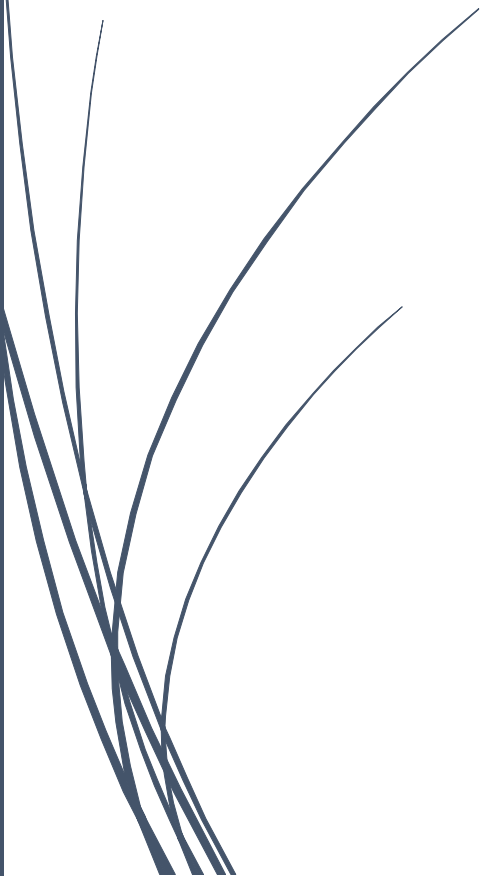
Student Registration System

Software Design Specification



Rajarata University of Sri Lanka
Faculty of Applied Sciences
INFORMATION AND COMMUNICATION TECHNOLOGY

Table of Content

1. Introduction

1.1 Purpose

This software design specification is made with the purpose of outlining the software architecture and design of the Student Registration System in detail. The document will provide developers an insight in meeting client's needs efficiently and effectively. Moreover the document facilitates communication and understanding of the system by providing several views of the system design.

1.2 Scope

The Software design document would demonstrate how the design will accomplish the functional and non- functional requirements captured in the Software Requirement specification (SRS). The document will provide a framework to the programmers through describing the high level components and architecture, sub systems, interfaces, database design and algorithm design. This is achieved through the use of architectural patterns, design patterns, sequence diagrams, class diagrams, relational models and user interfaces.

1.3 Intended audience

This document is mainly for the developers and the technical and academic staff of Rajarata university of Sri Lanka and student representatives of the University.

1.4 Glossary

Term	Description
Algorithm Design	Specific method to create a mathematical process in solving problems
Architectural Design	Establishing the overall structure of software system
AS	Applied Science
Compatible	Capable of orderly efficient integration and operation with other elements in a System with no modification
Database	A collection of stored related data
Encapsulate	To express or show the most important facts about something.
ER diagram	Entity Relationship Diagram; Data model for describing a data base in an abstract way
HPT	Health Promotion Technology
ICT	Information & Communication Technology
Instantiate	To represent (an abstract concept) by a concrete or tangible
SDS	System Design Specification
Sequence Diagram	An interaction diagram that shows how process interact with one another and in what order
SRS	Software Requirements Specification

1.5 References

- *Software Requirements Specification (SRS) of the Student Registration System*
- *Golden rules of user interface design by Theo Mandel*
- *Internet*

1.6 Document Overview

The next chapter of the document has described architectural design of the Student Registration System. The high level components and their interactions, suitable architectural patterns, physical arrangement of components and design decisions applied to the whole system.

The 3rd and final chapter of the System Design Specification is on Component and detailed design. Includes design patterns, sequence diagrams, class diagrams, database design in detail and user interface design with screen shots of the interfaces.

2. Architectural Design

2.1 High level components and interfaces

2.1.1 Components

- Student component

This is one of the key components of the Student Registration System. This is where the student subject selection option is implemented. This also includes the result preview, semester preview and profile functions.

- Authentication and user management component

This is the major sub system that is responsible for the security of the Student Registration system. It authenticate users and also handles the user management activities such as creating new user accounts, removing accounts from the system etc. Furthermore this component Implements the "control access privilege matrix".

- Subject component

This is the key component that implements the functions related to the subject operations of administrator such as adding a new subject, editing credits of an existing subject and removing subjects etc. Subject component is also responsible for displaying the available subject list for every semester.

- Publish component

Publish component is the component responsible for publishing notices created by the administrator and also the time tables. This component has the ability to publish multiple notices and time tables at the same time.

- Public component

This is a relatively small subsystem compared to the other components of the Student Registration System. This is the component which is responsible for the guest viewer (public) to view student results.

2.1.2 Interfaces

Student component

studentSelectSubjects: This interface will provide the available subject list for the semester. This allows the student to choose the subjects, hence a busy interface. This interface is the bridge between the student component and the subject component.

studentProfile: This interface is used for the student to view and edit personnel information.

Authentication and user management component

authenticateUser: This is the interface that allows the users to login to the system. This will guide the user to the relevant home page.

Subject component

newSubject: This interface is where administrator adds new subjects to the courses offered. The Subject and student components are connected.

editSubject: In this interface the administrator edit existing subjects. The Subject and student components are connected.

Publish component

Getnotice: This is the interface where the notifications are published. It is connected with the subject component

Gettimetable: This is the interface where the time table is published. This is connected with the subject component.

Public component

viewResults: This is the interface which shows the results of a student to a Guest user, therefore connected with the Student component.

2.2 Architectural Styles / Patterns

The Student Registration System will be developed under two main architectural styles/ patterns. Development of the project will be done in MVC architectural style and also 3 tier Client/Server Architecture. Client can browse the internet and access the Student Registration System provided within the local area network of the University.

2.2.1 MVC Architecture Style (Model – View – Controller)

MVC Style separates presentation and interaction from the system data. The system is structured into three logical components that interact with each other.

- The Model component -Manages the system data and associated operations on that data.
- The View component- Defines and manages how the data is presented to the user.
- The Controller component- Manages user interaction and passes these interactions to the View and the Model. .

We will use this MVC Style for the Student Registration System because, there are multiple ways to view and interact with data. Also used when the future requirements for interaction and presentation of data are unknown. In some software systems the code between the process logic and interface are mixed. This will reduce the modularity of application and make the system more difficult to maintain. To avoid this problem we have decided to use MVC architectural style to separate the application logic with the interface. The main advantage of this is style allows the data to change independently of its representation and vice versa. Support presentation of the same data in different ways with changes made in one representation shown all of them.

2.2.2 Three-Tier Client Server Architecture

In a client server architecture, the functionality of the system is organized into services, with each service delivered from separate server. Clients are users of these services and access servers to make use of them.
We will use this 3- Tier Client Server Architecture because, when data in a shared database has to be accessed from a range of locations. Because servers can be replicated, may also be used when the load on a system is a variable.

- Data Tire

The data tire maintains the applications data such as Users' data , Departments' data , subjects' data , courses' data , time tables' data and the SQL queries . It stores these data in a relational database management system (RDBMS). All the connections with the RDBMS are managed in this tier.

- Middle Tire

The middle tier (web / application server) implements the business logic , controller logic and presentation logic to control the interaction between the applications' clients and data. Business rules enforced by the business logic dictate how clients and cannot access application data and how applications process data.

- Client Tire

The client tire is the applications user interface connecting data entry forms and client side applications. It displays data to the user. User interact directly with the application through user inter-face. The client tier interacts with the web/ application server to make requests and to retrieve data from the database. It then displays to the user the data retrieved from the server.

Example of the 3-tier architecture in the Student Registration System

If a student needs to view his current semester subject combination, first he has to login to the system. Then he has to click "User" option, after he has to login "Summary" option. Then system will display the information. In this process, Login screen, users' main screen and Subject combination summary screen are defined into the Client tier, data for login information and profile information and SQL queries for those information are maintained into the Data tier and controller logic for login process and loading profile information from the database are defined in Middle tier.

2.2.3 Different process and their communication

In the Student Registration System, there are number of different processes, such as database server process, web server process, connections between above servers likewise. When sending mails there should run a mail server. HTTP protocol is using to communicate with web servers, SMTP protocol is using to communicate with mail servers. They should communicate each other well to perform the functions of whole application.

2.3 Physical arrangement of devices and Communication and installed software Components on devices.

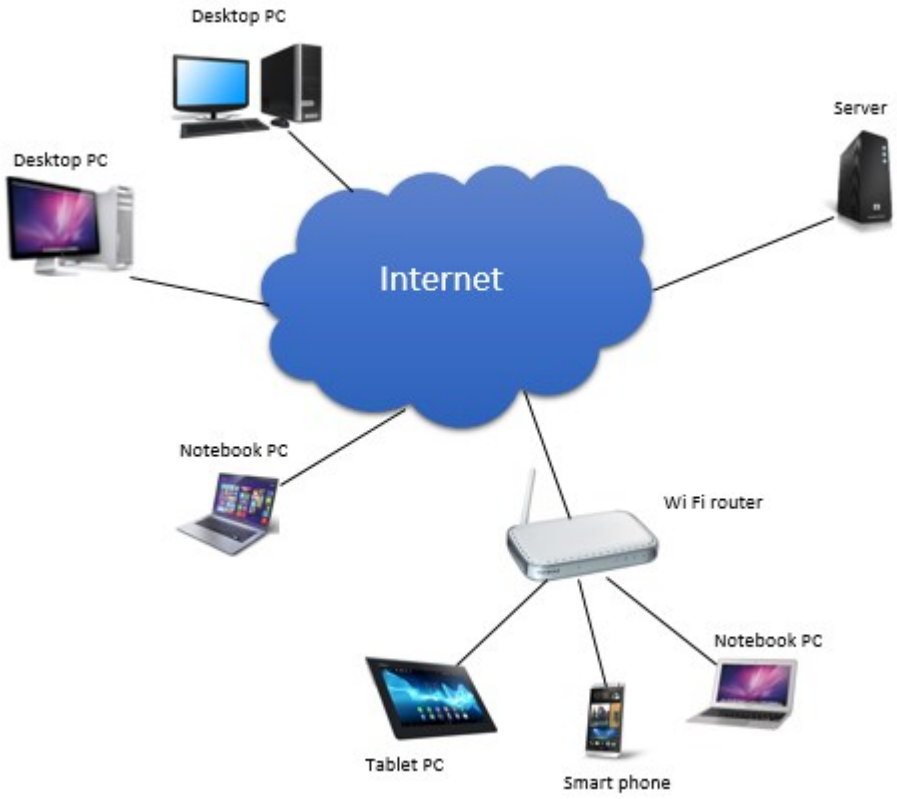
The Student Registration System deals mainly with hardware devices and installed software components on devices. The System performs many tasks. It consists of web based system used by Super Users, Administrators and Students of the university. The system helps to record students' personal details, publish time table, preview student result, select subjects for the semester. Therefore the web based part is expected to run on various operating system platforms. The client server architecture of the system requires to remotely connecting with client and server through the internet connection. The system has two nodes such as the Web server and Clients. The nodes can represent specific instances (workstations) or a class of computers (web server), which is a virtual machine. The applications of the system will run on the web server connected to the database server.

Internet is the worldwide interconnection of all smart communication devices that have a valid IP. There should be installed browser software to access internet. If the user accesses the system, directly through the internet connection the user has to install dongle or modem or relevant device and Wi-Fi or etc… to connect with the system. If the user accesses the system through the intranet connection, the server should install the relevant software. Most of intranet accessing modes refer to the website of the organization which can only be accessed by its employees who have a user name and password. So, considering that type of security purposes, it is better to access this Student Registration through the intranet, but it should be accessed through the internet directly also.

When generating reports such as students' result and subjects details there should be installed the crystal report software. And also to print the generated reports the user machine should be installed the software relevant to its connected printer.

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.



Arrangement of devices and serves

Our University Student Management System needs some specific set of servers and devices. Such as:

Server to host web applications and web service applications.
Database server to store and manage data.
Personal computer, note book, smart phone etc… to access the website.
Modem/ router/ switch/ hub/ Wi-Fi network/ cable network etc… and also need an Internet Service Provider to have the internet connectivity.

Communication among components

Above devices are communicating with each other. Personal computer communicates with web server and the database server through HTTP protocol. It communicates with mail server through SMTP protocol. Cable network or Wi-Fi network is also a communication method using in connecting different network components.

2.4 Design decisions applied whole application

2.4.1 Object oriented software development methods

Reasons:

- Improved software maintainability.
- Faster development
- Lower cost development
- Improved software development productivity
- Higher quality software

2.4.2 Three-Tier Client Server Architecture

Reasons:

- As more users access the system a three-tier solution is more scalable than the other solutions because you can add as many middle tiers as needed to ensure good performance.
- Security is also the best in the three-tier architecture because the middle layer protects the database tier.

2.4.3 MVC Architectural Pattern

Reasons:

- It should interact with other machines or users effectively.
- For more efficient interaction system should have flexible interfaces.
- 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 & intermediate logics.
- Ease to coding and provide well defined interfaces within each logic.

3. Component and Detailed Design

3.1 Design Patterns and Techniques used

3.1.1 Abstract factory Pattern

Abstract factory pattern is creational design pattern that provides a way to encapsulate a group of individual factories that have a common theme without specifying their concrete classes. Abstract factory pattern offers the interface for creating a family of related objects without explicitly specifying their classes.

Applications:
The design pattern will be applied in creating different user accounts which are the different factories. It will also be used to keep the system

3.1.2 Singleton Pattern

This is a creational design pattern and is one of the simplest patterns in the field of software engineering. It involves only one class which is responsible to instantiate itself, so that it creates no more than one instance. The singleton pattern is useful when access to limited resource needs to be controlled.

Applications:
In the student registration system this pattern will be used for database manager.

3.1.3 Observer Pattern

The observer pattern is a behavioral pattern which defines a one-to-many dependency between objects where a state change in one object results with all its dependents being notified and updated automatically. This pattern may be used in all situations where more than one display format for state information is required and where it is not necessary for the object that maintains the state information to know about the specific display formats used.

Applications:
Observer pattern will be used in the Student Registration system for the operations of the system users.

3.1.4 Adapter Pattern

The adapter pattern is a structural pattern that translates one interface for a class to a compatible interface. This will convert the interface of a class in to another interface that the user expects. Adapter gives the opportunity for the classes with incompatible interfaces to work together.

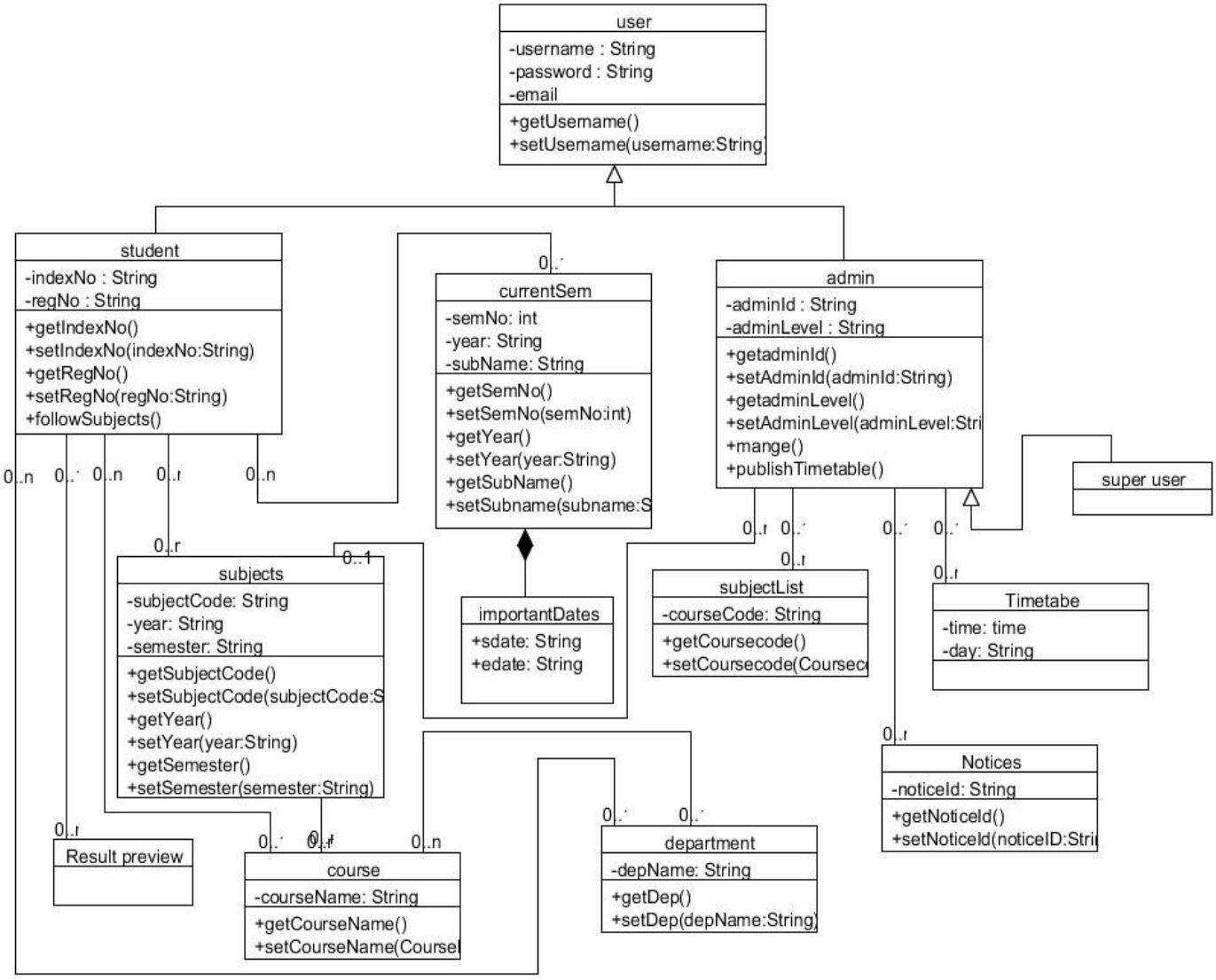
Application:
This pattern will be used in the Student Registration system when displaying information from the database.

3.1.5 Techniques Used

Prototyping

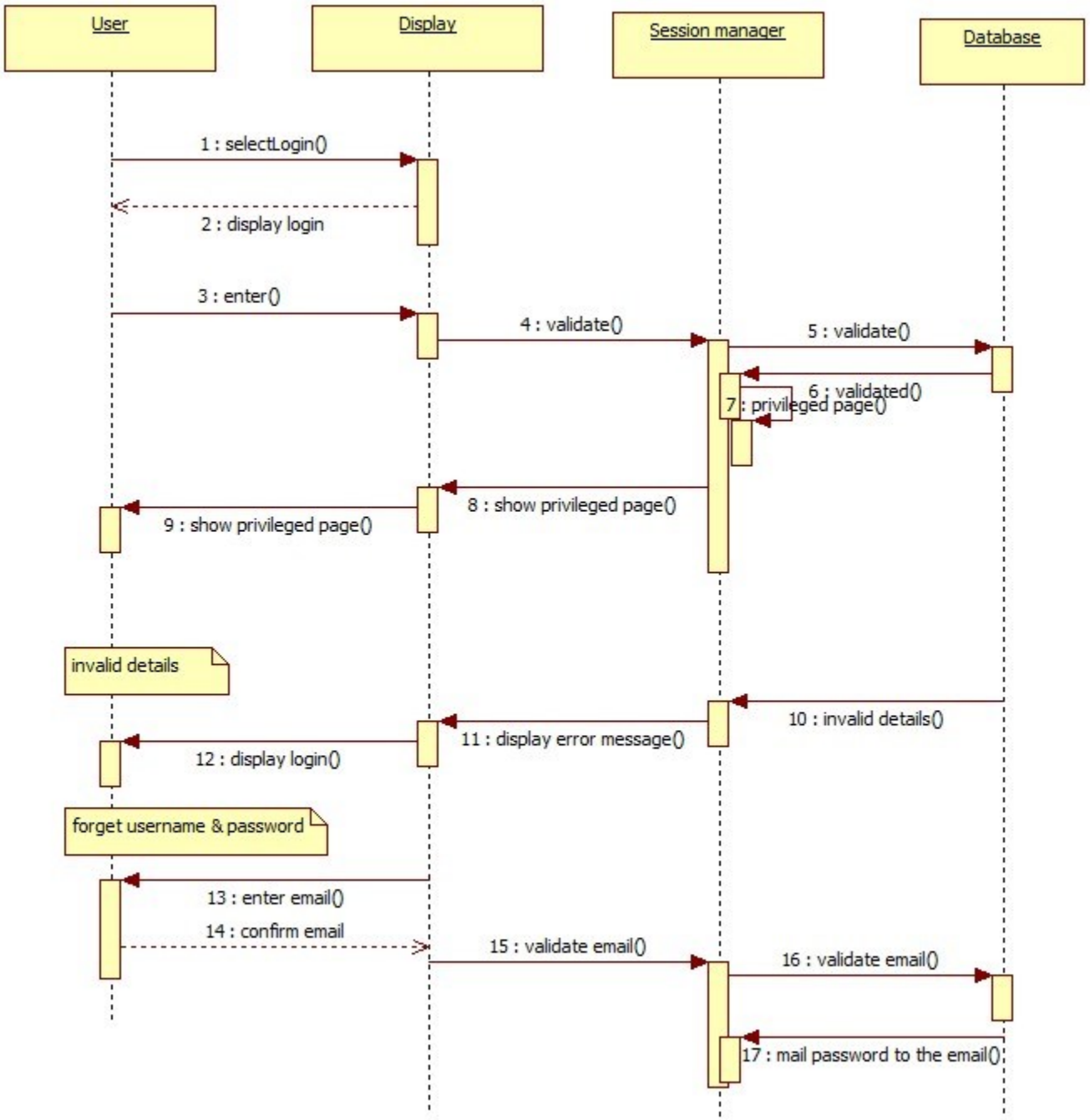
In designing the Student registration System prototyping will be used to demonstrate underpinning concepts of the designing and for user interfaces. This technique will provide the opportunity for the system users to experiment the software to a certain extent during the development process.

3.2 Class diagram

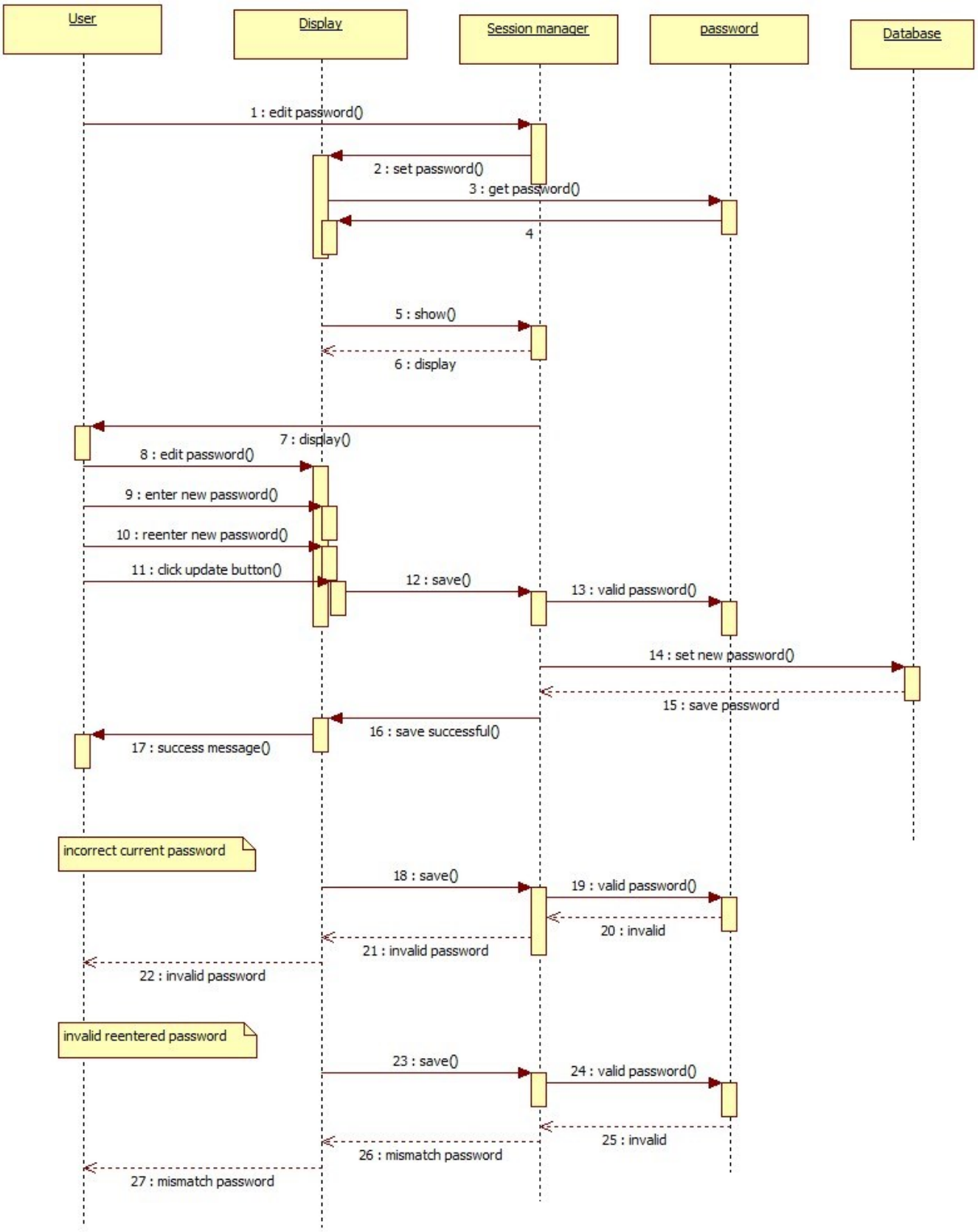


3.3 Sequence diagrams

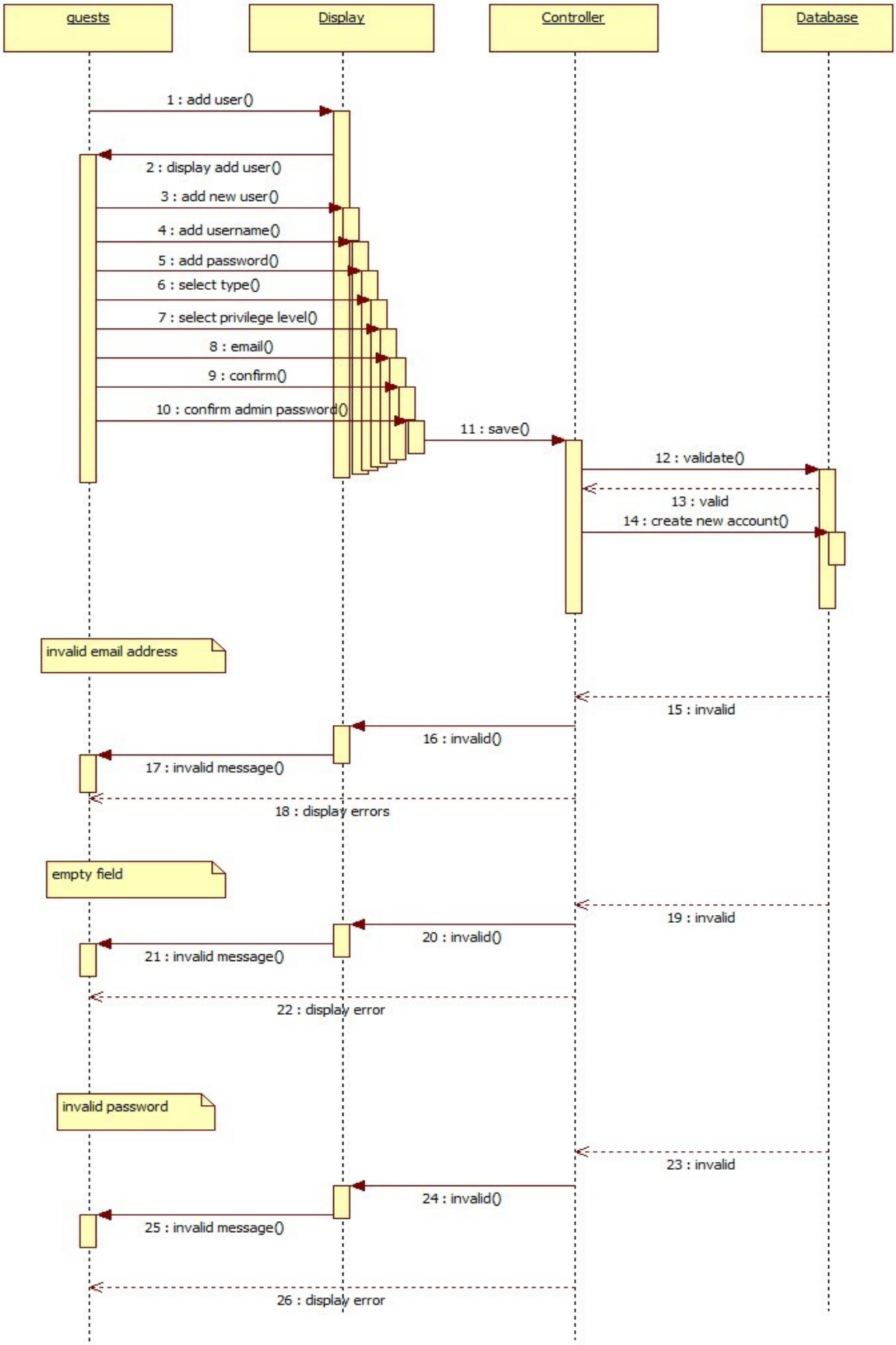
3.3.1 Sequence diagram 1: Login into the system



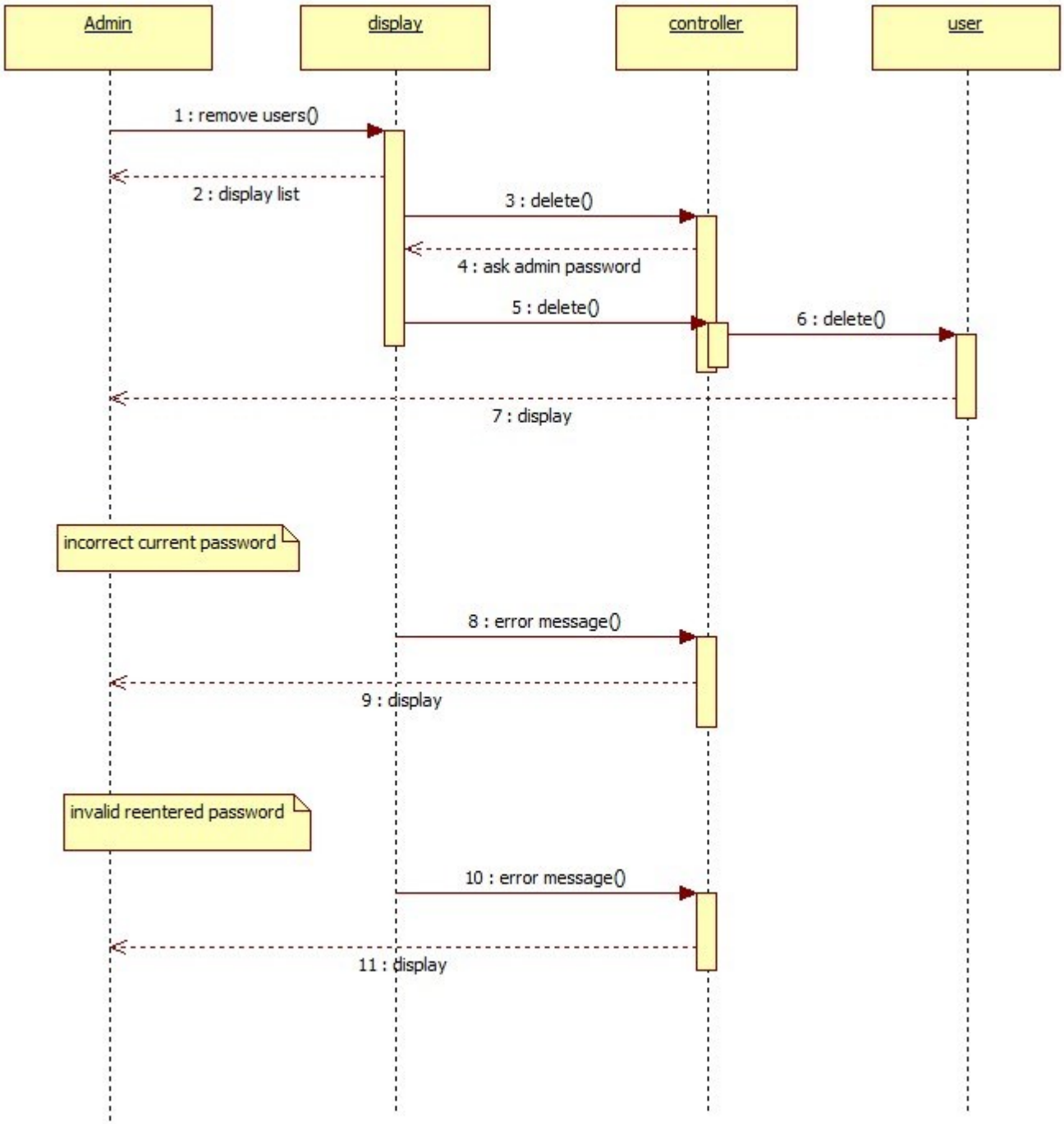
3.3.2 Sequence diagram 2 : Change user password



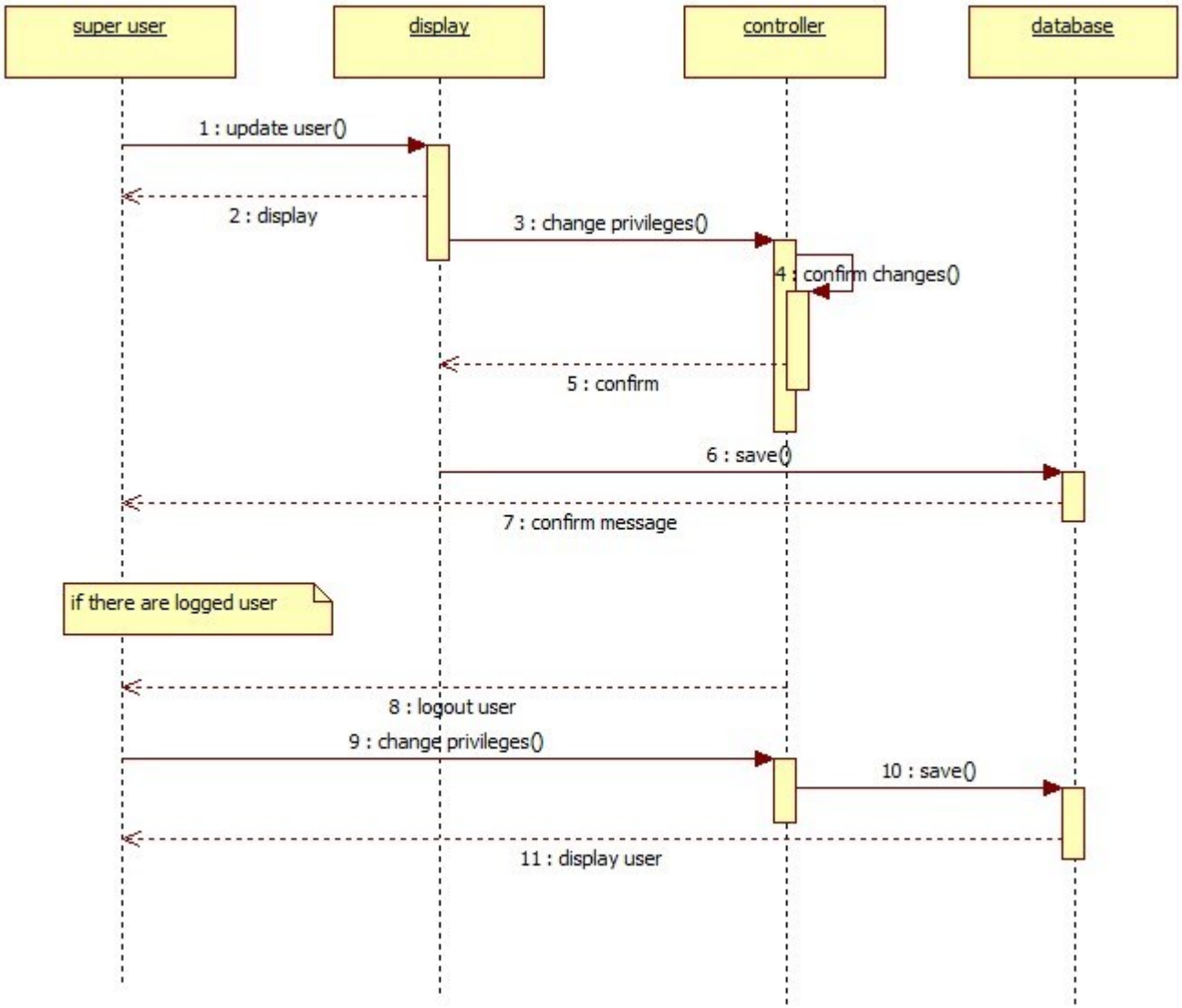
3.3.3 Sequence diagram 3: Add users



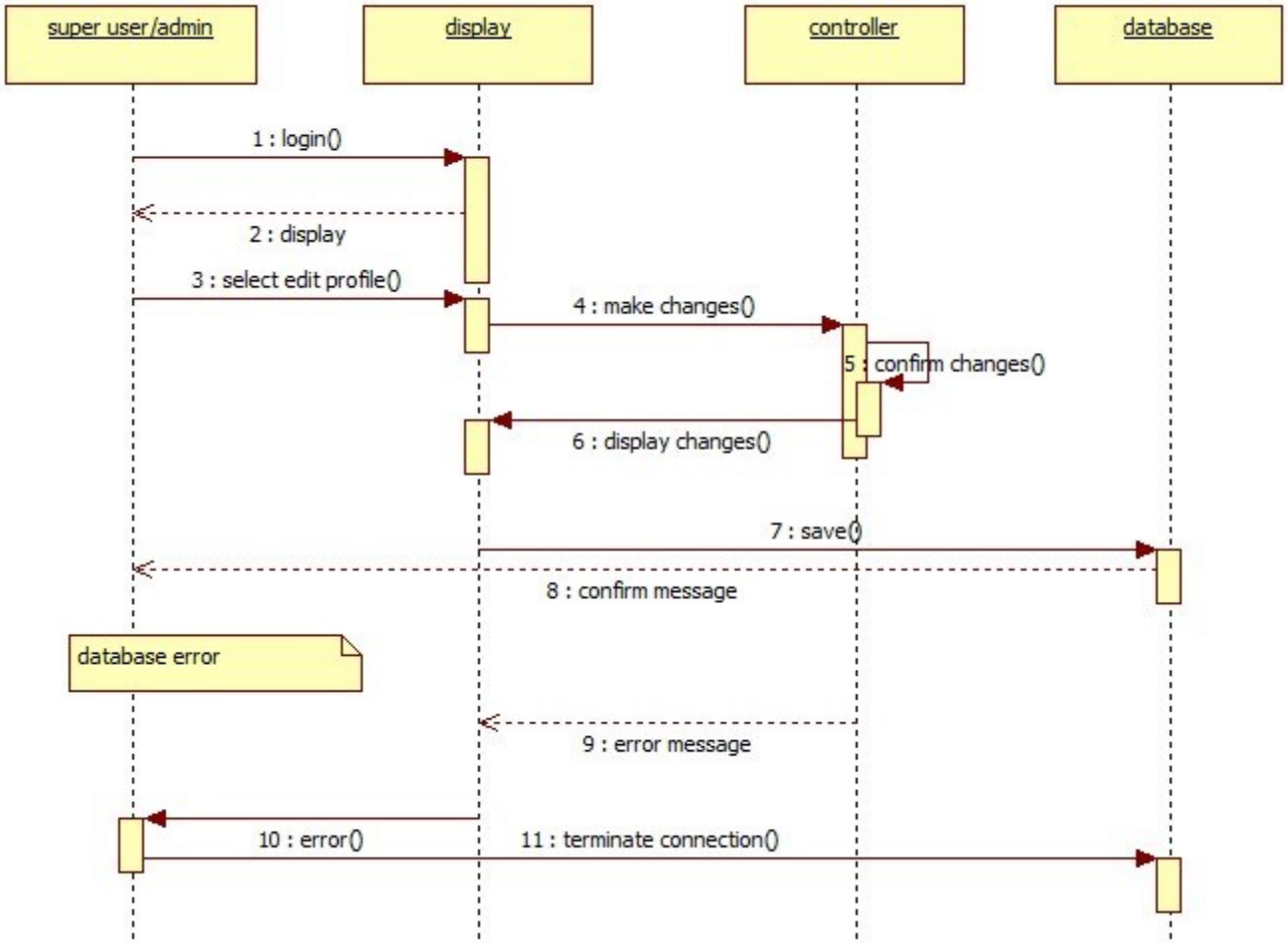
3.3.4 Sequence diagram 4: Remove users



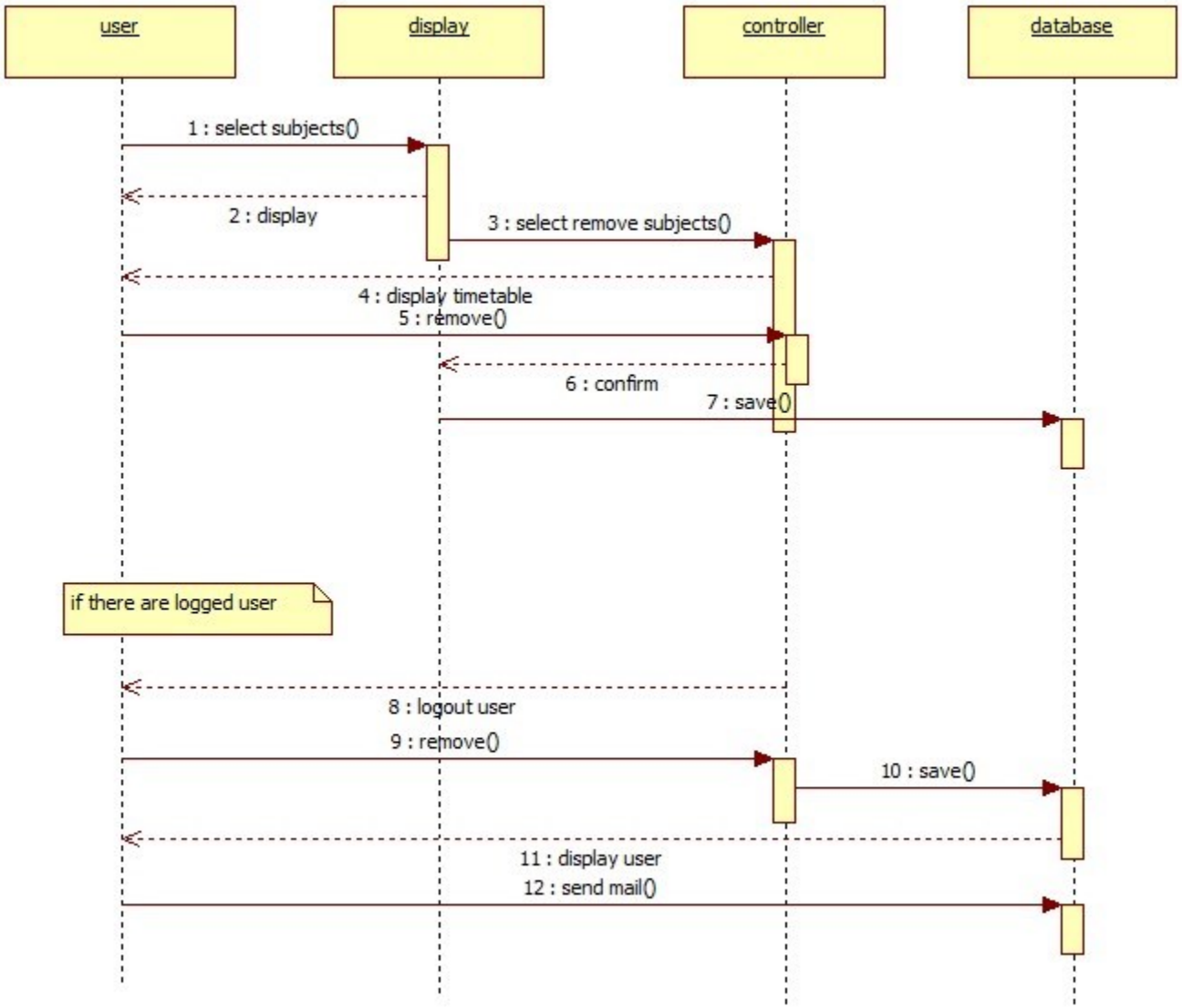
3.3.5 Sequence diagram 5: Manage users



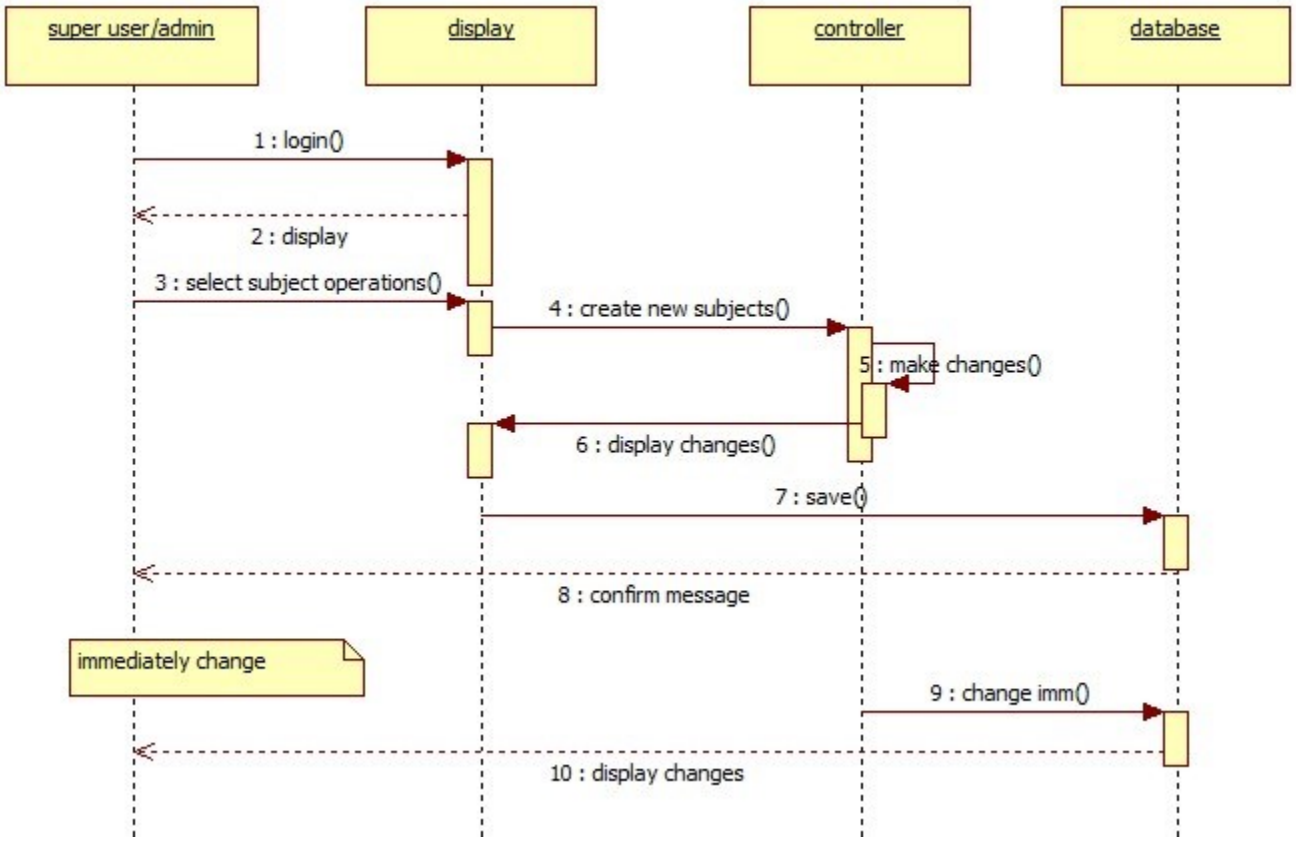
3.3.6 Sequence diagram 6: Edit profile information



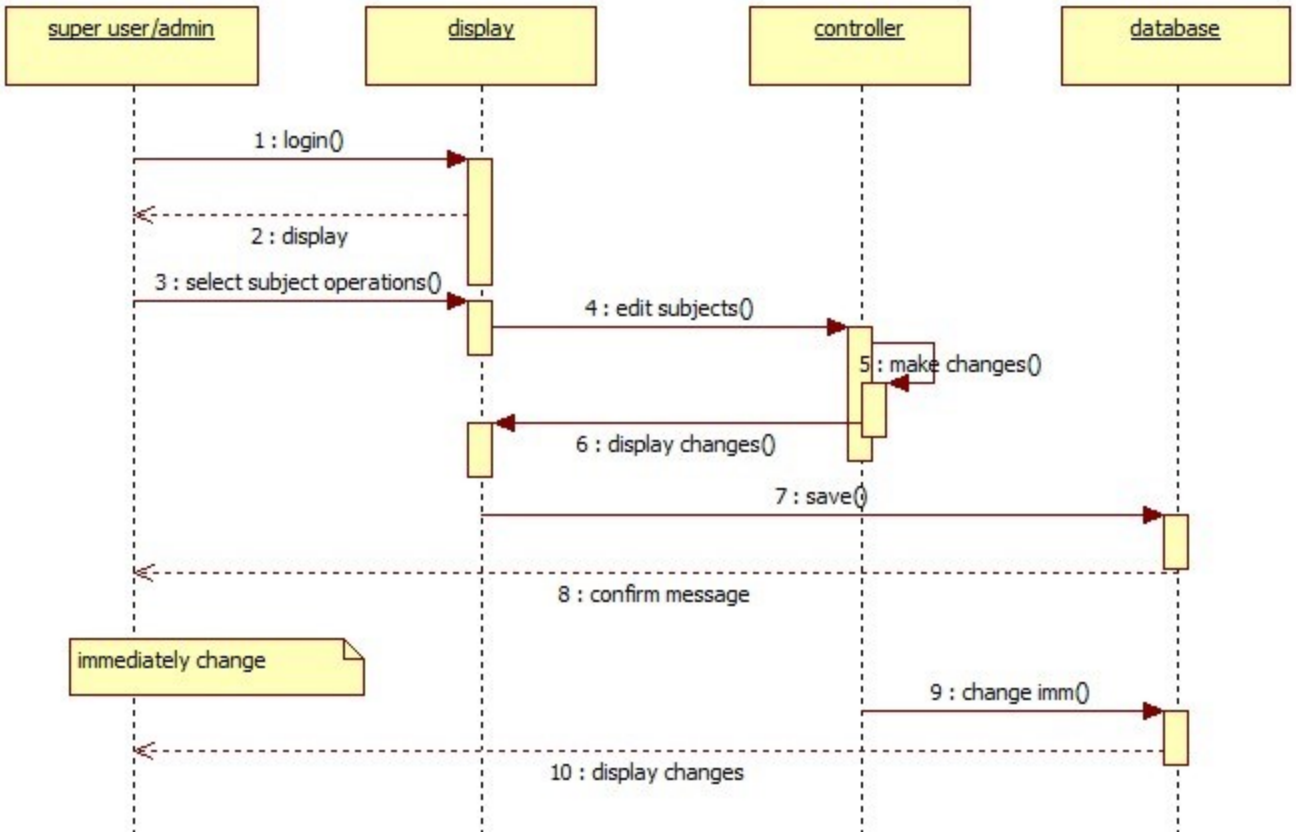
3.3.7 Sequence diagram 7: Remove selected subjects from student's selections



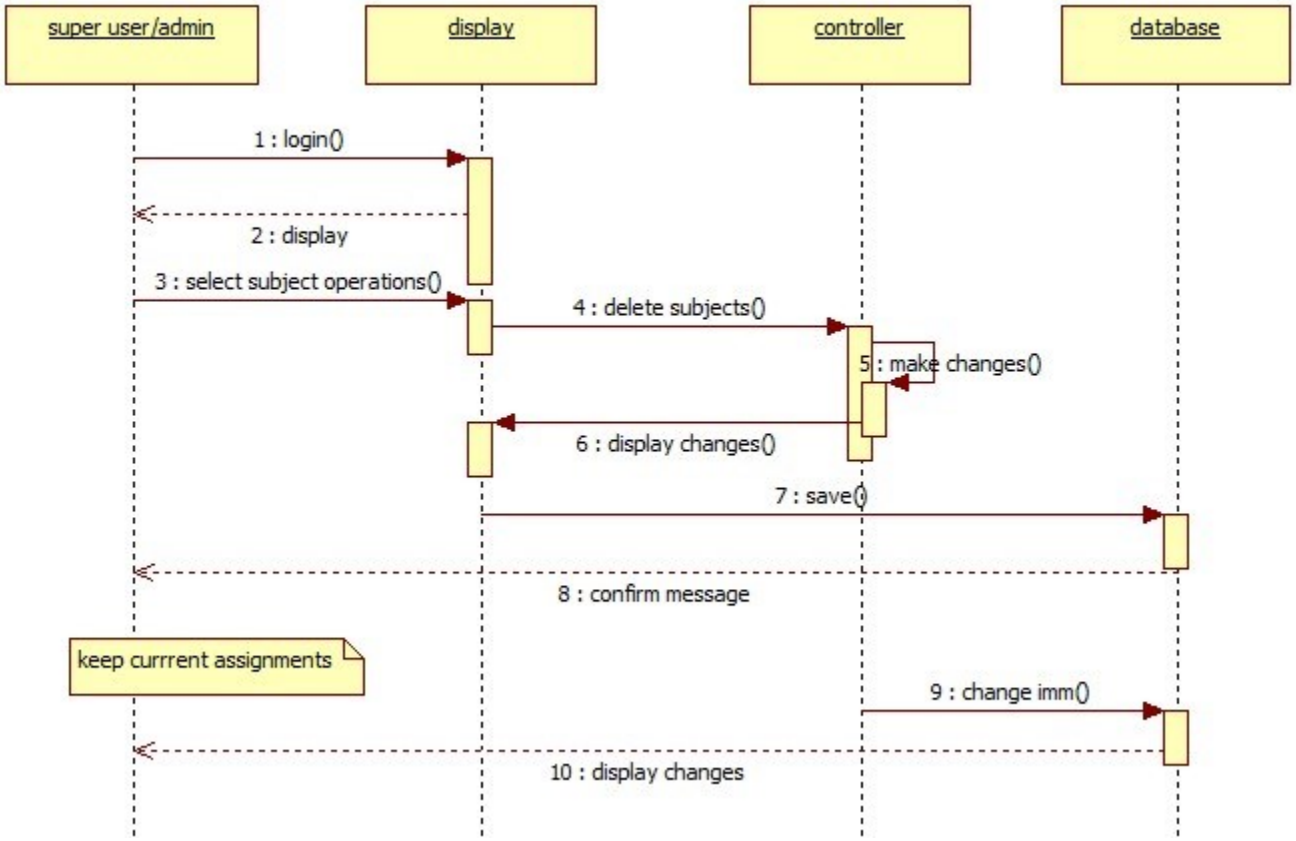
3.3.8 Sequence diagram 8: Create new subjects



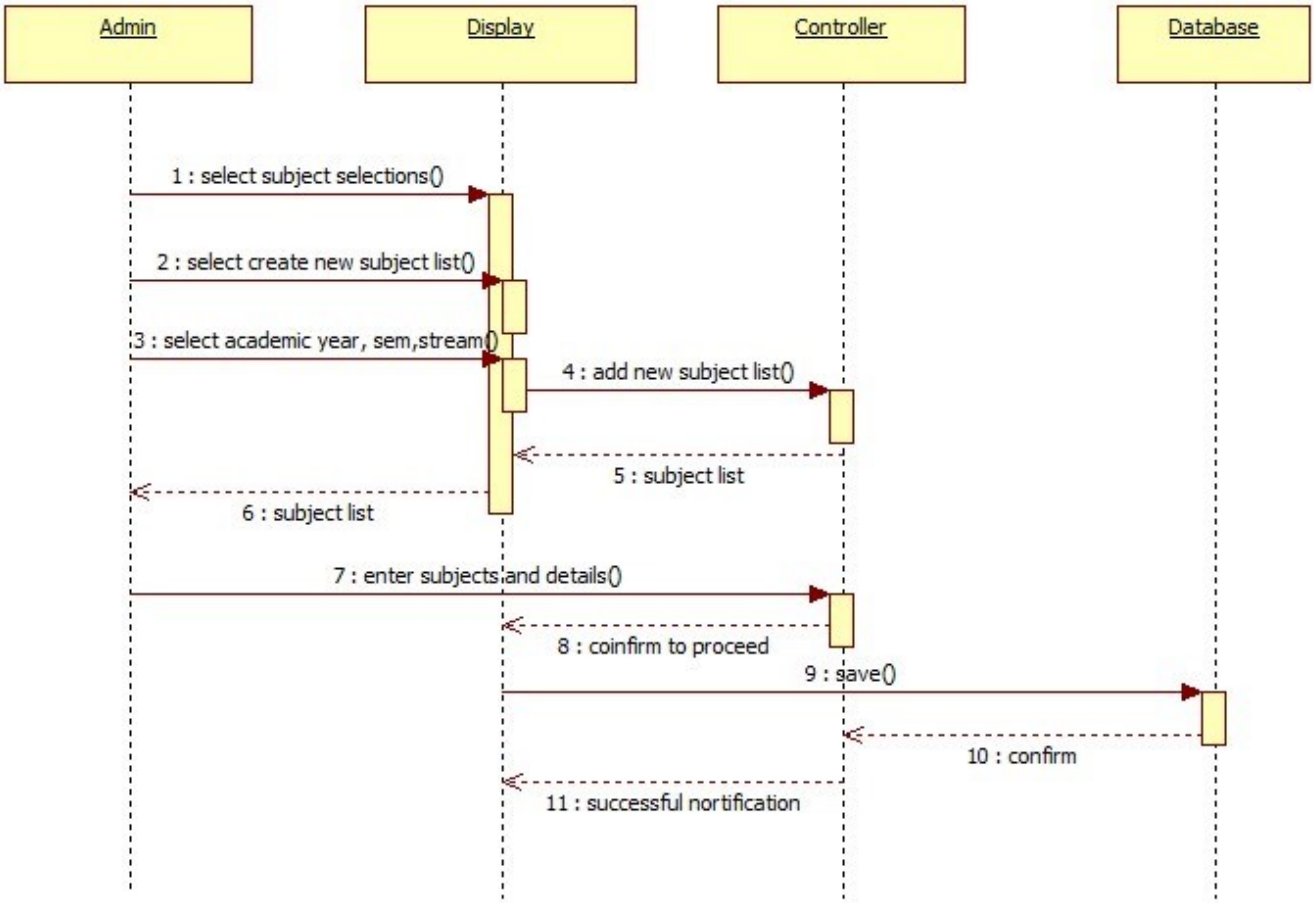
3.3.9 Sequence diagram 9: Edit existing subjects



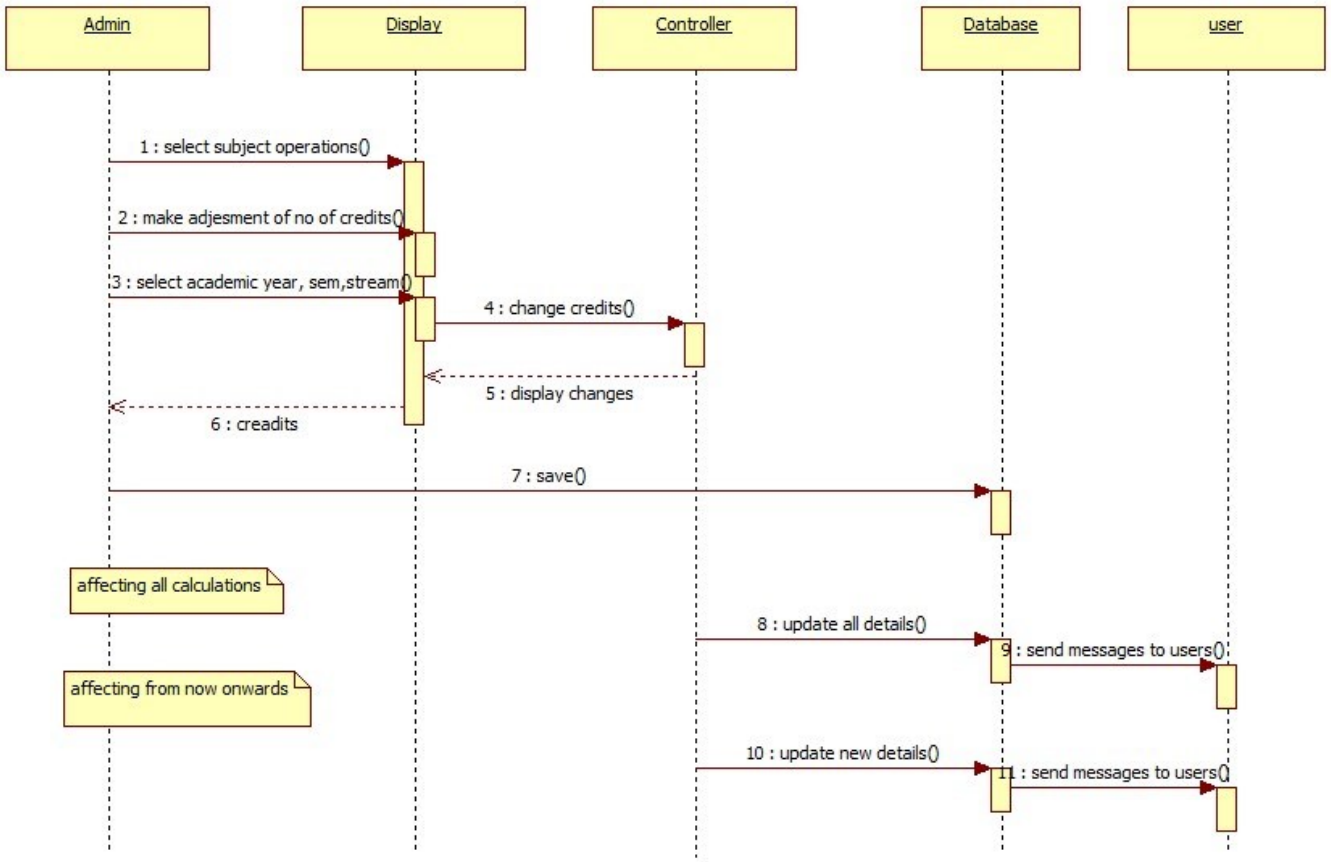
3.3.10 Sequence diagram 10: Delete subjects



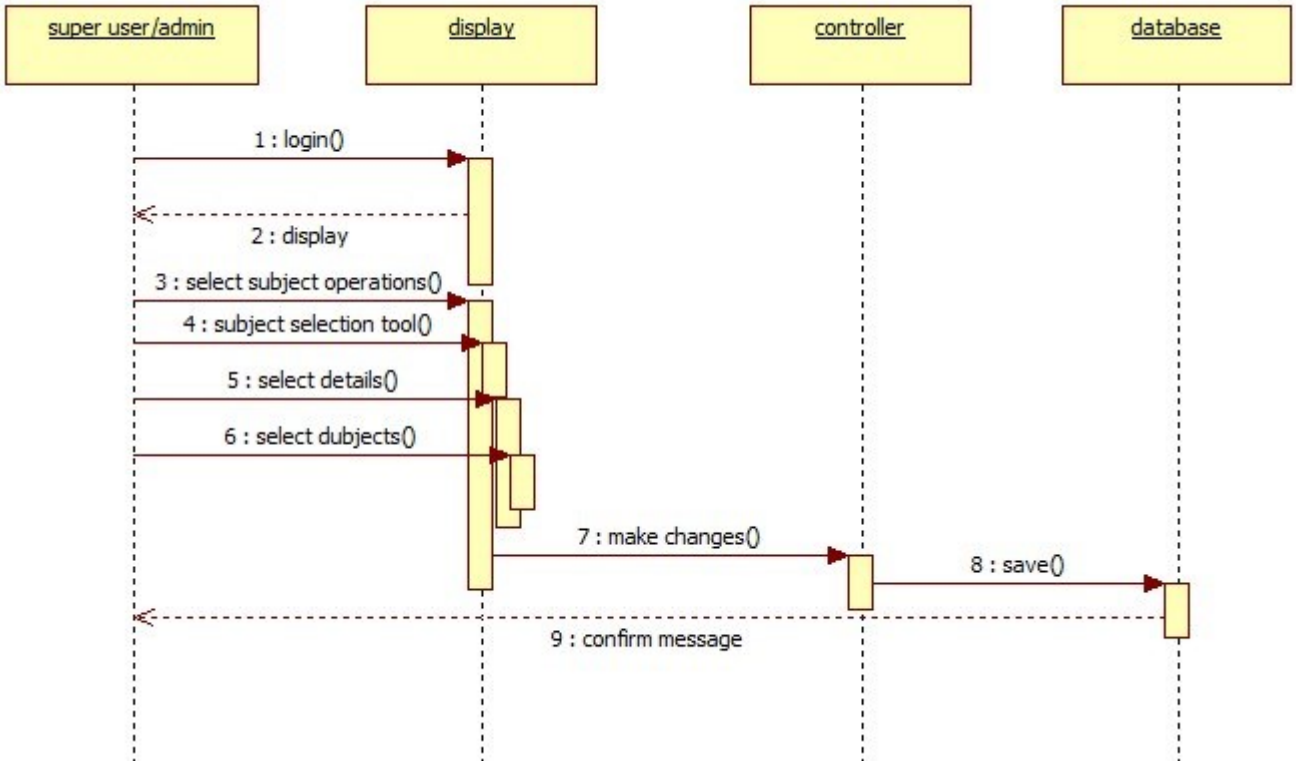
3.3.11 Sequence diagram 11: Create new subject lists



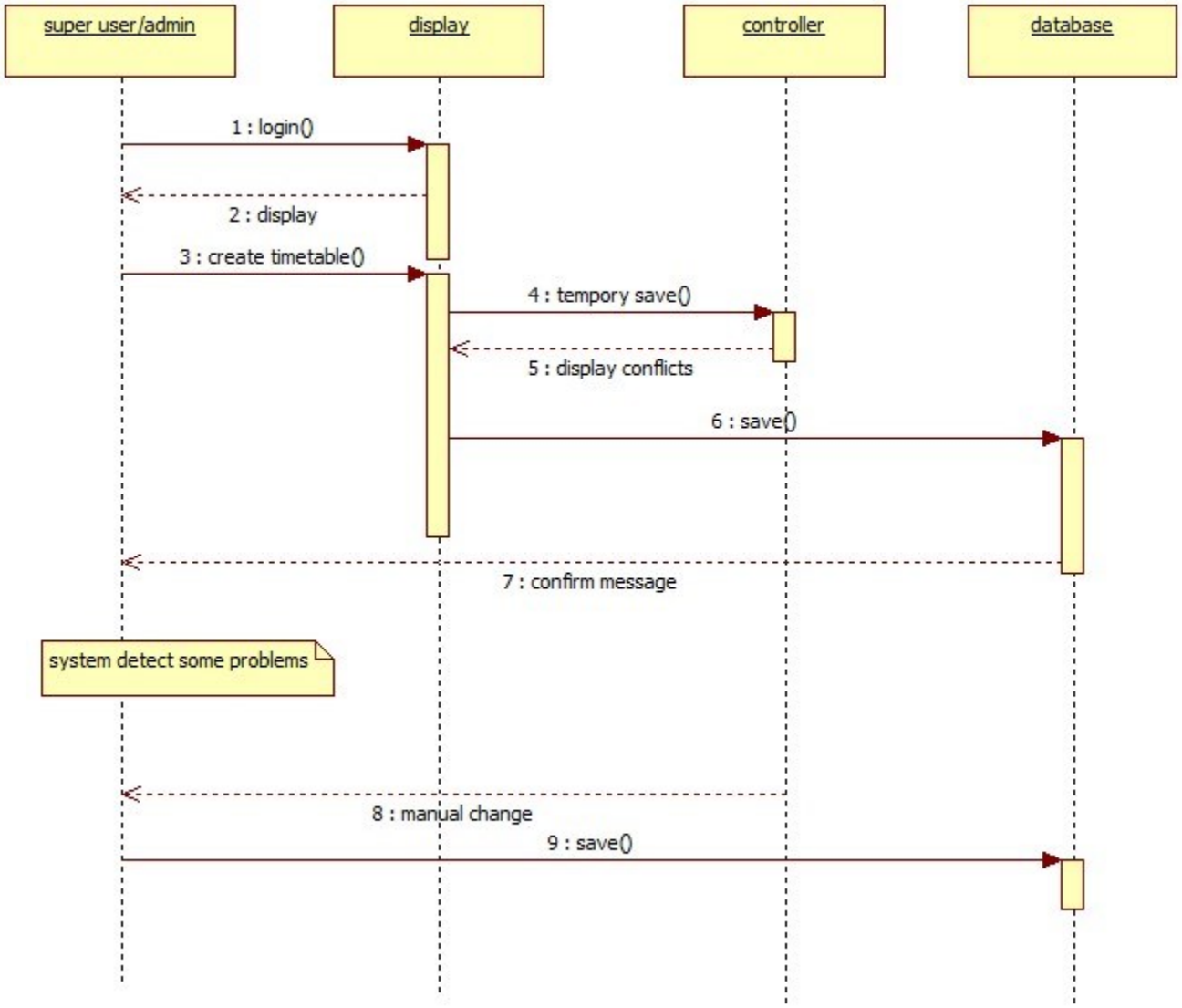
3.3.12 Sequence diagram 12: Make adjustment of credits per subject.



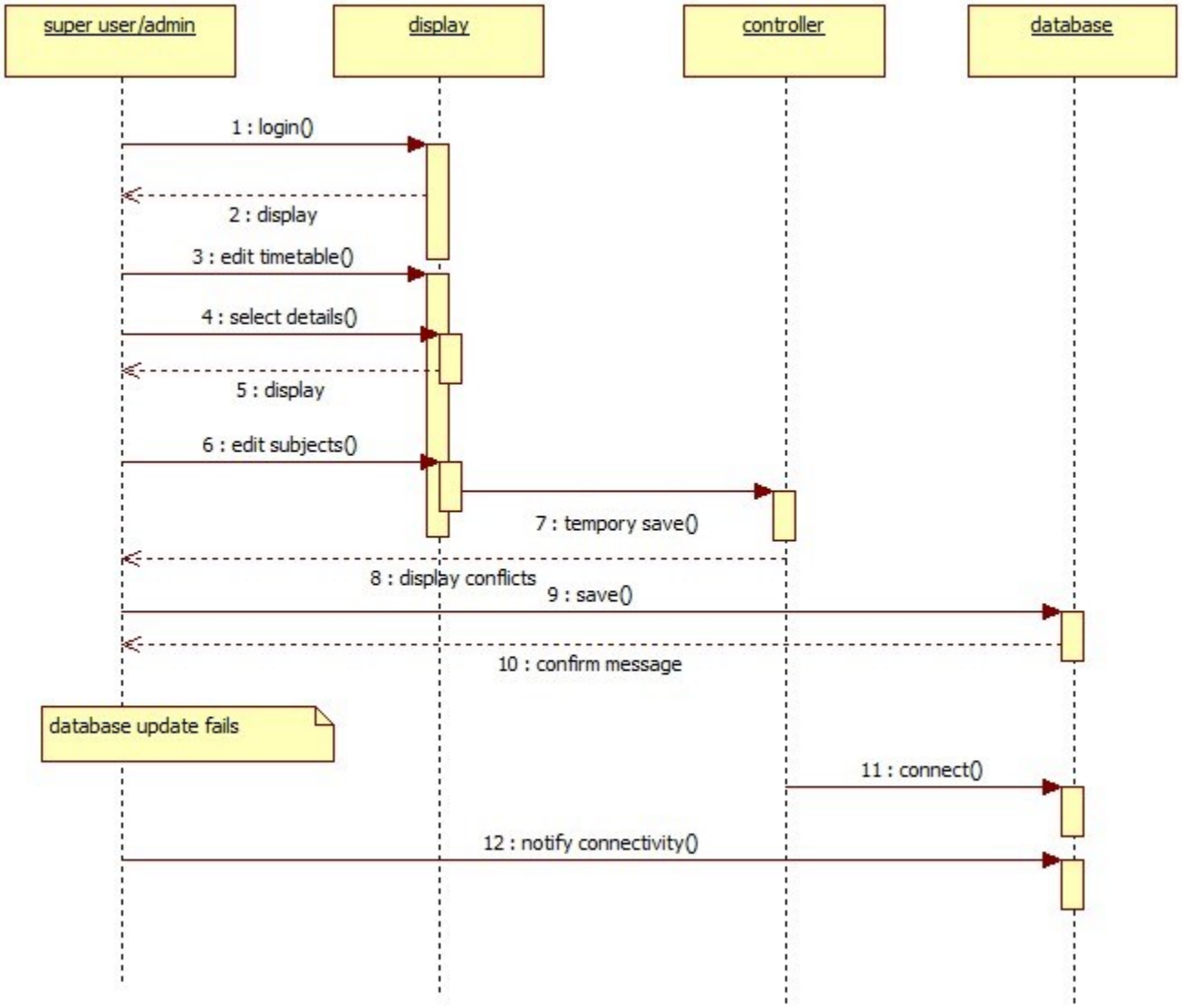
3.3.13 Sequence diagram 13: Subject combination tool



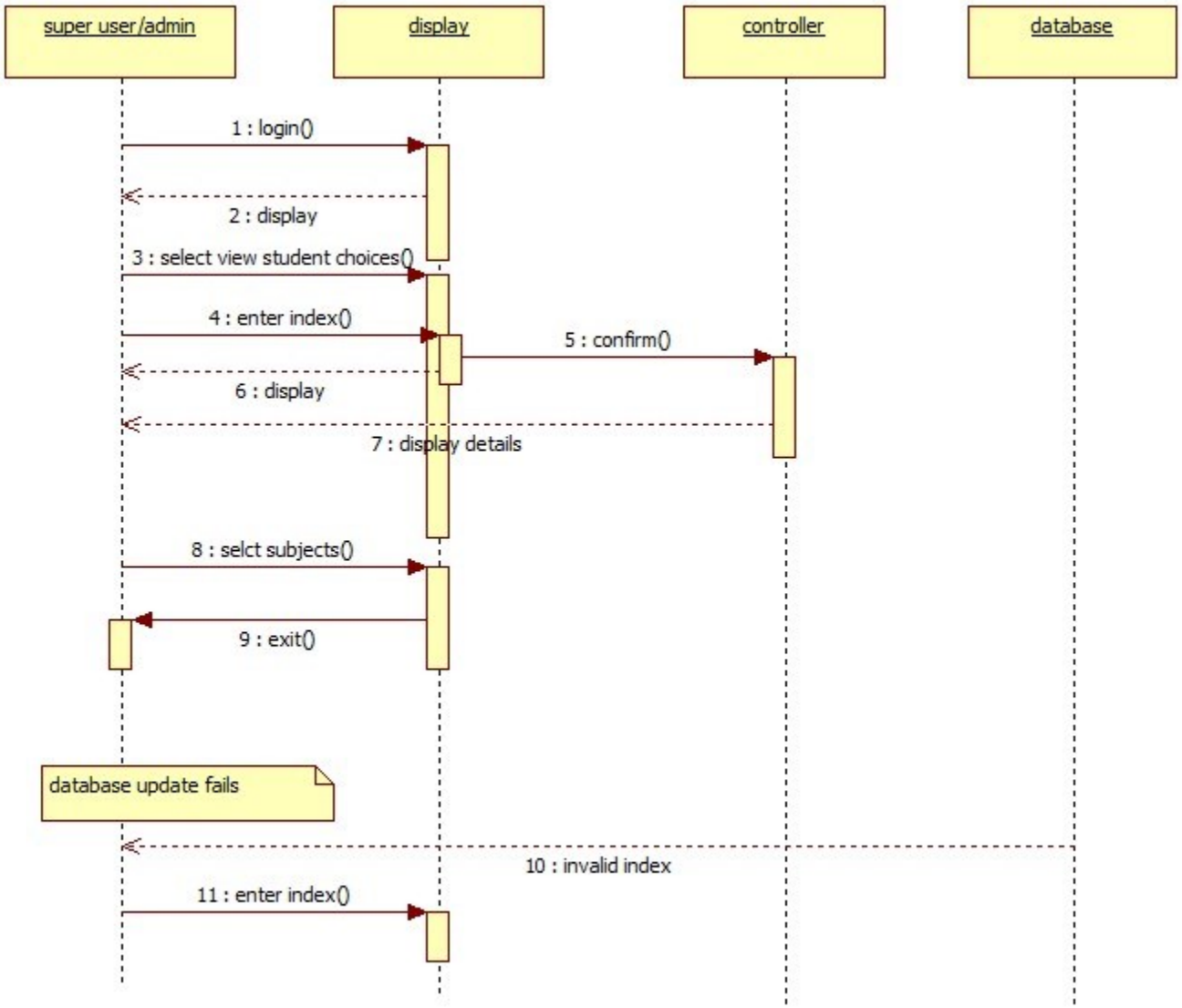
3.3.14 Sequence diagram 14: Publish time table



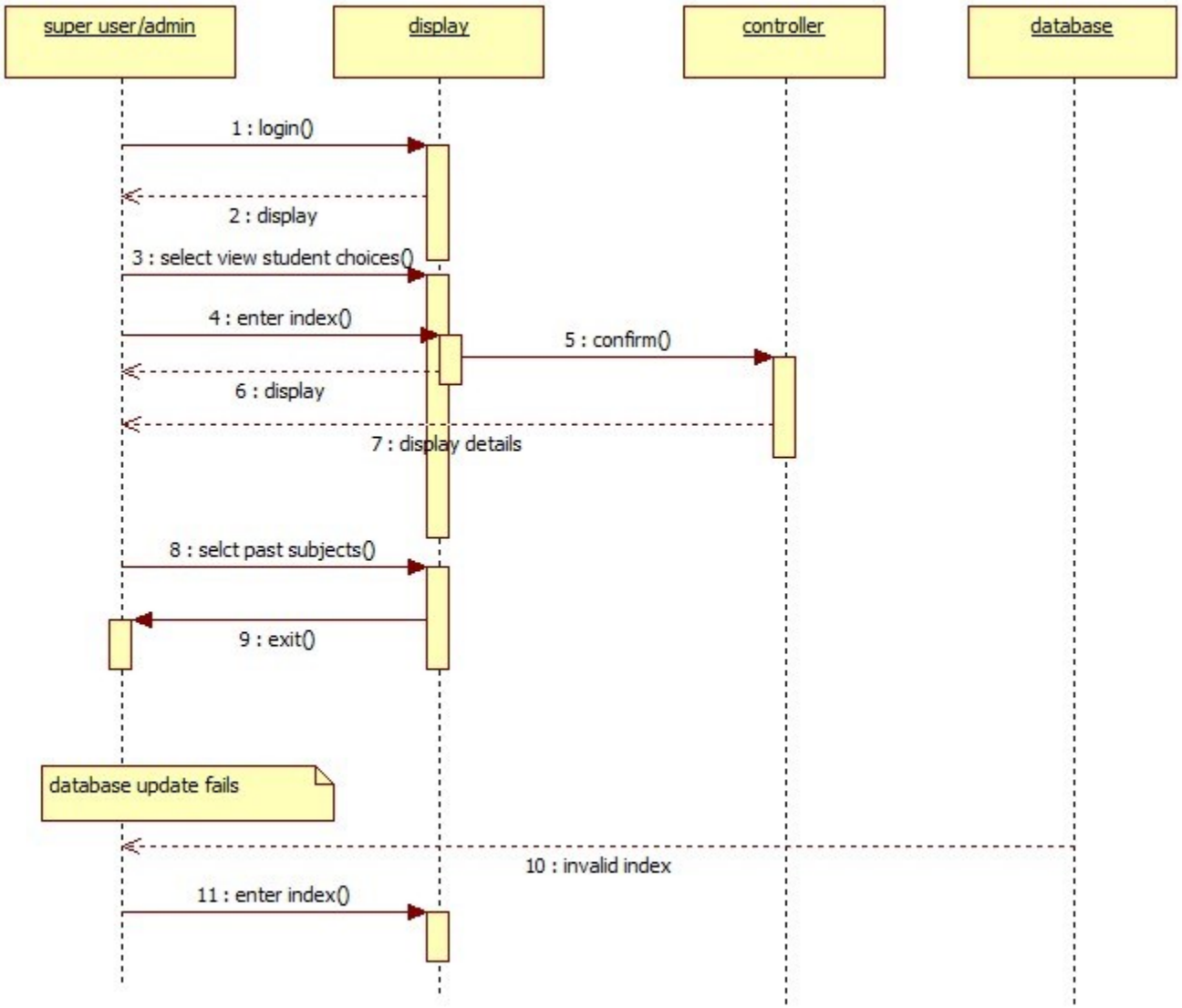
3.3.15 Sequence diagram 15: Update time table



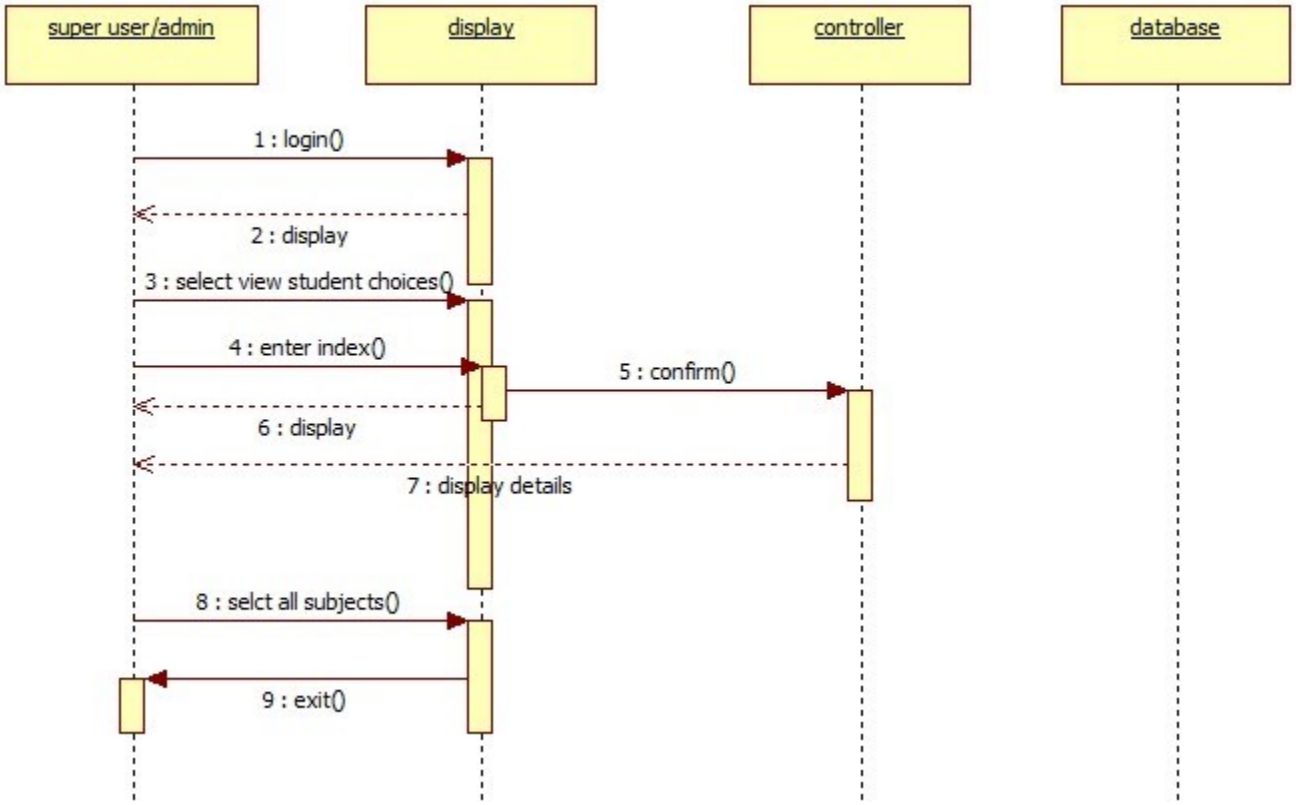
3.3.16 Sequence diagram 16: Current subjects by subject vice



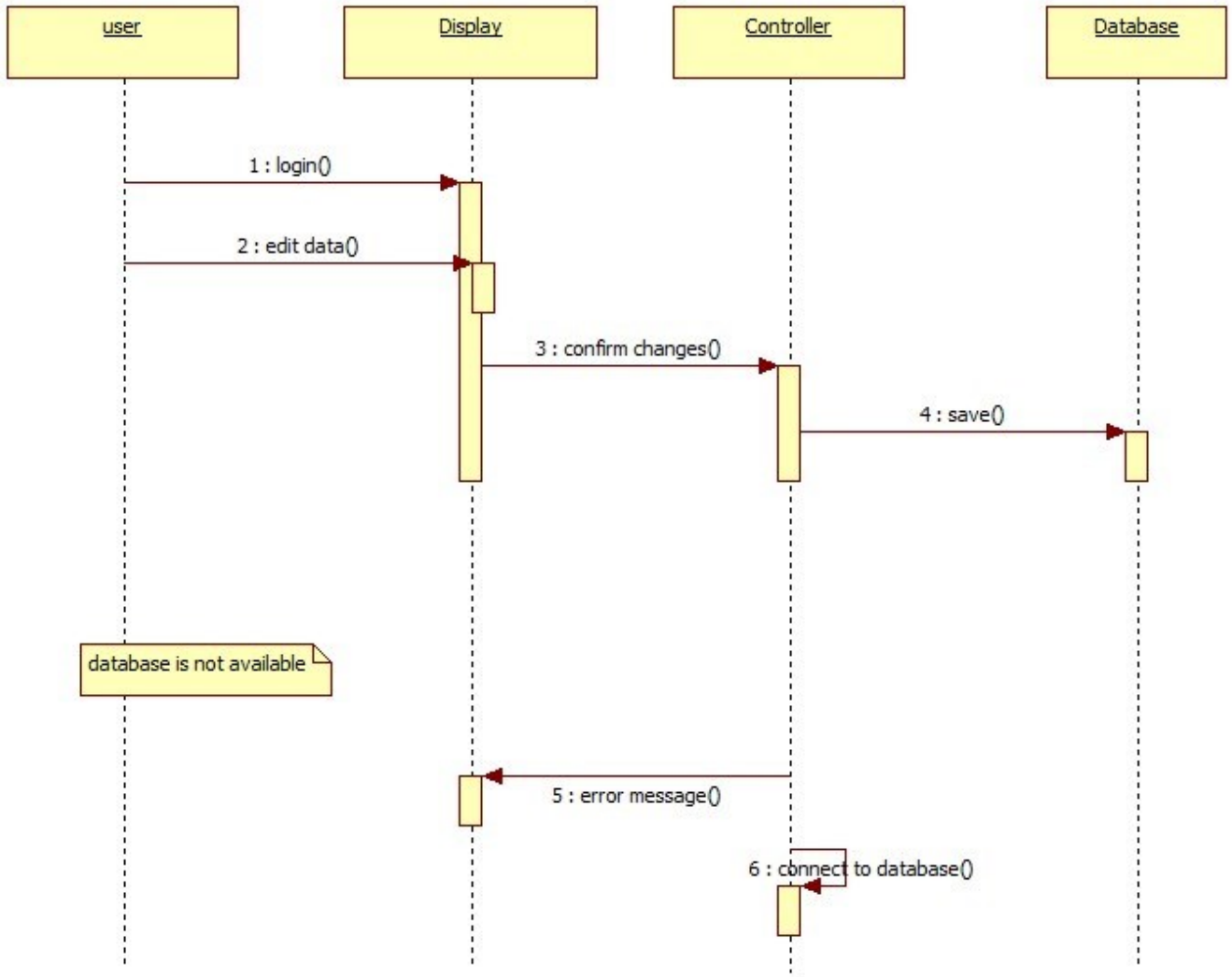
3.3.17 Sequence diagram 17: Past subject selections by subject vice



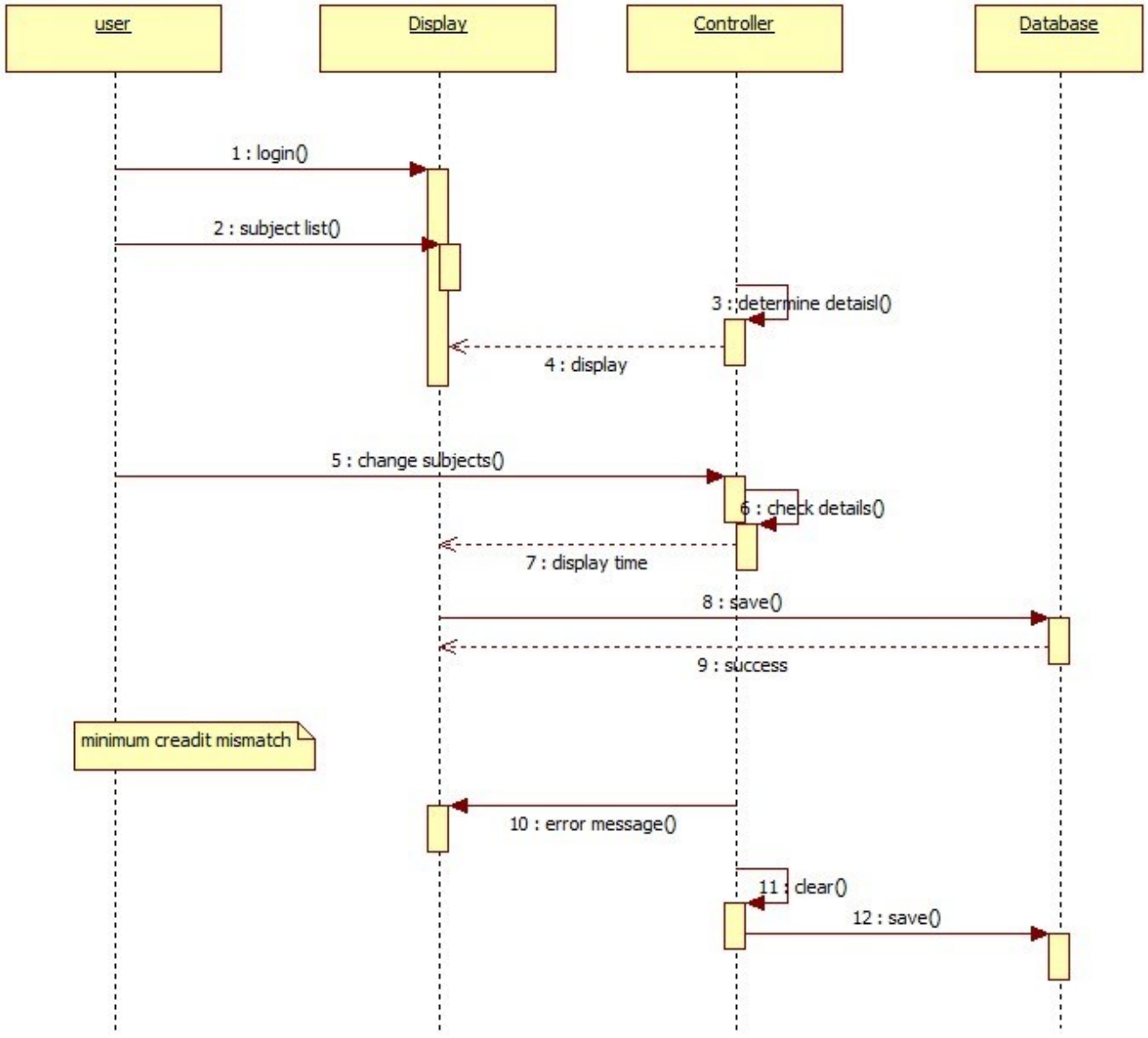
3.3.18 Sequence diagram 18: View the past & present subject selections of a student



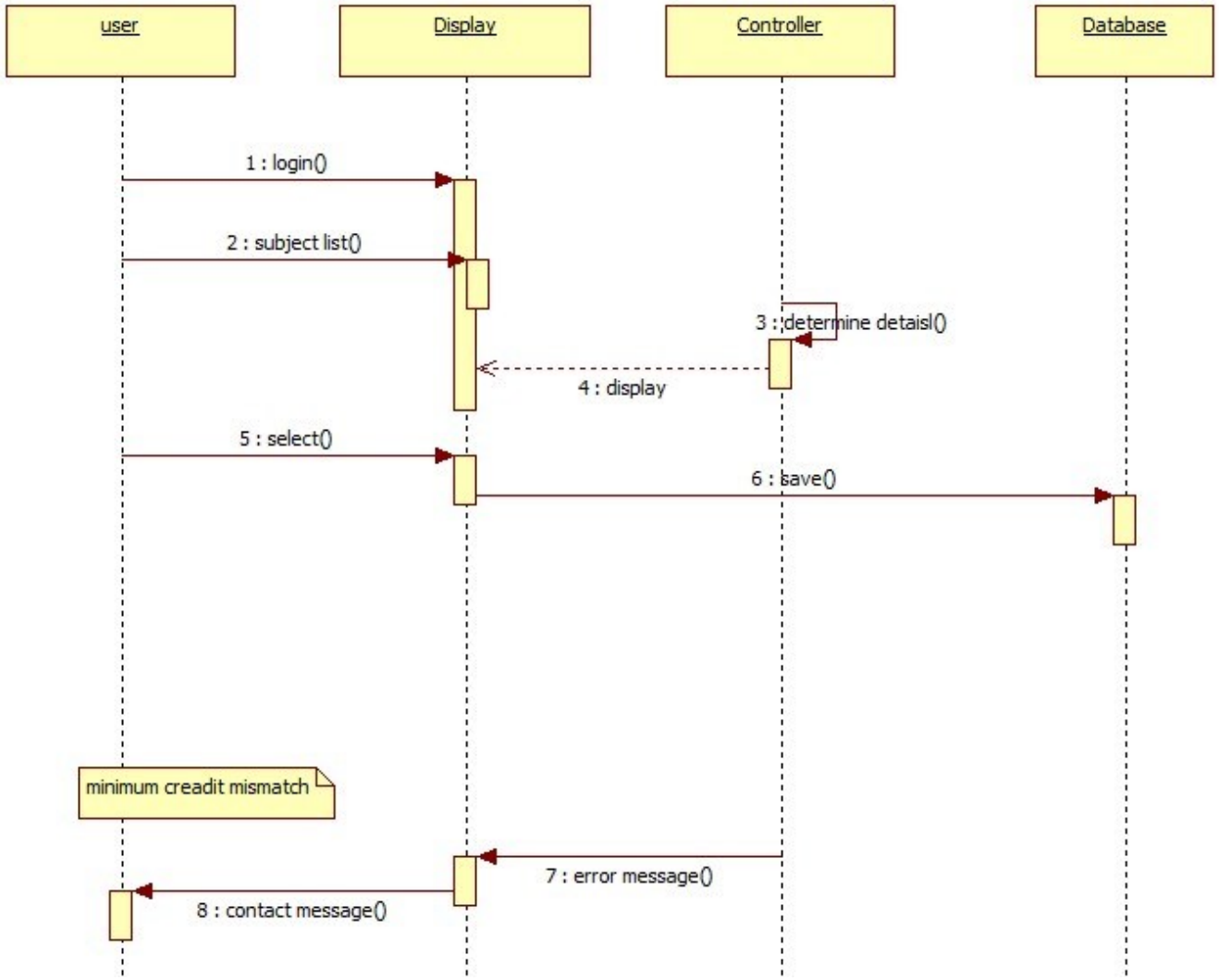
3.3.19 Sequence diagram 19: Edit student's profile



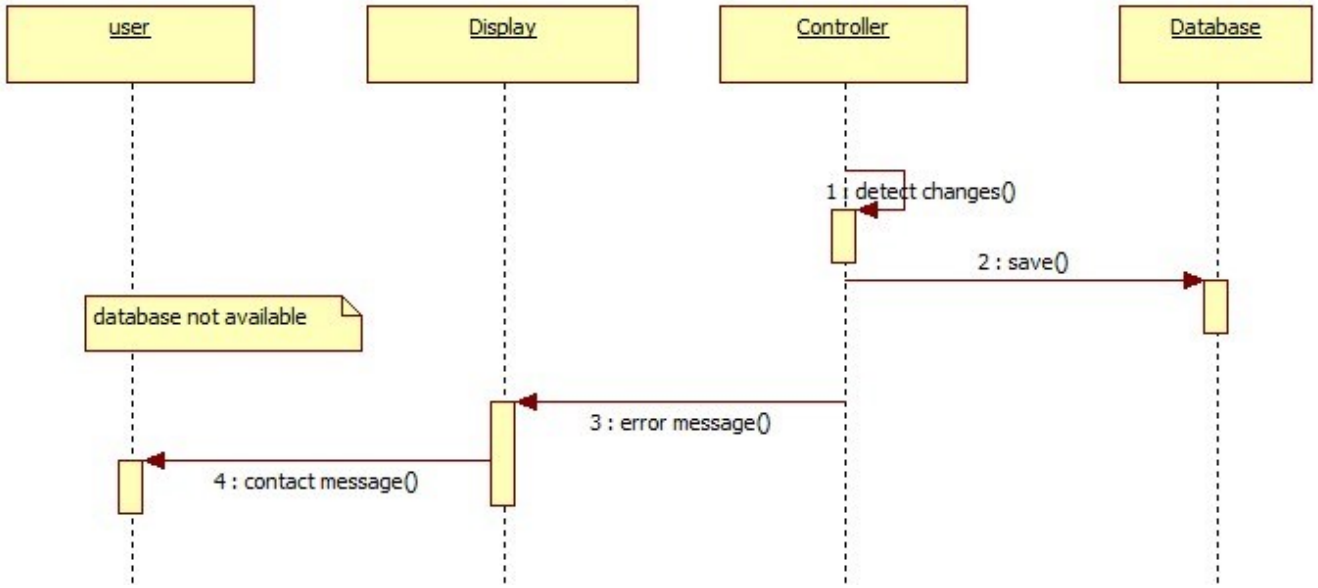
3.3.20 Sequence diagram 20: Select subjects



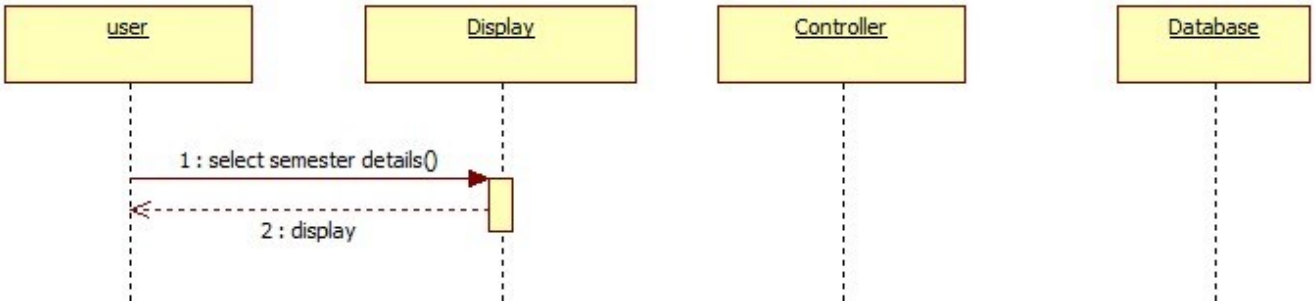
3.3.21 Sequence diagram 21: Remove subjects from the list



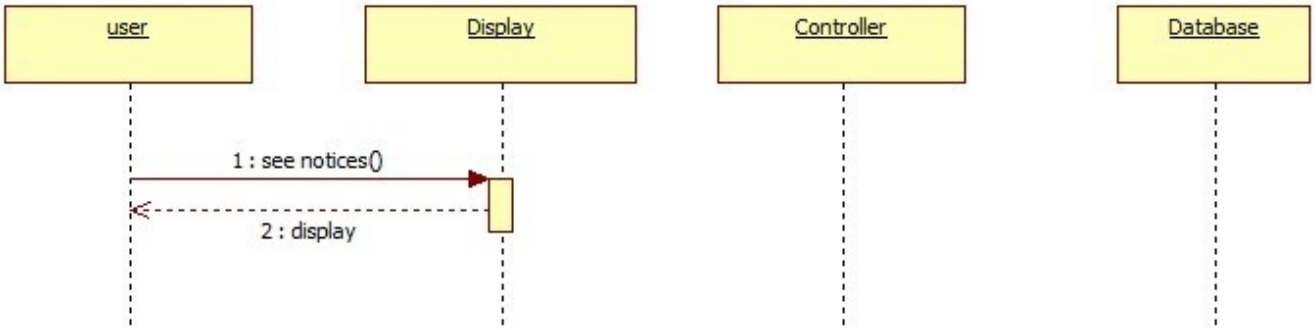
3.3.22 Sequence diagram 22: Save



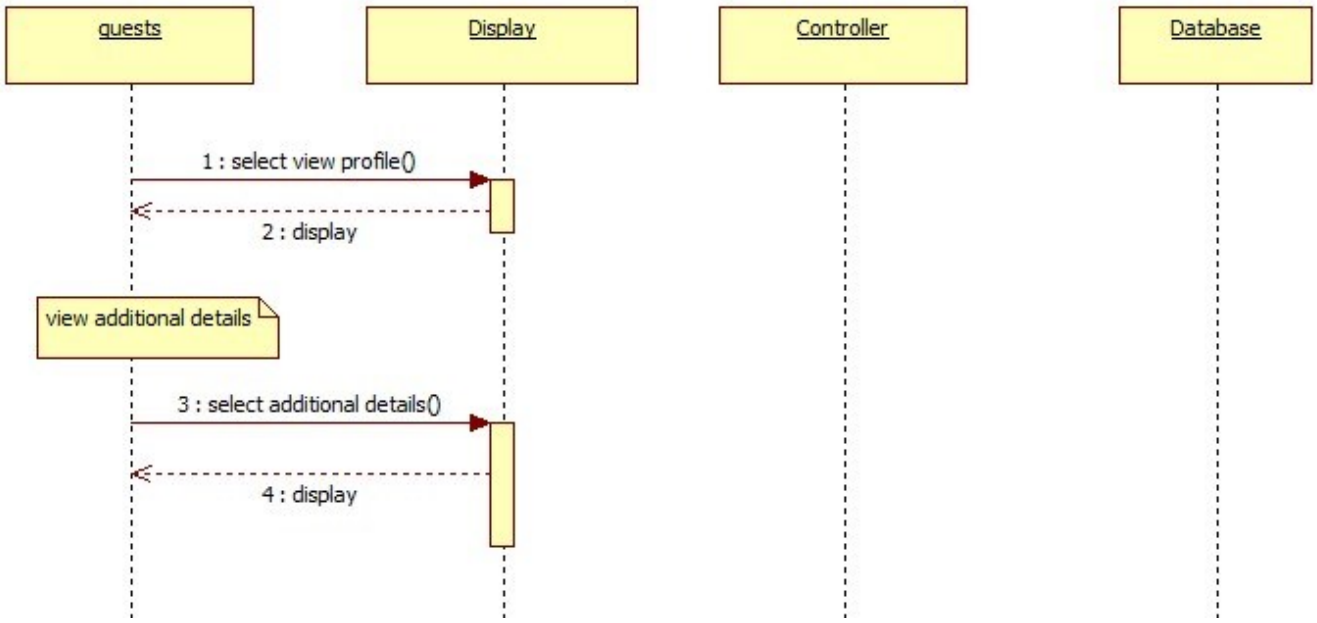
3.3.23 Sequence diagram 23: View previous semester information



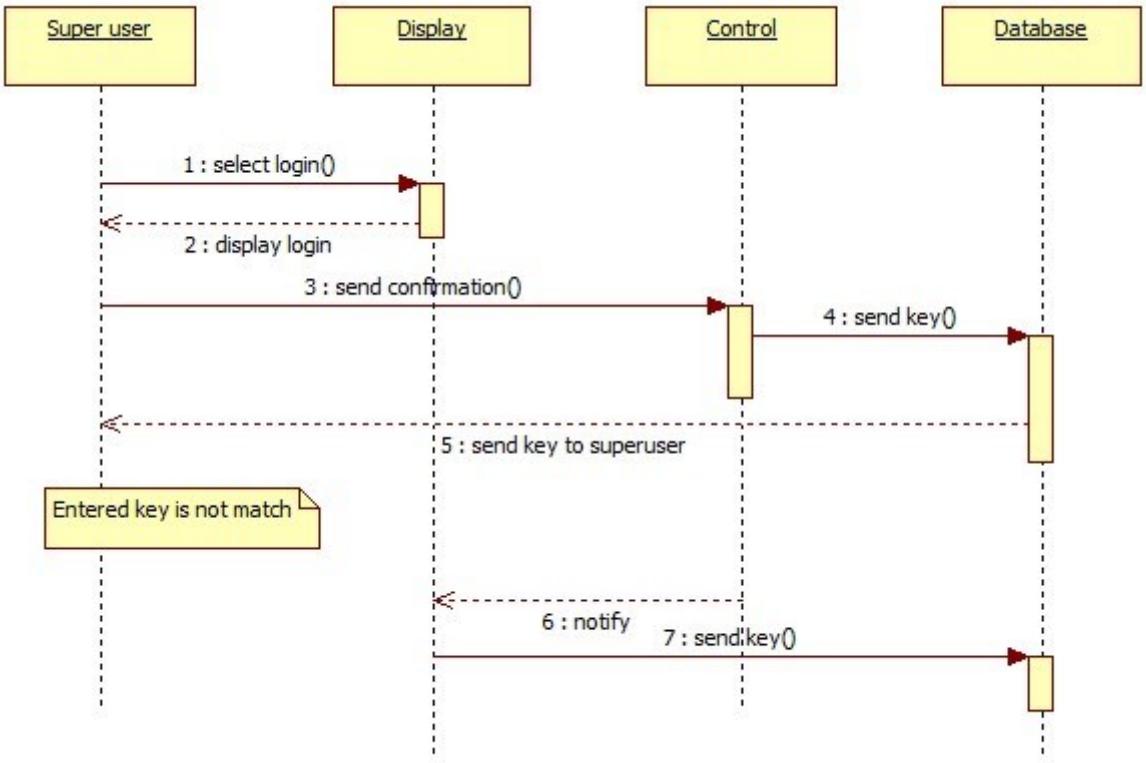
3.3.24 Sequence diagram 24: View Notices



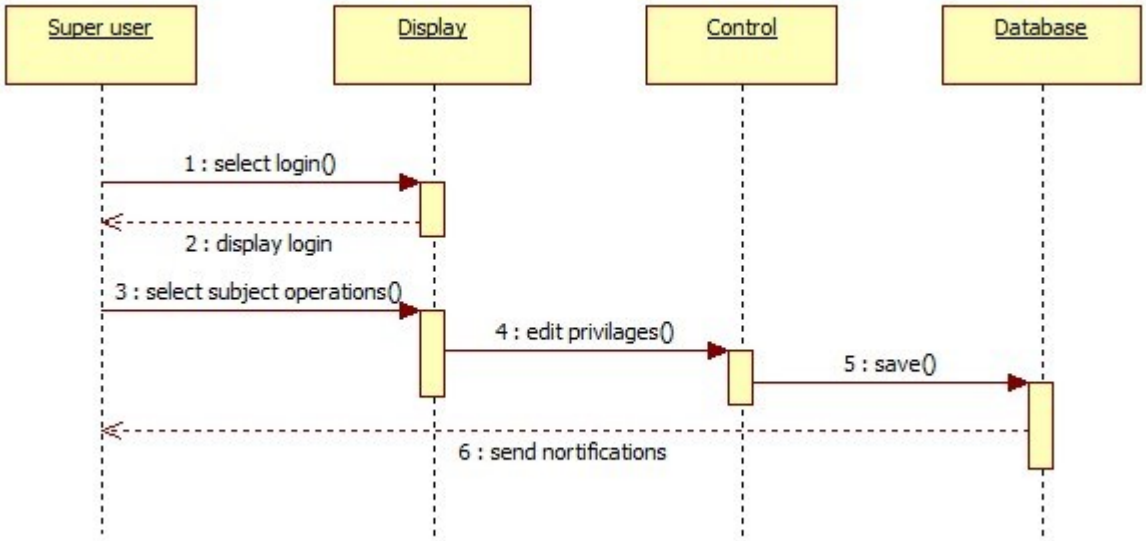
3.3.25 Sequence diagram 25: Guest view of the student's profile



3.3.26 Sequence diagram 26: Super user confirmation



3.3.27 Sequence diagram 27: Access Control Matrix



3.4 Algorithm Design

Login

1. Get Username and Password
2. If user name is equal to the entered Username & the password is equal to the entered Password
3. Then login successful
4. Else login failed
5. End If.

```
Get Username Get Password
IF   FILE EXIST THEN
   READ Password   FROM   FILE
        IF FILE. Password ==   Entered Password
             Login   successful
                 ELSE
                    PRINT "incorrect Username or password "
        END IF
ELSE
      PRINT "incorrect Username or password "
END IF
```

Student - Edit Profile

1. Student login to account
2. Select student personal details option

```
While Edit Profile
{
Add details
Remove details
}
Save changes
EXIT
```

Student - View Previous Semesters Information

1. Student login account
2. Select exam details option
3. Then view previous semester result preview
4. EXIT

Student select & remove subjects

1. Student login account
2. Select student subject option
3. Select academic year and the semester
4. Then user selects subject from the appeared list as preferred

IF wrong selection
Remove Subjects
ENDIF
Confirm selections
EXIT

Guest - Student Performance Preview

1. Guest login account
2. Enter relevant Index number and Password
3. IF Index number is equal to the entered Index number and the Password is equal to entered Password
4. Then preview student performance
5. ELSE Print incorrect password or index
6. Back to step 2
7. EXIT

Administrator - Subject Operation

1. Admin login to account
2. Select admin selection option
3. Then go to subject managing tool

IF required privileges given

{
Select relevant course, year, and the semester
While {
Viewing relevant subject information
Edit Subject
Remove Subject
}
Select update information

}
ELSE
Print privileges not given
EXIT

Administrator Account Operations

1. Admin login to account
2. Select login management tool

IF required privileges
{
IF exiting account
{
IF select remove user
{
Confirm
}
ELSE select Manage User
}
IF Suspend account
{
Enter relevant details
Confirm
}
ELSE

{
Select login Issuing Tool
Select stream and year

Enter default and unique option details
Create account
}
}
ELSE
Print "Required Privileges not given "

EXIT

Administrator - Time Table Publishing

1. Admin login account
2. Select admin selection option

IF required privileges given
{
Then go to Time Table Preview page
Select course name, academic year and semester
View relevant time table
IF already
Exit a time table
Then update as admin prefer
ELSE
Create new time table
}
ELSE
Print "Required Privileges not given"
EXIT

Administrator - View Student Choices

1. Admin login to account
2. Select admin selection option

IF Required privileges given
{

Then go to student choices page
Enter relevant index number and registration number
IF change subjects

{
Edit subject choices
Save
}
ELSE

{View past and present subjects}
}
ELSE

{Print "Privileges not given"}
EXIT

Administrator - View Users' Profile

1. Admin login to account
2. Select login management option

IF required privileges given
{

Enter user's registration number
View users profile
}
ELSE
{
Print "Privileges not given"
}
EXIT

Super User privilege Access Control Matrix

1. Super user login to system
2. Select privilege access control matrix

IF select edit

{
Edit privileges
Save changes
}
ELSE

Selected view matrix
EXIT

Super user - Account Operation

1. Super user login to system
2. Select account operations

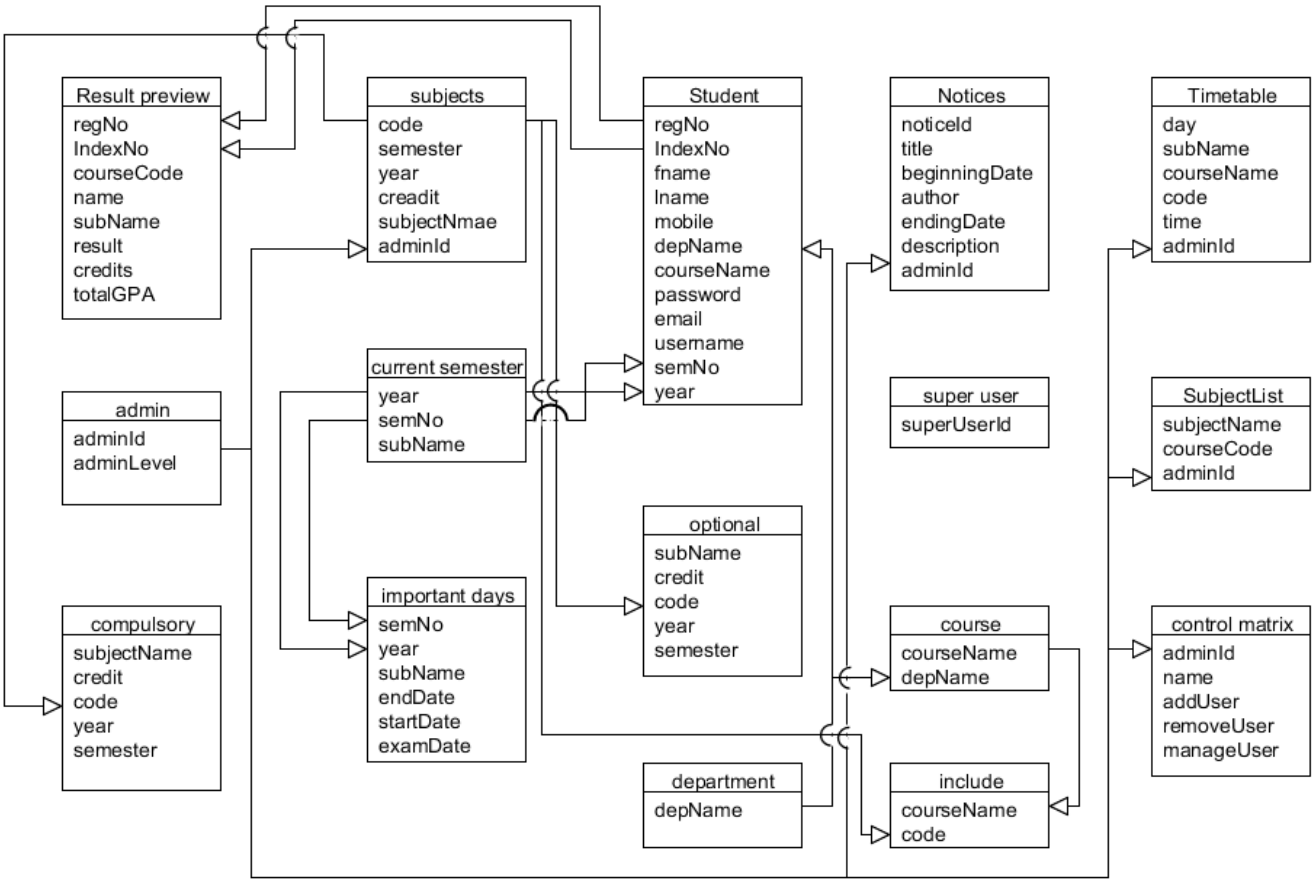
IF select remove user
{

Enter user type
Enter relevant ID or Registration number
Go to relevant user profile
Confirm removal

}
ELSE select add user
{
Select user type
Enter details
Create account
}
EXIT

3.5 Database design

3.5.1 Relational model



3.5.2 Normalization / DE normalization

Database normalization is the process of organizing the fields and tables of a relational database to minimize data redundancy and dependency. Normalization usually involves dividing a database into two or more tables and defining relationships between the tables. The objective is to isolate data so that additions, deletions, and modifications of a field can be made in just one table and then propagated through the rest of the database via the defined relationships.

There are three main normal forms, each with increasing levels of normalization:

- First Normal Form (1NF): Each field in a table contains different information. For example, in a Student list, each table would contain only one email field.
- Second Normal Form (2NF)**:** Each field in a table that is not a determiner of the contents of another field must itself be a function of the other fields in the table.
- Third Normal Form (3NF)**:** No duplicate information is permitted. So, for example, if two tables both require an email field, the email information would be separated into a separate table, and the two other tables would then access the email information via a password field in the student table. Any change to a email would automatically be reflect in all tables that link to the student table.

3.5.2.1 Anomalies

When there are too many attributes it is called as anomalies.

For example,

STUDENT (Reg-no,Index no,Fname,Lname,Mobile,Dept-name,Course name,Password,E-mail)

There are basically three types of anomalies.

1. **Deletion anomalies** – Deletion anomalies occur when we delete data from flawed schema.
2. **Insertion anomalies** – Insertion anomalies are occur when we try to insert data into flawed Table.
3. **Update anomalies** – Update anomalies occur when we change data in a flawed schema.

3.5.2.2 Normalization in our System

In this table it is satisfied requirements which should have for the first normalization. In second Normalization tables have changed as following.

Subject table has normalized in two tables as “subject” and “subject-credits”.

- Subject (sub-code, Sub-name)
- Subject-credits (Sub-code,Semester,Year,Credit)

Student table has normalized in to three tables as “Student”, “Student-course name” and “student year”.

- Student (Reg-no,Index-no,Fname,Lname,Mobile)
- Student-year (Reg-no,Course name,Password,Email,User name)
- Student-year (Index-no,Sem-no,Year)

Result preview tabel has normalized in to two tables as “Result preview” and “Result total GPA” .

- Result preview (Reg-no,Index-no,Course code,Name,Sub-Name)
- Result total GPA (Reg-no,Results,Credits,Total GPA)

Likewise we can normalizing our tables into sub tables. One table divide into several tables. For the third normalization also requirements have satisfied. There for now all the tables have satisfied the requirements in First, Second, and third normalization.

3.5.3 Data Types Identification

3.5.3.1 Indexes

Result preview		
Field	**Data type**	**Size of Disk**
Reg-no	**Int(11)**	**4 bytes**
Index no	**Int(11)**	**4 bytes**
Course code	**Varchar(50)**	**50 bytes**
Name	**Varchar(200)**	**200 bytes**
Sub- name	**Varchar(200)**	**200 bytes**
Result	**Varchar(50)**	**50 bytes**
Credits	**Varchar(50)**	**50 bytes**
Total GPA	**Varchar(50)**	**50 bytes**

Course		
Field	**Data type**	**Size on disk**
Course Name	**Varchar(50)**	**50 bytes**
Dept-name	**Varchar(100)**	**100 bytes**

Student		
Field	**Data type**	**Size on disk**
Reg-no	**Int(20)**	**4 bytes**
Index-no	**Int(20)**	**4 bytes**
Fname	**Varchar(50)**	**50 bytes**
Lname	**Varchar(50)**	**50 bytes**
Mobile	**Varchar(20)**	**20 bytes**
Dept-name	**Varchar(20)**	**20 bytes**
Course name	**Varchar(20)**	**20 bytes**
Password	**Varchar(20)**	**20 bytes**
E-mail	**Varchar(20)**	**20 bytes**
Name	**Varchar(20)**	**20 bytes**
Sem-no	**Varchar(20)**	**20 bytes**
Year	**Varchar(20)**	**20 bytes**

Subject		
Field	Data type	Size on disk
Code	Varchar (20)	20 bytes
Semester	Varchar (20)	20 bytes
Year	Varchar (20)	20 bytes
Credit	Varchar (20)	20 bytes
Subject Name	Varchar (20)	20 bytes

Important days		
Field	Data type	Size on Disk
Sem-no	Varchar(20)	20 byes
Year	Varchar(20)	20 bytes
Sub name	Varchar(20)	20 bytes
End date	Date	
Start date	Date	
Exam date	Date	

Compulsory		
Field	Data type	Size On Disk
Subject name	Varchar(20)	20 bytes
Credit	Varchar(20)	20 bytes
Code	Varchar(20)	20 bytes
Year	Varchar(20)	20 bytes
Semester	Varchar(20)	20 bytes

Optional		
Field	Data type	Size On Disk
Subject name	Varchar(20)	20 bytes
Credit	Varchar(20)	20 bytes
Code	Varchar(20)	20 bytes
Year	Varchar(20)	20 bytes
Semester	Varchar(20)	20 bytes

Notices		
Field	Data type	Size On Disk
Notice-Id	Int(20)	4 bytes
Title	Varchar(100)	100 bytes
Beginning date	Date	
Author	Varchar(50)	50 bytes
Ending date	Date	
Description	Varchar(500)	500 bytes
Admin-Id	Int(20)	4 bytes

Include		
Field	Data type	Size On Disk
Course name	Varchar(20)	20 bytes
Code	Varchar(20)	20 bytes

Current Semester		
Field	Data type	Size On Disk
Sem-no	Varchar(50)	50 bytes
Year	Varchar(50)	50 bytes
Sub name	Varchar(50)	50 bytes

Current Semester		
Field	Data type	Size On Disk
Sem-no	Varchar(50)	50 bytes
Year	Varchar(50)	50 bytes
Sub name	Varchar(50)	50 bytes

Admin		
Field	Data type	Size On Disk
Admin-ID	Int(20)	4 bytes
Admin level	Varchar(50)	20 bytes

Department		
Field	Data type	Size On Disk
Department-name	Varchar(20)	20 bytes

Time table		
Field	Data type	Size On Disk
Day	Varchar(20)	20 bytes
Subject name	Varchar(20)	20 bytes
Course name	Varchar(20)	20 bytes
Code	Varchar(20)	20 bytes
Time	Varchar(20)	20 bytes
Admin-Id	Int(20)	4 bytes

Subject list		
Field	Data type	Size On Disk
Subject name	Varchar(50)	50 bytes
Course code	Varchar(20)	20 bytes
Admin-ID	Int(20)	4 bytes

Control matrix		
Field	Data type	Size On Disk
Admin-ID	Int(20)	4 bytes
Name	Varchar(50)	50 bytes
Add users	Boolean	
Remove users	Boolean	
Manage users	Boolean	
View users Profile	Boolean	
Remove selected subject from selection	Boolean	
Create new subjects	Boolean	
Edit existing subjects	Boolean	
Delete subjects	Boolean	
Create new subject lists	Boolean	
Make adjustments of no of credits	Boolean	
Subject combination tool	Boolean	
Time table management	Boolean	
Create time table	Boolean	
Update time table	Boolean	
View student choices	Boolean	

3.5.6 Triggers

In a DBMS, a *trigger* is a SQL procedure that initiates an action when an event (INSERT, DELETE or UPDATE) occurs .The SQL CREATE TRIGGER statement provides a way for the database management system to actively control, monitor, and manage a group of tables whenever an insert, update, or delete operation is performed. The statements specified in the SQL trigger are executed each time an SQL insert, update, or delete operation is performed. An SQL trigger may call stored procedures or user-defined functions to perform additional processing when the trigger is executed.

The CREATE TRIGGER statement defines a trigger at the current server.

Ex:

```
CREATE TRIGGER fname_trigger
BEFORE UPDATE ON student_table
REFERENCING NEW ROW AS n, OLD ROW AS o
FOR EACH ROW
IF n.fname <> o.fname THEN

END IF;
```

3.6 User interface designing

3.6.1 Rules and guidelines for user interface designing

3.6.1.1 User input validation methods

This system will use java script for basic form validations. But for further validations functions we use php. Using php greatly increases security and usability.

JavaScript will detect empty fields of a form, and php will be used to determine whether inputs are met some per-defined rules to insure only desired inputs are stored in the database. Detecting and avoid accepting erroneous inputs before storing in the database, will make system more efficient.

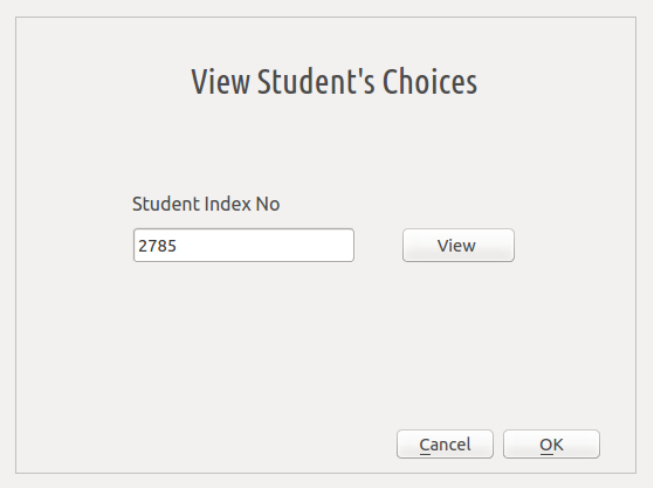
E.g.

Rules to make sure user enters a correct E-mail address:

1. Input e-mail address must contain a @ sign and at least one dot (.)
2. Also, the @ must not be the first character of the email address, and the last dot must be present after the @ sign and minimum 2 characters before the end.Php will be used to ensure that these rules are met, while entering an E-mail address.

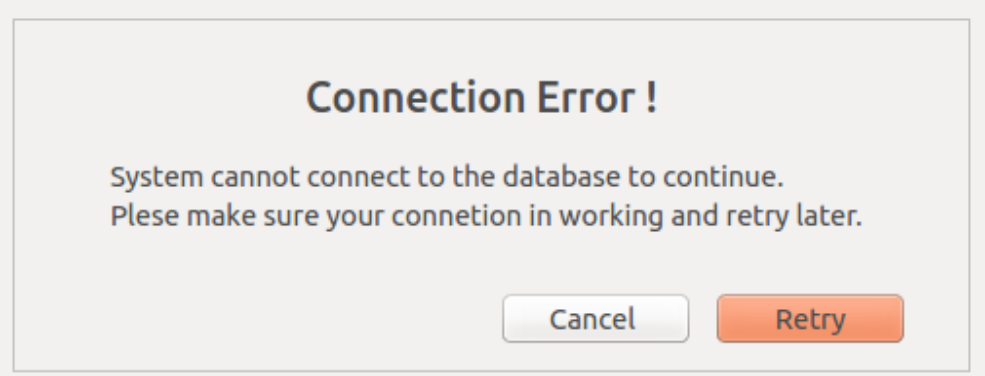
Few instances we hope to use validations

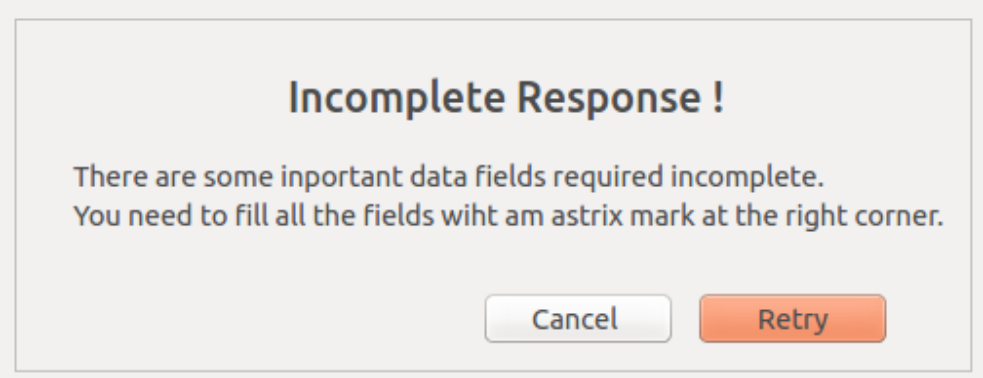
1. On all text inputs, check boxes to ensure user fill all the required fields.
2. E-mail address validations (mentioned above)
3. Input date validation, to make sure entered date is not likely to be an unacceptable one.



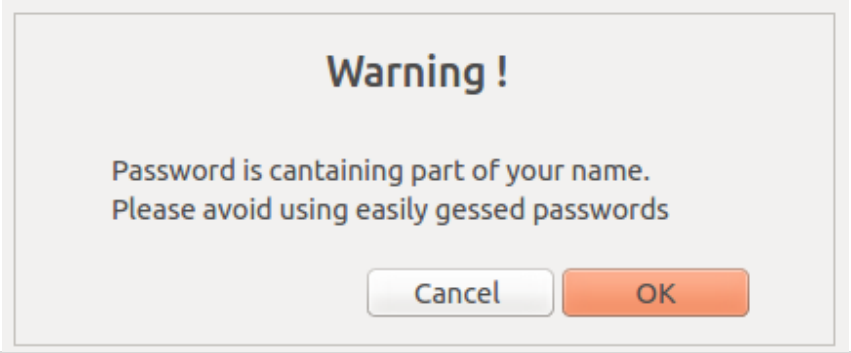
3.6.1.2 Guideline for error messages, warnings & supportive information.

System will use simple but informative error messages to inform user what is wrong with a certain operation. The type of the error will be stated top of the error message, enabling user to understand quickly what caused the problem.

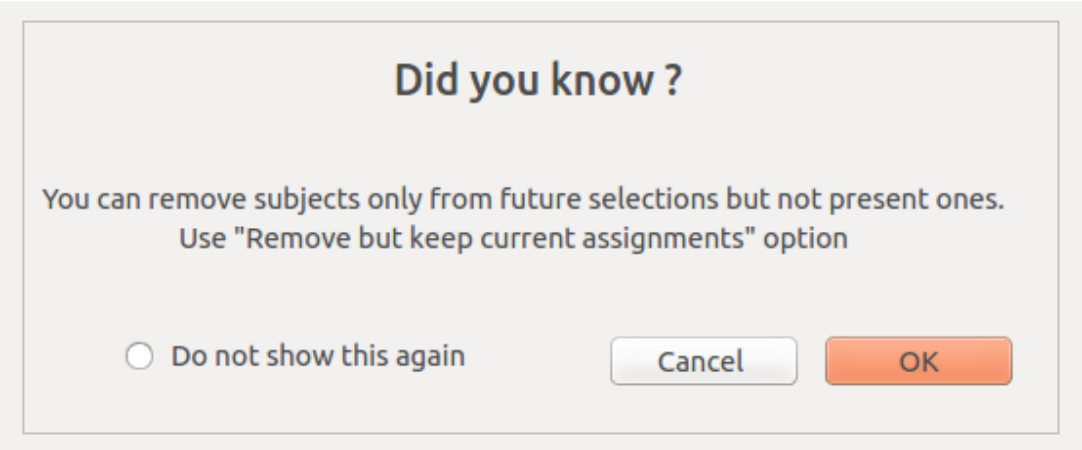




Warnings will contain the same characteristics of Error messages but will clearly display the type of it. Occupationally system will display some useful tips and trick for the user. This will reduce user's time to figure out everything on their own.



A tips will look like Warning.



3.6.1.3 Guideline for Interface designing.

1. Allow users to use either the keyboard or mouse (flexible)

   Users will be able enter data and navigate using any method, either keyboard or mouse. (Scrolling, Selecting)

2. Allow users to change focus (interruptible)

   Users will open several windows once but focus working on only one interface. This will open different interfaces in different web browser tabs, making it easier to change among tasks. (Navigation will be convenient)

3. Display descriptive messages and text (Helpful)

   Tool tips will be displayed to help users to understand further what important operations and buttons do. User will be able to view them when mouse hover above the relevant texts, buttons.

4. Provide immediate and reversible actions, and feedback (forgiving)

   Most of the time users will be given help to revise their selections in the same interface where they make changes.
   E.g. there will be options to remove subjects on the interface where users select their subject list, making it easier to do changes quickly.

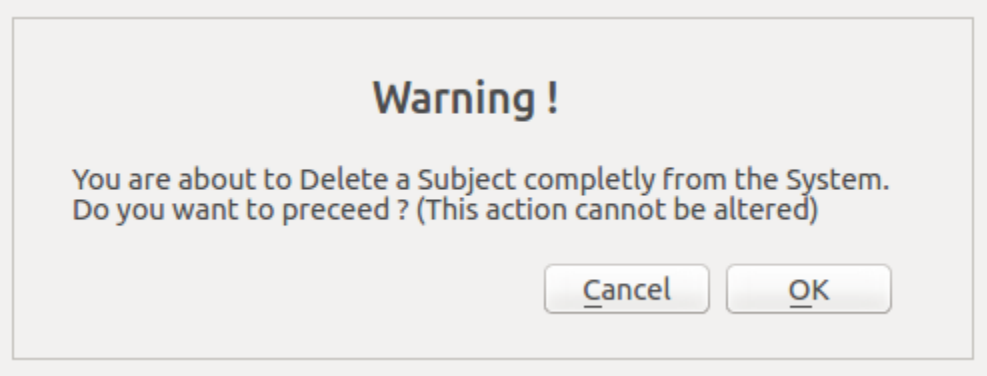
5. Provide meaningful paths and exits (navigable)

Finding a relevant interface and exiting from it should be easy and sensible. Operations should be able to understand without much technical knowledge for all users. Users should be able to guess where to find certain interfaces without complex procedures.

6. Accommodate users with different skill levels (accessible)

System will be used by users with many skill levels. And all of them should be able get their job done without specific technical support. This system will be designed with simple interfaces, easy to understand operations and many tool tips. And occasional warnings will be displayed if the users is likely to do a risky operations to aware the user about its danger.

E.g. Users will be displayed warnings when they choose to delete a subject from database or if a student choose to remove a subject and it makes some issue.es. (Not covered an enough credit level)



7. Principals that make the user interface consistent

a. Sustain the context of users' tasks (continuity)

Users will be provided points of reference as they navigate through a product interface. Window titles, navigation maps and trees, and other visual aids give users an immediate, dynamic view of where they are and where they've been.

b. Maintain consistency within and across products (experience)

One of the most important aspects of the interface is to enable users to learn general concepts about system and products and then apply what they've learned to new situations in different programs or different parts of the system. Consistency in behavior means that the way an object works is the same everywhere.

c. Keep interaction results the same (expectations)

Interaction technique consistency is also important. The same shortcut keys should work in similar programs. Consistency in interface behavior is very important. If users experience different results from the same action, they tend to question their own behavior rather than the product's behavior.

d. Provide aesthetic appeal and integrity (attitude)

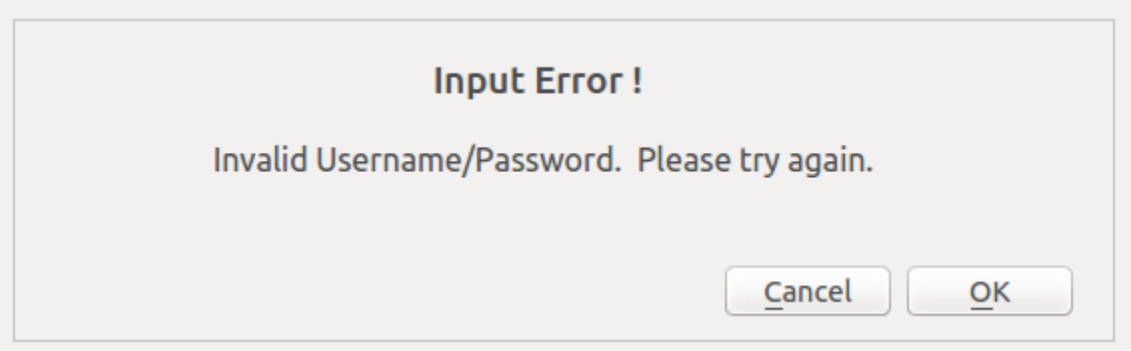
The software interfaces will be clean and professional looking but also it will not contain any high resource consuming visual effects and tricks. Because as we hope to implement this on a campus server, we try to use minimal resources as much as possible. Otherwise when there are many students are logged into the system, it will not be able to handle all user requests.

e. Encourage exploration (predictable)

Interface will be clean and user friendly as described above, but we try to make all the interfaces and options more predictable and user finds it is fun to use the software as it is a simple. User should be tempted to use new features and explore them, as there are no use of being quiet.

3.6.1.4 Guideline for error messages, warnings & supportive information.

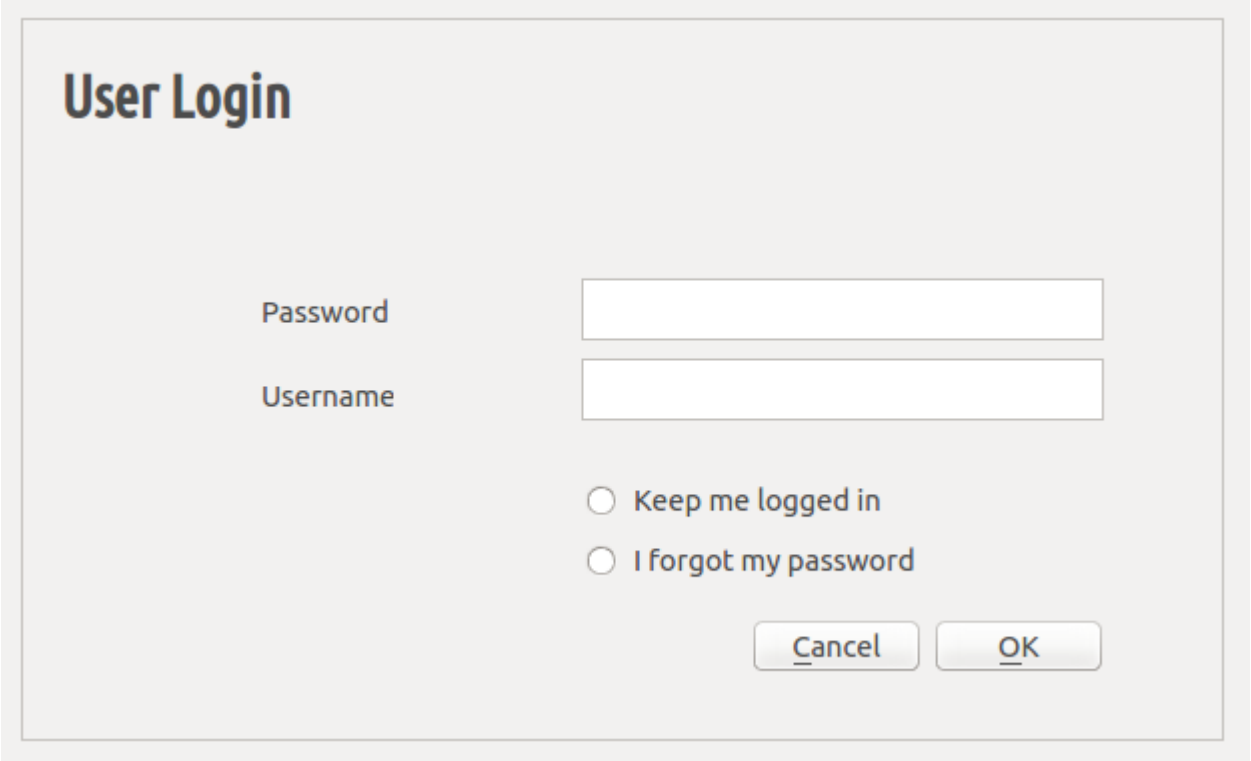
We use simple but informative error messages to inform user what is wrong with a certain operation. The type of the error will be stated top of the error message, enabling user to understand quickly what caused the problem.



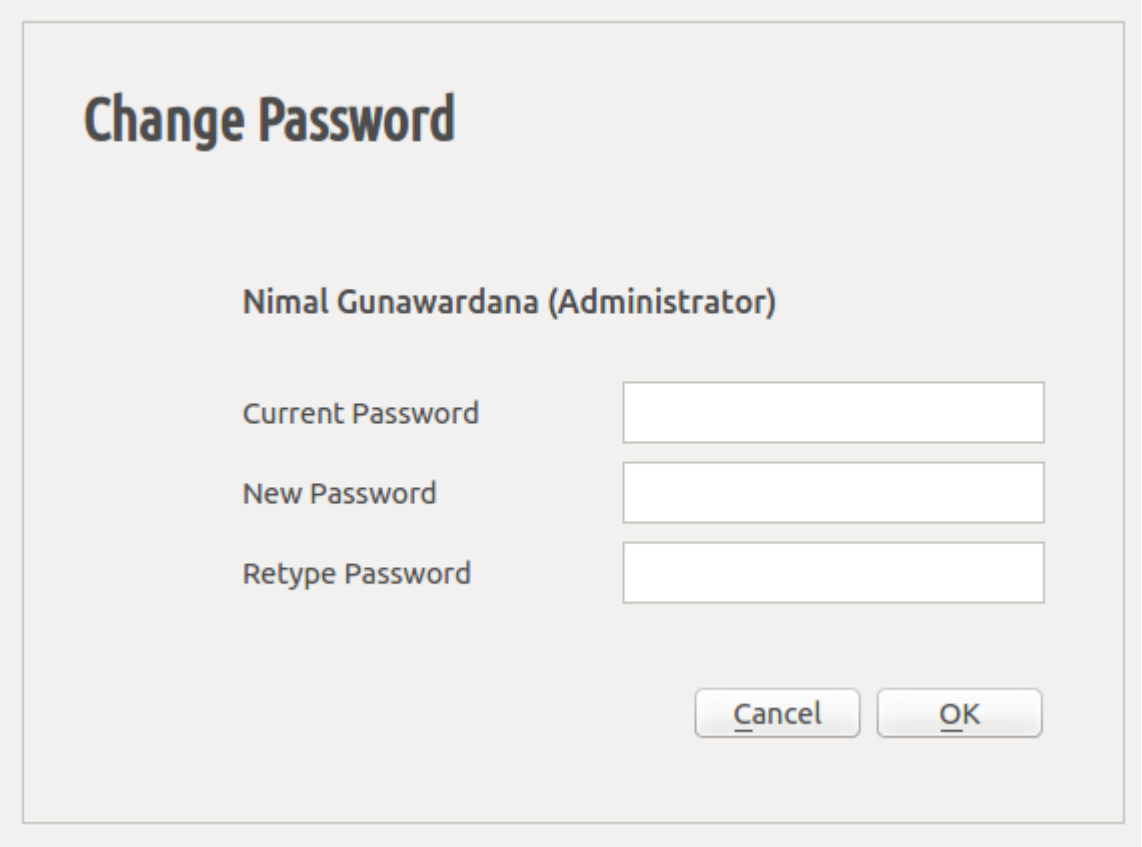
(Reference: Golden rules of user interface design by Theo Mandel)

3.6.2 User interfaces for each use case

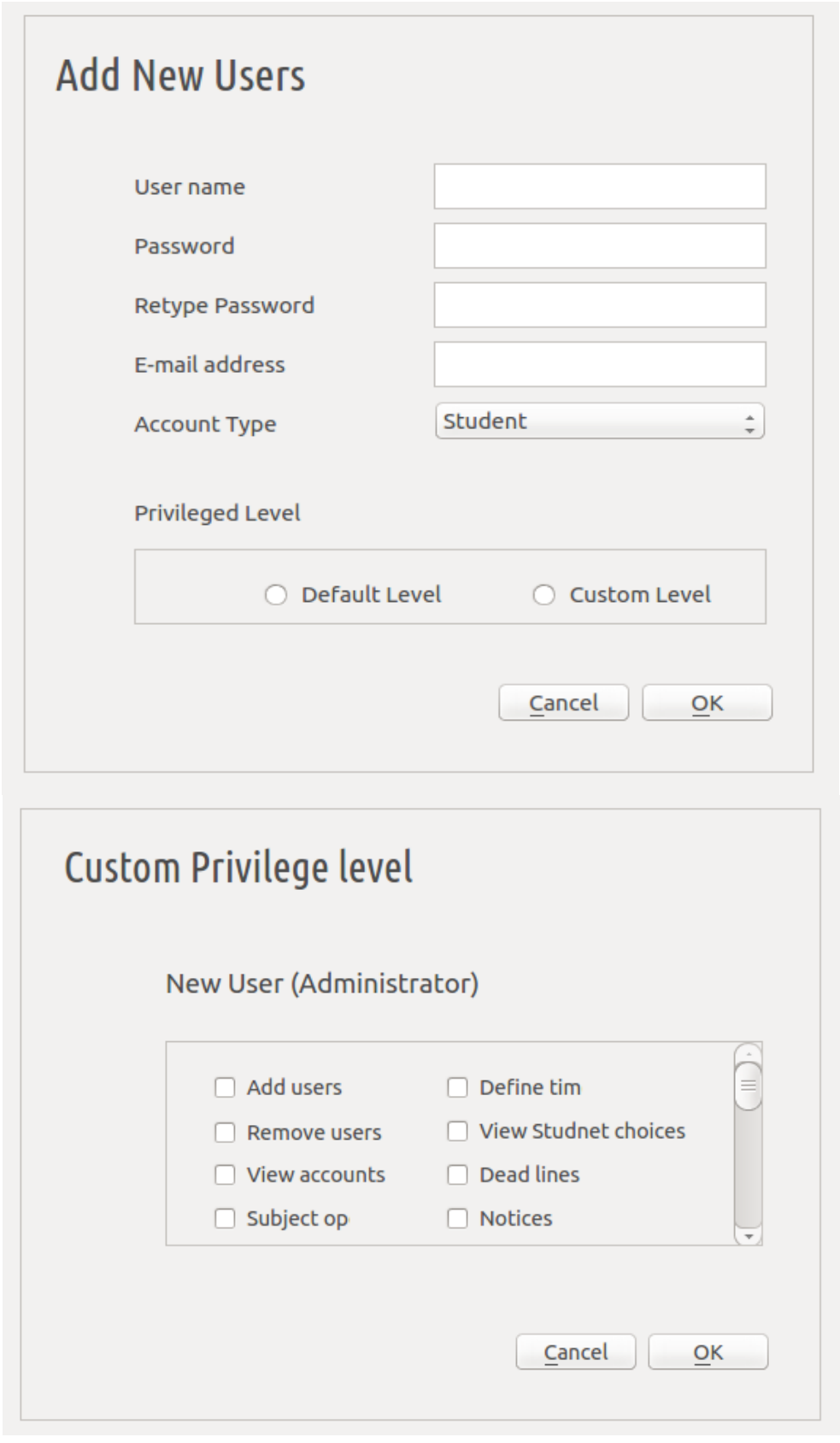
Use case 01 : Login into the system.



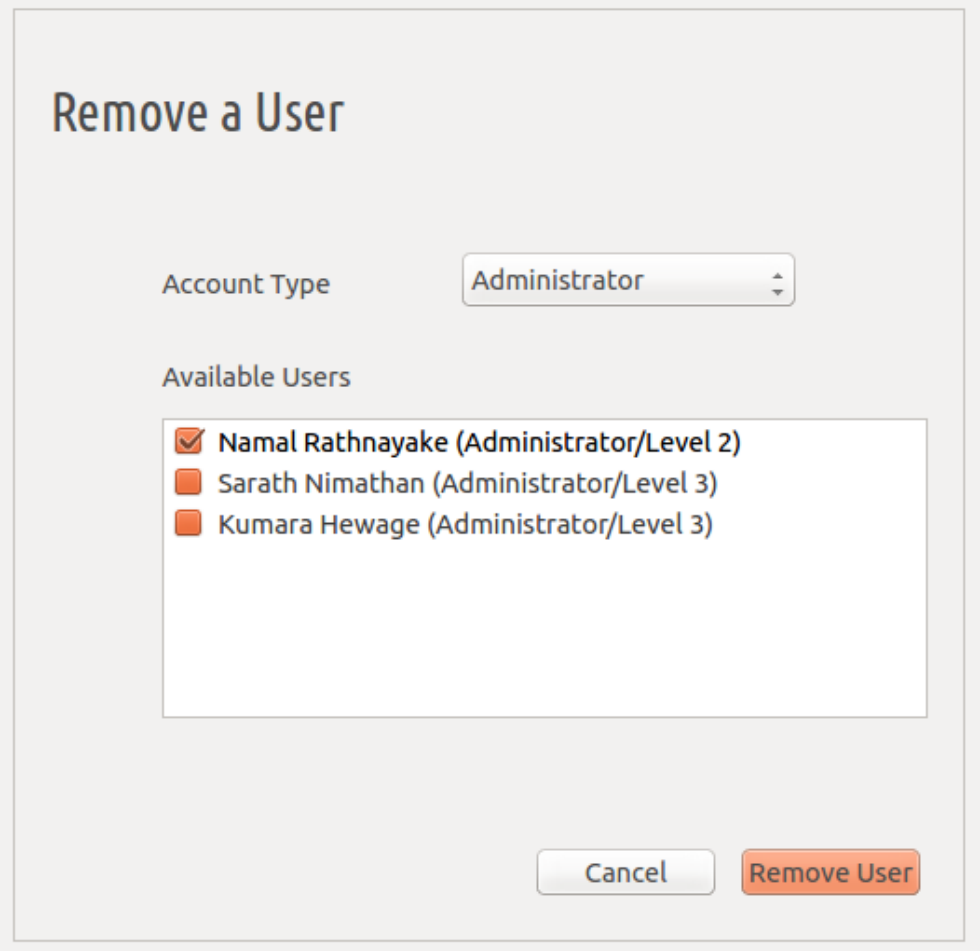
Use case 02	:	Change user password

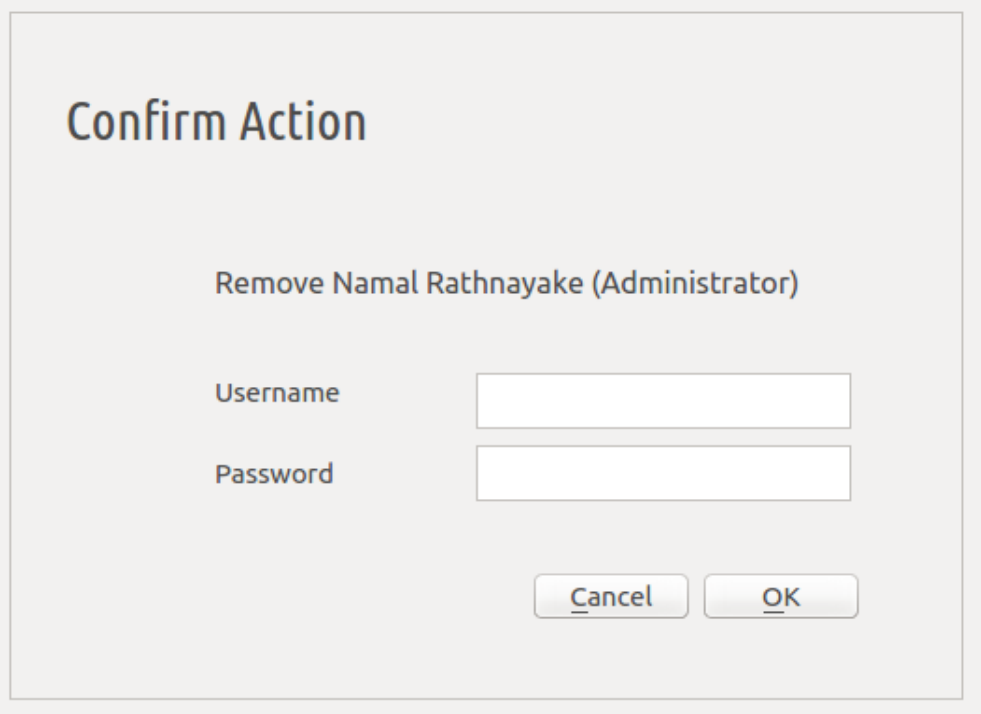


Use case 03	:	Add users

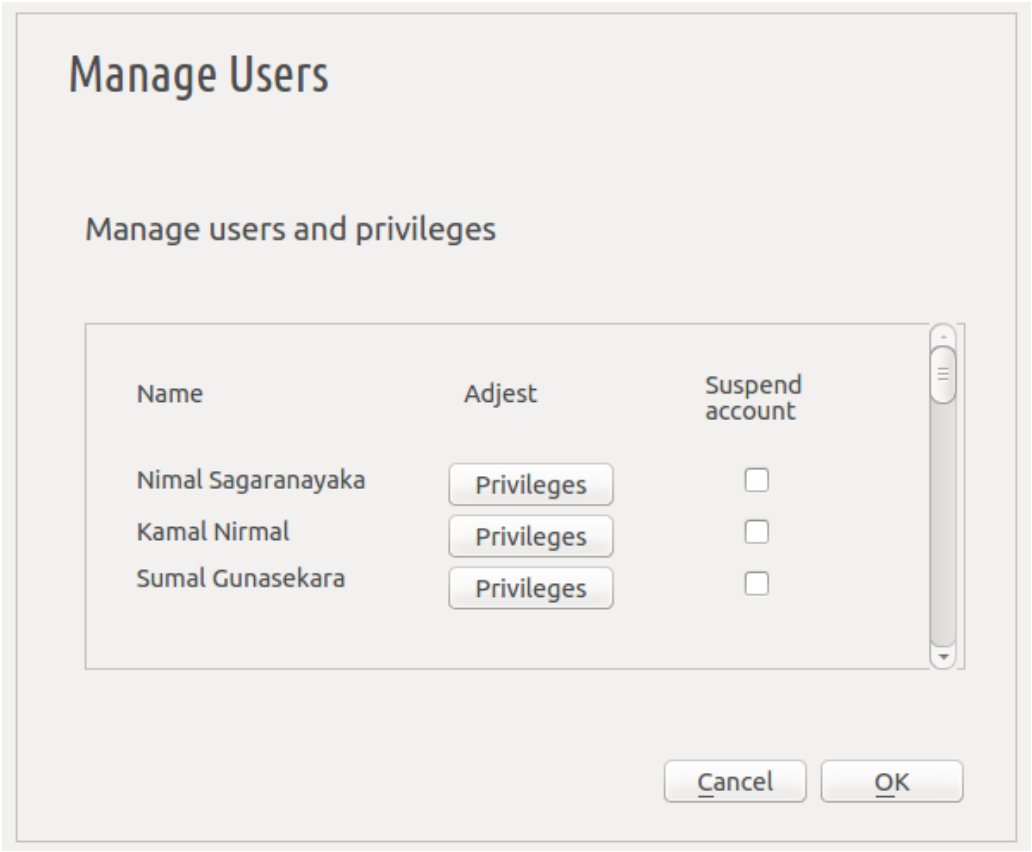


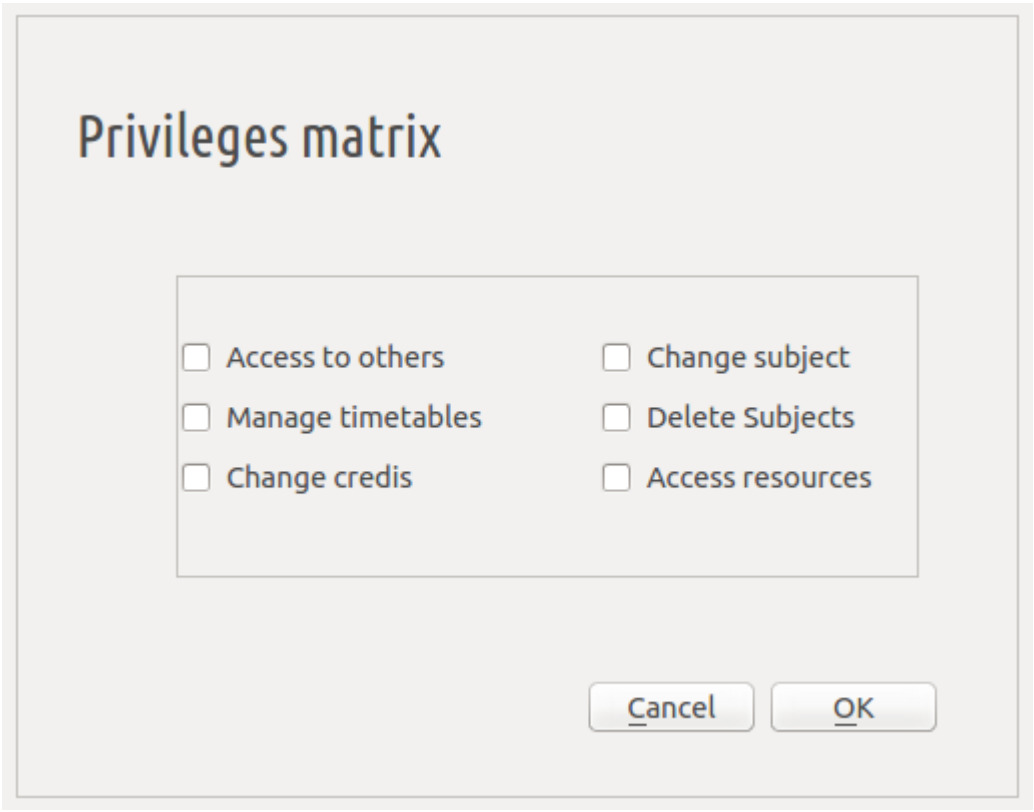
Use case 04 : Remove users



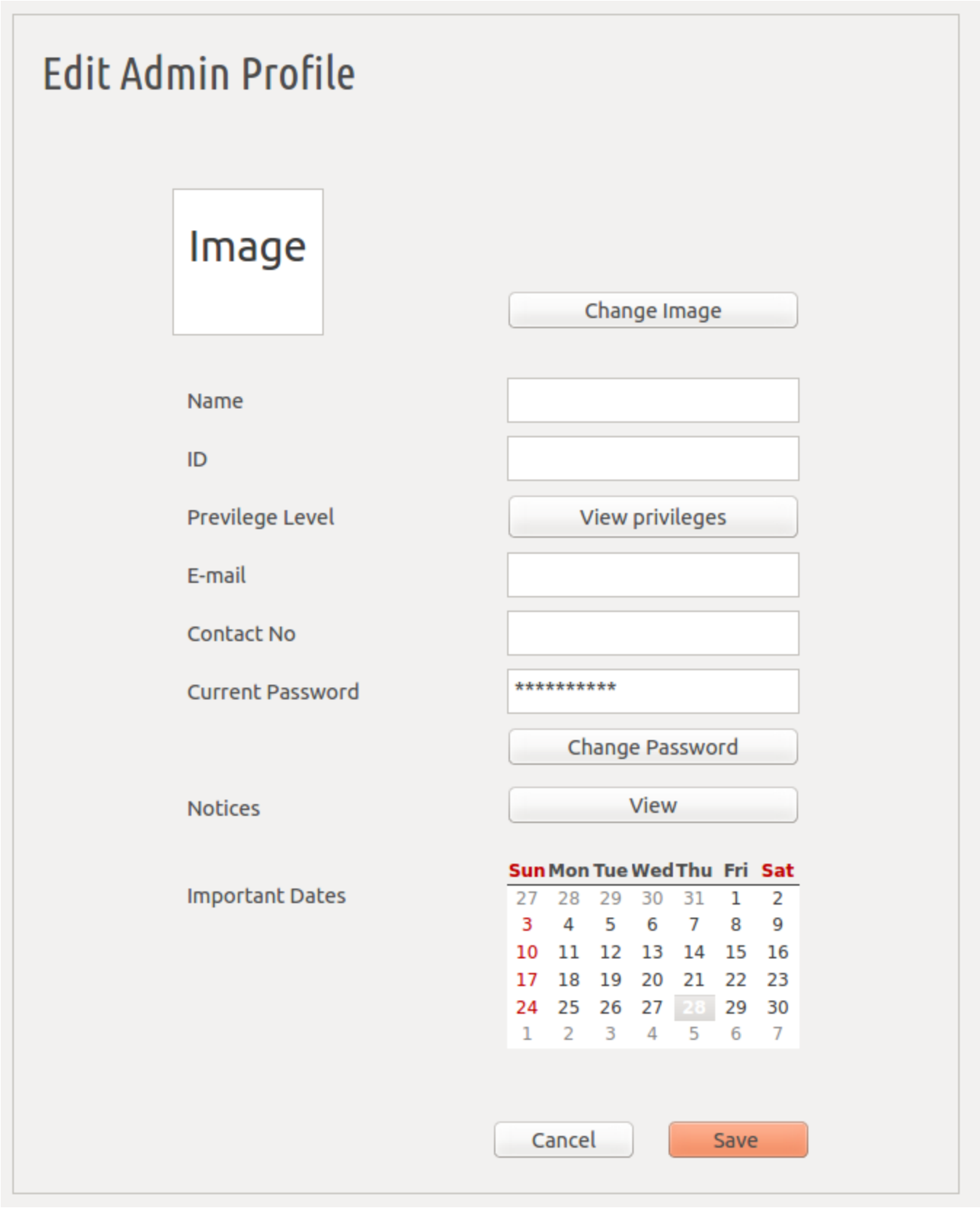


Use case 05	:	Manage users

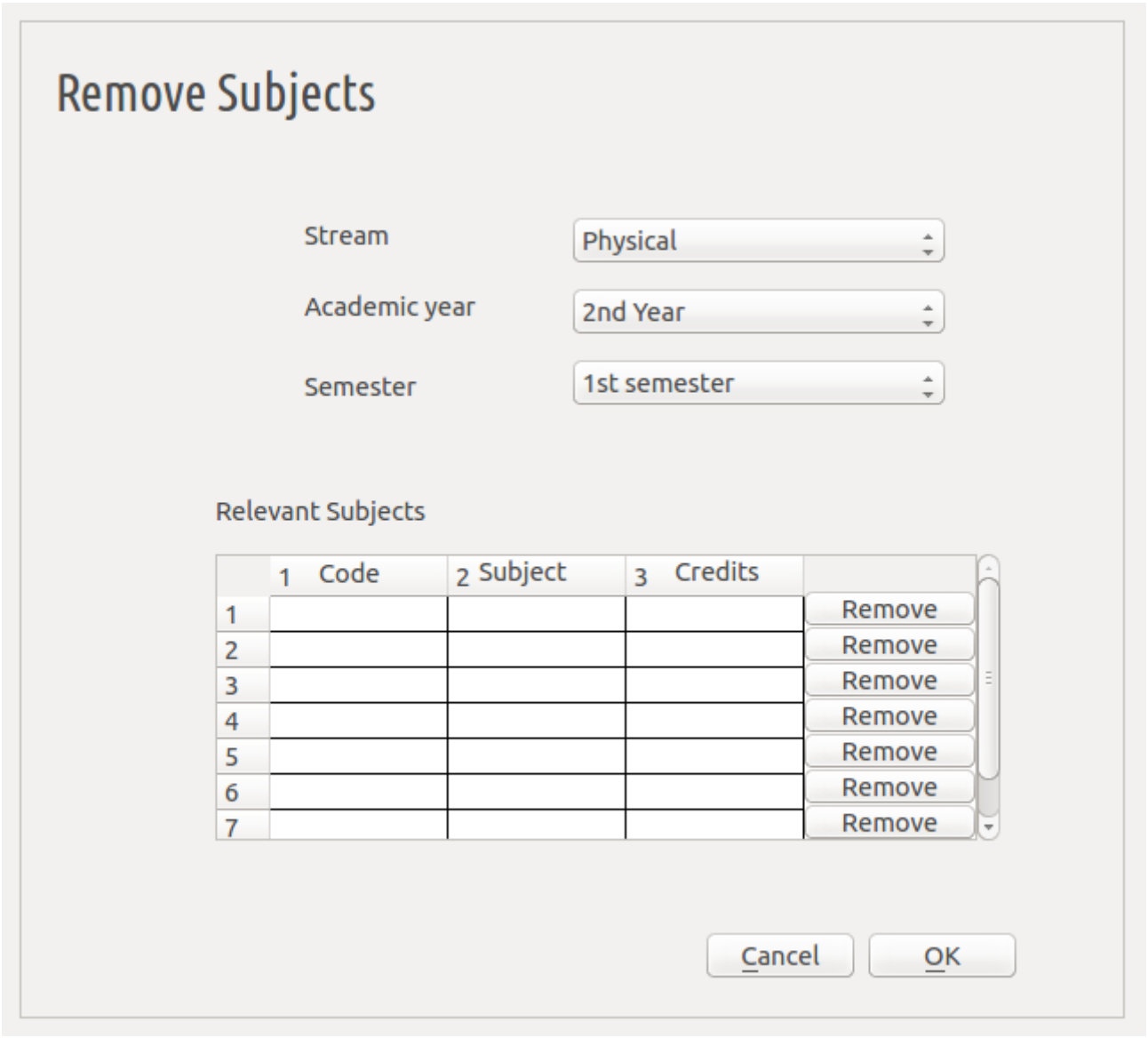




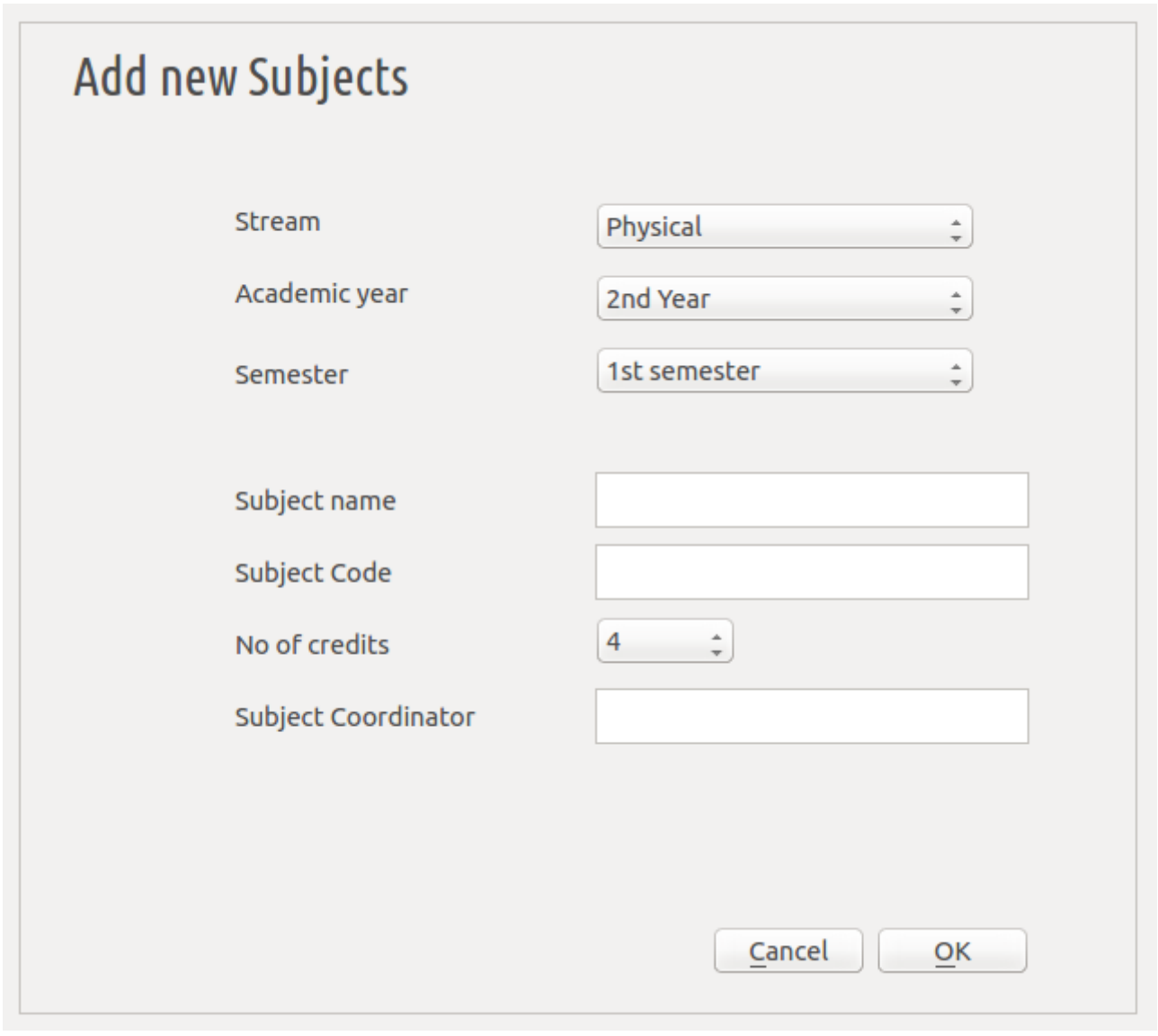
Use case 06	:	Edit profile information



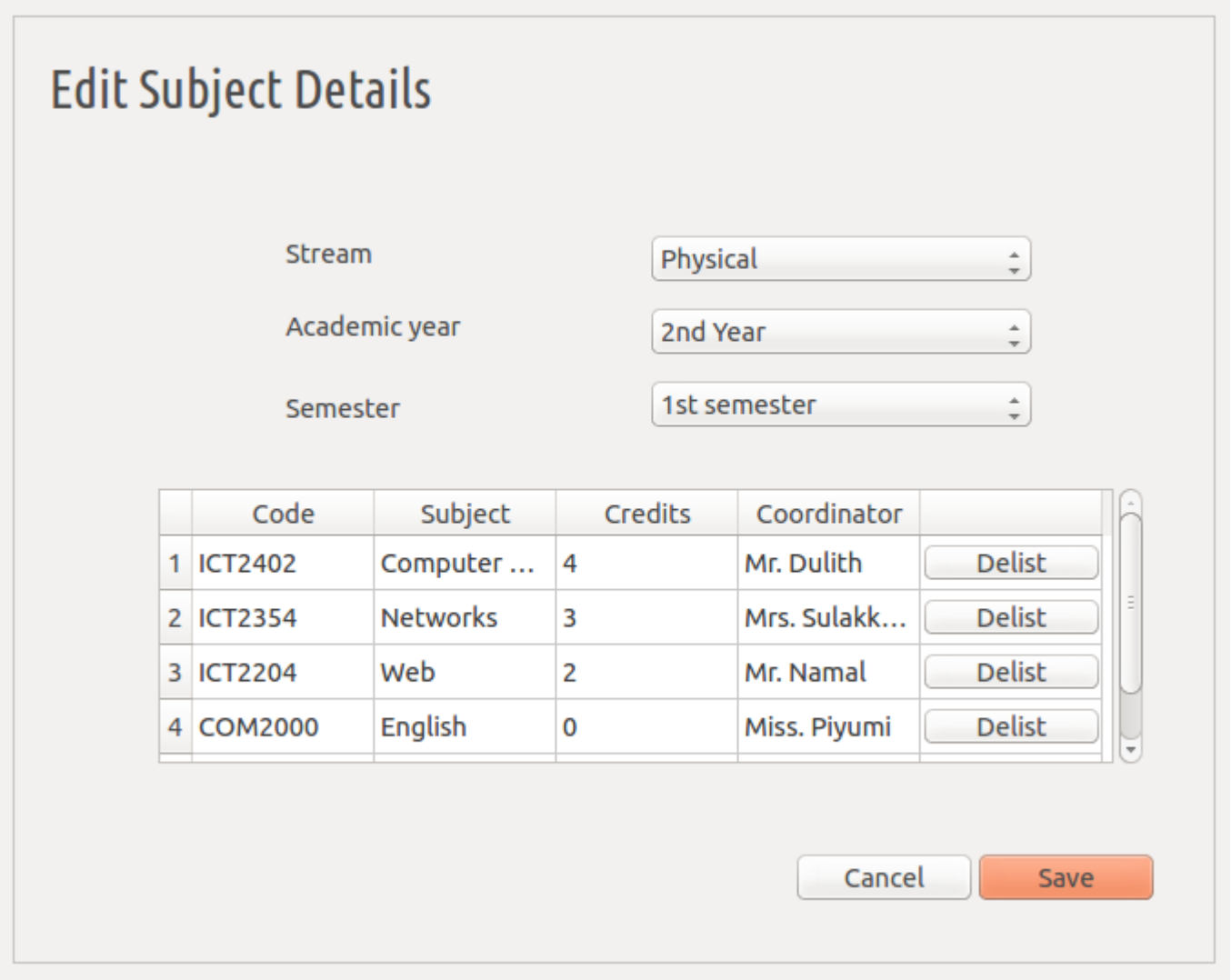
Use case 07	:	Remove selected subjects from student's selections



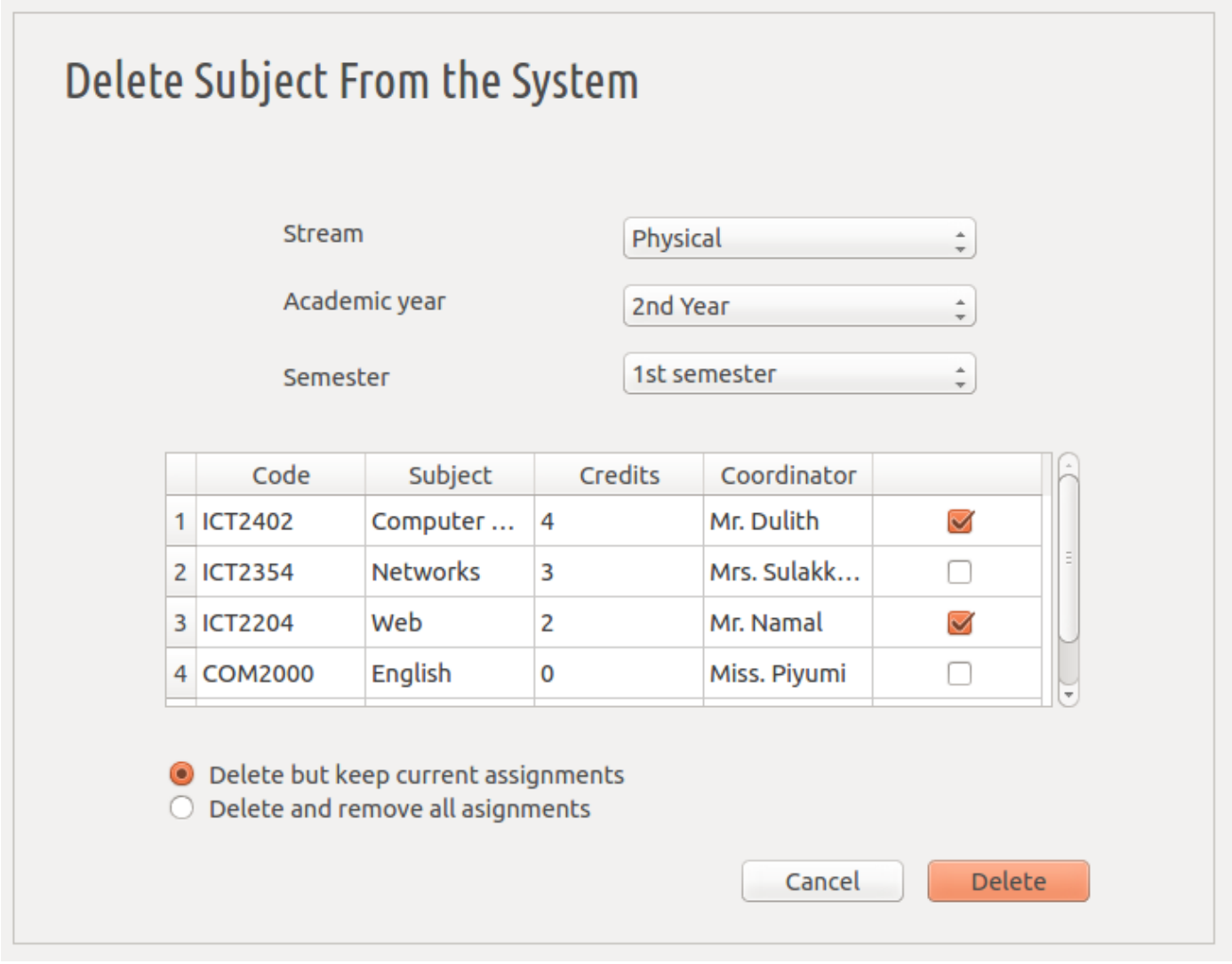
Use case 08 : Create new subjects



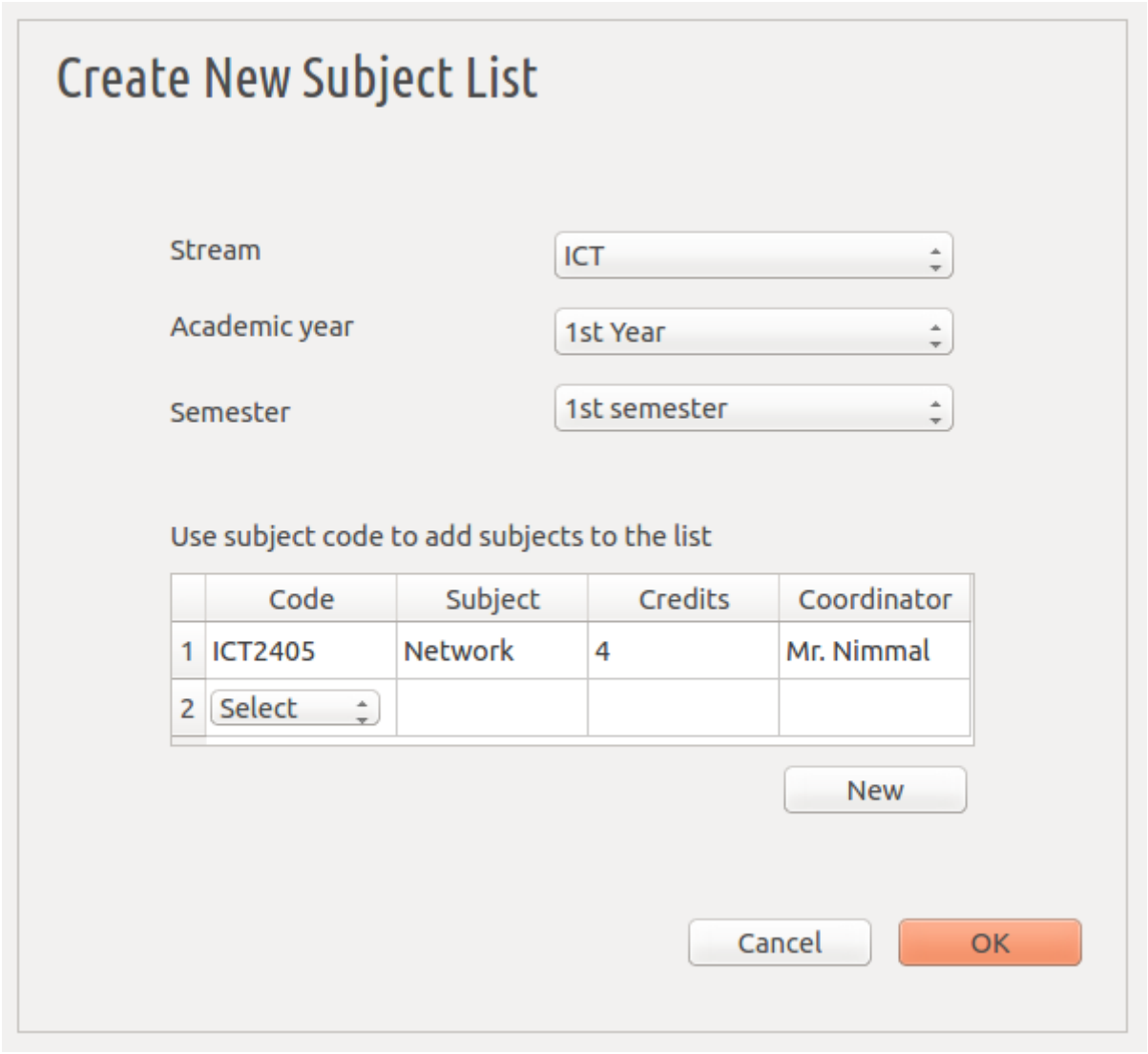
Use case 09	:	Edit existing subjects



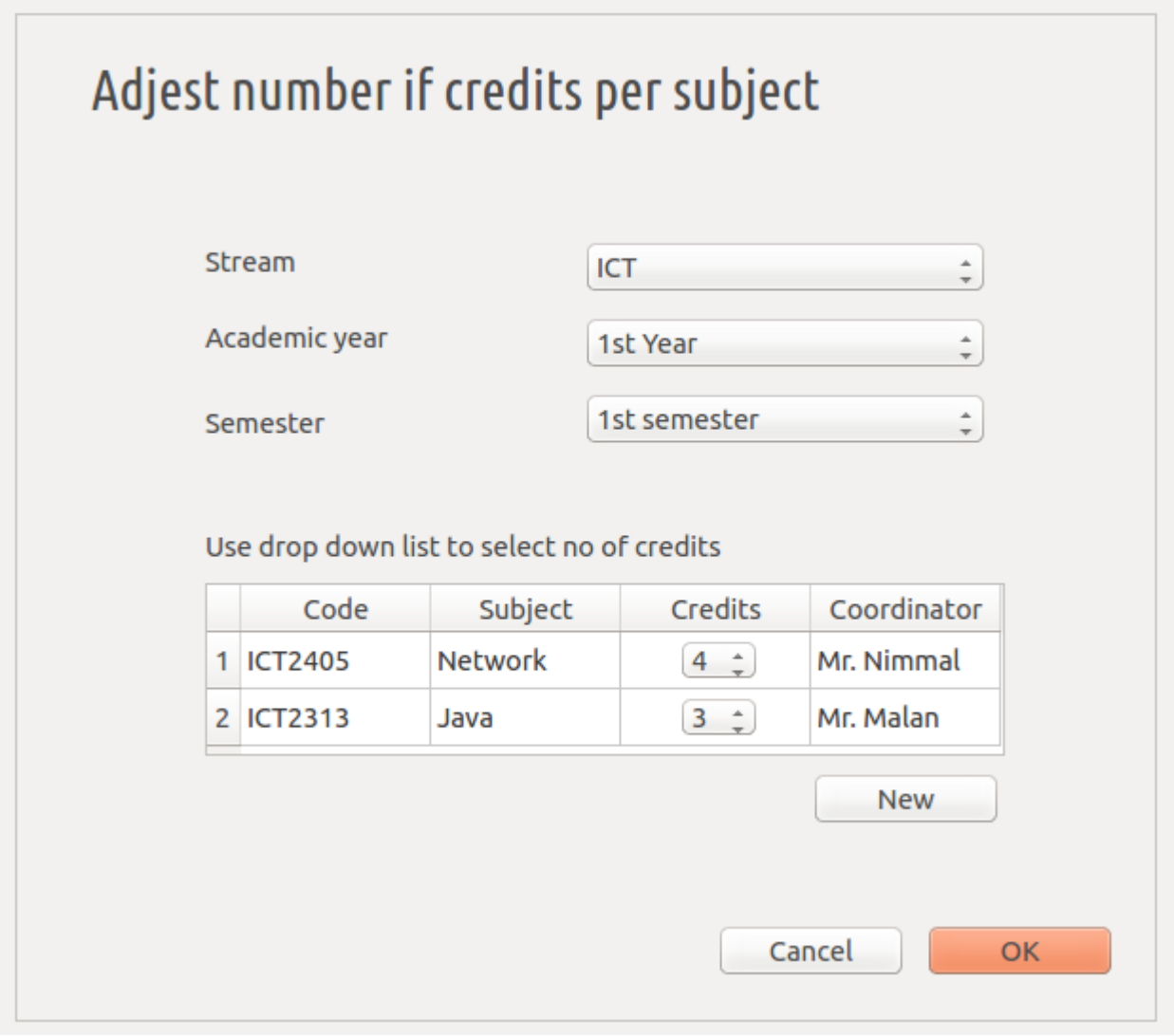
Use case 10	:	Delete subjects



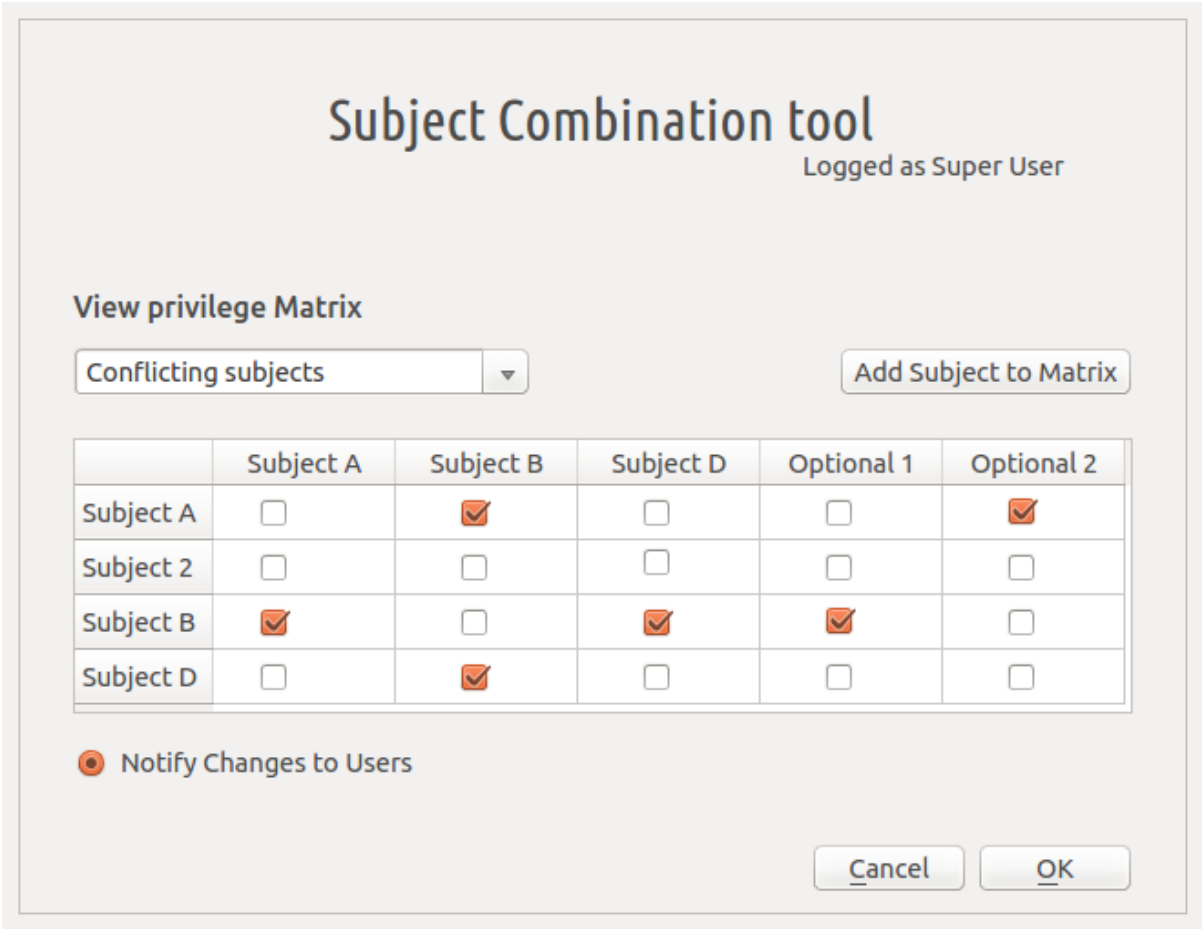
Use case 11	:	Create new subject lists

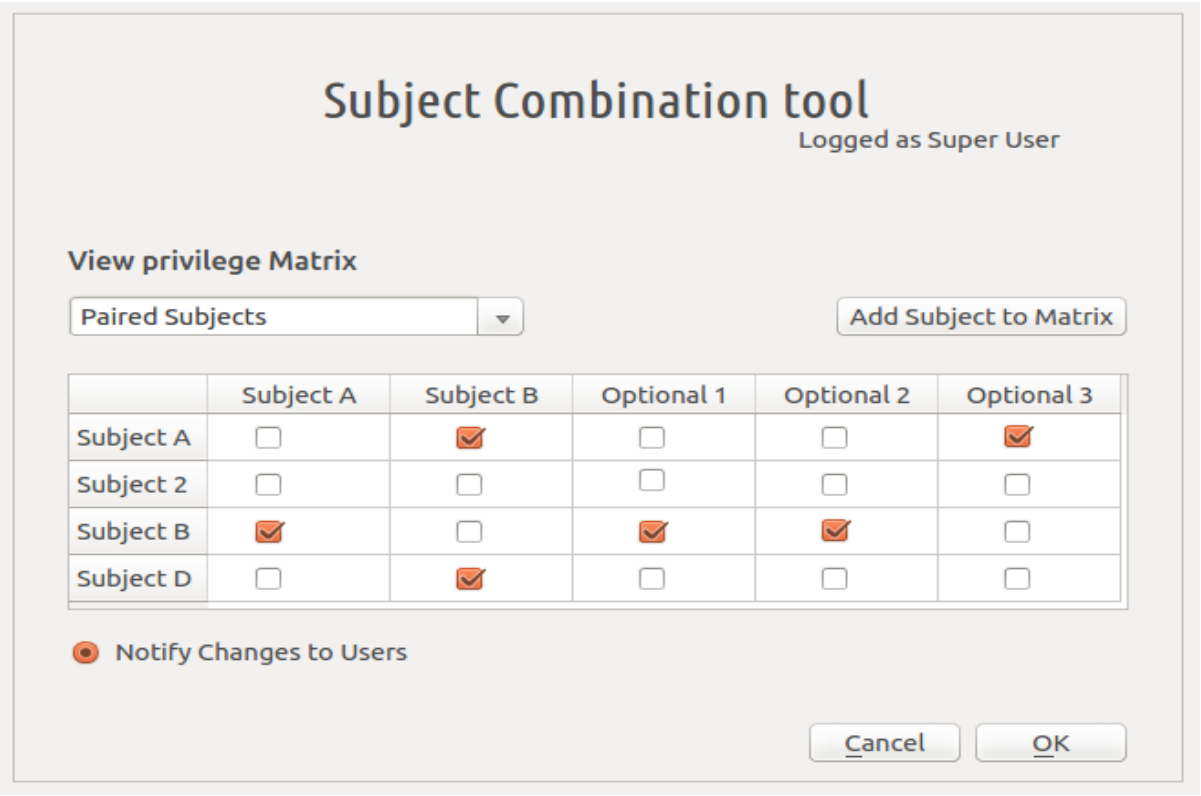


Use case 12	:	Make adjustment of credits per subject.

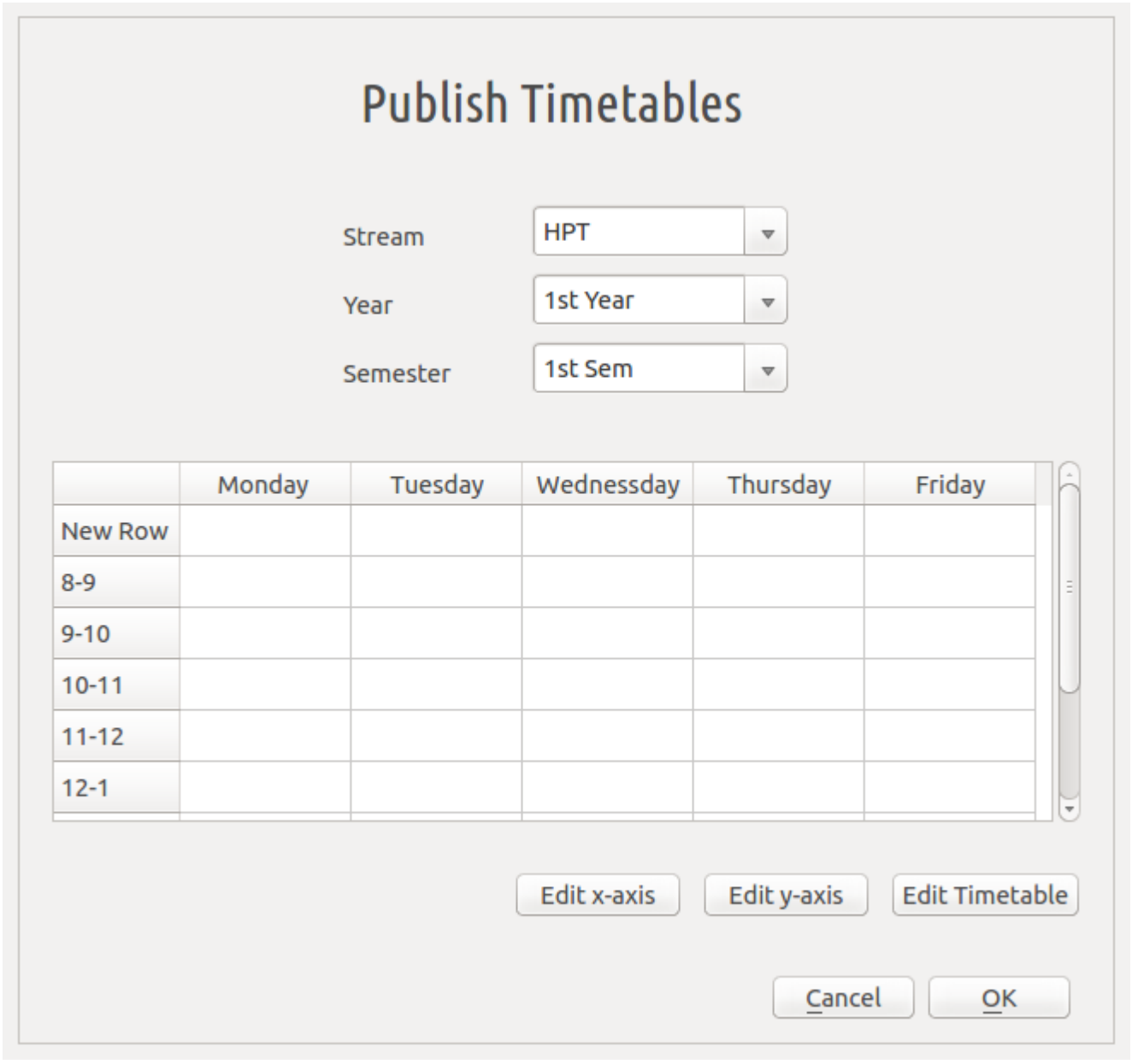


Use case 13 : Subject combination tool

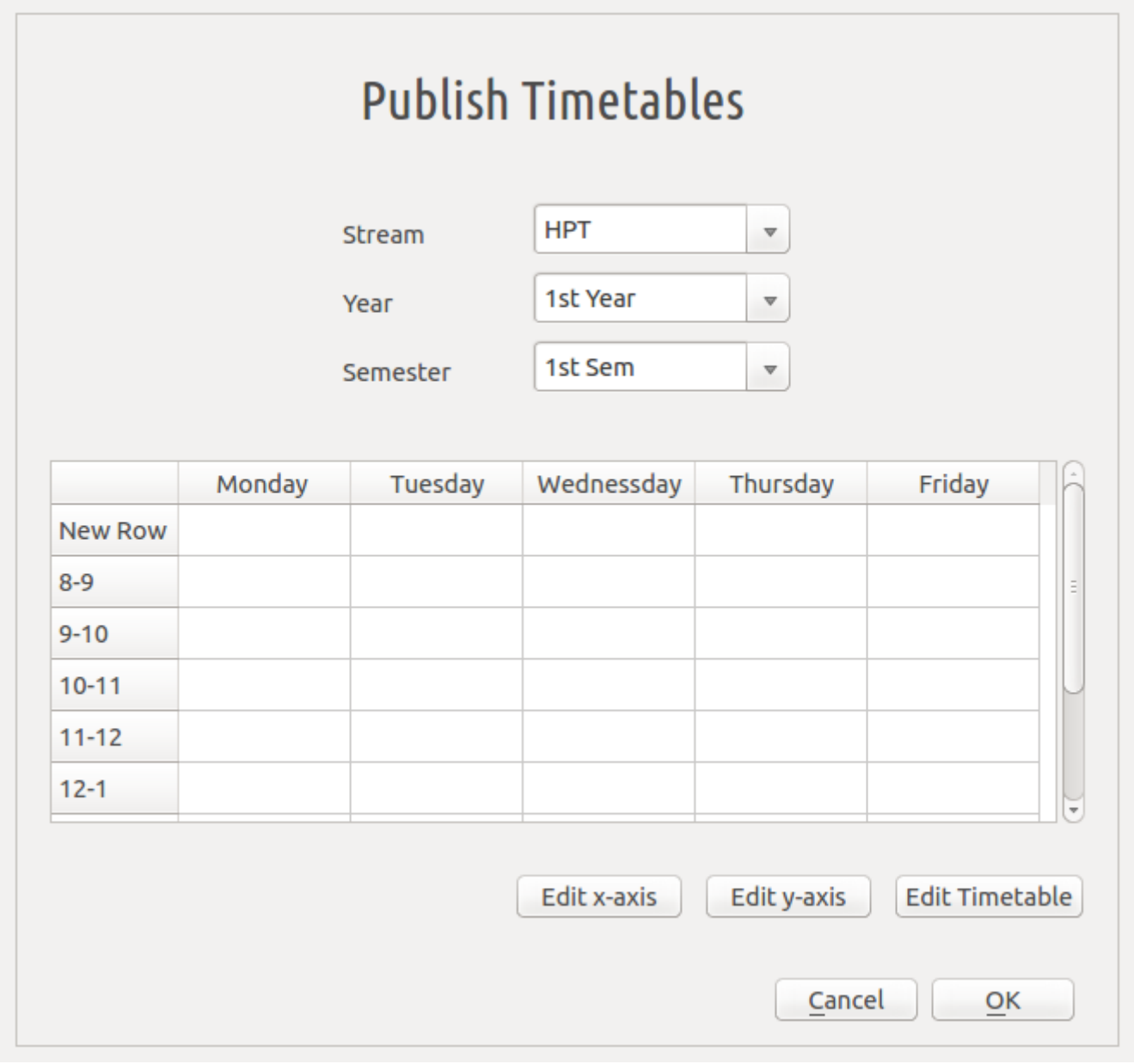




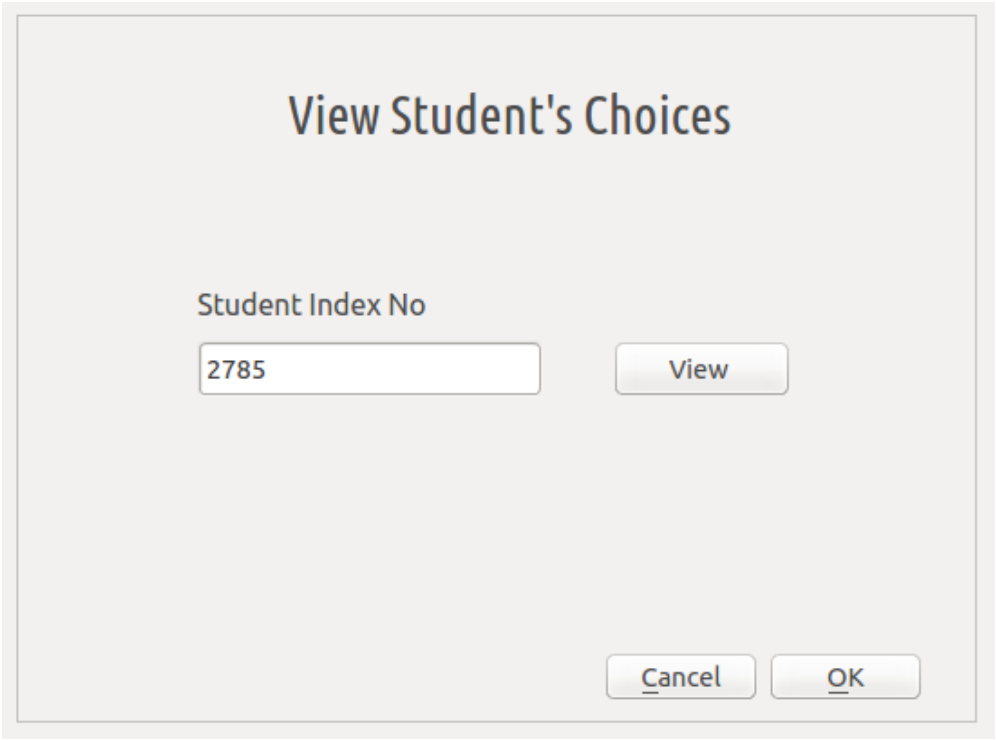
Use case 14	:	Publish time table

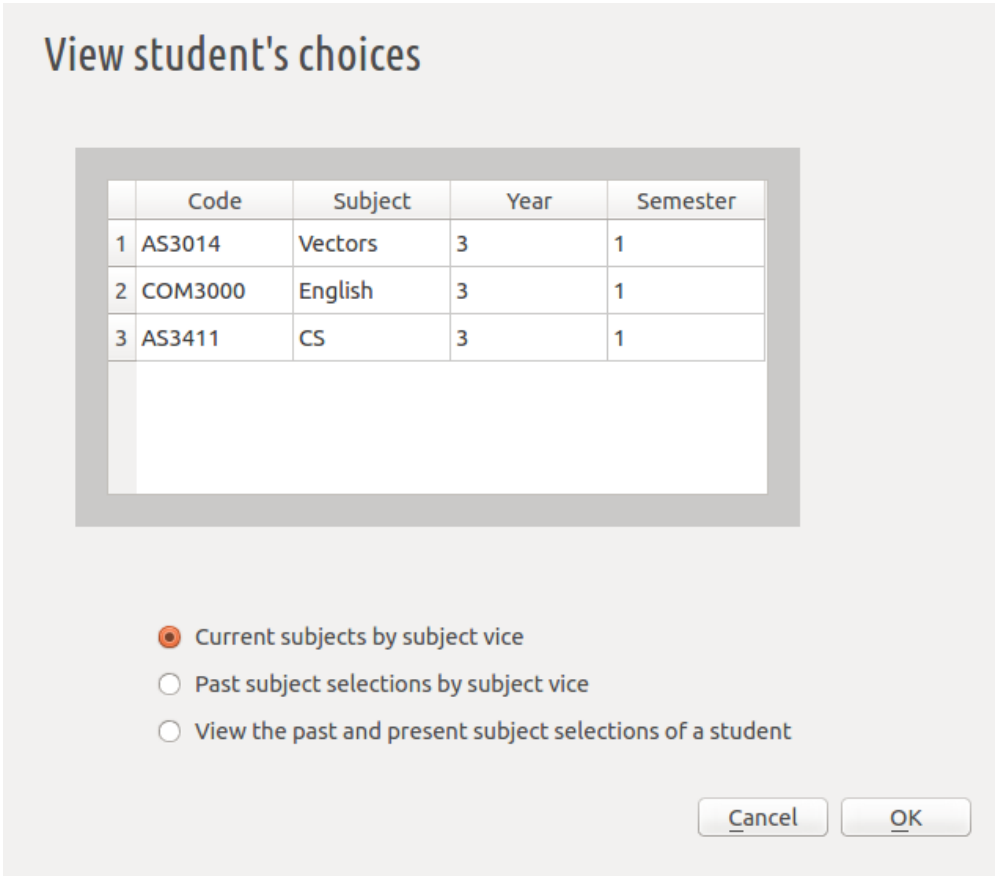


Use case 15	:	Update time table

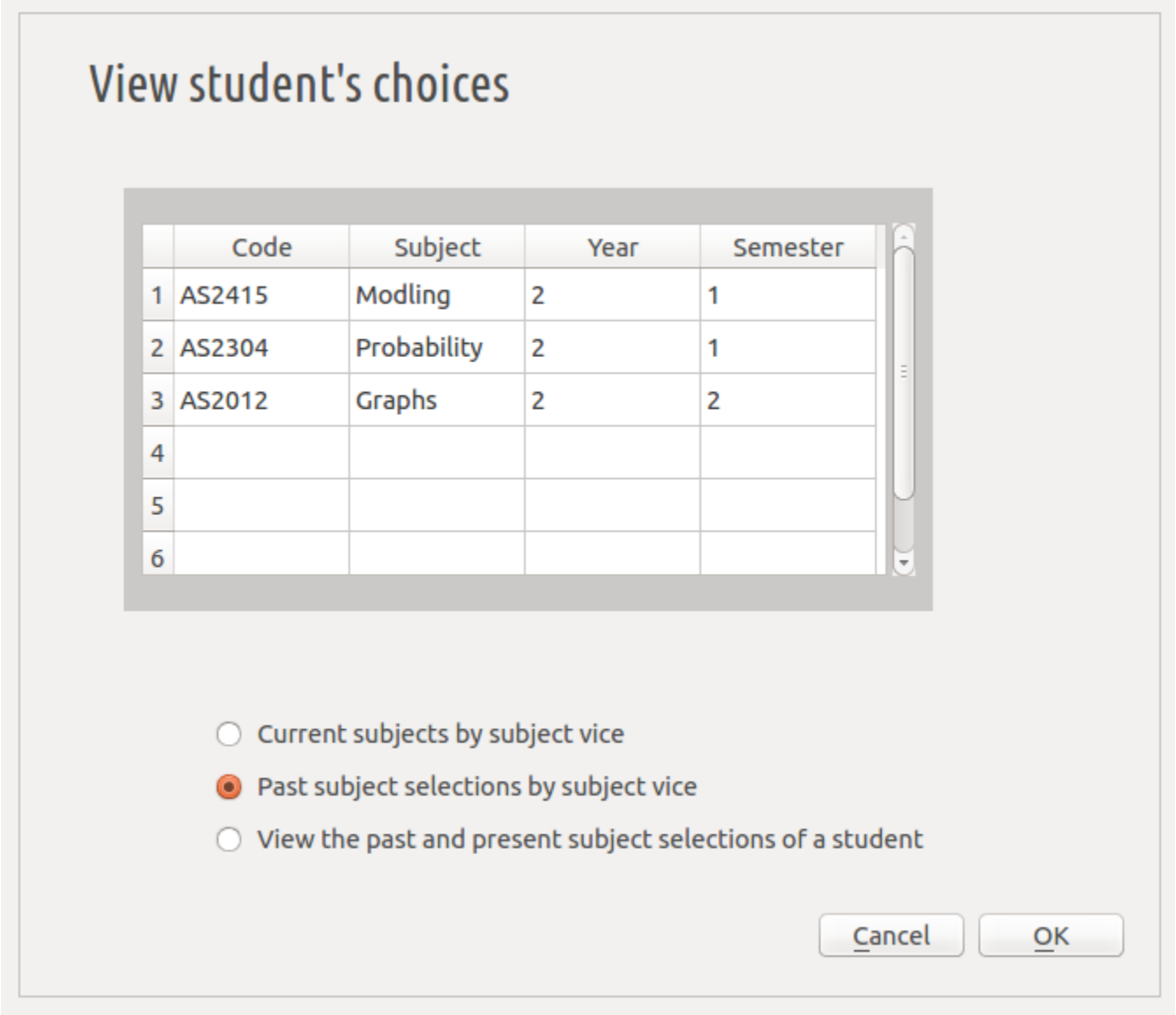


Use case 16	:	Current subjects by subject vice

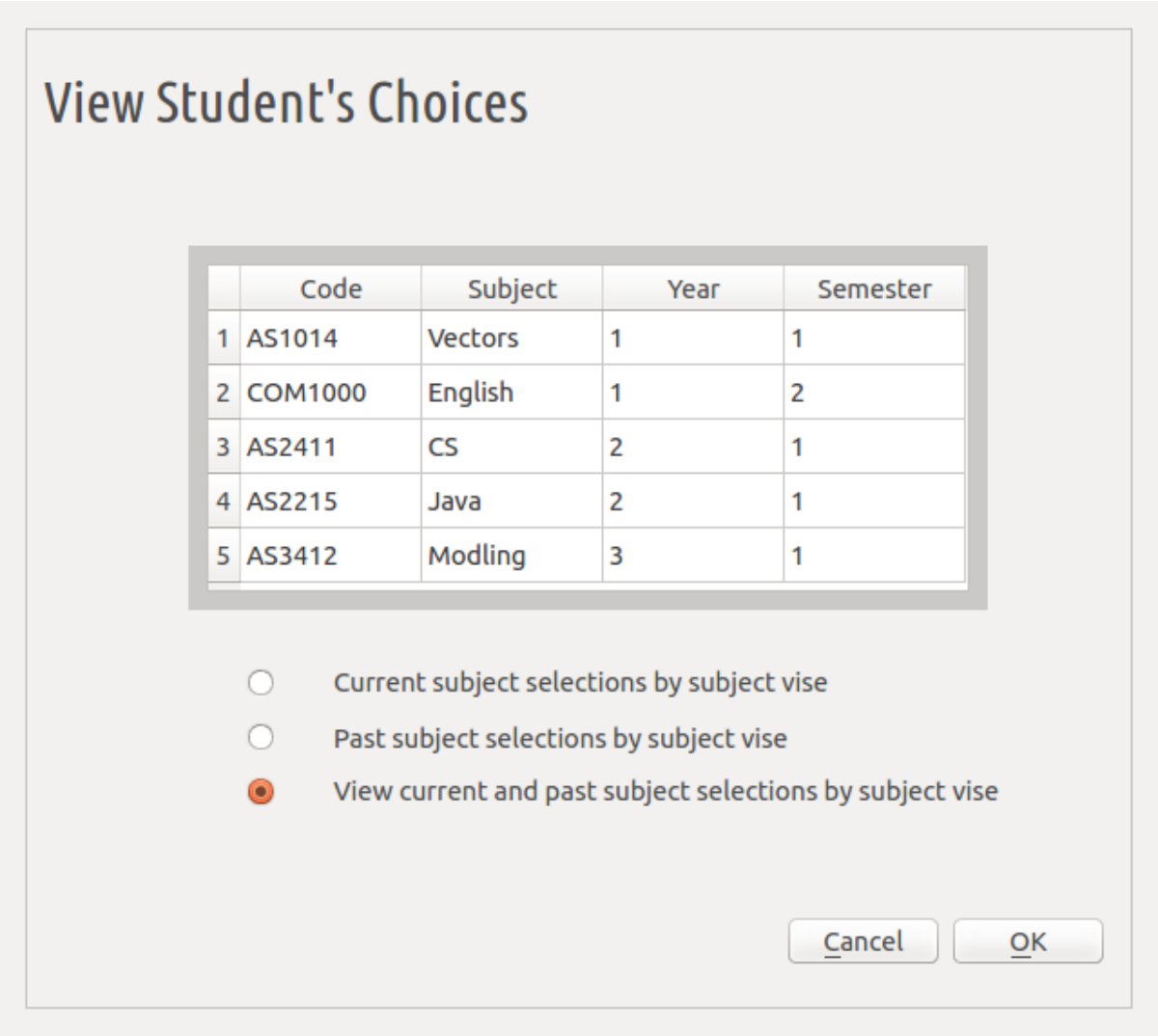




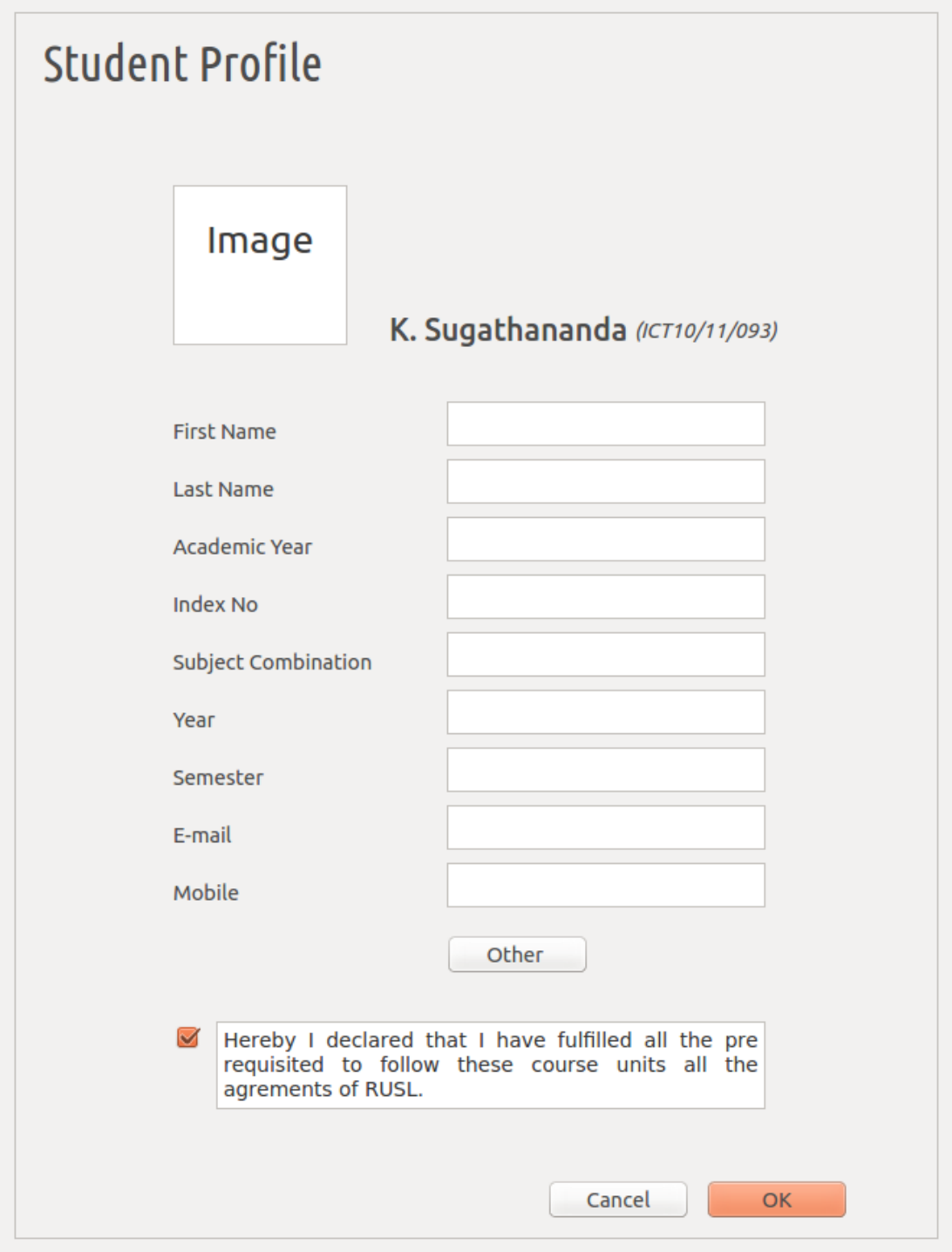
Use case 17	:	Past subject selections by subject vice



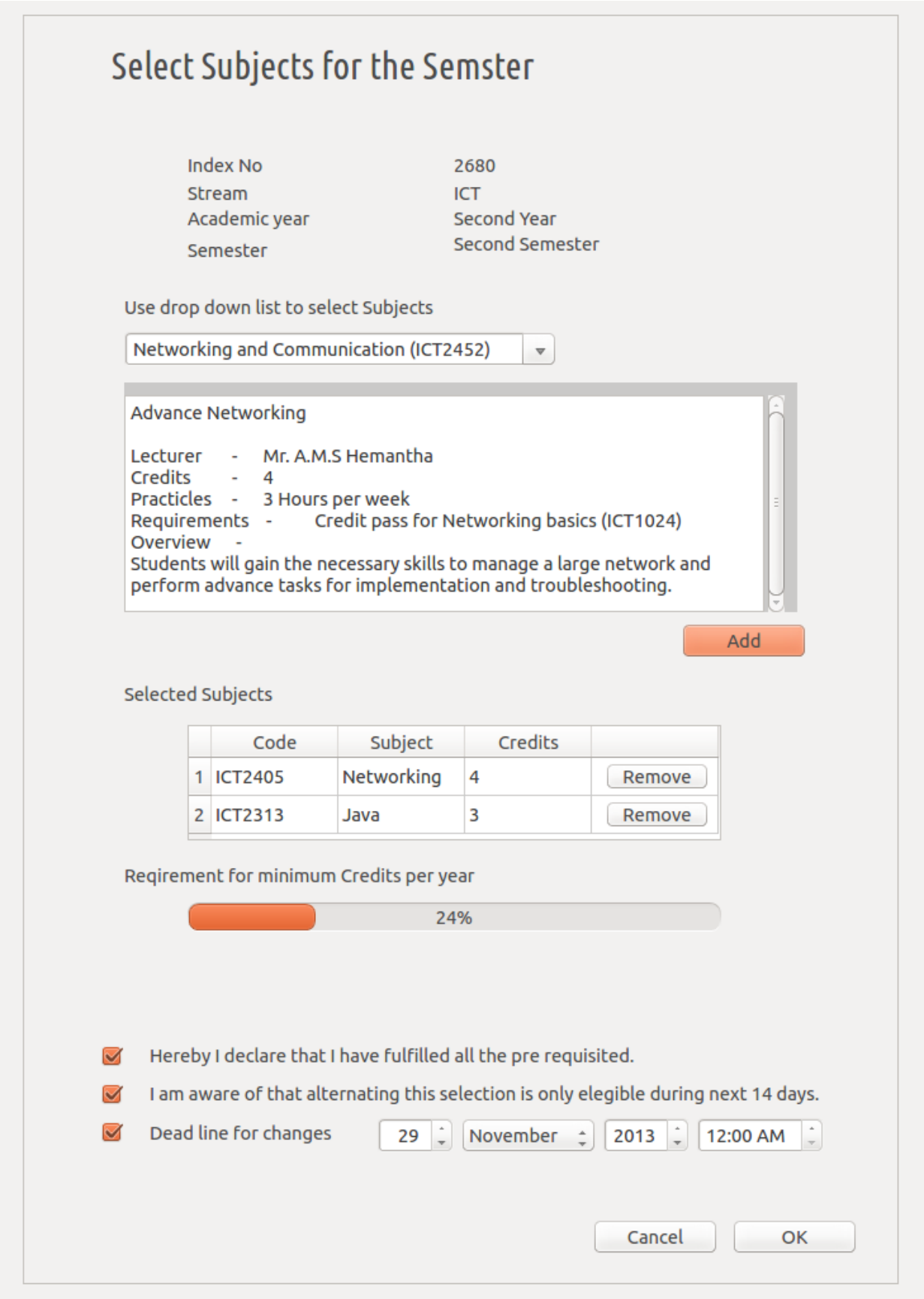
Use case 18	:	View the past and present subject selections of a student



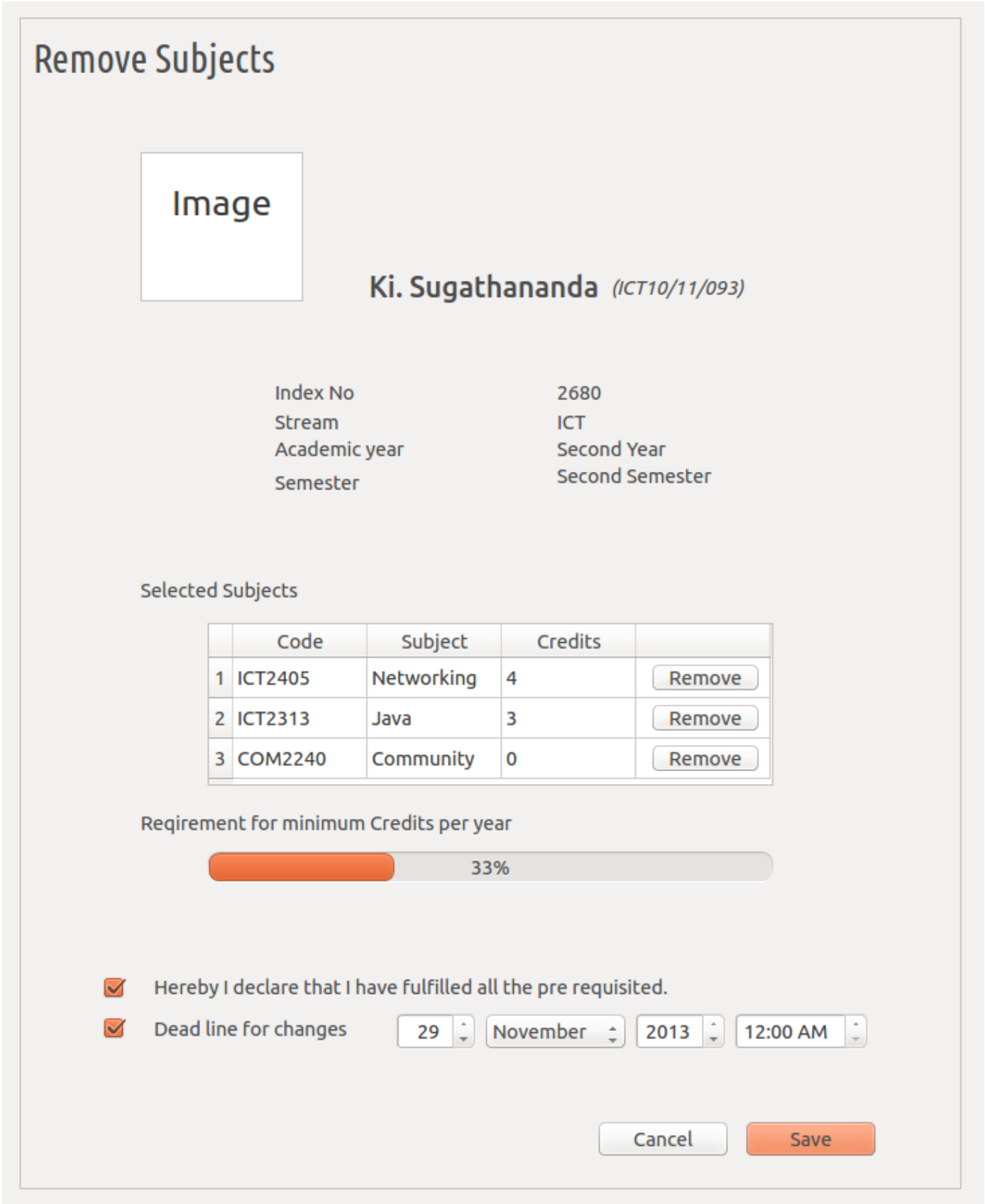
Usecase19	:	Edit student's profile



Use case 20 : Select subjects



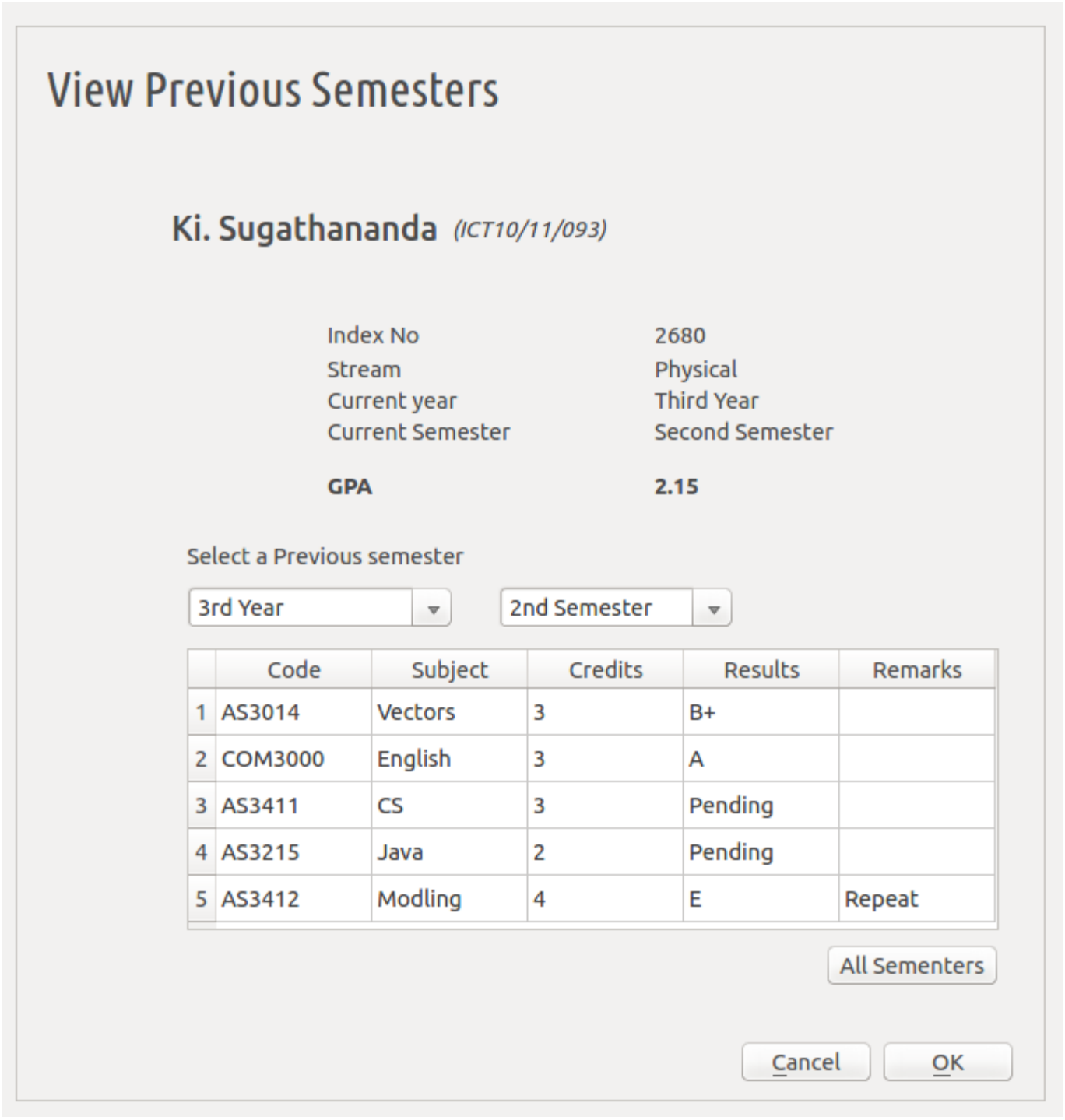
Use case 21 : Remove subjects from the list



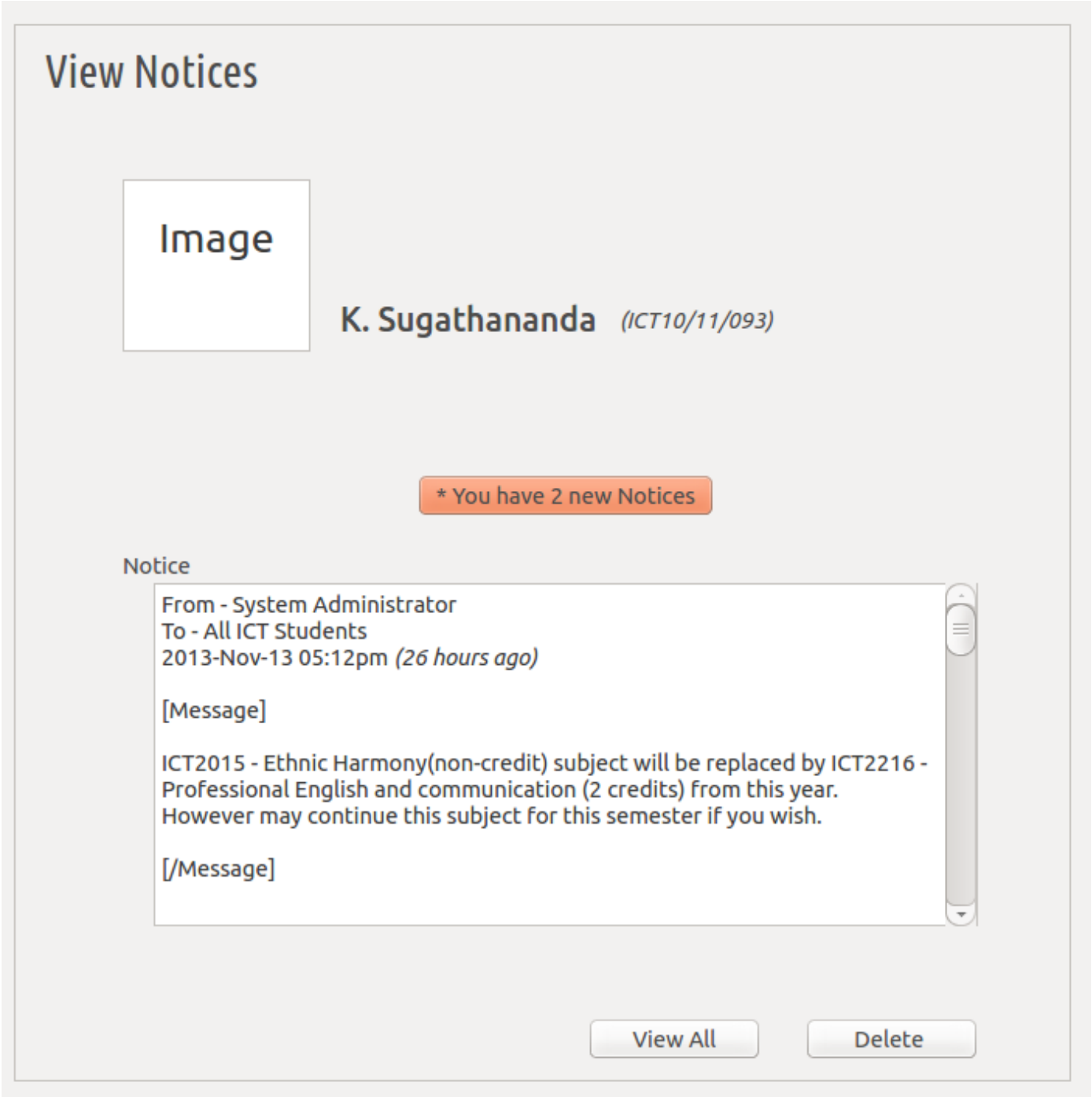
Use case 22 : Save

This use case is for save document details on server.

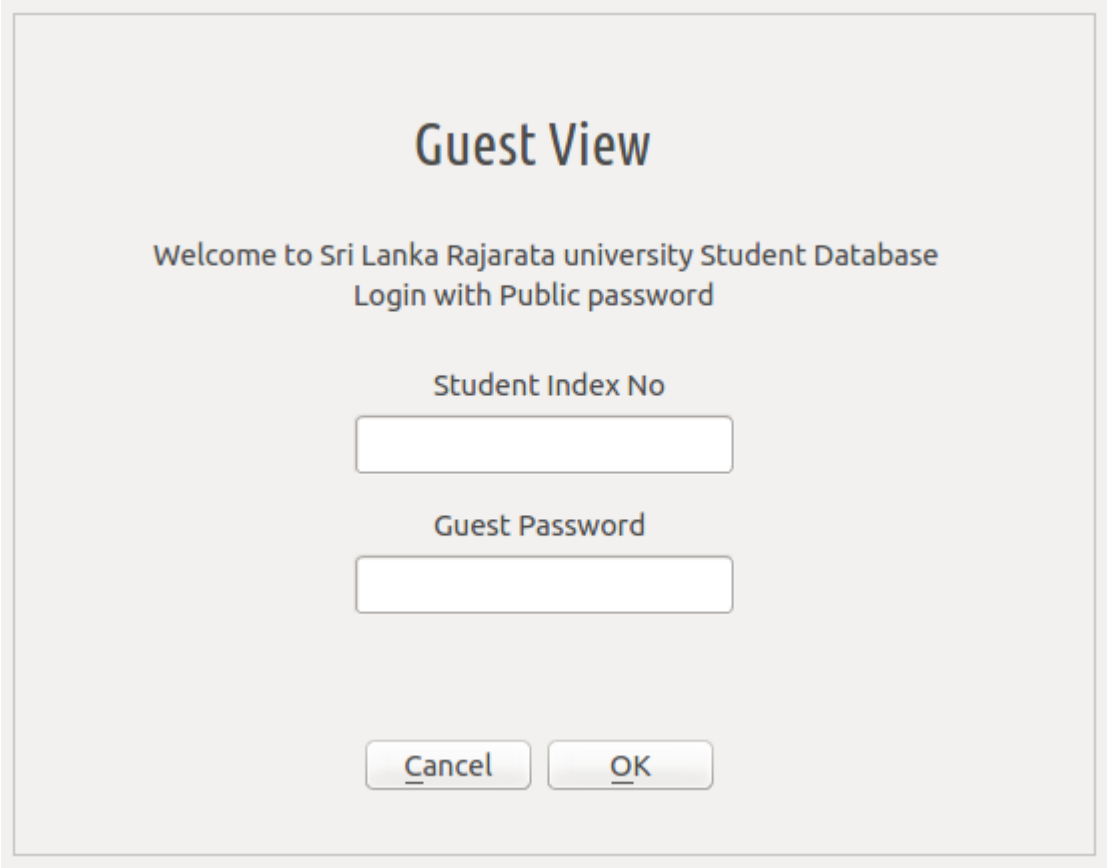
Use case 23 : View previous semester information



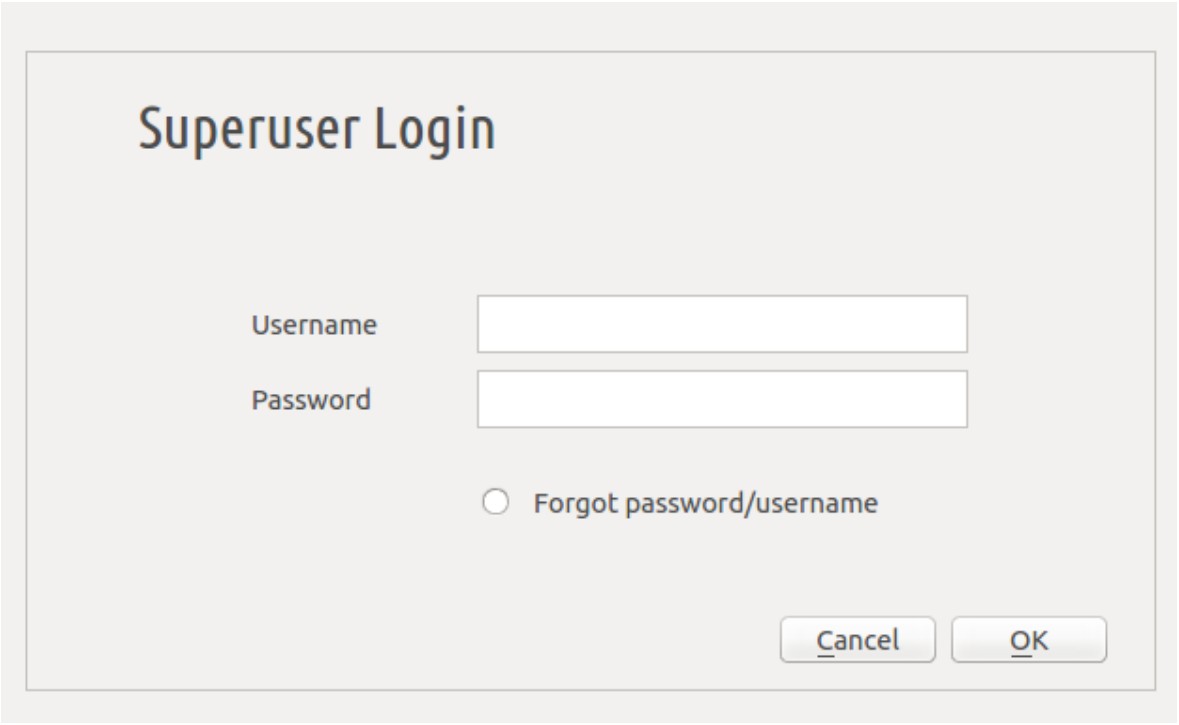
Use case 24	:	View Notices

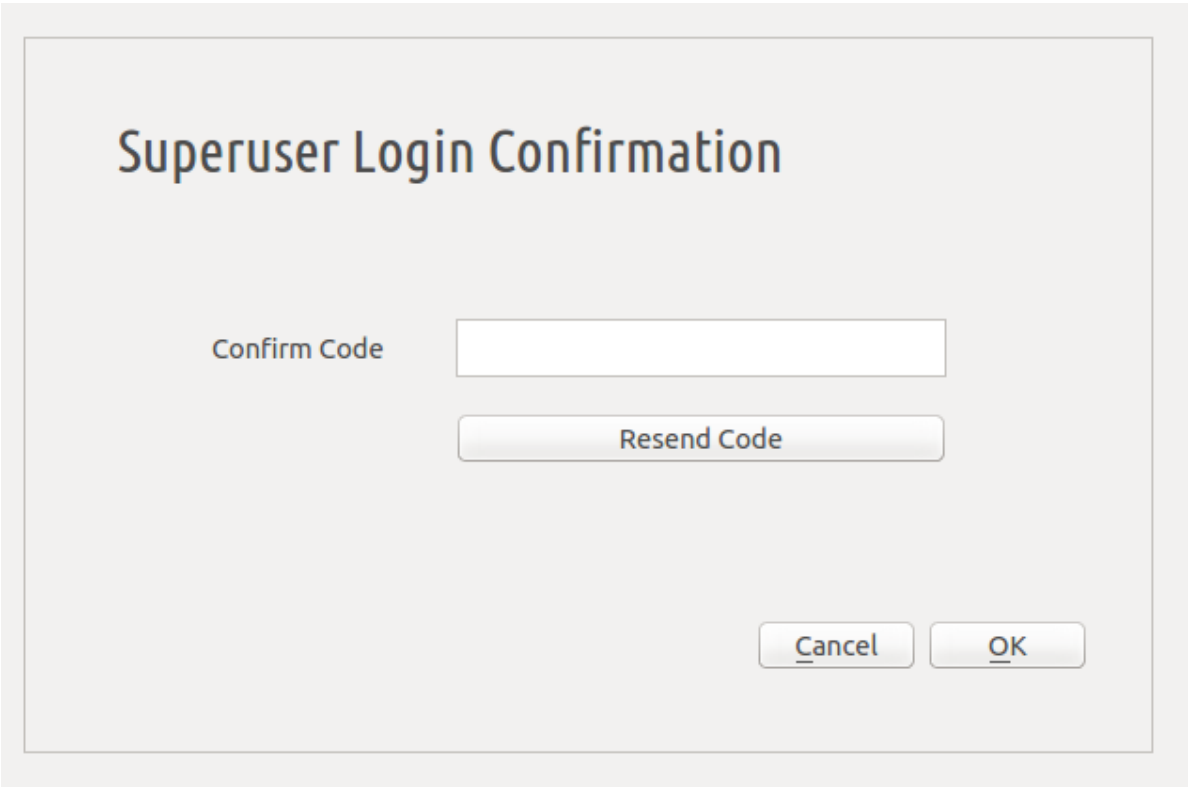


Use case 25	:	Guest view of the student's profile

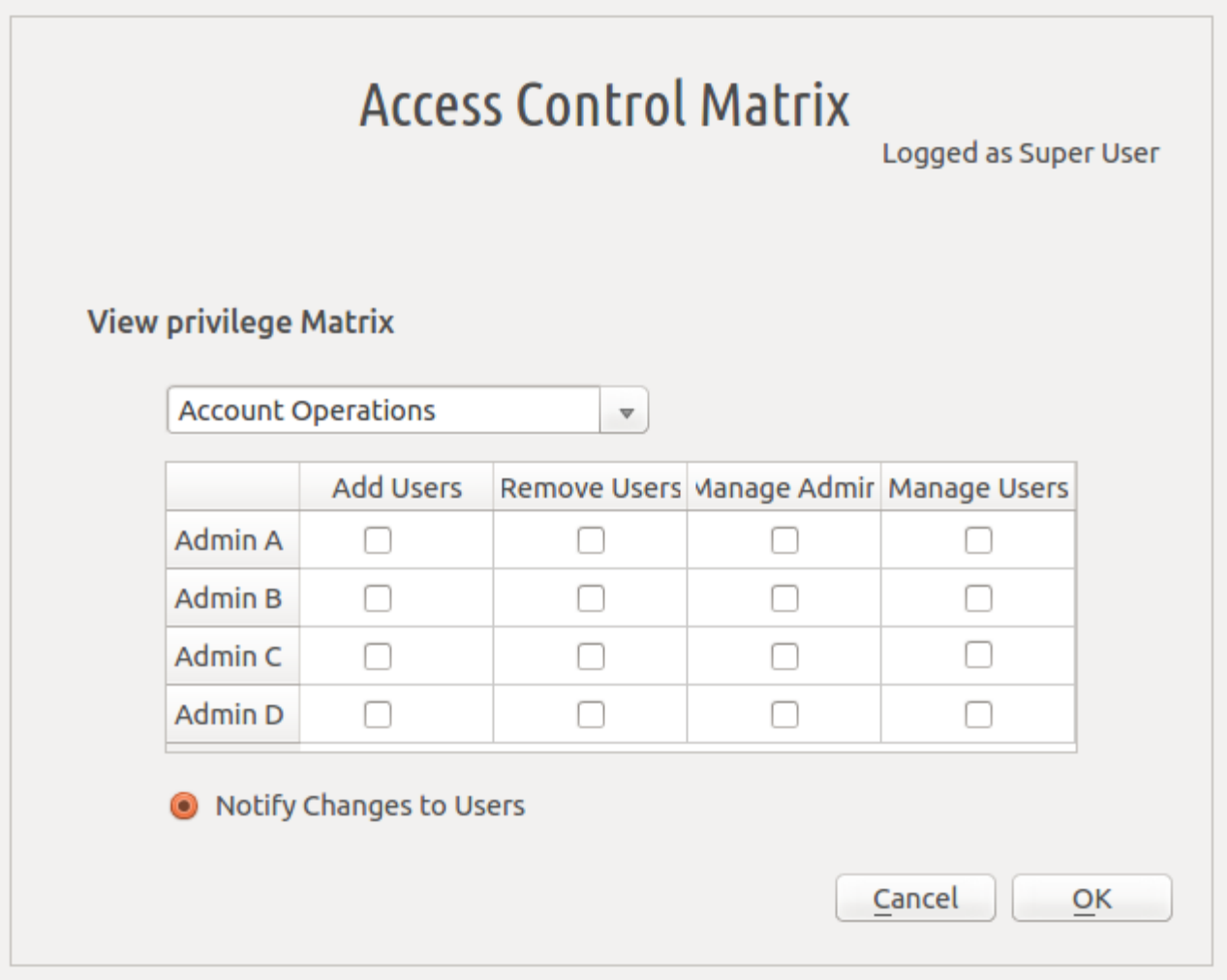


Use case 26	:	Super user confirmation





Use case 27	:	Access Control Matrix



Team name : ***Humming Bird***
Team leader : W.D.R.Y. Jayasundara

Members :

Y.M.Y.T.K Yaparatne	ICT/10/11/050	2687
K.Gunawardana	ICT/10/11/008	2640
K.A.D.C.P. Kaluarachchi	ICT/10/11/057	2650
S.D. Thrimawithana	ICT/10/11/032	2675
W.D.R.Y. Jayasundara	ICT/10/11/013	2648