255)	Yes	
city	Varchar(255)	Yes	
freeze_state	Boolean	No	0- Profiles are in unfreezed state , meaning students can edit profile 1- Profiles are in unfreezed state , meaning students cann't edit profile
industry_ready	Boolean	No	0- Not ready for industry view. Industry can't view profile data. 1- Ready for industry view. Industry can view profile data.
approved_state	Varchar(5)	No	Three approval states available 0- Initial approval 1- Industry preview approval 2- Approval for calculation Use to identify ,from which phase the profiles are approved

linkedIn_ref	Varchar(255)	Yes	Link to the LinkedIn profile to grab relevant details
study_programme_ID	Varchar(11)	No	
batch_ID	Int(11)	No	

Article			
Field	**Data Type**	**Null**	**Description**
article_ID	int(11)	No	
article_title	Varchar(255)	Yes	
article_content	Text	Yes	
publish_state	Boolean	No	Specify , 0- publish the article to notice board 1- only to save it and not publish
sys_user_ID	Int(11)	No	

Notice			
Field	**Data Type**	**Null**	**Description**
notice_ID	int(11)	No	
notice_title	Varchar(255)	Yes	
date_start	Date	No	Starting date of the notice
date_end	Date	No	End date of the notice
notice_description	Text	Yes	
publish_state	Boolean	No	0- publish the article to notice board 1- only to save it and not publish
sys_user_ID	Int(11)	No	
article_ID	Int(11)	Yes	

extra_activity			
Field	**Data Type**	**Null**	**Description**
activity_ID	int(11)	No	
activity_name	Varchar(255)	No	Ex: School achievements, sports,certifications,positions, etc.
activityCat_description	Varchar(255)	Yes	

student_extra			
Field	**Data Type**	**Null**	**Description**
activity_ID	int(11)	No	
std_user_ID	Int(11)	No	
mark	Int(3)	Yes	Marks allocate for extra activities

batch			
Field	**Data Type**	**Null**	**Description**
batch_ID	int(11)	No	
academic_year	char(4)	No	Ex: 2008/2009 , 2010/2011
registration_on_state	Boolean	No	0- Registration option is not visible 1- Registration option is visible to the particular batch

batch_programme			
Field	**Data Type**	**Null**	**Description**
batch_ID	int(11)	No	
study_programme_ID	Varchar(11)	No	
freeze_state	Boolean	No	0- Profiles are in unfreezed state , meaning students can edit profile 1- Profiles are in unfreezed state , meaning students cann't edit profile
industry_ready	Boolean	No	0- Not ready for industry view. Industry can't view profile data. 1- Ready for industry view. Industry can view profile data.
approval_phase	Varchar(5)	yes	Three approval states available 1. primary 2. Initial choice 3. Finalizing choice based on the phase the content of the student profile varies.

subject			
Field	**Data Type**	**Null**	**Description**
subject_ID	Varchar(11)	No	Ex: ICT,MAA,CHE
subject_name	Varchar(255)	No	ICT - Information and Communication Technology MAA- Applied Mathematics
subject_description	Varchar(255)	Yes	

follow			
Field	**Data Type**	**Null**	**Description**
course_unit_ID	Varchar(11)	No	Ex: ICT1205, HPT 1103
std_user_ID	int(11)	No	
grade	char(2)	Yes	Ex: A+,A

course_unit			
Field	**Data Type**	**Null**	**Description**
course_unit_ID	Varchar(11)	No	Ex: ICT1205, HPT 1103
course_unit_name	Varchar(255)	No	Ex: ICT 1205 – Fundamentals of Computer
credit	Int	Yes	Ex: ICT 1205 → 2 credits
academic_year	Char(4)	Yes	
course_unit_syllabus	Text	Yes	
subject_ID	Varchar(11)	No	Ex: ICT,MAA,CHE

study_programme			
Field	**Data Type**	**Null**	**Description**
study_programme_ID	Varchar(11)	No	
student_reg_identifier	Varchar(255)	No	Ex: ICT,BIO,PHY

interested_area			
Field	**Data Type**	**Null**	**Description**
area_ID	int(11)	No	
area_name	Varchar(255)	No	Ex: software engineering, networking
area_description	Varchar(255)	Yes	

studyPro_interestArea			
Field	**Data Type**	**Null**	**Description**
study_programme_ID	Varchar(11)	No	
area_ID	Int(11)	No	Ex: ICT,BIO,PHY

study_course			
Field	**Data Type**	**Null**	**Description**
study_programme_ID	Varchar(11)	No	
course_unit_ID	Varchar(11)	No	Ex: ICT1205, HPT 1103

Organization			
Field	**Data Type**	**Null**	**Description**
org_ID	int(11)	No	
org_name	Varchar(255)	No	
address	Varchar(255)	Yes	
email	Varchar(255)	Yes	
logo	Varchar(255)	Yes	Link to an image
org_bio	Text	Yes	Small description on organization

feedback			
Field	**Data Type**	**Null**	**Description**
feedback_ID	int(11)	No	
date	Date	No	
feedback_content	Text	No	
std_user_ID	Int(11)	No	To identify the student related to the feedback
org_ID	Int(11)	No	To identify the organization related to the feedback

cv			
Field	**Data Type**	**Null**	**Description**
cv_ID	int(11)	No	
std_user_ID	Int(11)	No	
reviewed_state	Varchar(5)	No	Whether authorized person has checked the CV, the CV is not reviewed CV is rejected due to insuffiant information
upload_time	Date/time	No	Time stamp to identify the latest CV uploaded related to a choice

ARO			
Field	**Data Type**	**Null**	**Description**
id	Int(10)	No	
parent_ID	Int(10)	Yes	
model	Varchar(255)	Yes	
foreign_key	Int(10)	Yes	
alias	Varchar(255)	Yes	
left	Int(10)	Yes	
right	Int(10)	Yes	

ACO			
Field	**Data Type**	**Null**	**Description**
id	Int(10)	No	
parent_ID	Int(10)	Yes	
model	Varchar(255)	Yes	
foreign_key	Int(10)	Yes	
alias	Varchar(255)	Yes	
left	Int(10)	Yes	
right	Int(10)	Yes	

ARO_ACO			
Field	**Data Type**	**Null**	**Description**
aro_id	Int(10)	No	
aco_id	Int(10)	No	

assignment			
Field	**Data Type**	**Null**	**Description**
area_ID	Int(11)	No	
org_ID	Int(11)	No	
std_user_ID	Int(11)	No	
priority	Varchar(3)	Yes	1st ,2nd and 3rd choice
state	Varchar(5)	Yes	Related to the result on student assignment to the organization 0- Pending result 1- Selected 2- Rejected 3- Not considered

opportunity			
Field	**Data Type**	**Null**	**Description**
area_ID	Int(11)	No	
org_ID	Int(11)	No	
batch_ID	Int(11)	No	
slots	Int(2)	Yes	Available opportunities granted by the organization
additional_requests	Varchar(255)	Yes	Special request field for requesting students at the very beginning

group			
Field	**Data Type**	**Null**	**Description**
group_ID	Int(11)	No	
group_name	Varchar(255)	No	
created	date	No	
modified	date	Yes	
aro_ID	Int(10)	No	

3.4.4 Decisions made to manage transactions concurrent

- In our project we use MySQL InnoDB Lock Mode for transaction locking. We maintain an InnoDB database engine.
- InnoDB implements standard row-level locking where there are two types of locks:
- A shared (S) lock permits a transaction to read a row.
- An exclusive (X) lock permits a transaction to update or delete a row.
- Additionally, InnoDB supports multiple granularity locking which permits coexistence of record locks and locks on entire tables.
- We don't use MyISAM engine because MyISAM doesn't support transactions.
- Concurrency is achieved by multiple-indexes as well.

3.4.5 Indexes

system_user			
Keyname	**Type**	**Unique**	**Column**
PRIMARY	BTREE	Yes	user_ID
FNAME	BTREE	No	first_name
SNAME	BTREE	No	last_name
EMAIL	BTREE	Yes	email

student			
Keyname	**Type**	**Unique**	**Column**
PRIMARY	BTREE	Yes	user_ID
FNAME	BTREE	No	first_name
SNAME	BTREE	No	last_name
EMAIL	BTREE	Yes	email
REGNO	BTREE	Yes	std_reg_no

article			
Keyname	**Type**	**Unique**	**Column**
PRIMARY	BTREE	Yes	article_ID
AUTHOR	BTREE	No	sys_user_ID

notice			
Keyname	**Type**	**Unique**	**Column**
PRIMARY	BTREE	Yes	notice_ID
AUTHOR	BTREE	No	sys_user_ID

extra_activity			
Keyname	**Type**	**Unique**	**Column**
PRIMARY	BTREE	Yes	Activity_ID

student_extra			
Keyname	**Type**	**Unique**	**Column**
PRIMARY	BTREE	Yes	activity_ID
			std_user_ID

batch			
Keyname	**Type**	**Unique**	**Column**
PRIMARY	BTREE	Yes	batch_ID
ACADEMIC_YEAR	BTREE	Yes	academic_year

batch_programme			
Keyname	**Type**	**Unique**	**Column**
PRIMARY	BTREE	Yes	batch_ID
			study_programme_ID

subject			
Keyname	**Type**	**Unique**	**Column**
PRIMARY	BTREE	Yes	subject_ID
SUB_NAME	BTREE	yes	subject_name

follow			
Keyname	**Type**	**Unique**	**Column**
PRIMARY	BTREE	Yes	course_unit_ID
			std_user_ID

course_unit			
Keyname	**Type**	**Unique**	**Column**
PRIMARY	BTREE	Yes	course_unit_ID
CU_NAME	BTREE	Yes	course_unit_name
RELATED_SUB	BTREE	Yes	subject_ID

study_programme			
Keyname	**Type**	**Unique**	**Column**
PRIMARY	BTREE	Yes	study_programme_ID

interested_area			
Keyname	**Type**	**Unique**	**Column**
PRIMARY	BTREE	Yes	area_ID
AREA	BTREE	Yes	area_name

studyPro_interestArea			
Keyname	**Type**	**Unique**	**Column**
PRIMARY	BTREE	Yes	study_programme_ID
			area_ID

study_course			
Keyname	**Type**	**Unique**	**Column**
PRIMARY	BTREE	Yes	study_programme_ID
			course_unit_ID

organization			
Keyname	**Type**	**Unique**	**Column**
PRIMARY	BTREE	Yes	org_ID
ORG_NAME	BTREE	Yes	org_Name

feedback			
Keyname	**Type**	**Unique**	**Column**
PRIMARY	BTREE	Yes	feedback_ID
COMMENTOR	BTREE	Yes	std_user_ID
FEED_OWNER	BTREE	Yes	org_ID

cv			
Keyname	**Type**	**Unique**	**Column**
PRIMARY	BTREE	Yes	cv_ID
CV_OWNER	BTREE	No	std_user_ID

ARO			
Keyname	**Type**	**Unique**	**Column**
PRIMARY	BTREE	Yes	id

ACO			
Keyname	**Type**	**Unique**	**Column**
PRIMARY	BTREE	Yes	id

ARO_ACO			
Keyname	**Type**	**Unique**	**Column**
PRIMARY	BTREE	Yes	aro_ID
			aco_ID

assignment			
Keyname	**Type**	**Unique**	**Column**
PRIMARY	BTREE	Yes	area_ID
			org_ID
			std_user_ID

opportunity			
Keyname	**Type**	**Unique**	**Column**
PRIMARY	BTREE	Yes	area_ID
			org_ID
			batch_ID

group			
Keyname	**Type**	**Unique**	**Column**
PRIMARY	BTREE	Yes	group_ID
MEMBERS	BTREE	No	group_name

3.5. USER INTERFACES

3.6. RULES AND GUIDELINES FOR INTERFACE DESIGN

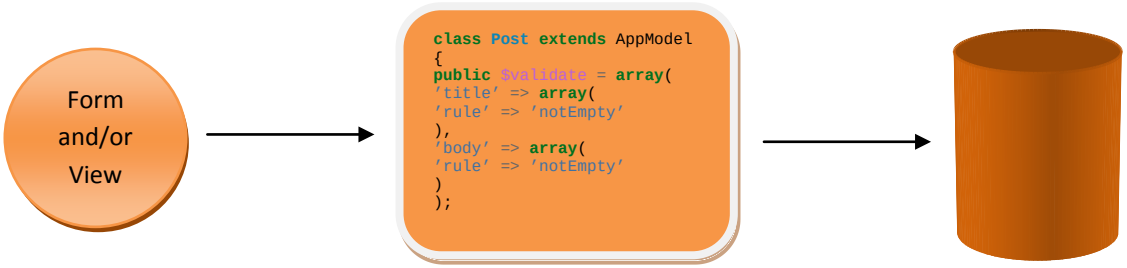
3.6.1 User interface design framework

In this project we'll be using *TwitterBootstrapper (http://twitter.github.io/bootstrap/),* a full comprehensive web interface design framework designed by Twitter *(http://twitter.com)* developers.

Reason: - This framework is very much helpful for a clean and professional design. Since this is a very popular framework and developed by a reputed company, framework documentation is well formed and there are many places that we can get help.

3.6.2 User Input validation methods

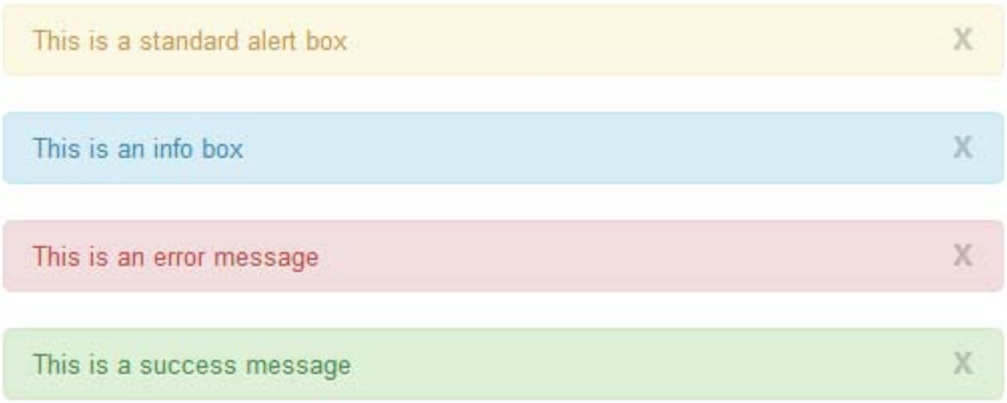
Basically this system will do all the form and data validation in the Model level. Given below is an illustration for the basic form validation process via Model.



Reason: - The PHP framework runs on top of an Active Record design pattern to connect with the database. So that feature enables us to have this kind of validation architecture which is more secure and readable than JavaScript validation.

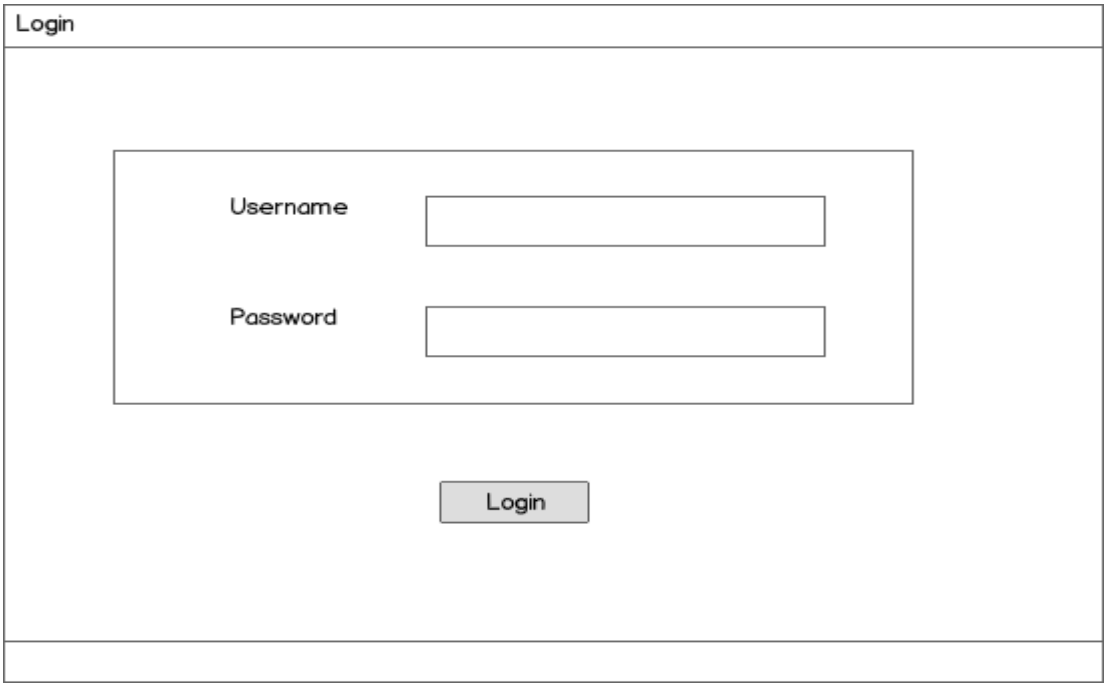
3.6.3 Alert messages decomposition

We'll be using following type of alert scheme for all over the system. So that people who are using this system will identify the type of the message just using the color of it.

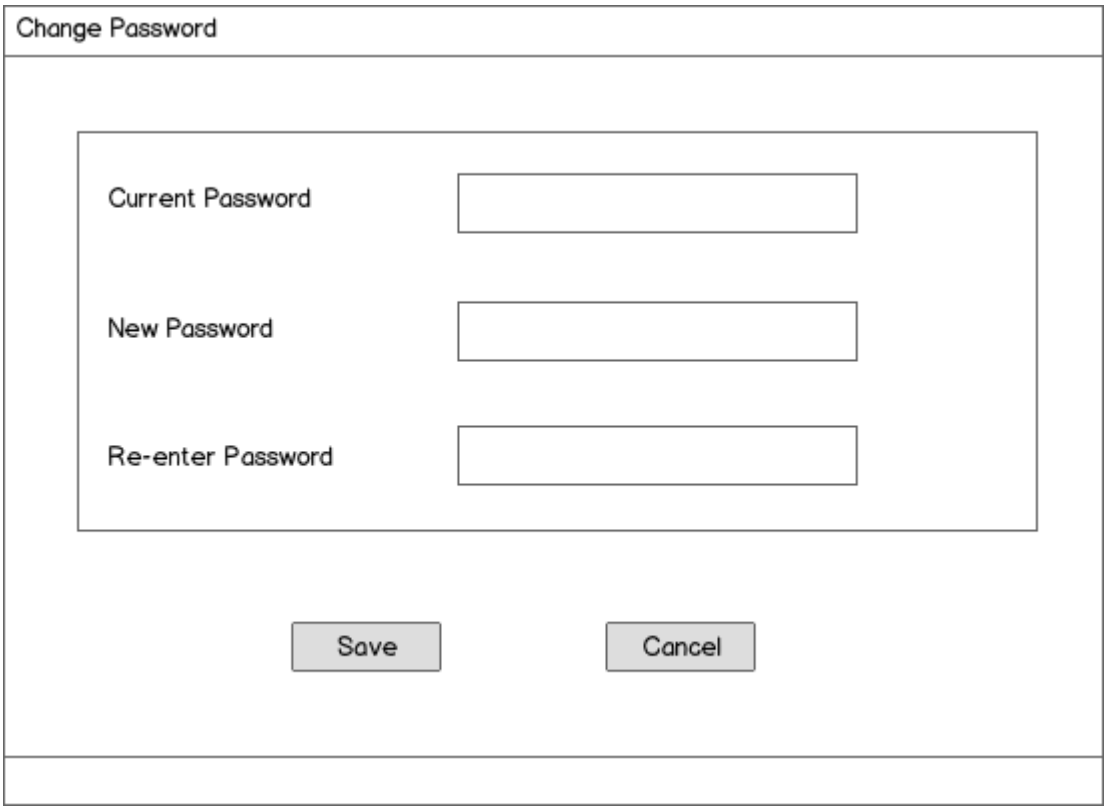


3.7. USER INTERFACES DESIGN

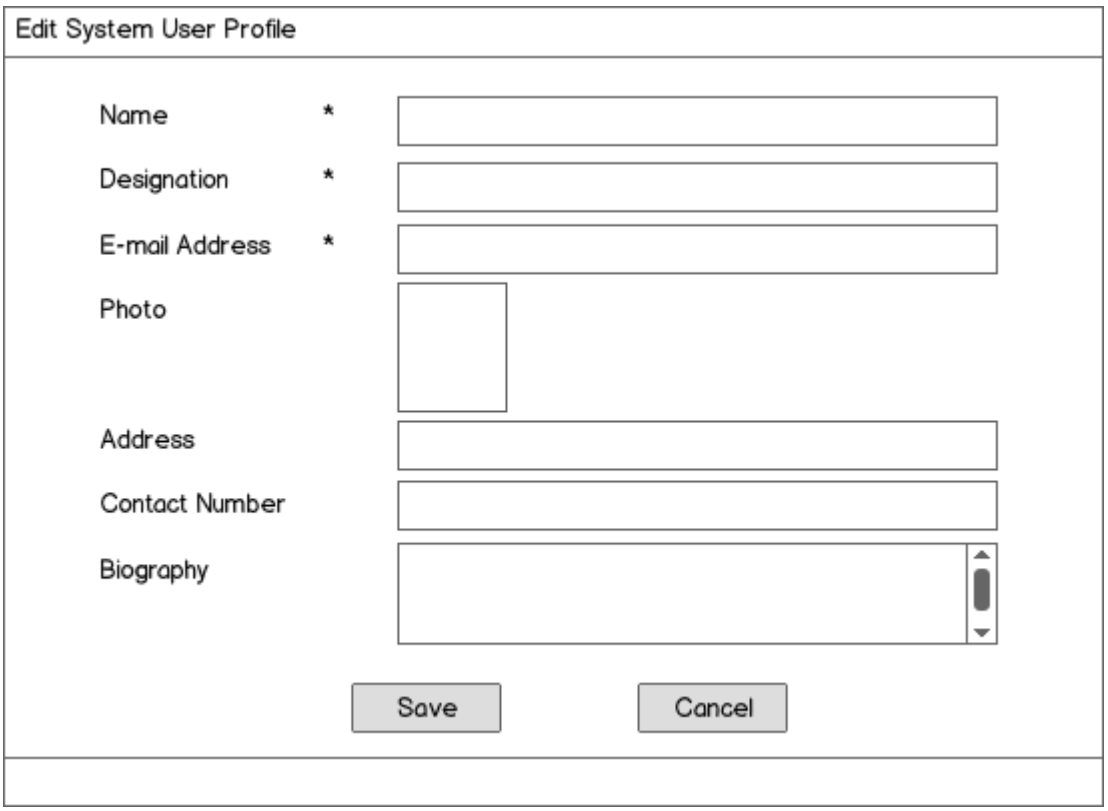
3.7.1 Use case 1: Login

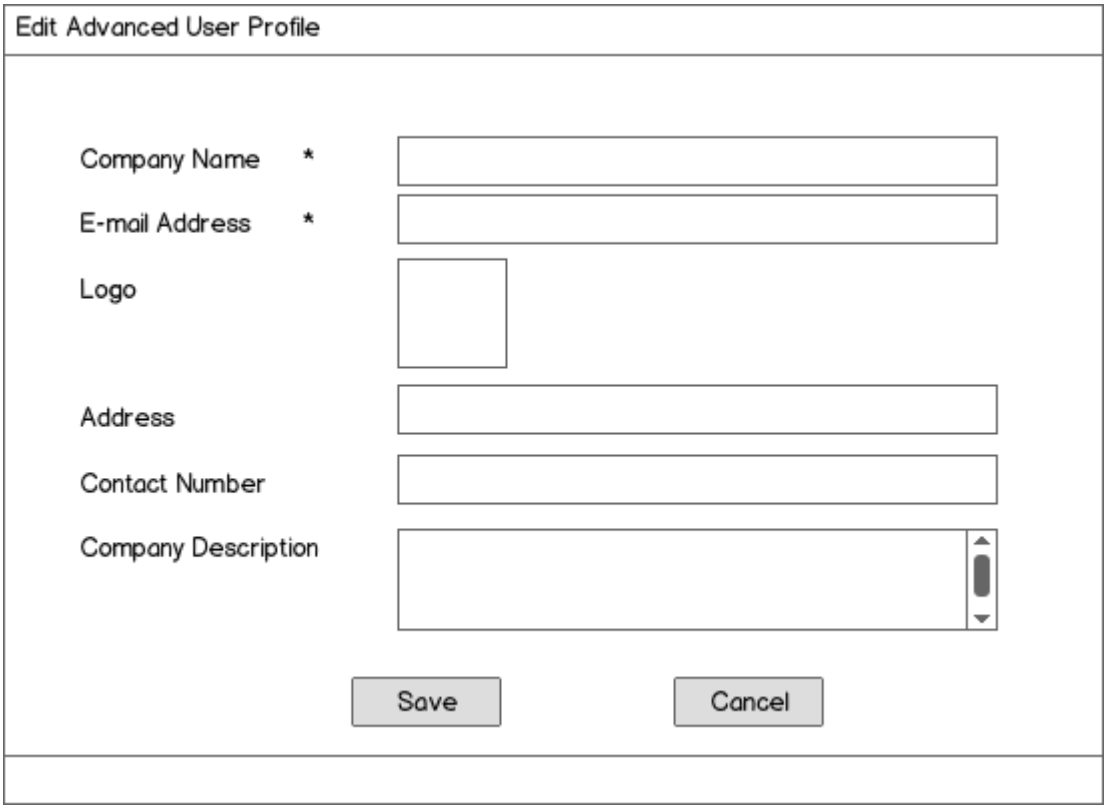


3.7.2 Use case 2: Change password

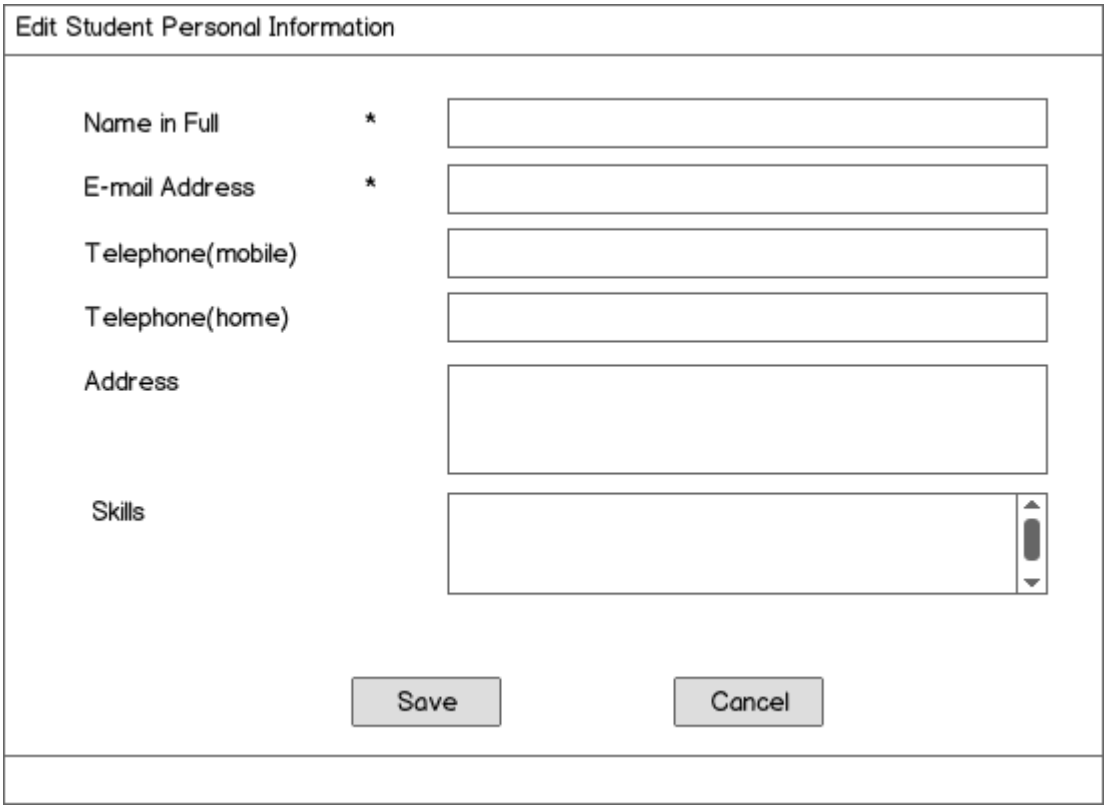


3.7.3 Use case 3: Edit System User Profile

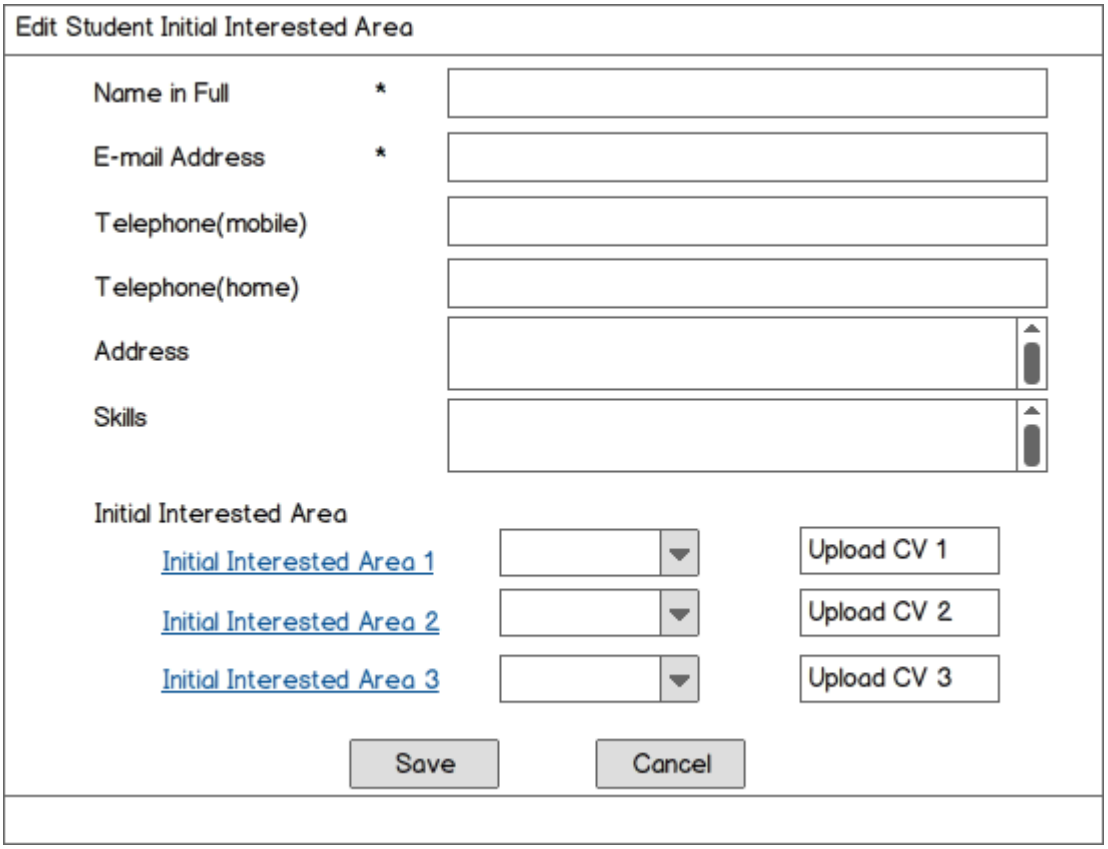




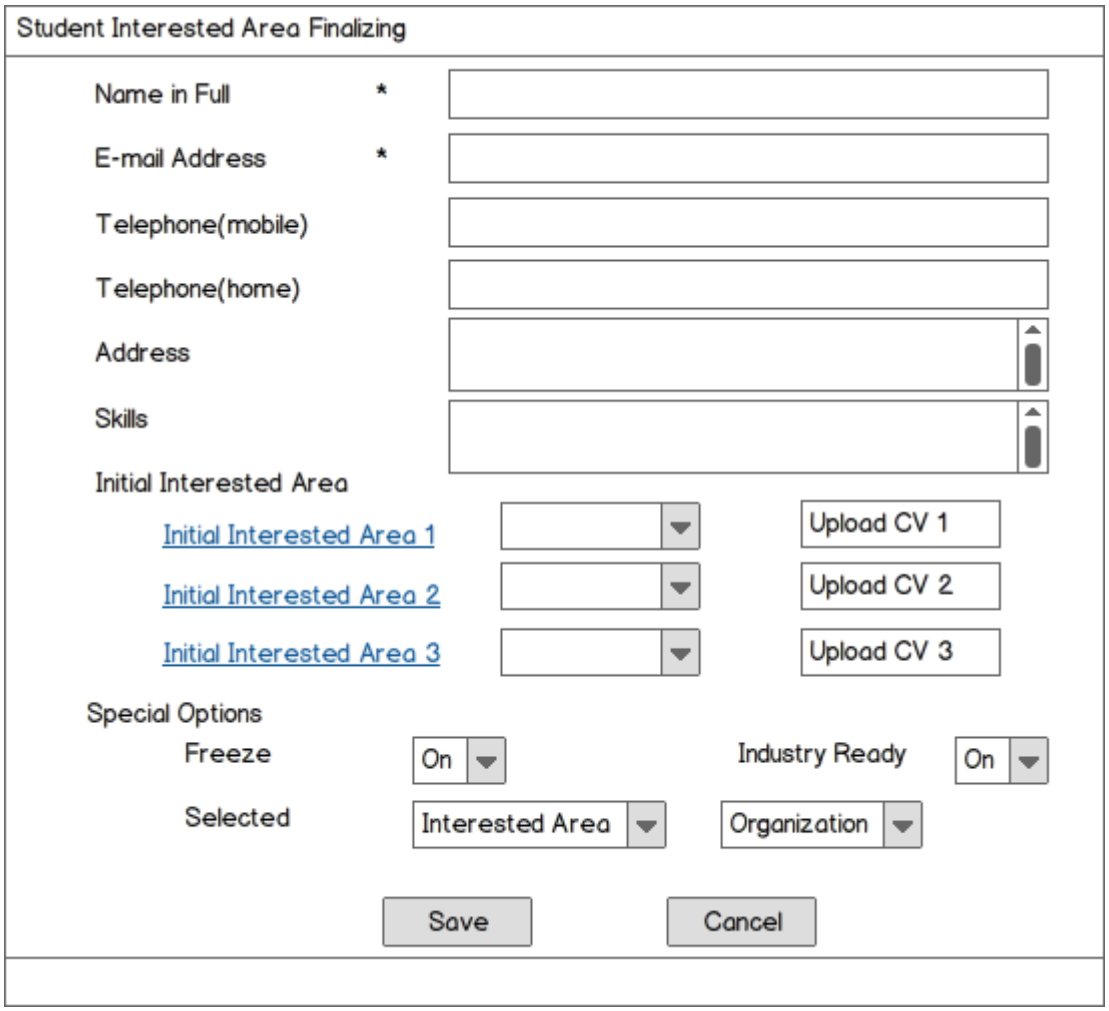
3.7.5 Use case 5: Edit Student Personal Information



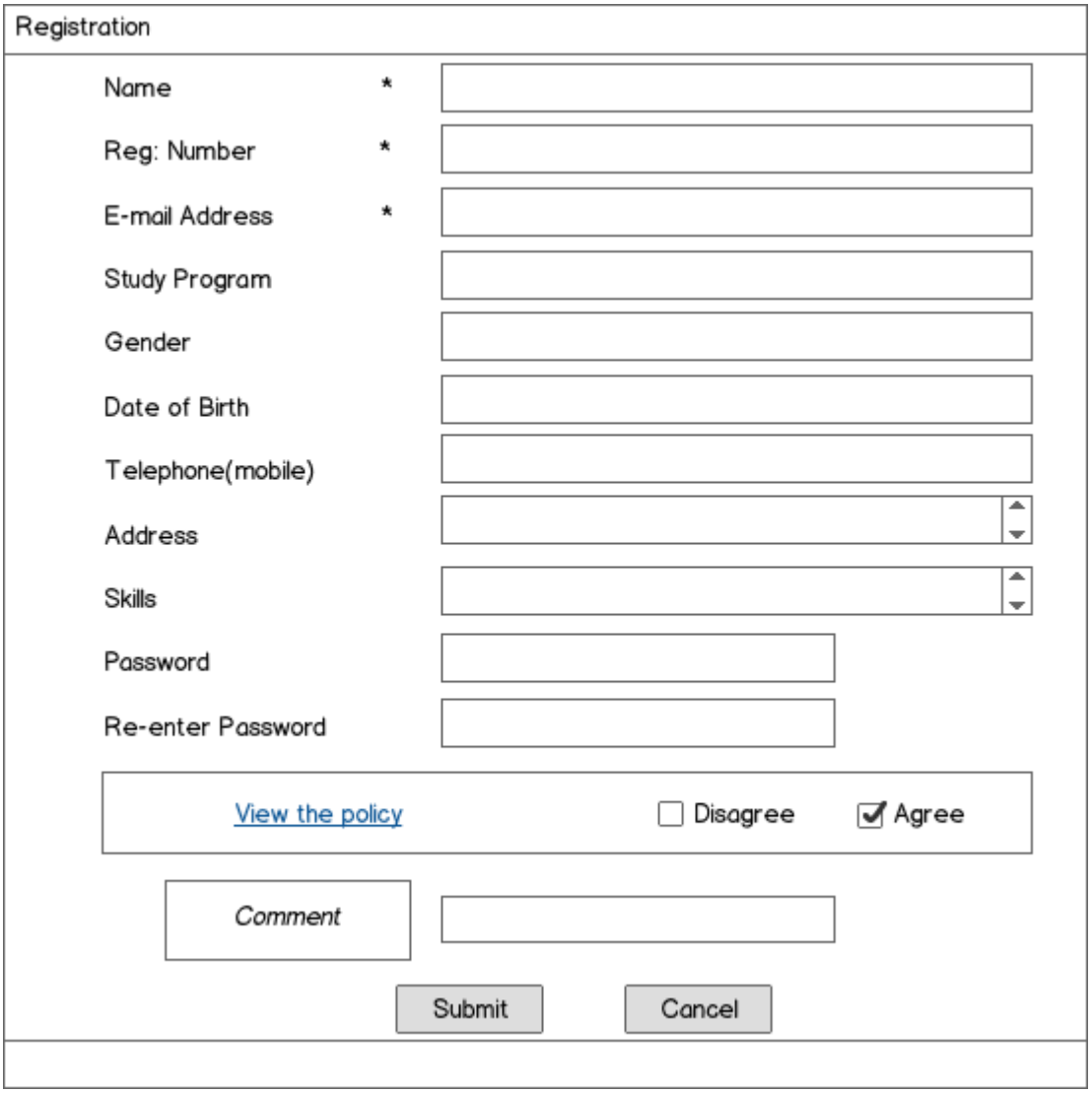
3.7.6 Use case 6: Edit Student Initial Choices



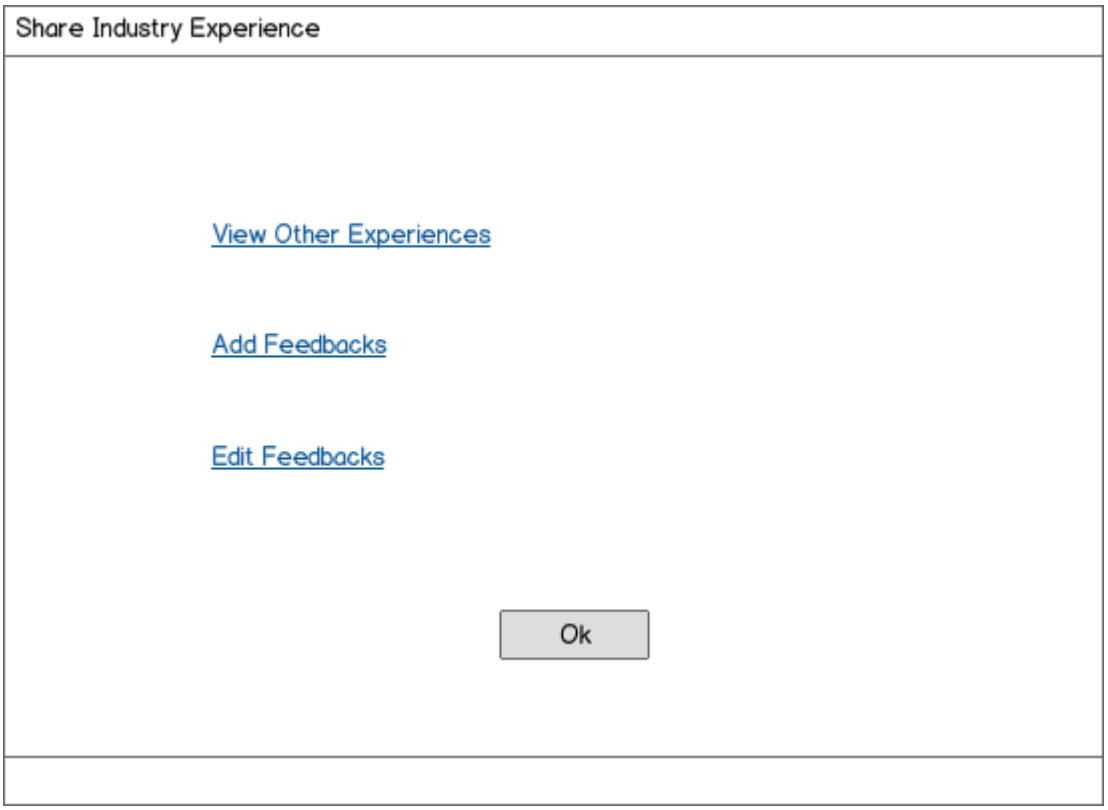
3.7.7 Use case 7: Student Choices Finalizing



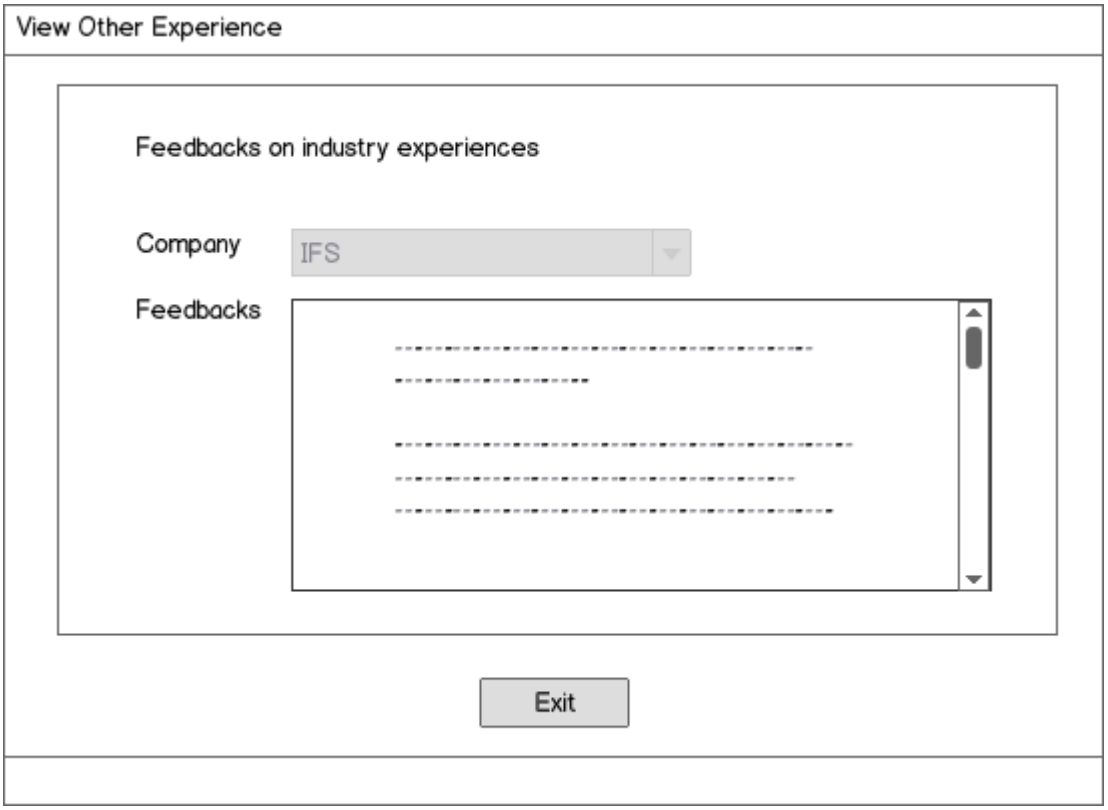
3.7.8 Use case 8: Registration



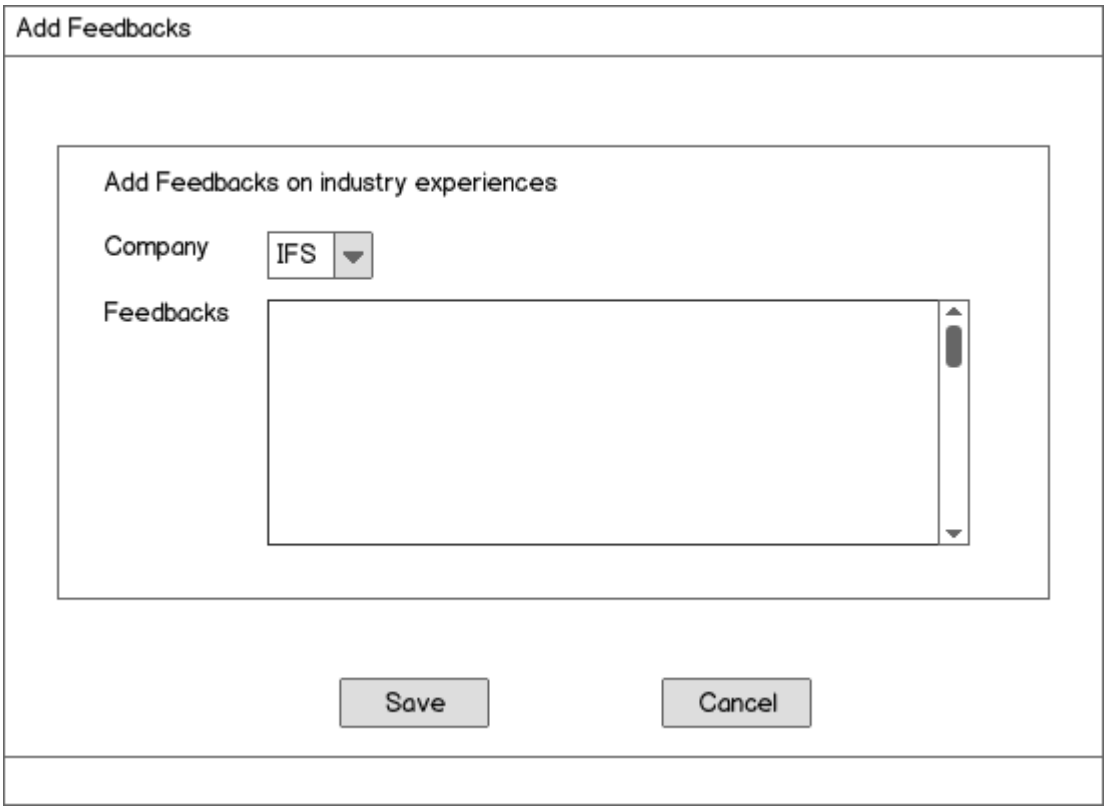
3.7.9 Use case 9: Share Industry Experience



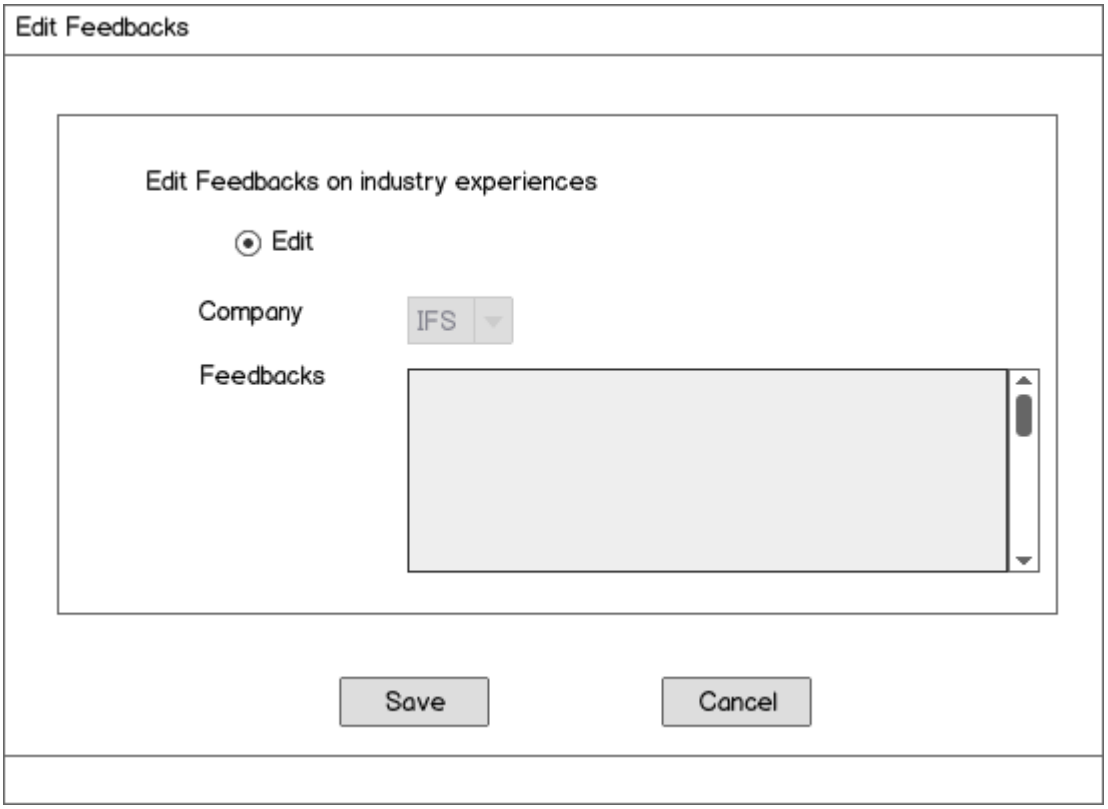
3.7.10 Use case 10 : View others experience



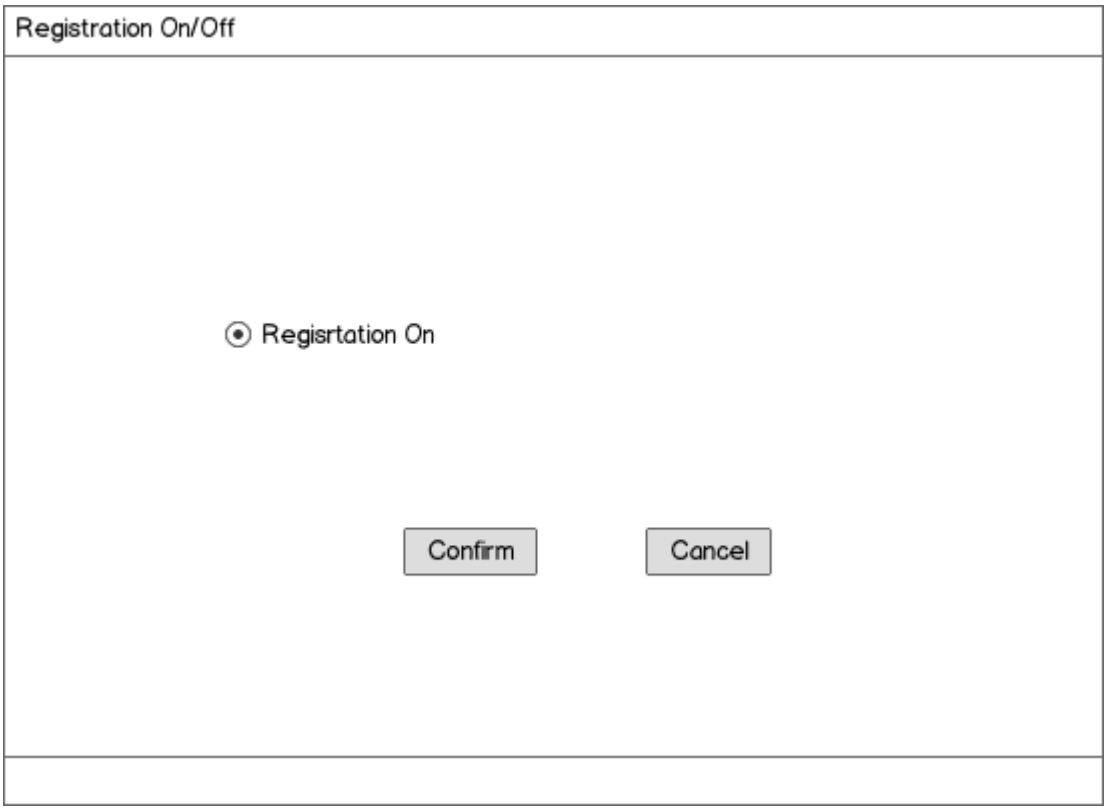
3.7.11 Use case 11 : Add feedbacks



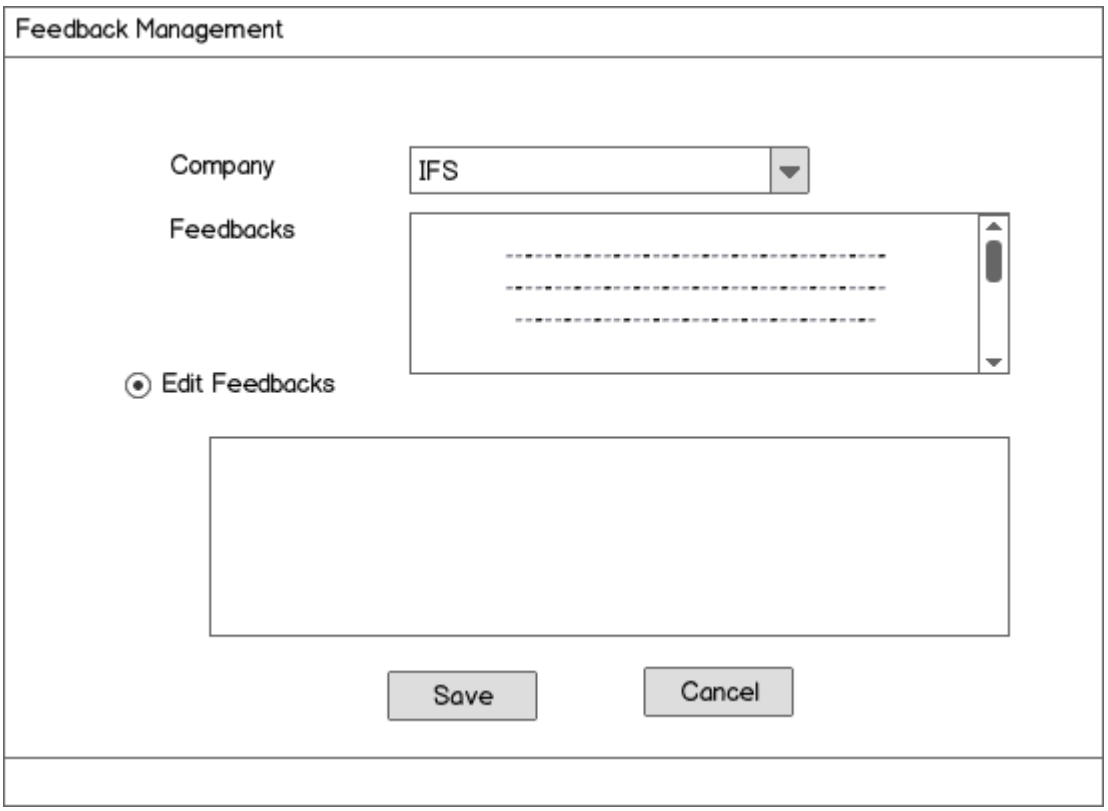
3.7.12 Use case 12 : Edit feedback



3.7.13 Use case 13 : Registration on/off

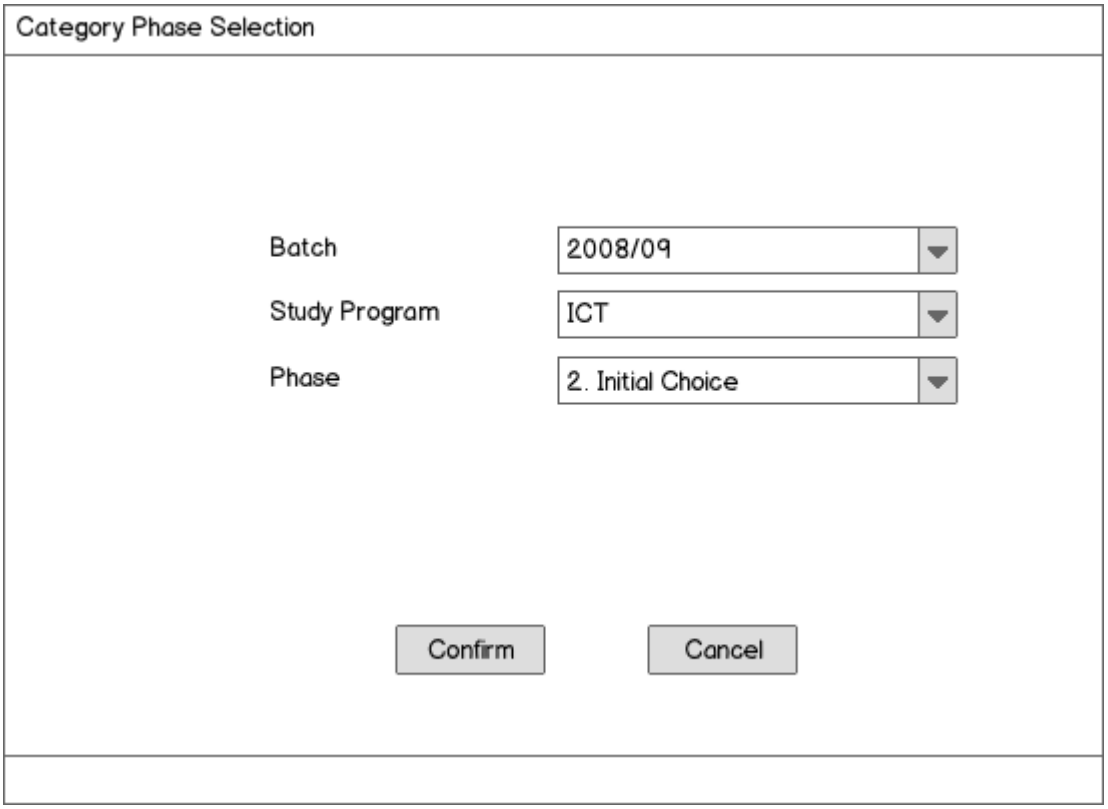


3.7.14 Use case 14 : Feedback management

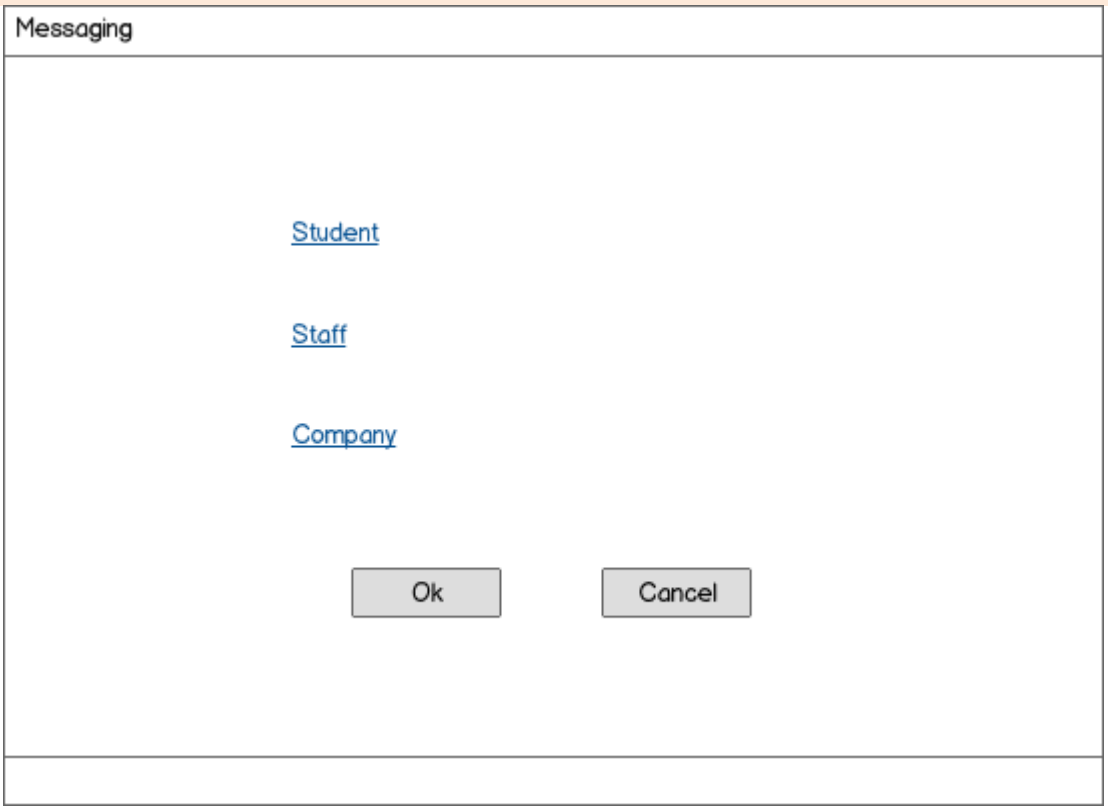


3.7.15 Use case 15 : Industry Grant Opportunity

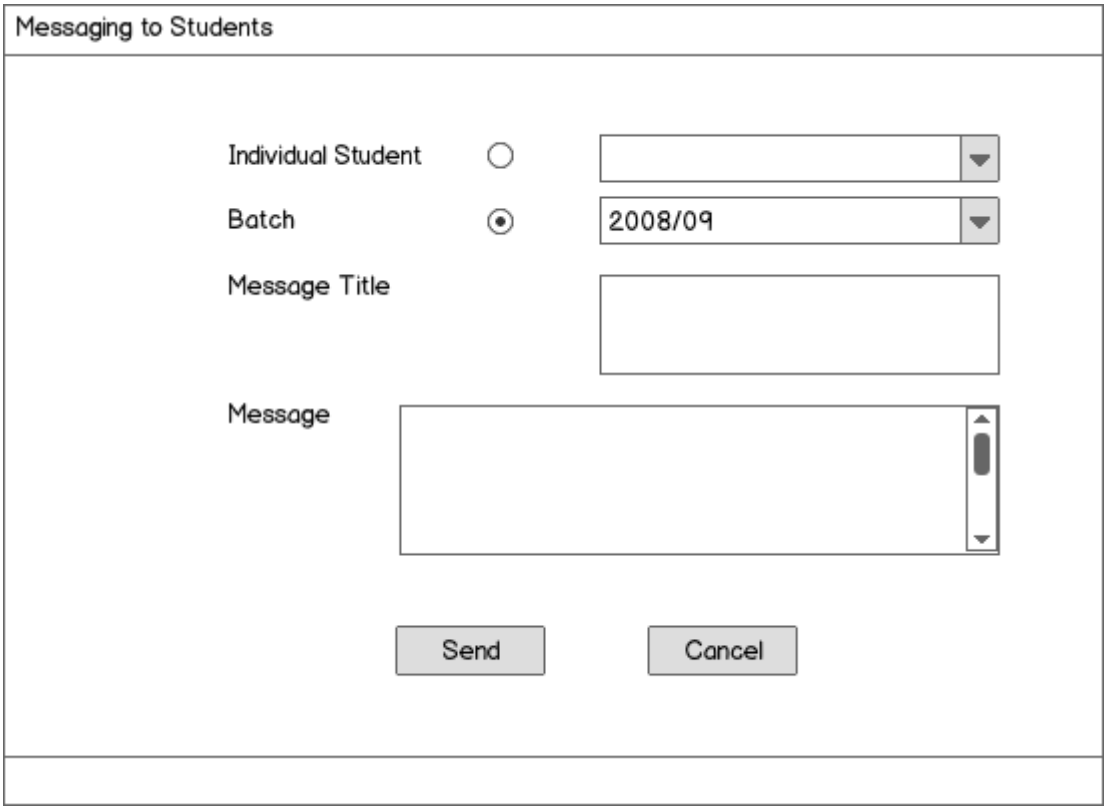
3.7.16 Use case 16 : Category phase selection



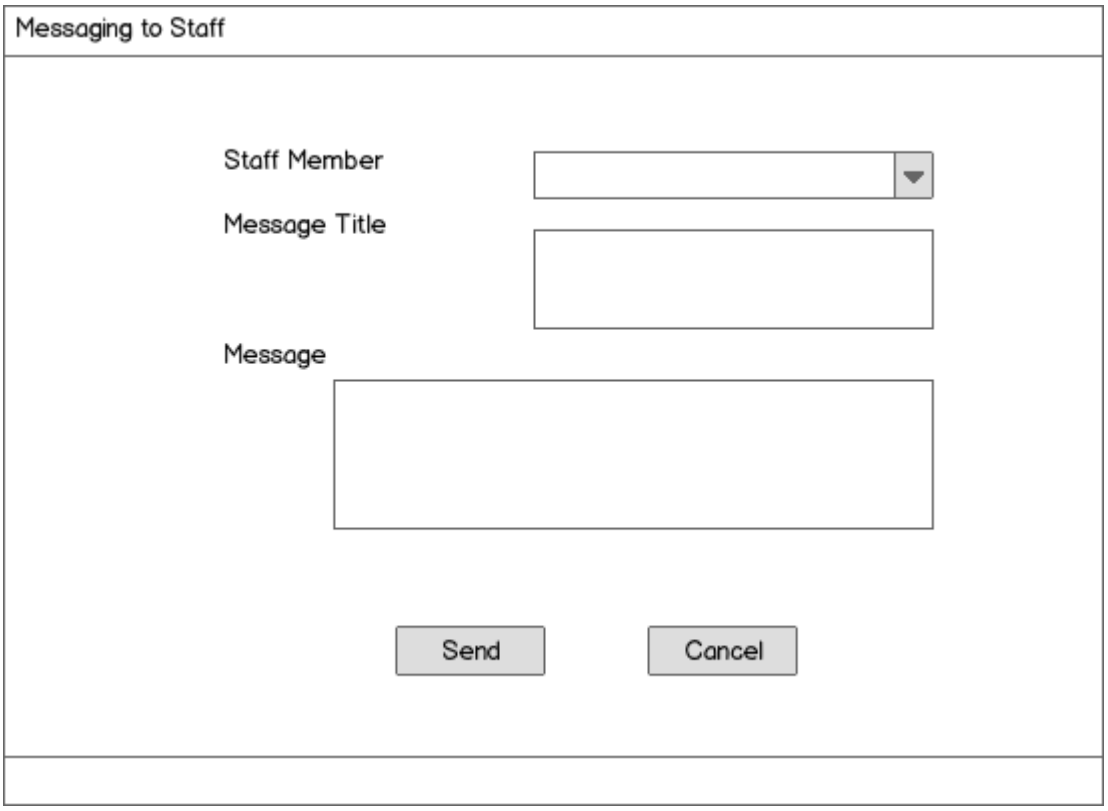
3.7.17 Use case 17 : Messaging



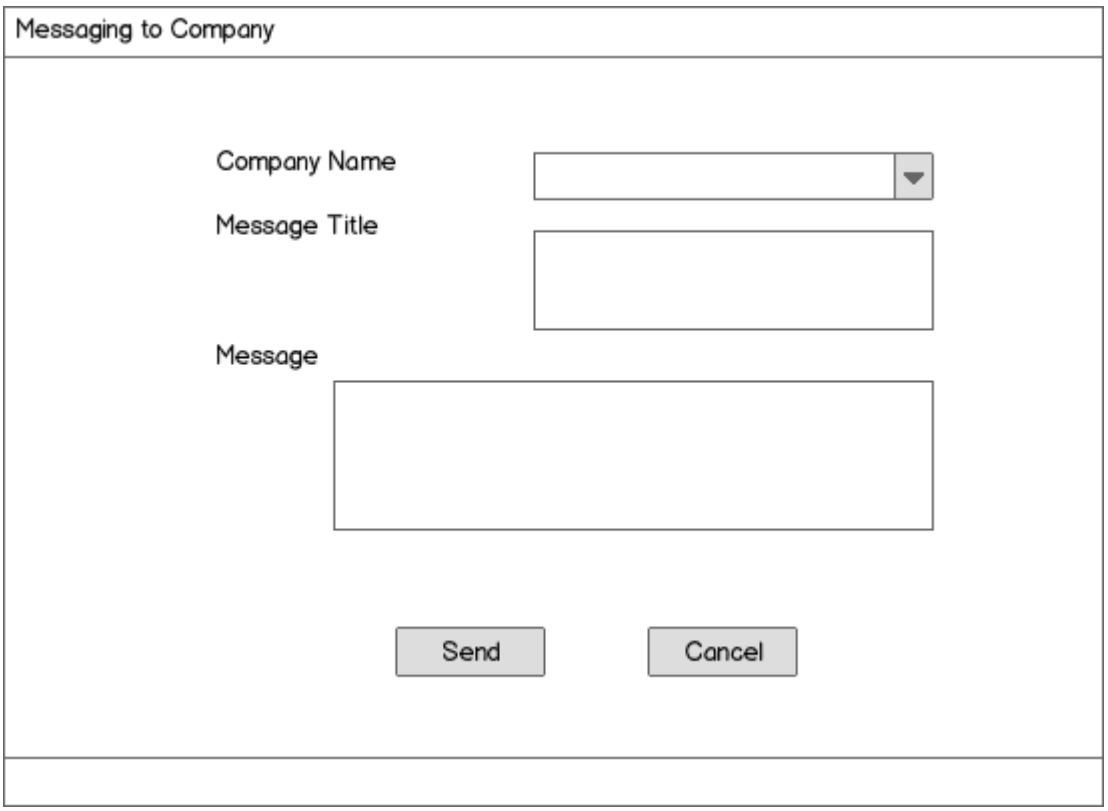
3.7.18 Use case 18 : Messaging to students



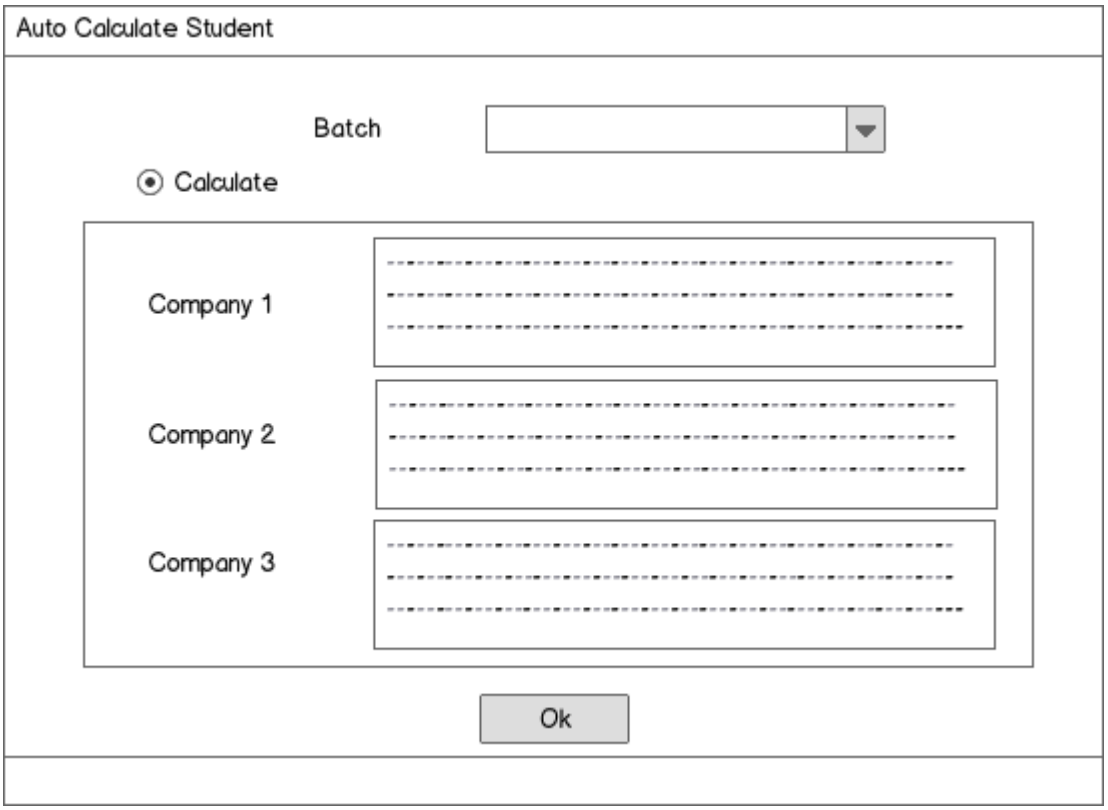
3.7.19 Use case 19 : Messaging to staff



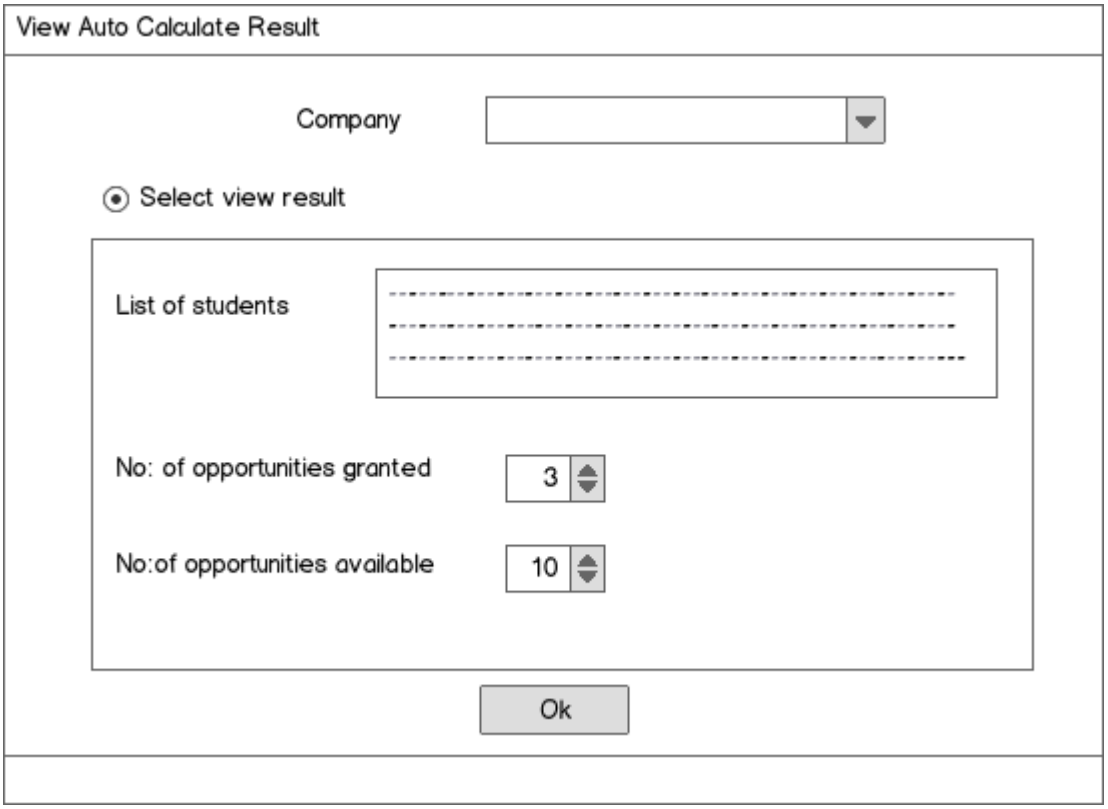
3.7.20 Use case 20 : Messaging to company



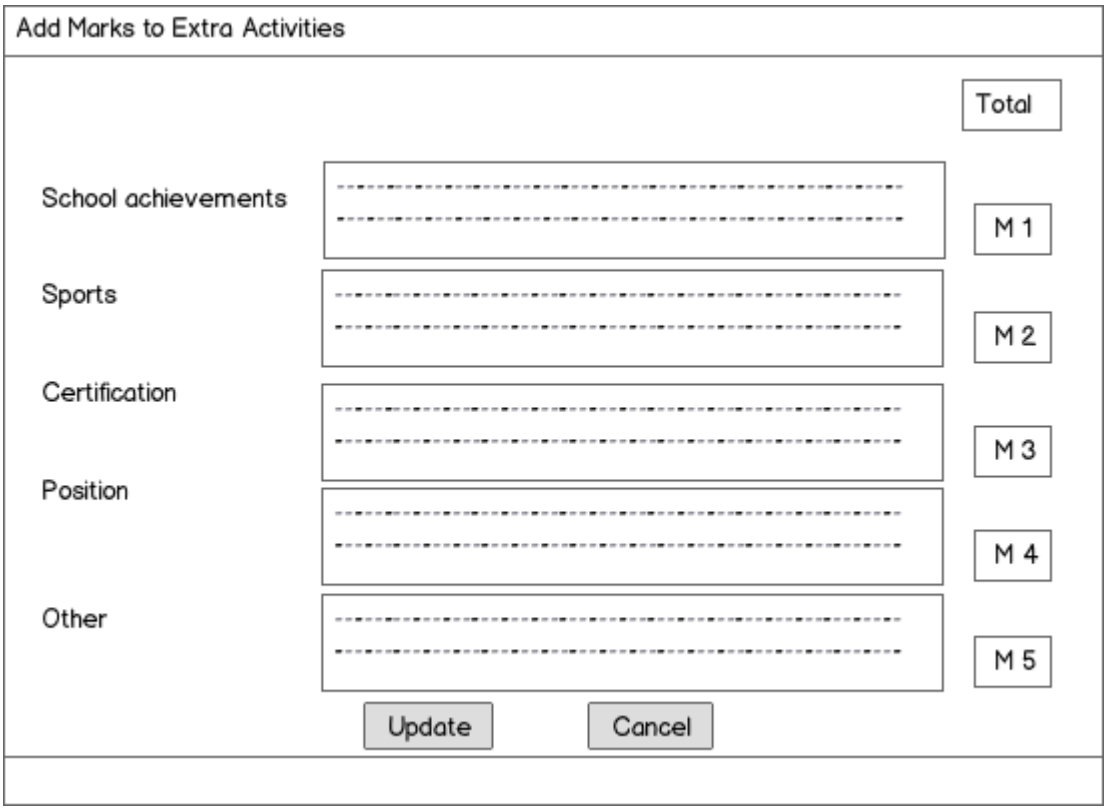
3.7.21 Use case 21 : Auto calculate student



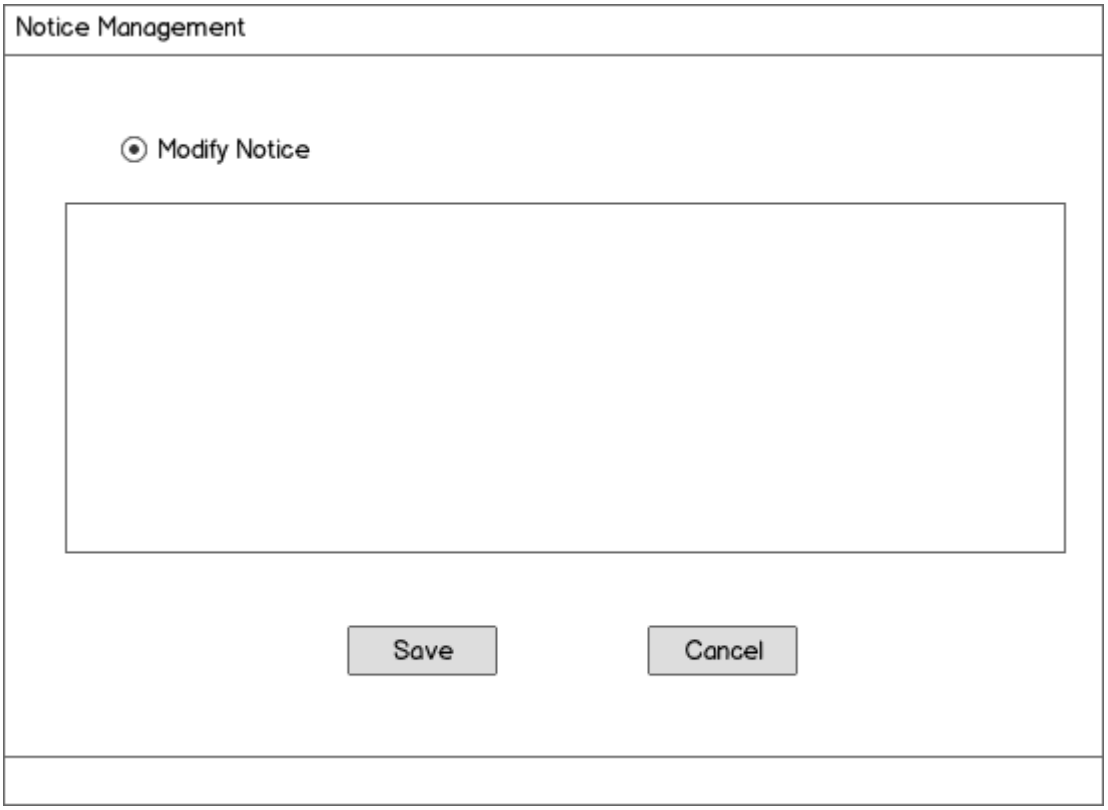
3.7.22 Use case 22 : View auto calculated result



3.7.23 Use case 23 : Add marks to extra activities

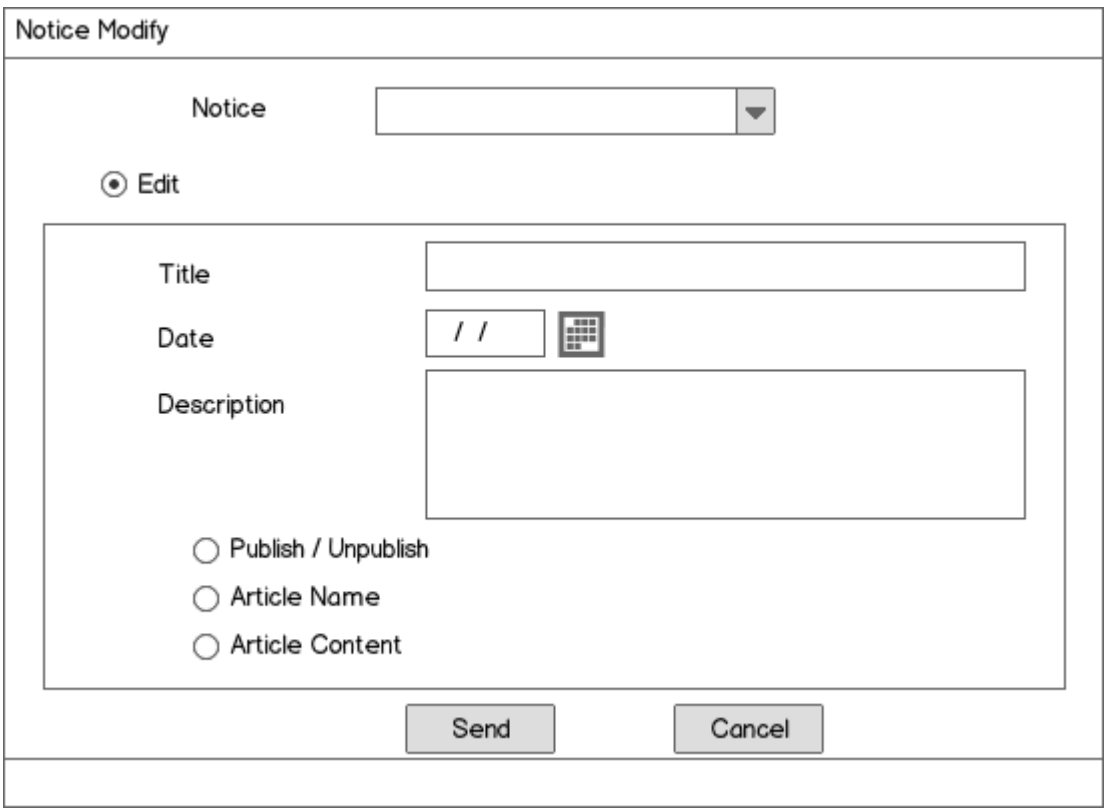


3.7.24 Use case 24 : Notice management

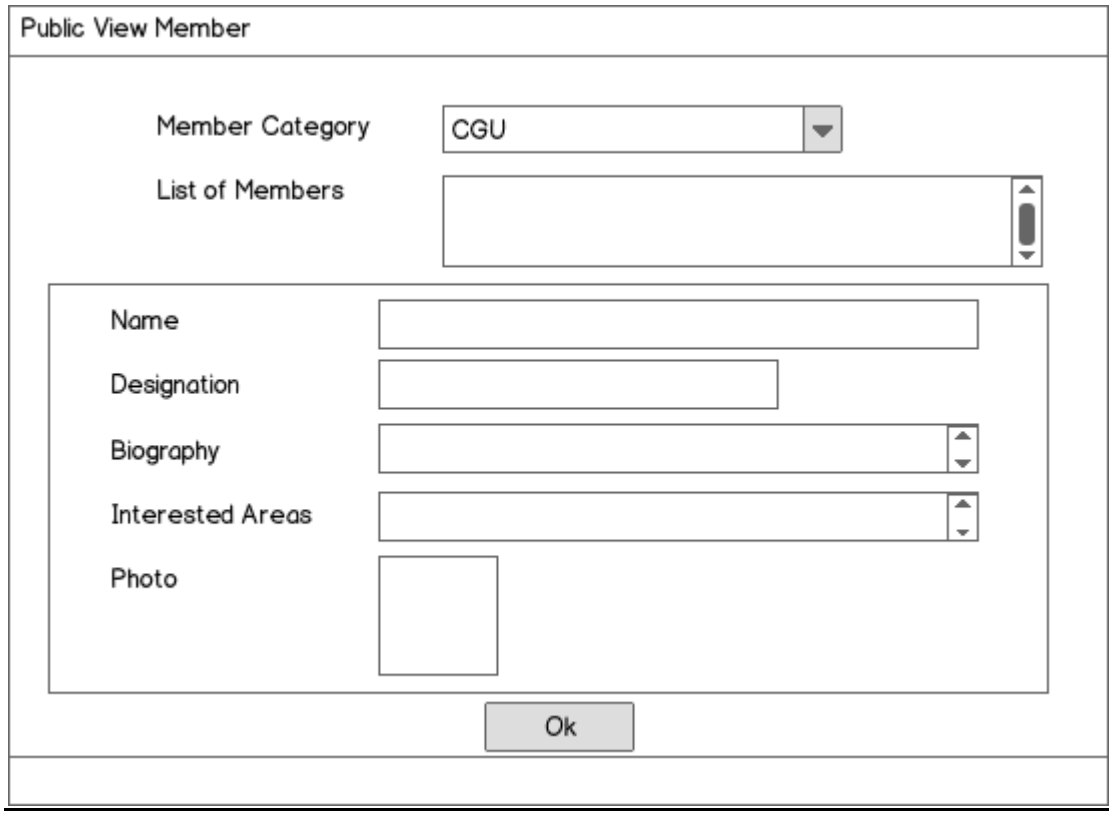


3.7.25 Use case 25 : Notice creation

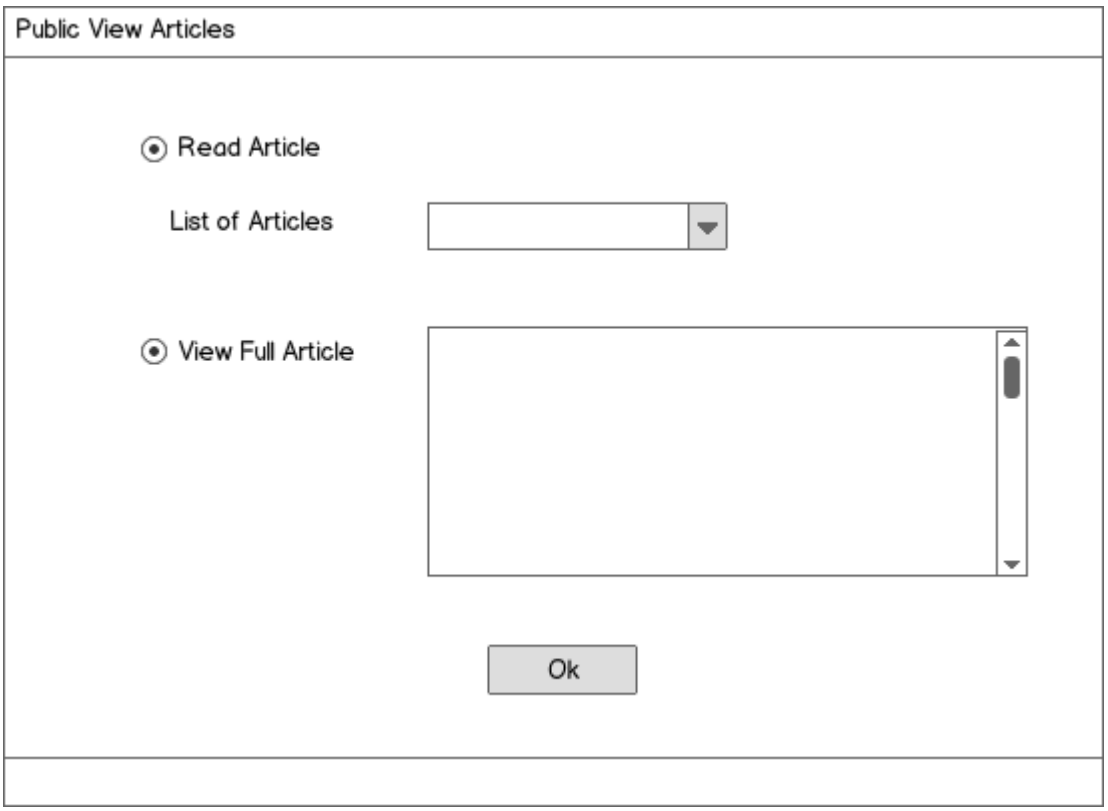
3.7.26 Use case 26 : Notice modify



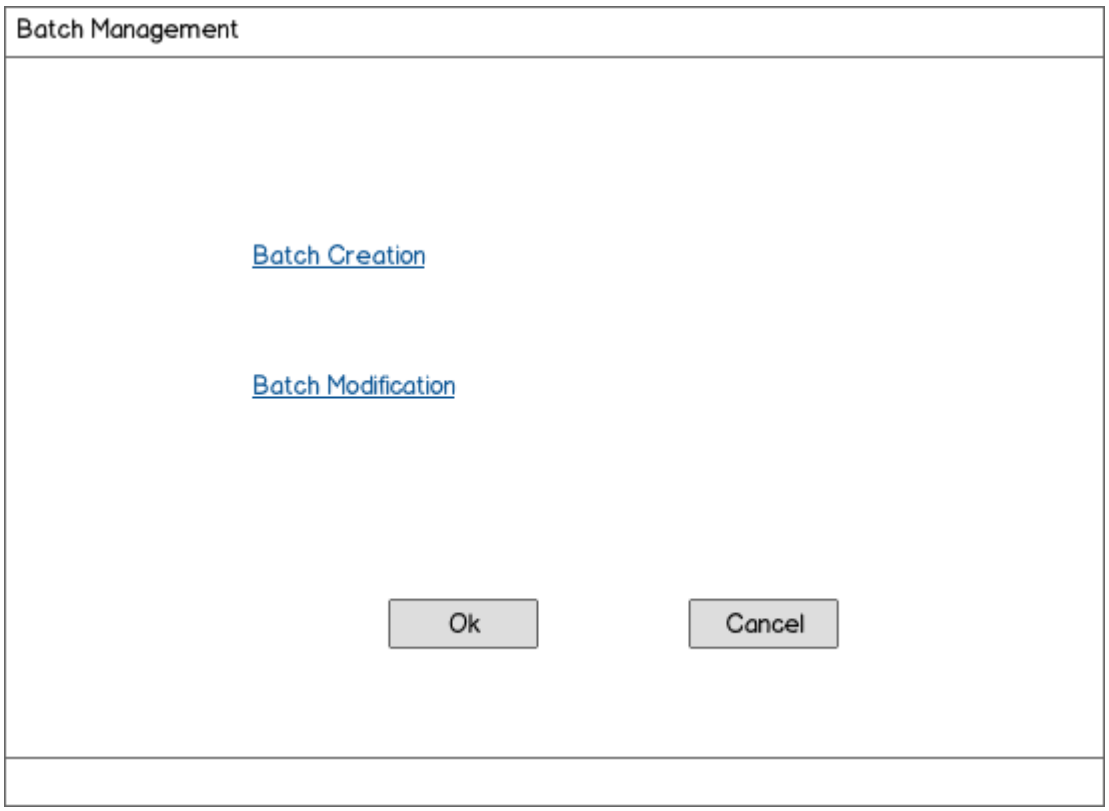
3.7.28 Use case 28 : Public view members



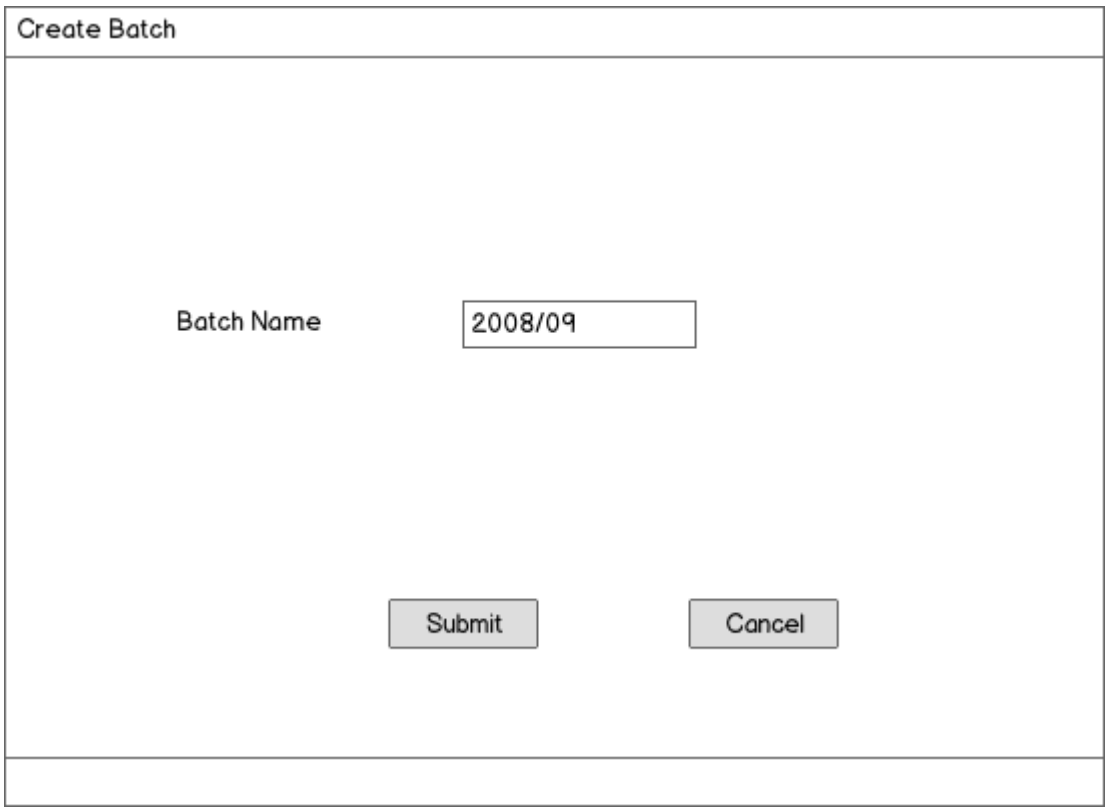
3.7.29 Use case 29 : Public view articles



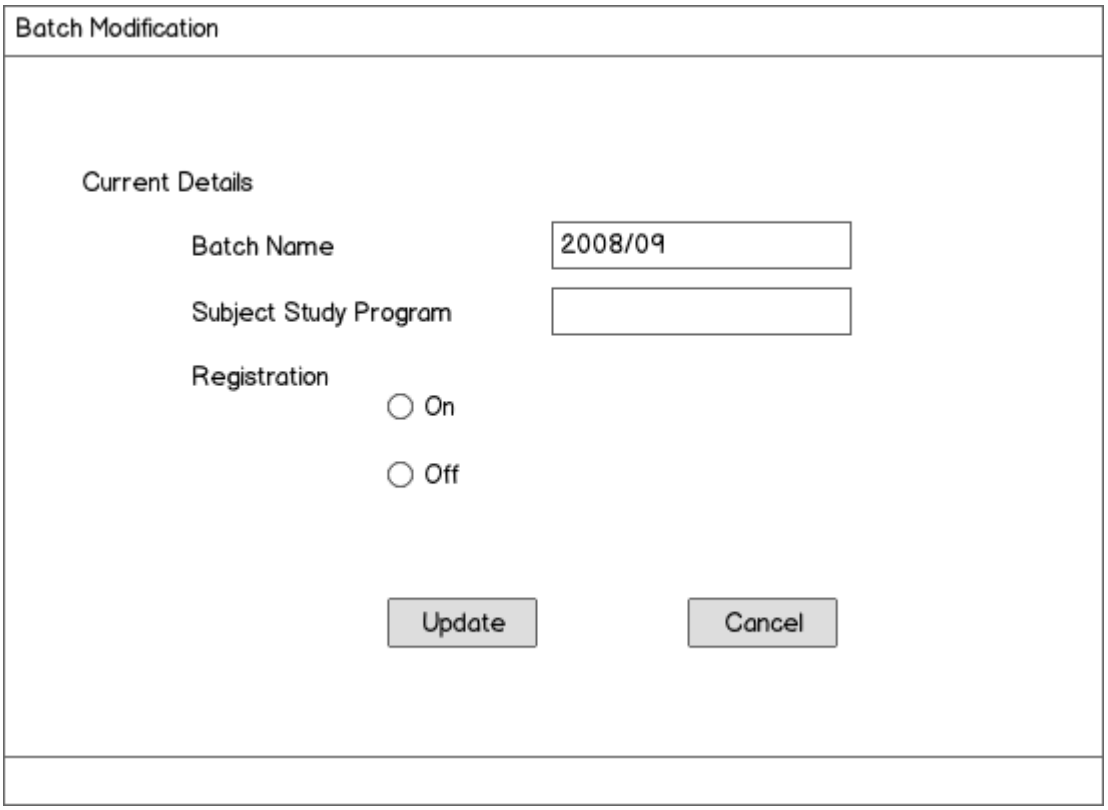
3.7.30 Use case 30 : Category Management



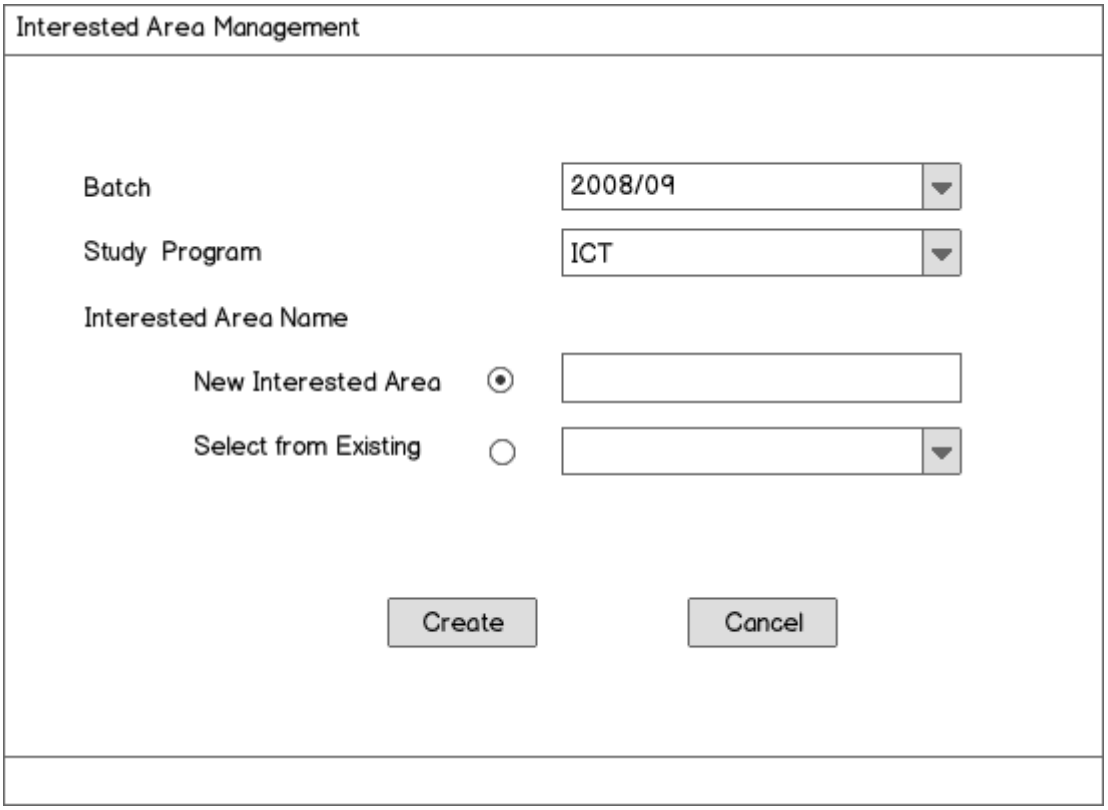
3.7.31 Use case 31 : Create Category



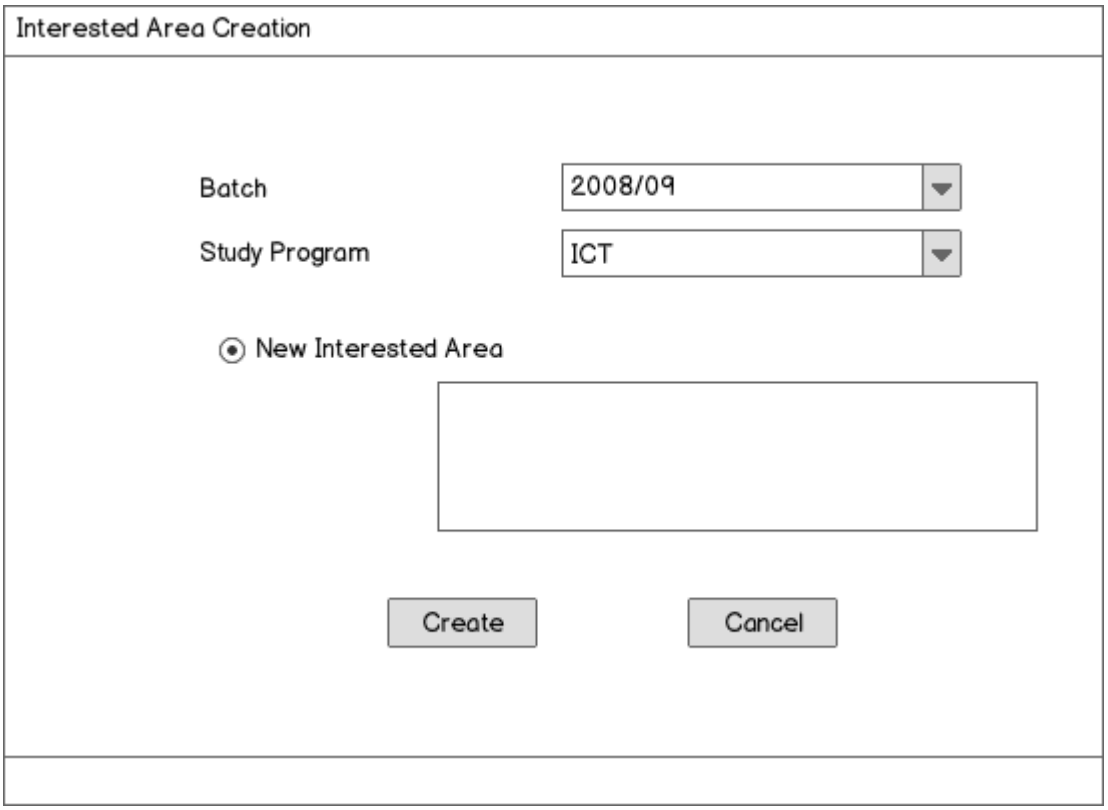
3.7.32 Use case 32 : Category Modification



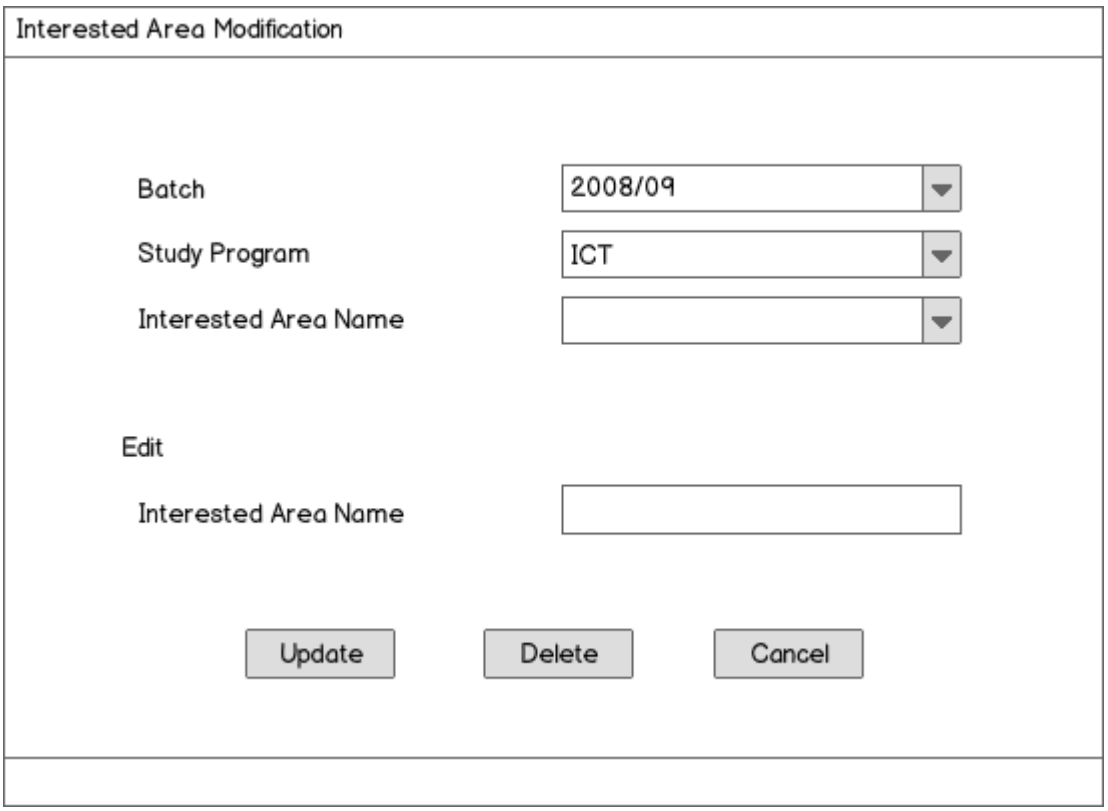
3.7.33 Use case 33 : Choice Management



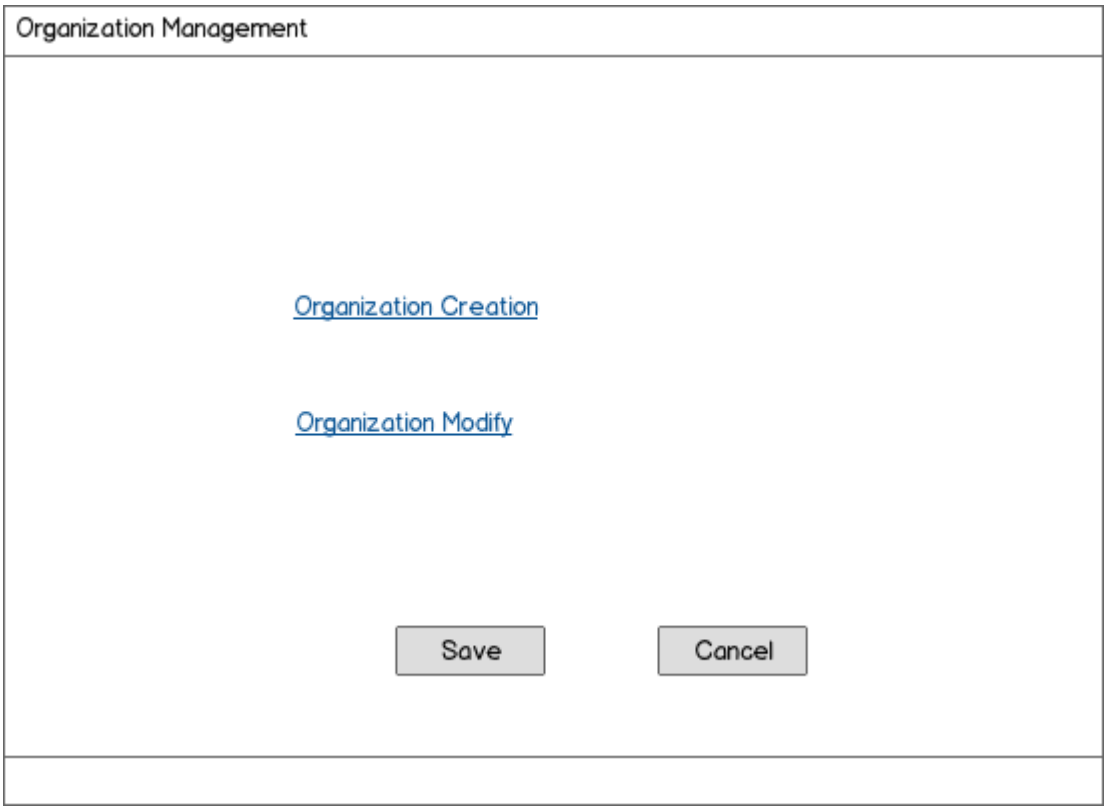
3.7.34 Use case 34 : Choice Creation



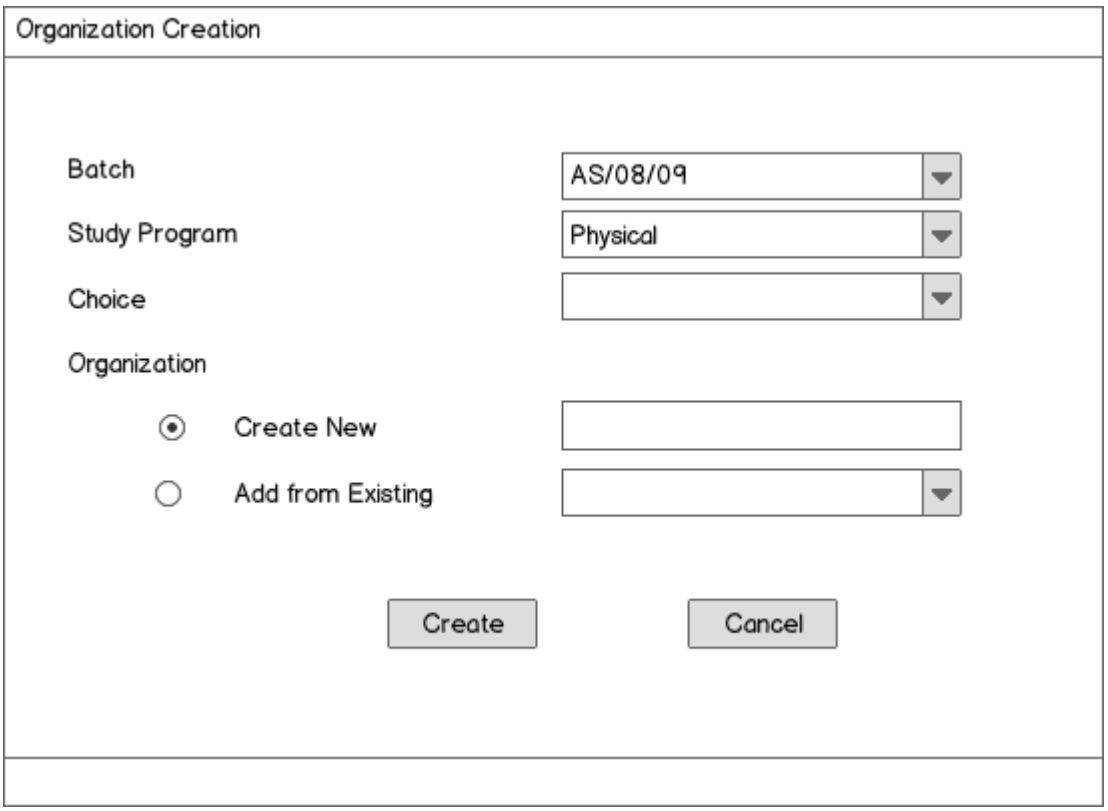
3.7.35 Use case 35 : Choice Modify



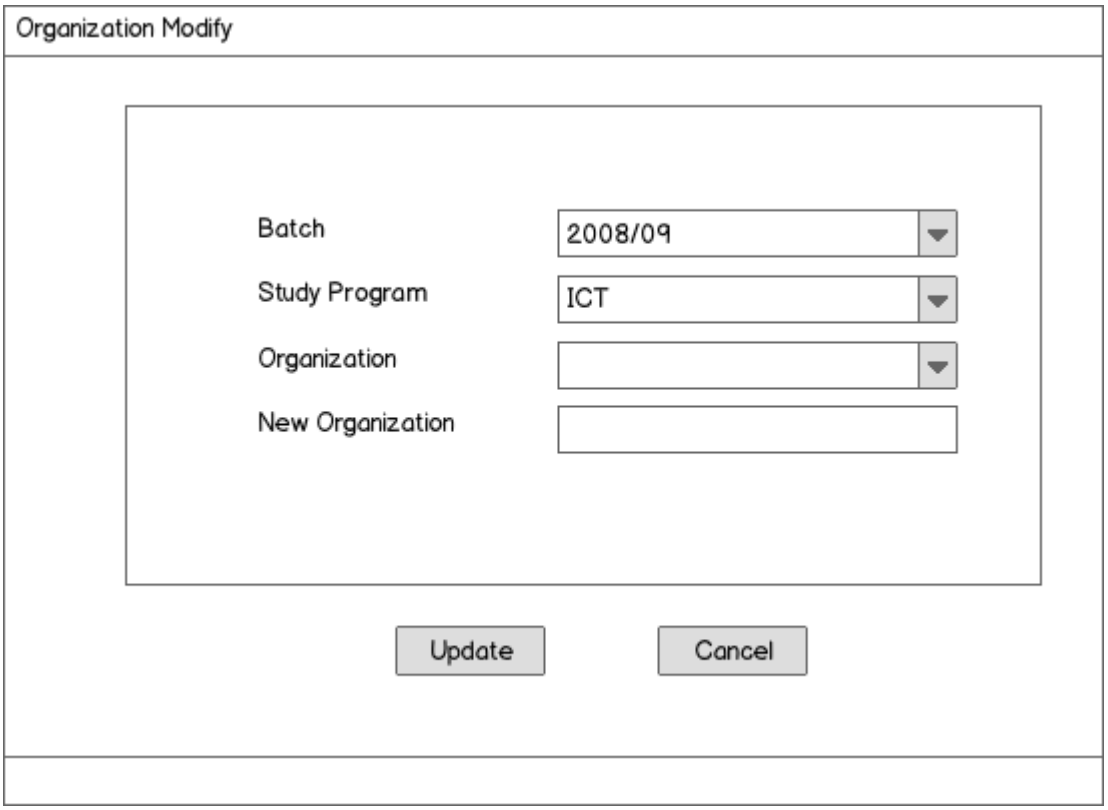
3.7.36 Use case 36 : Organization Management



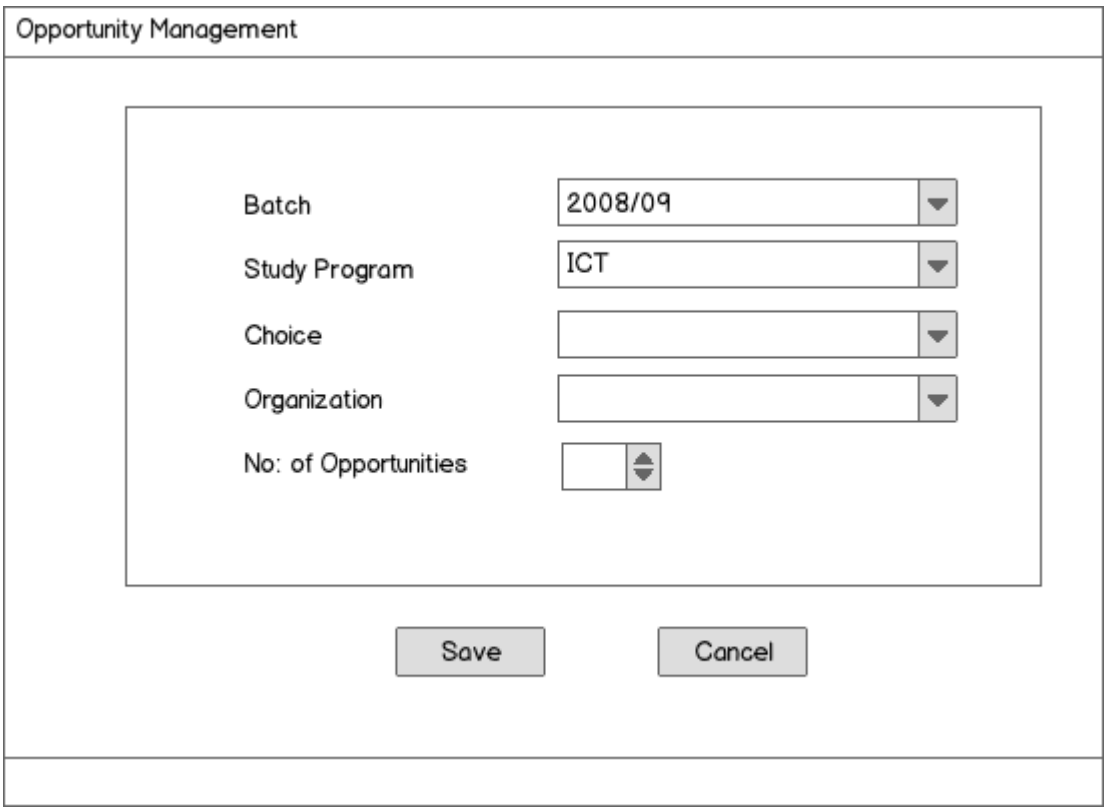
3.7.37 Use case 37 : Organization Creation



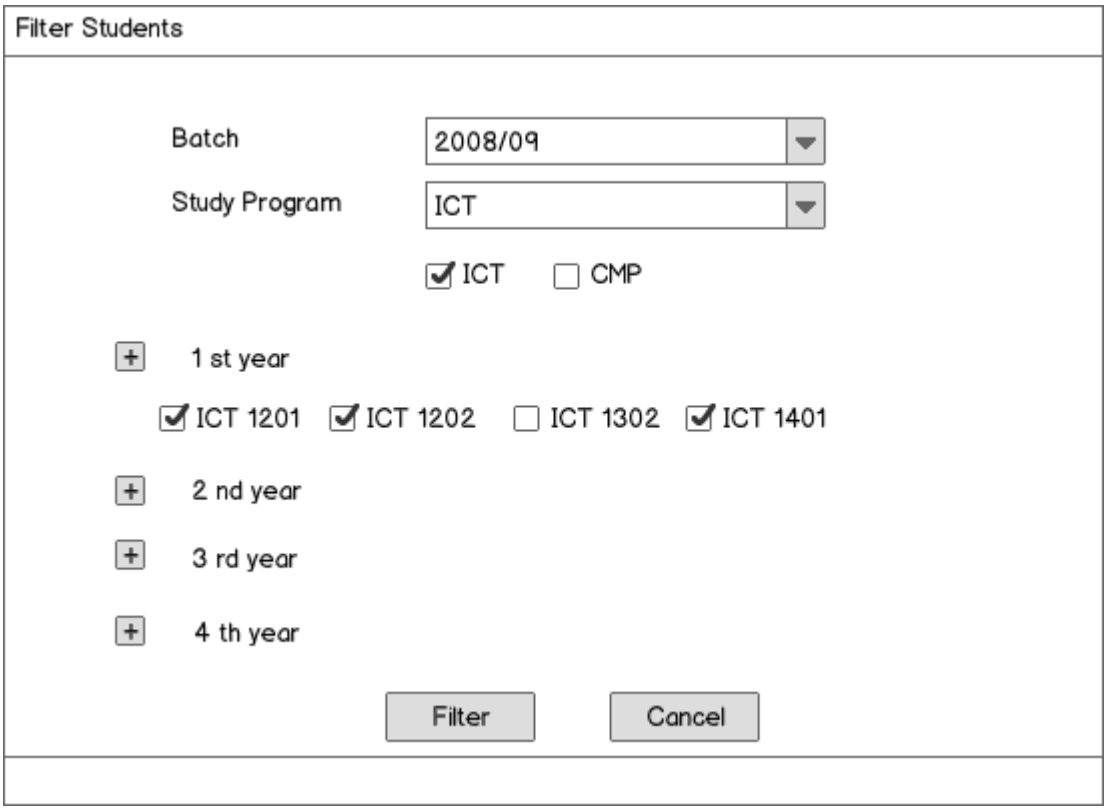
3.7.38 Use case 38 : Organization Modify



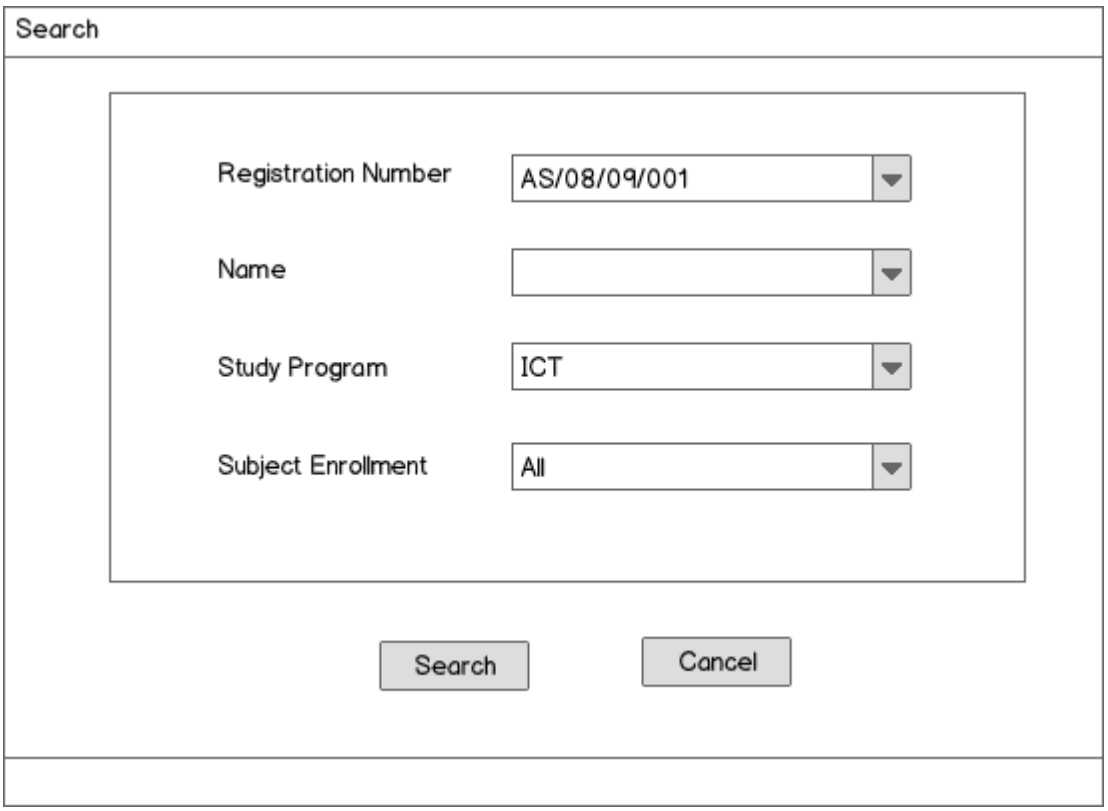
3.7.39 Use case 39 : Opportunity Management



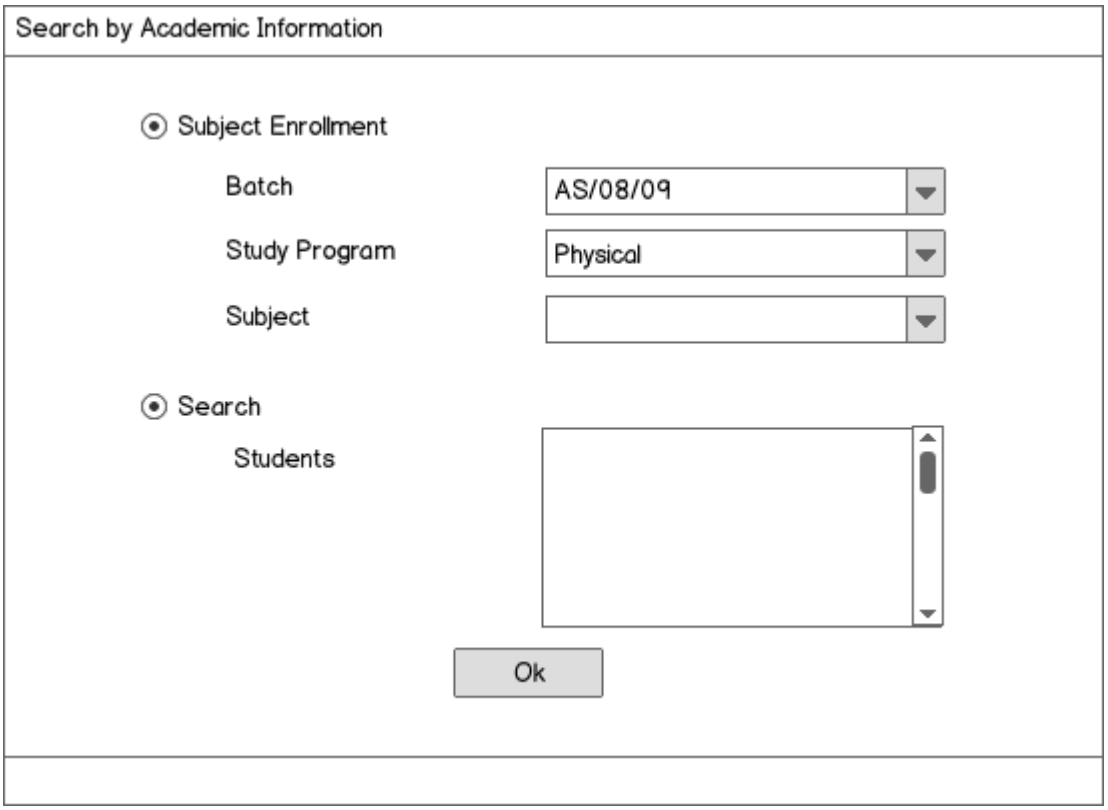
3.7.40 Use case 40 : Filter Students



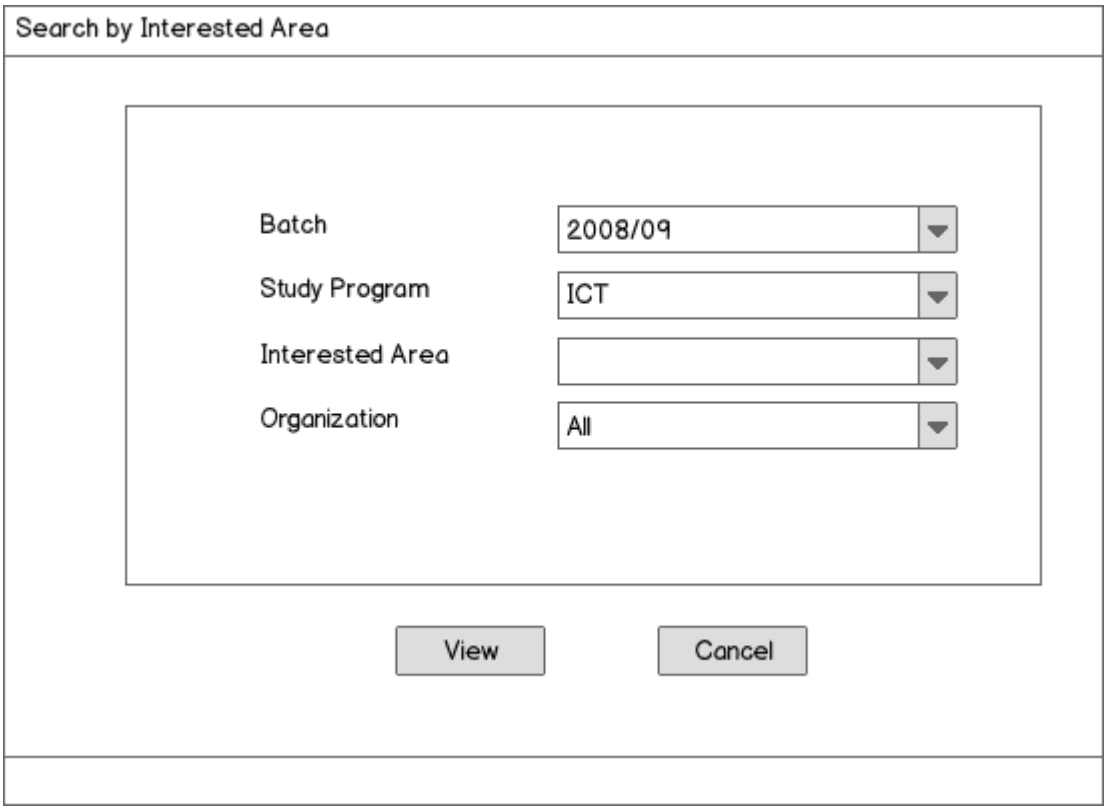
3.7.41 Use case 41 : Search



3.7.42 Use case 42 : Search by academic information

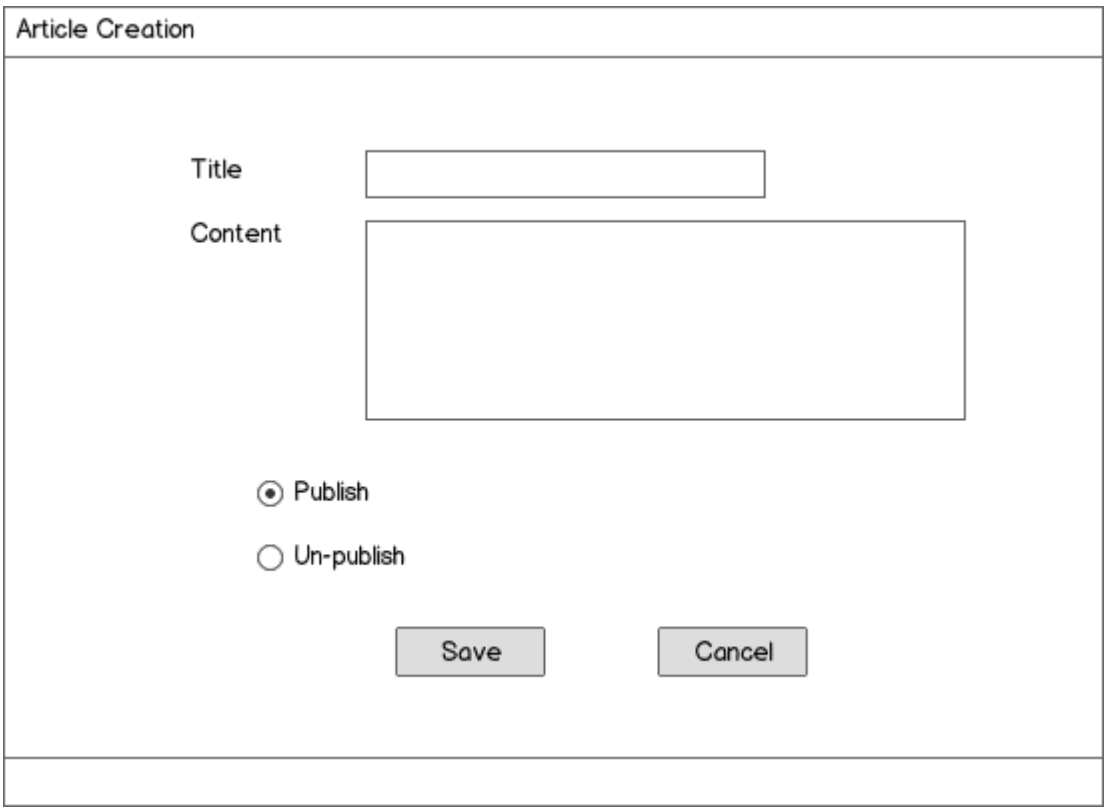


3.7.43 Use case 43 : Search by choice

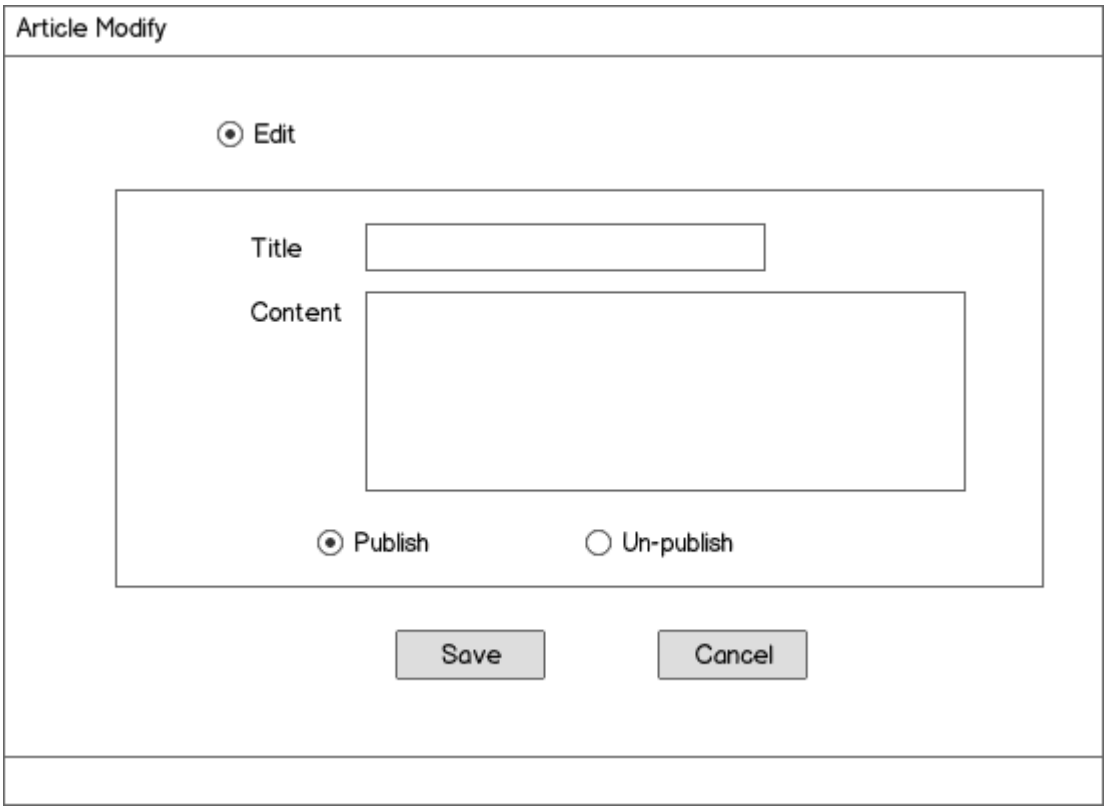


3.7.44 Use case 44 : Article management

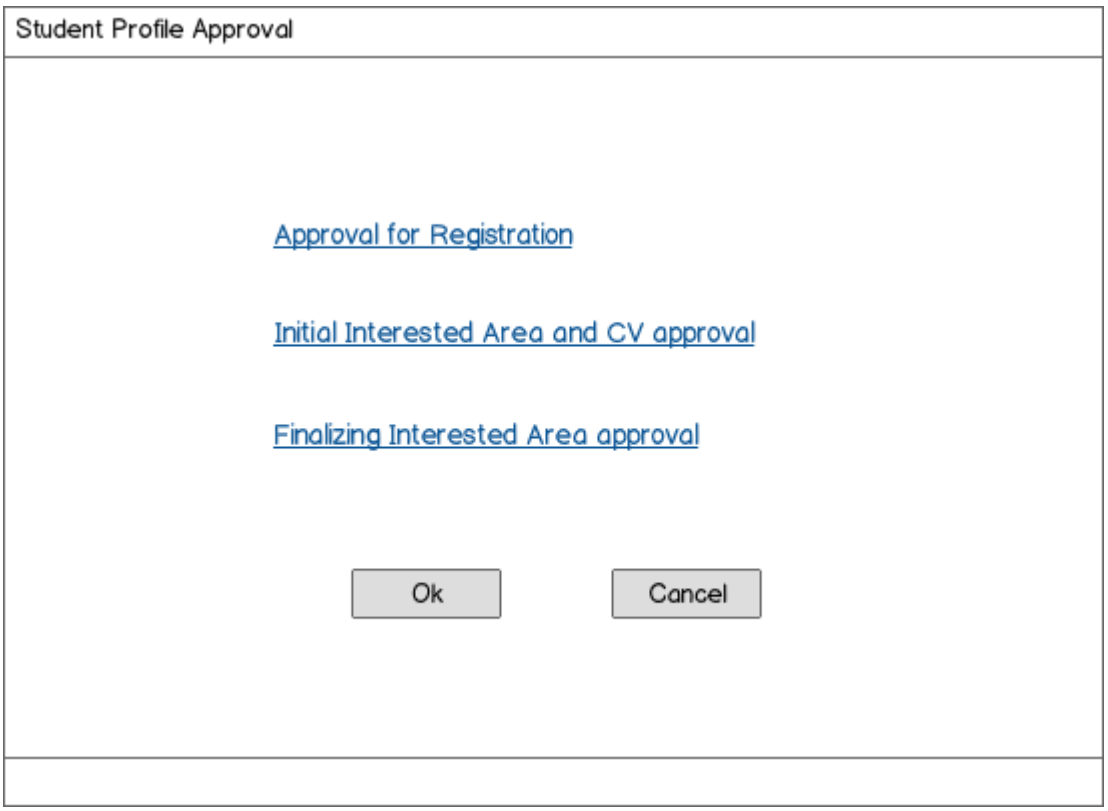
3.7.45 Use case 45 : Article creation



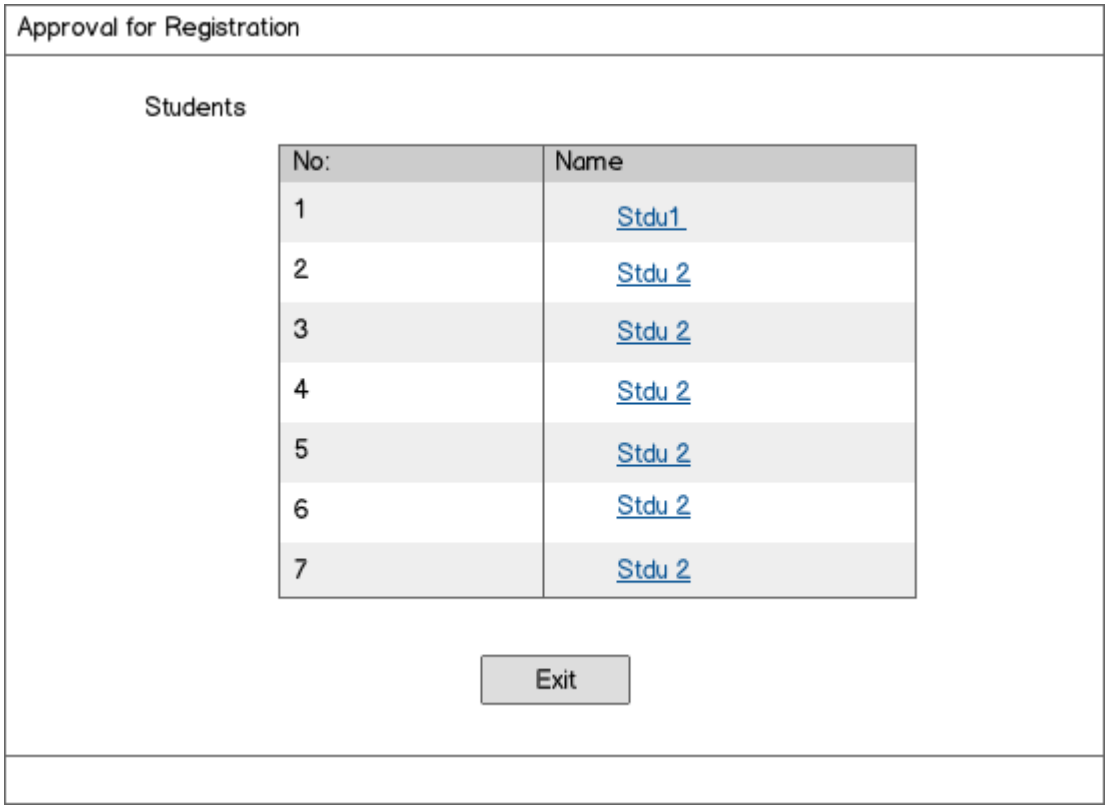
3.7.46 Use case 46 : Article modify



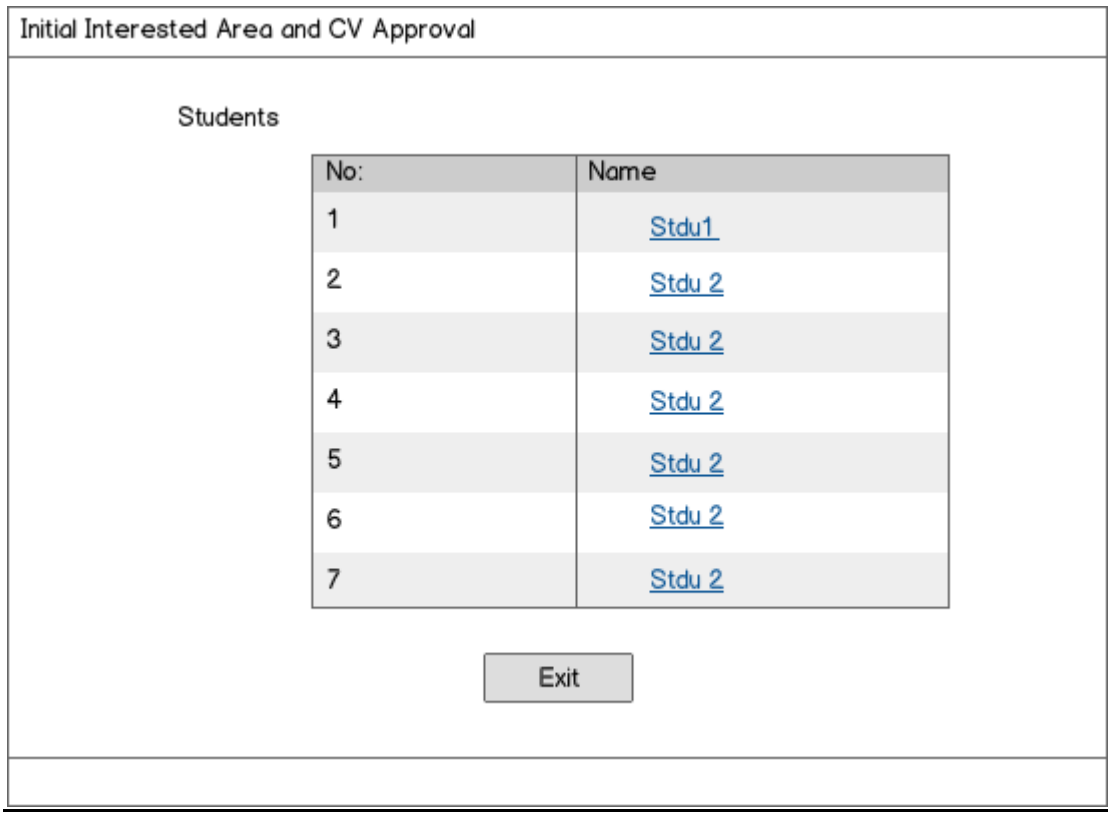
3.7.47 Use case 47 : Student profile approval



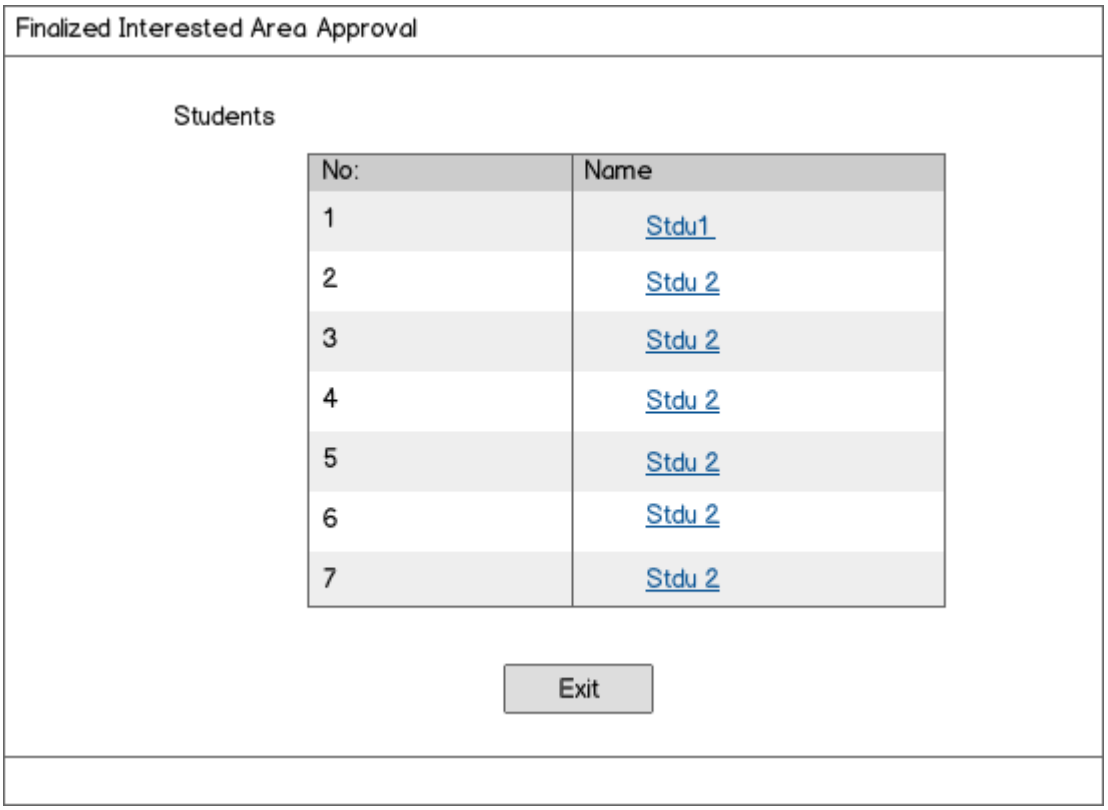
3.7.48 Use case 48 : Approval for Registration



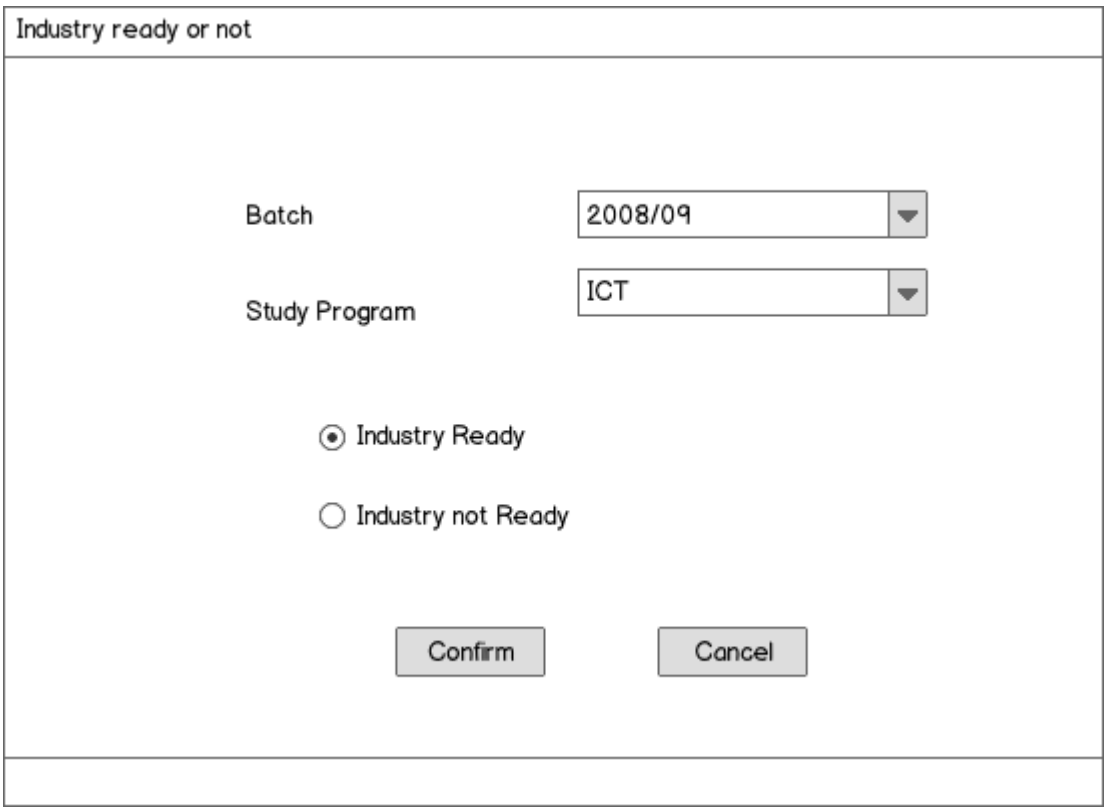
3.7.49 Use case 49 : Initial choice and CV approval



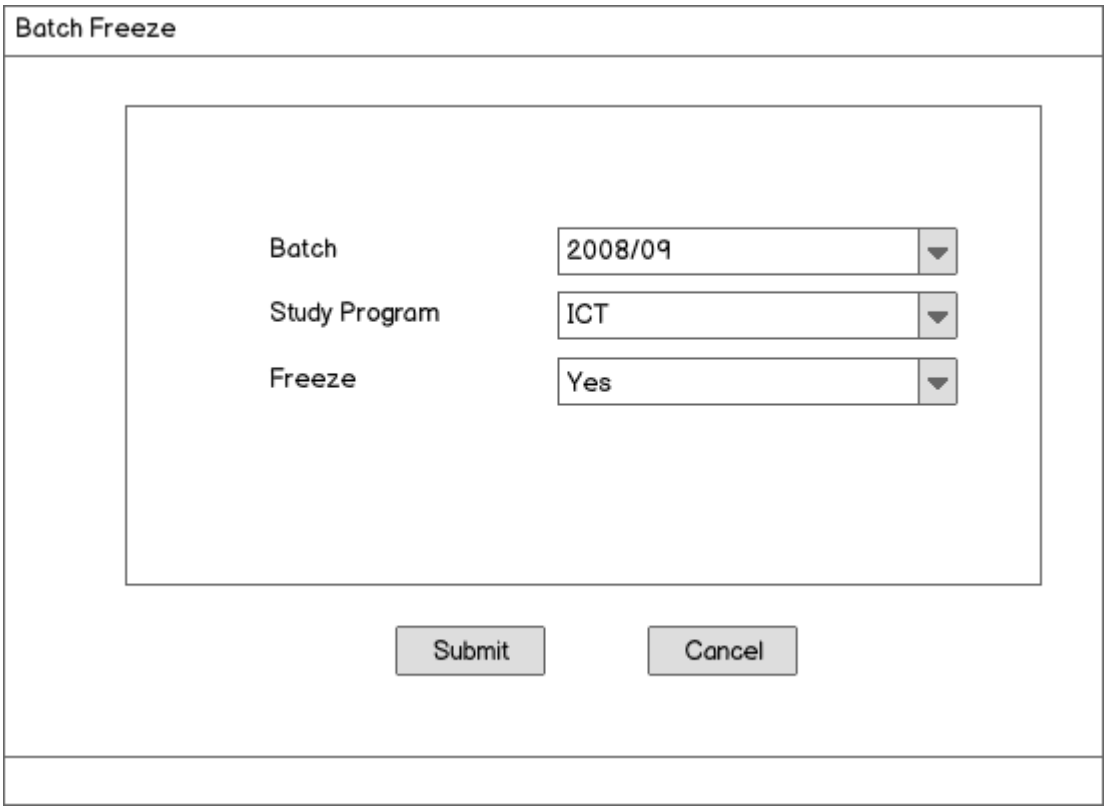
3.7.50 Use case 50 : Finalized choice approval



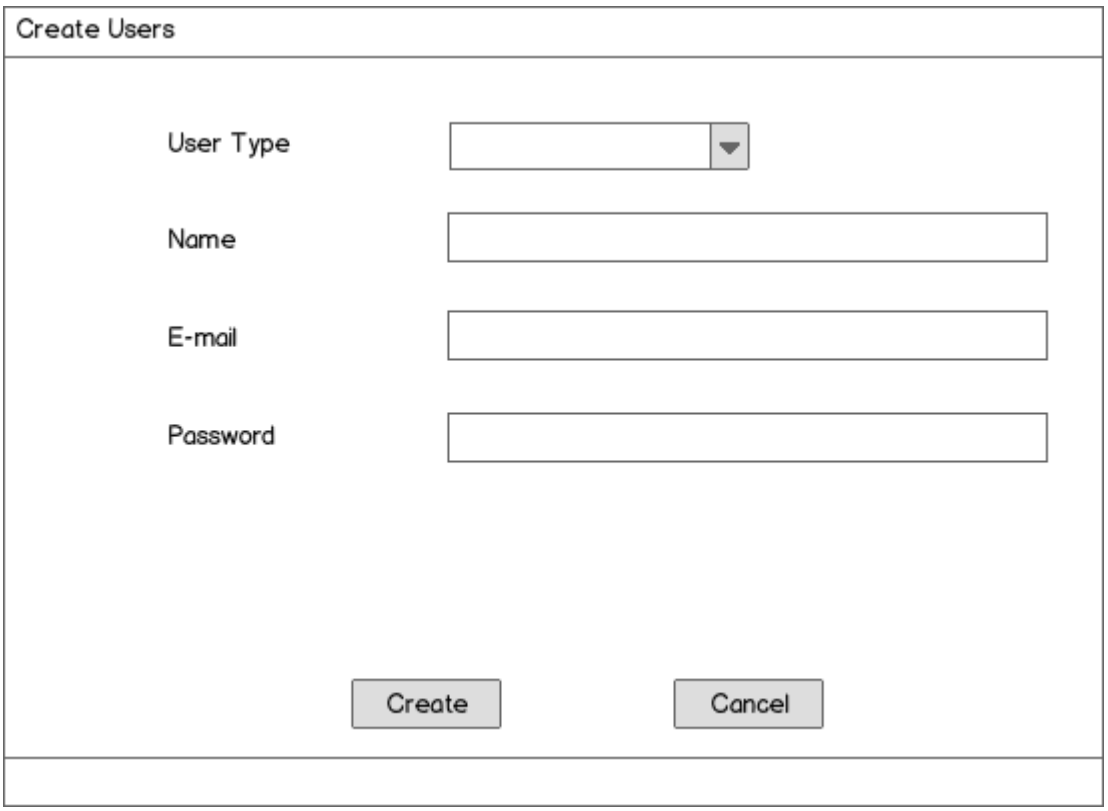
3.7.51 Use case 51 : Industry ready or not



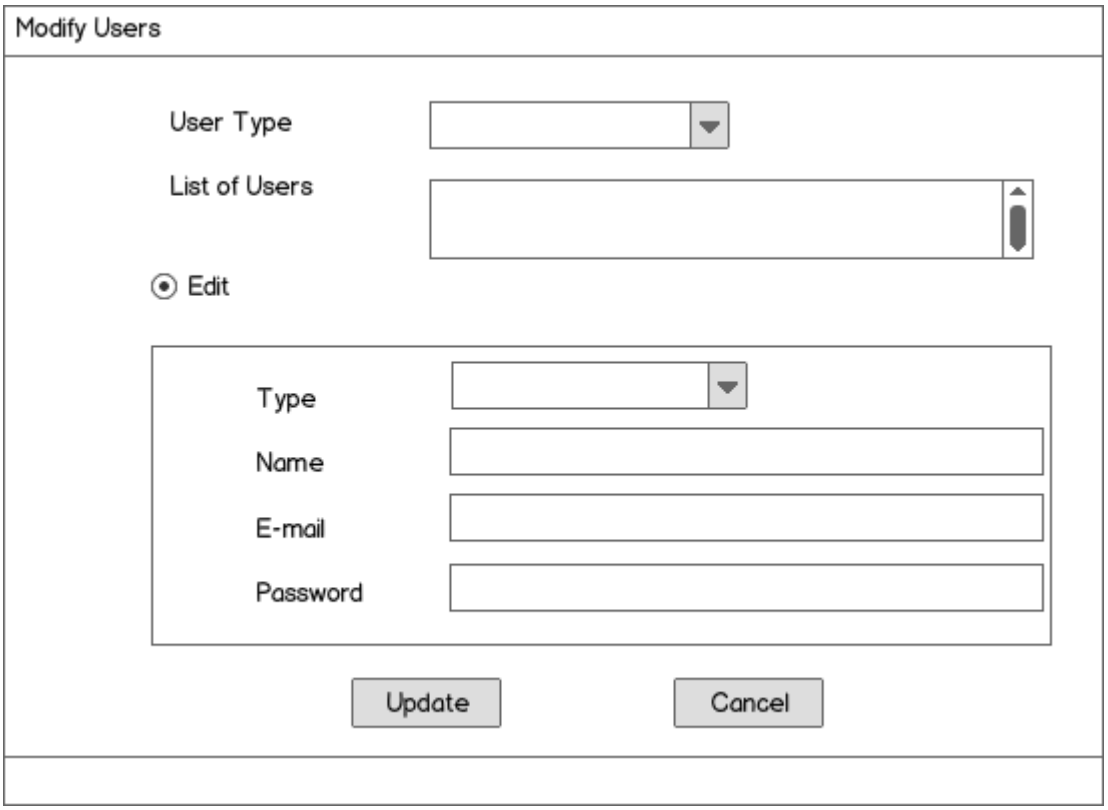
3.7.52 Use case 52 : Category freeze



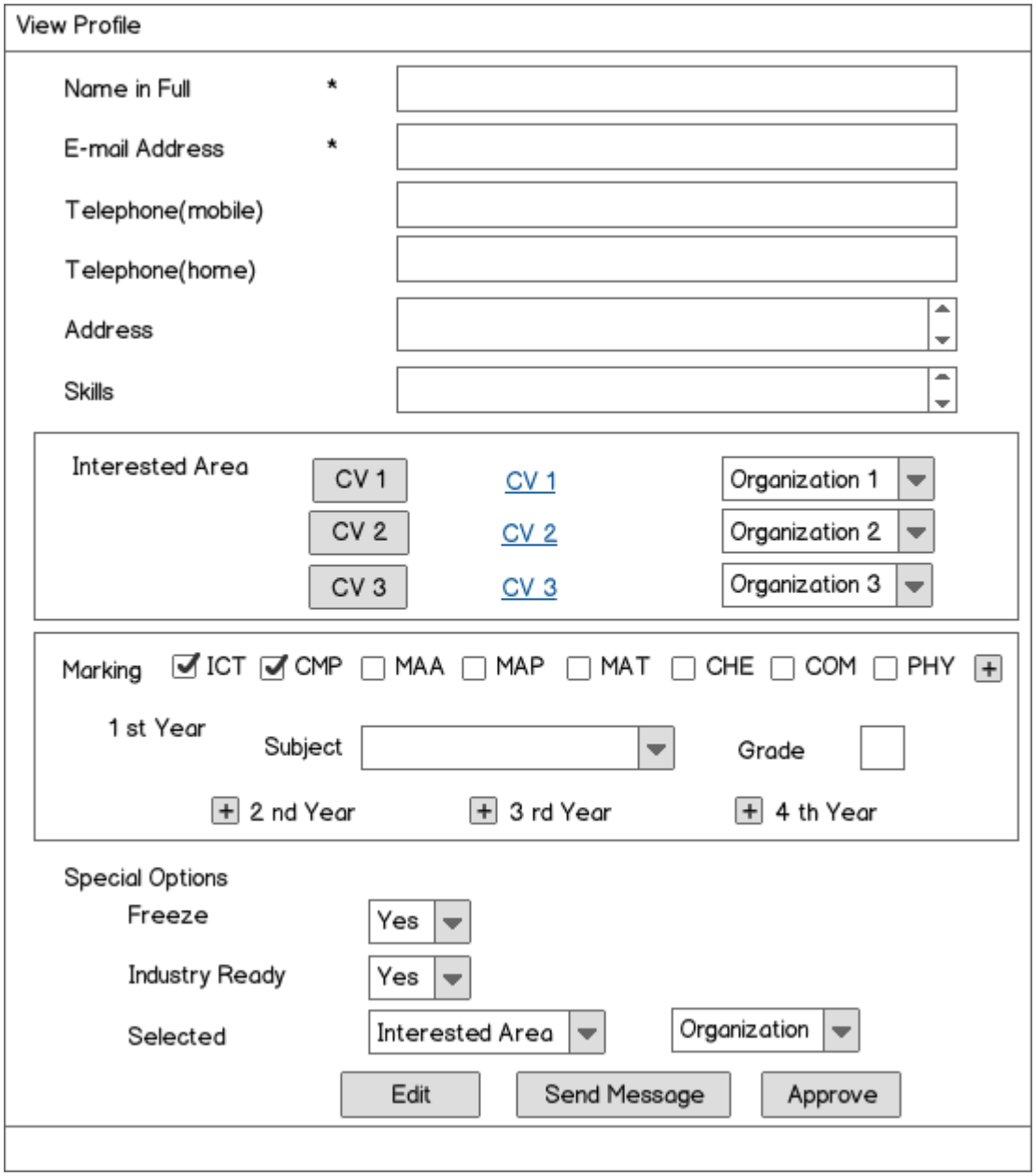
3.7.54 Use case 54 : Create users



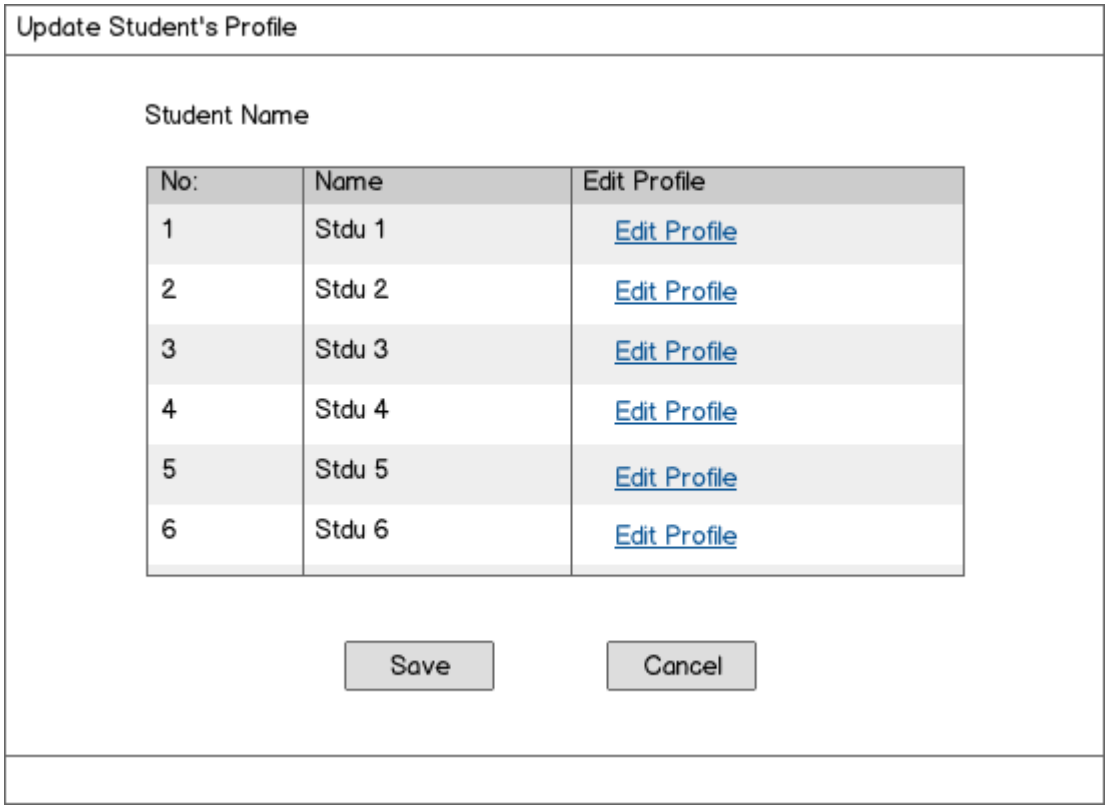
3.7.55 Use case 55 : Modify users



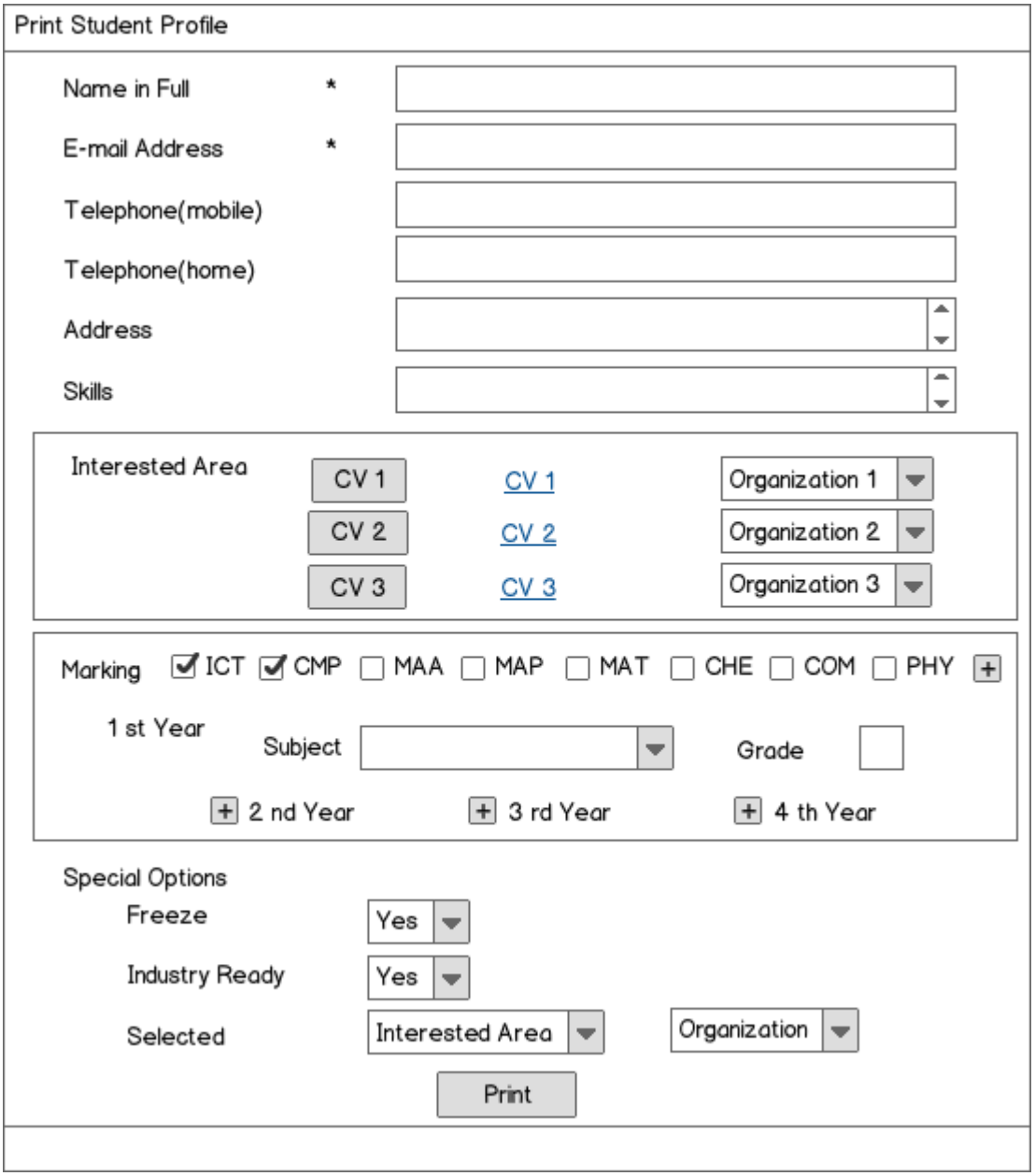
3.7.56 Use case 56 : View profile



3.7.57 Use case 57 : Update student's profile



3.7.59 Use case 59 : Print Student Profile



3.7.60 Use case 60 : Add Course