Cairo University  
Faculty of Computers and Information



**CS251**

**Software Engineering I**

Project Name

Software Design

Team Names

Month & Year

Contents

[Instructions [To be removed] 3](#_Toc434318348)

[Team 3](#_Toc434318349)

[Document Purpose and Audience 3](#_Toc434318350)

[System Models 3](#_Toc434318351)

[I. System Decomposition 3](#_Toc434318352)

[II. Class diagrams 6](#_Toc434318353)

[Important Algorithm 7](#_Toc434318354)

[III. Sequence diagrams 8](#_Toc434318355)

[Class - Sequence Usage Table 9](#_Toc434318356)

[IV. Physical Entity-Relationship Diagram 10](#_Toc434318357)

[V. User Interface Design 10](#_Toc434318358)

[VI. Bonus: State Diagram and OCL 12](#_Toc434318359)

[Ownership Report 12](#_Toc434318360)

[Policy Regarding Plagiarism: 12](#_Toc434318361)

[References 12](#_Toc434318362)

[Authors 12](#_Toc434318363)

# Instructions [To be removed]

* **IMPORTANT. Rename this document to CS251-LeaderID-SDDocument.docx**

**(e.g. CS251-20040752-SDDDocument.docx)**

* **Remove the following notes and any red notes**
* **This document is the template document for your Software Design.**

# Team

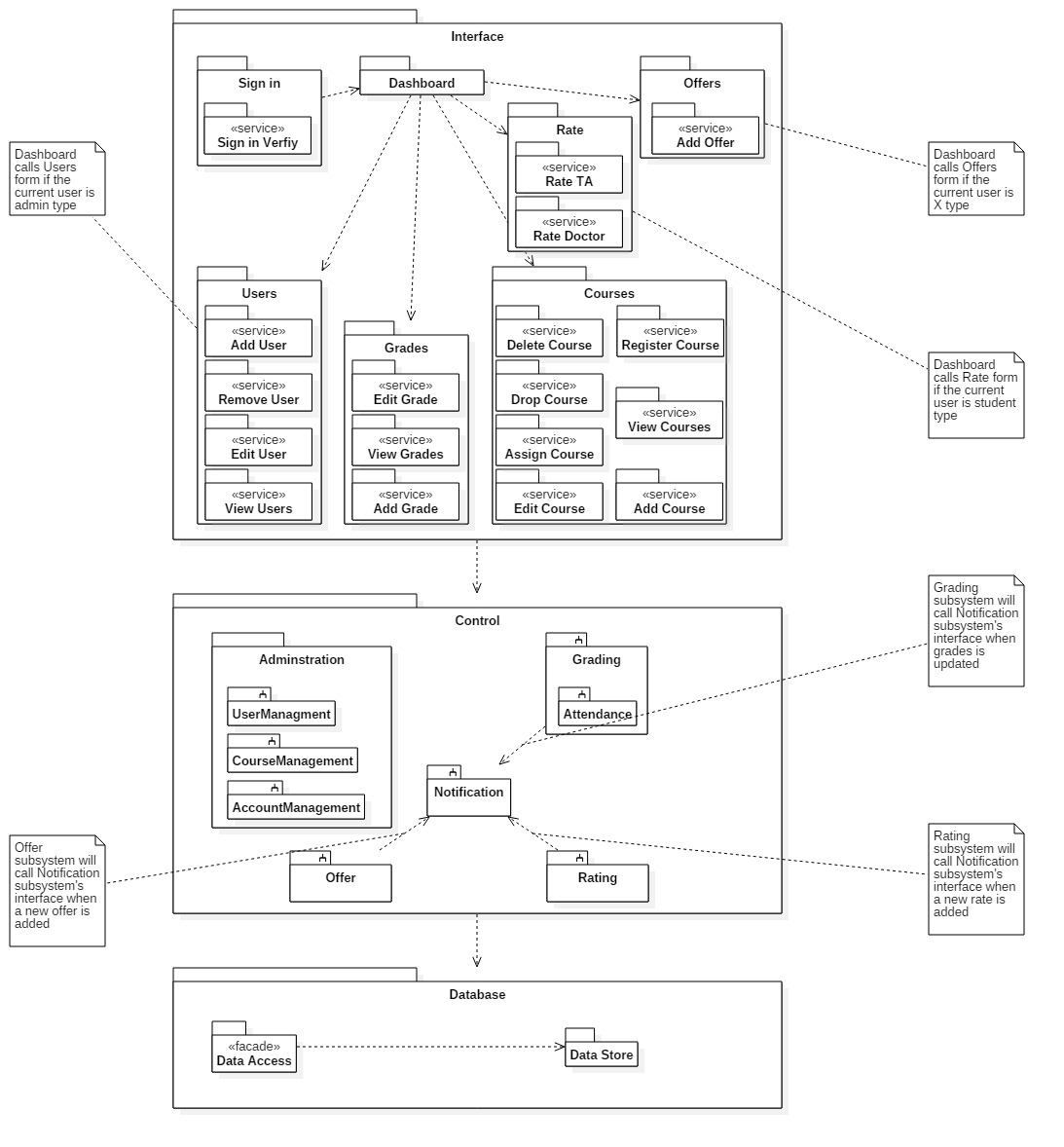
|  |  |  |  |
| --- | --- | --- | --- |
| **ID** | **Name** | **Email** | **Mobile** |
|  | 1st name is team leader |  |  |
|  |  |  |  |
|  |  |  |  |

# Document Purpose and Audience

* **Any document should tell the reader 2 things: What is this document? Who is expected to read it?**
* **Write in simple notes: what this document is.**
* **List the target audience to read this document (e.g. CEO? Project Manager? Customer...?)**

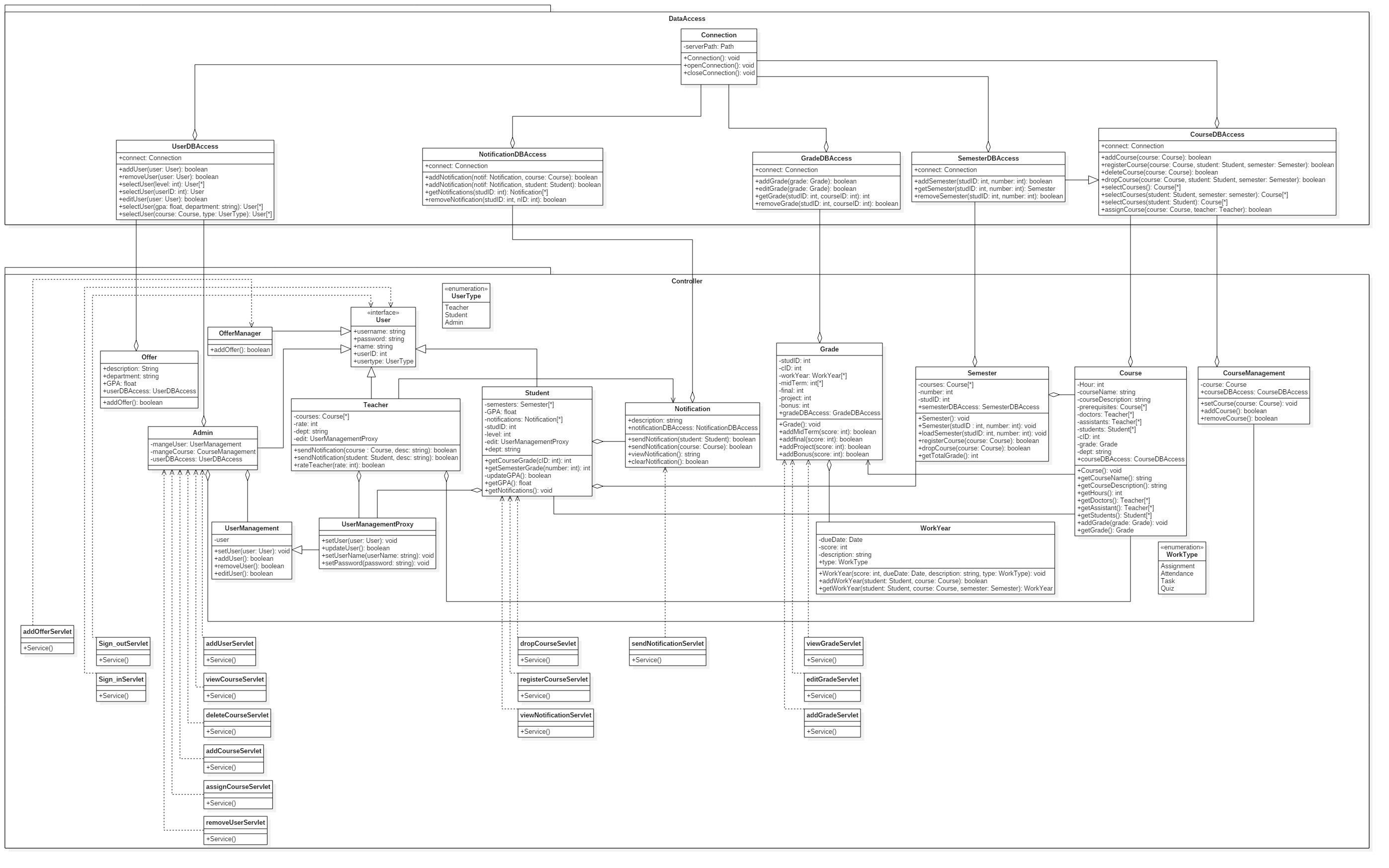
# System Models

## I. System Decomposition



## 

## II. Class diagrams



**List down your classes and describe them**

| **Class ID** | **Class Name** | **Subsystem ID** | **Description & Responsibility** |
| --- | --- | --- | --- |
| 0 | UserDBAccess | 2 | Access the database , Retrieve , modify the user tables in the database |
| 1 | NotificationDBAccess | 2 | Access the database , Retrieve , modify the Notification tables in the database |
| 2 | GradeDBAccess | 2 | Access the database , Retrieve , modify the grades tables in the database |
| 3 | SemesterDBAccess | 2 | Access the database , Retrieve , modify the semester tables in the database |
| 4 | CourseDBAccess | 2 | Access the database , Retrieve , modify the course tables in the database |
| 5 | Offer | 1 | For adding offers by the Offer Administrator |
| 6 | User | 1 | Interface for the usertypes of the systems |
| 7 | Admin | 1 | For managing the Admin responsibilities |
| 8 | Teacher | 1 | For managing the Teacher responsibilities |
| 9 | Student | 1 | For managing the Student responsibilities |
| 10 | Notification | 1 | For managing the notification entity instantiated by the student, teacher and admin |
| 11 | Grade | 1 | For managing the grade entity instantiated by the course |
| 12 | Semester | 1 | For managing the semester entity instantated as a list in the Student |
| 13 | Course | 1 | For managing the course entity instantiated as a list by the semester |
| 14 | UserManagement | 1 | For managing users by the admin |
| 15 | CourseManagement | 1 | For managing courses by the admin |
| 16 | UserManagementProxy | 1 | Proxy for the user to edit their profile |
| 17 | WorkYear | 1 | Work subject for the grades |
| 18 | OfferManager | 1 | Adding offers to students |
| 19 | Connection | 2 | For opening the connection between web application and database server |

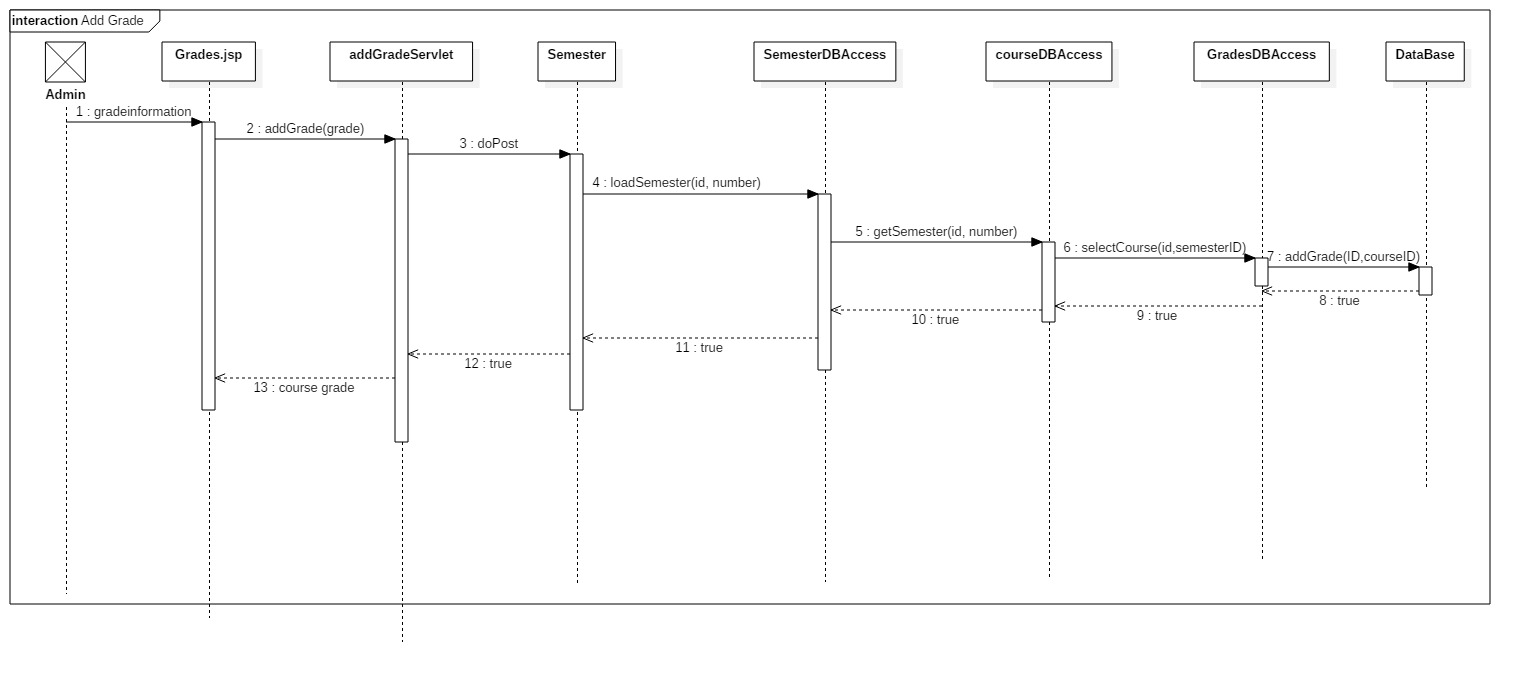
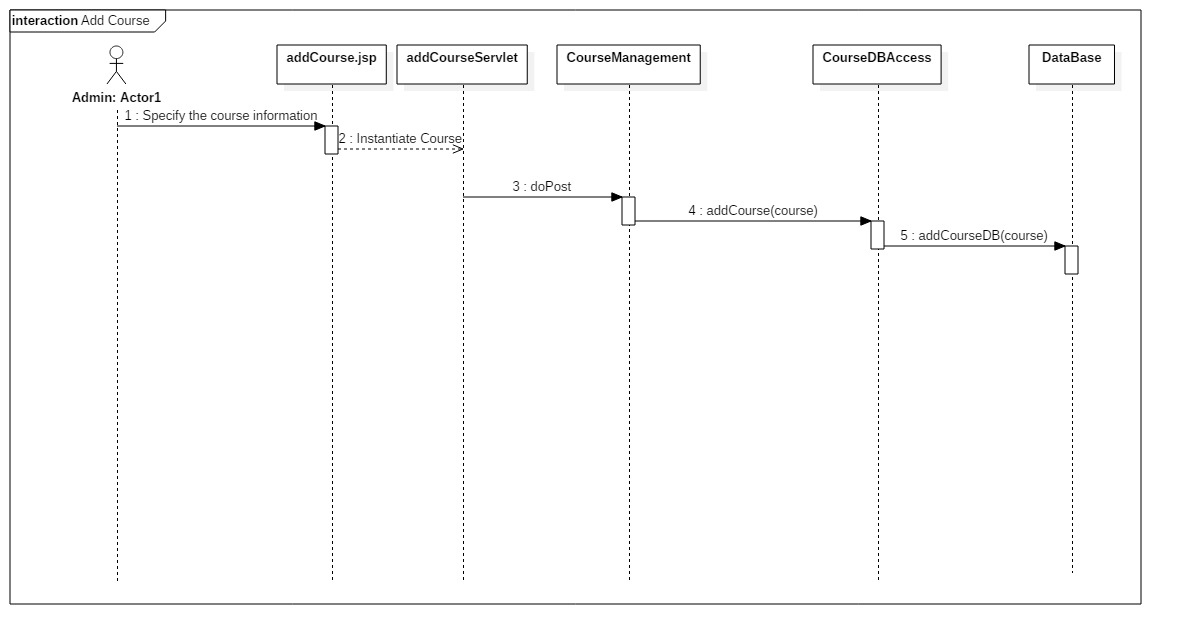
* **In the above table make sure that each class belongs to a subsystem.**
* **In the above table ALL classes should belong to subsystems. And each subsystem should at least contain one class.**

### Important Algorithm

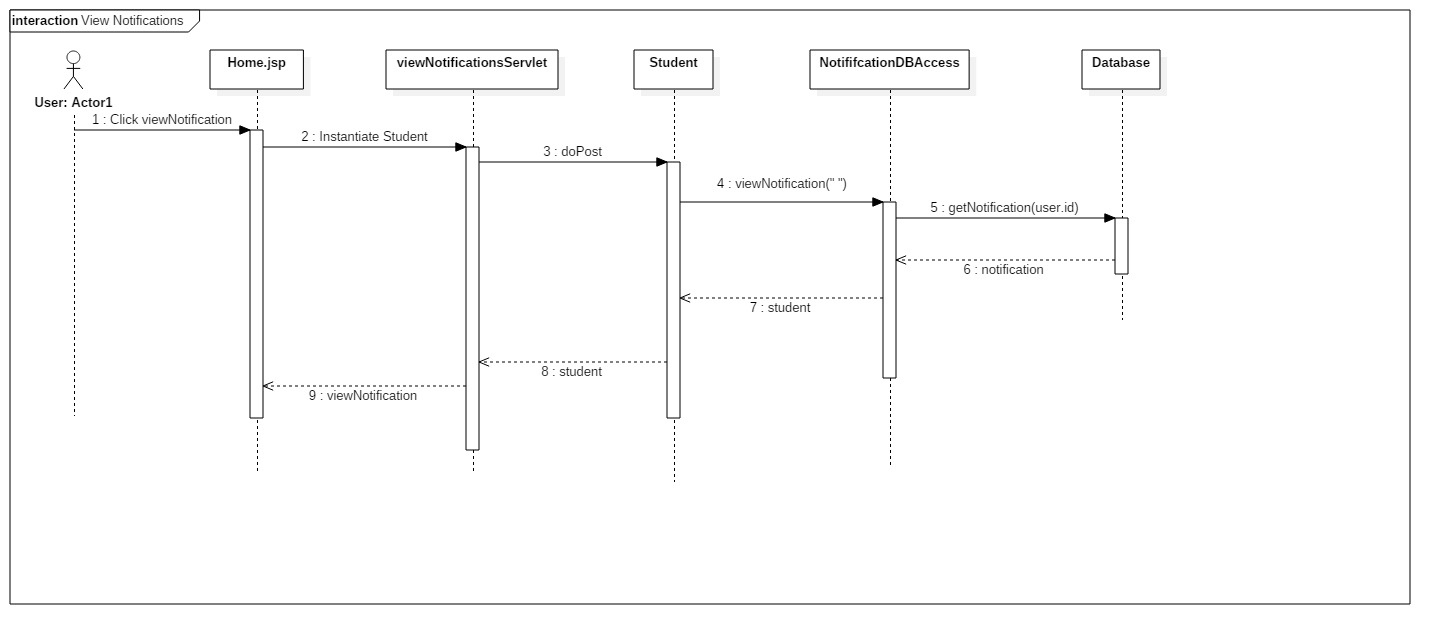
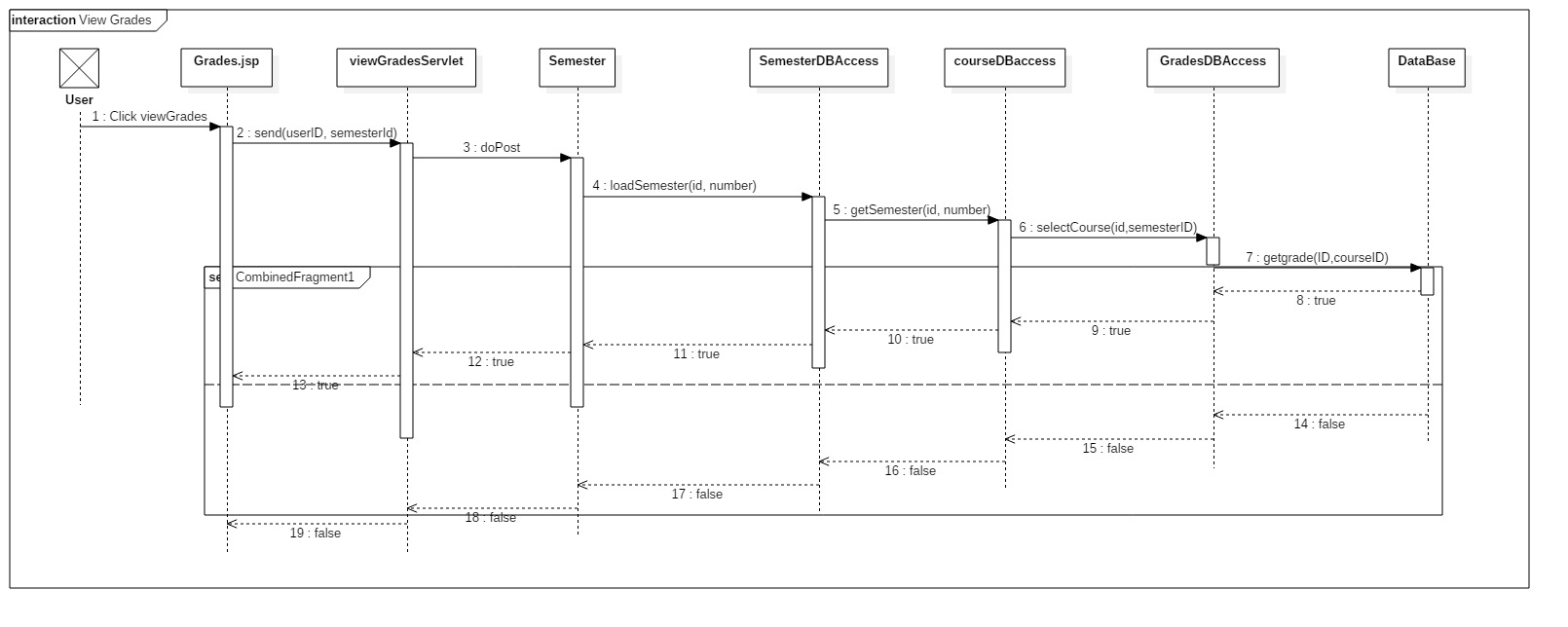
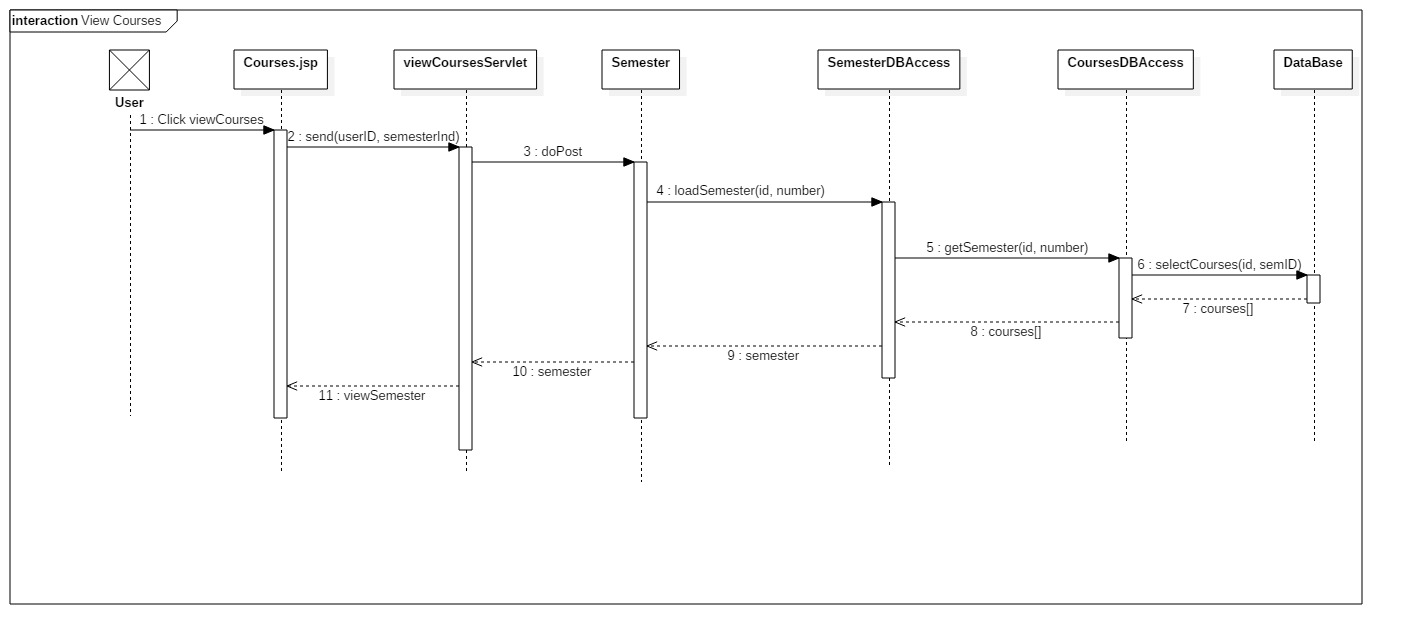
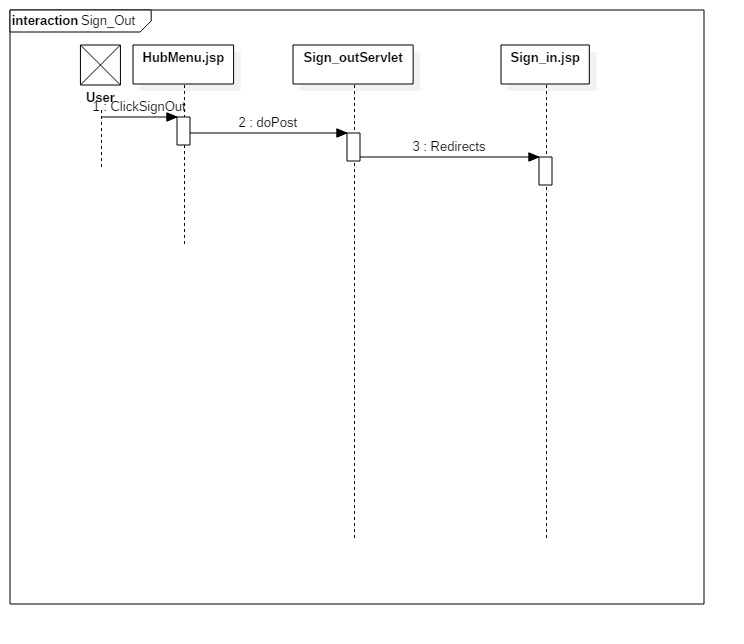
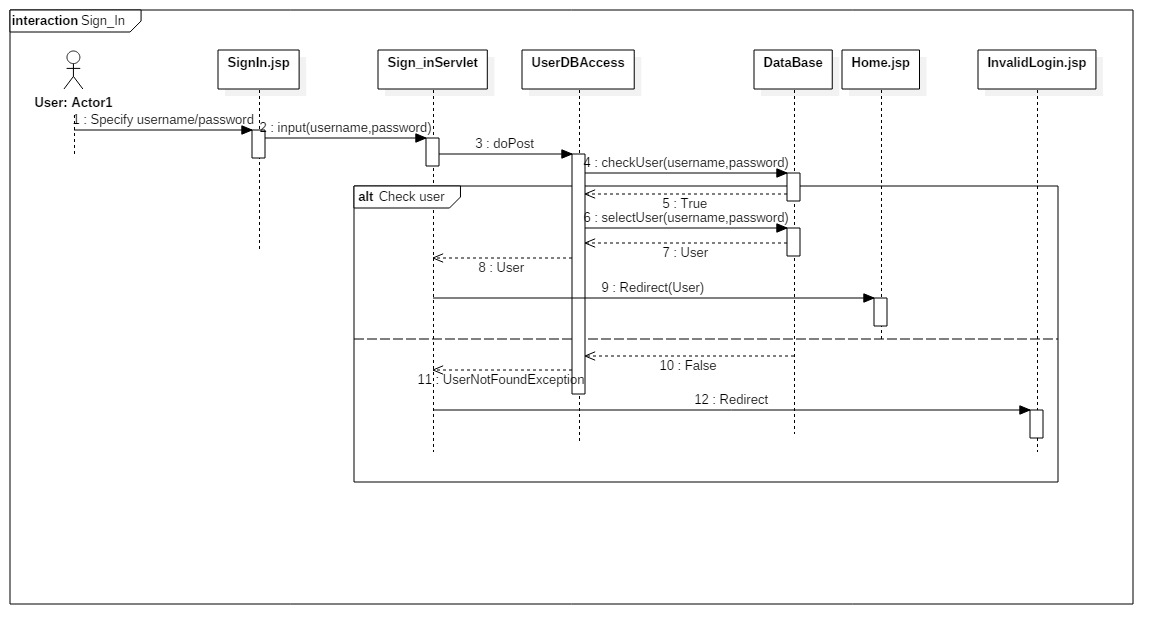
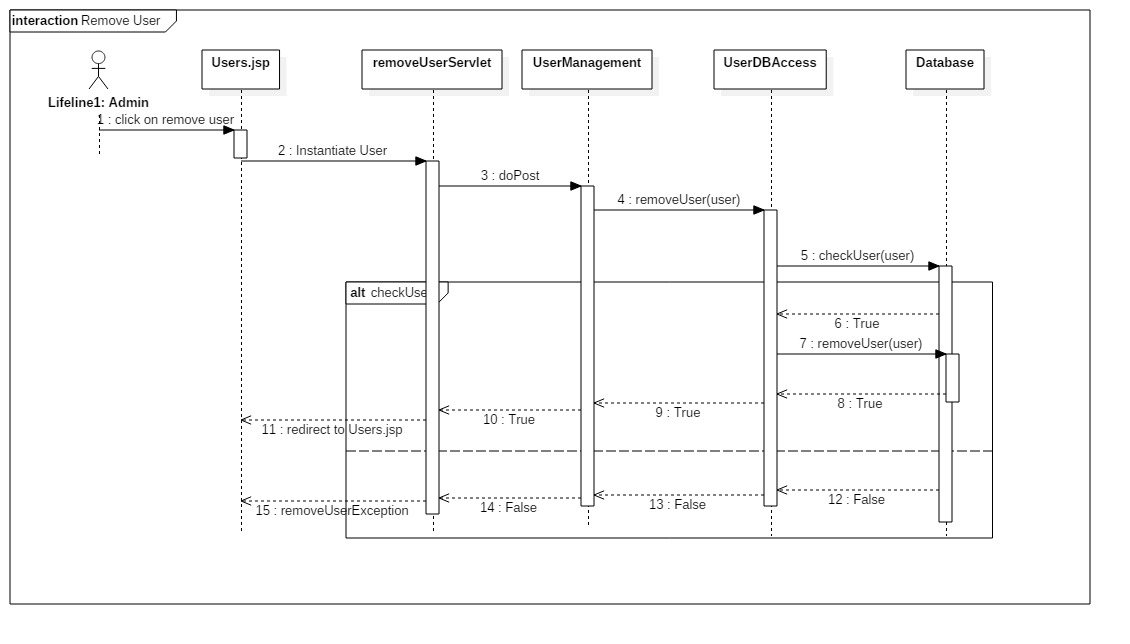
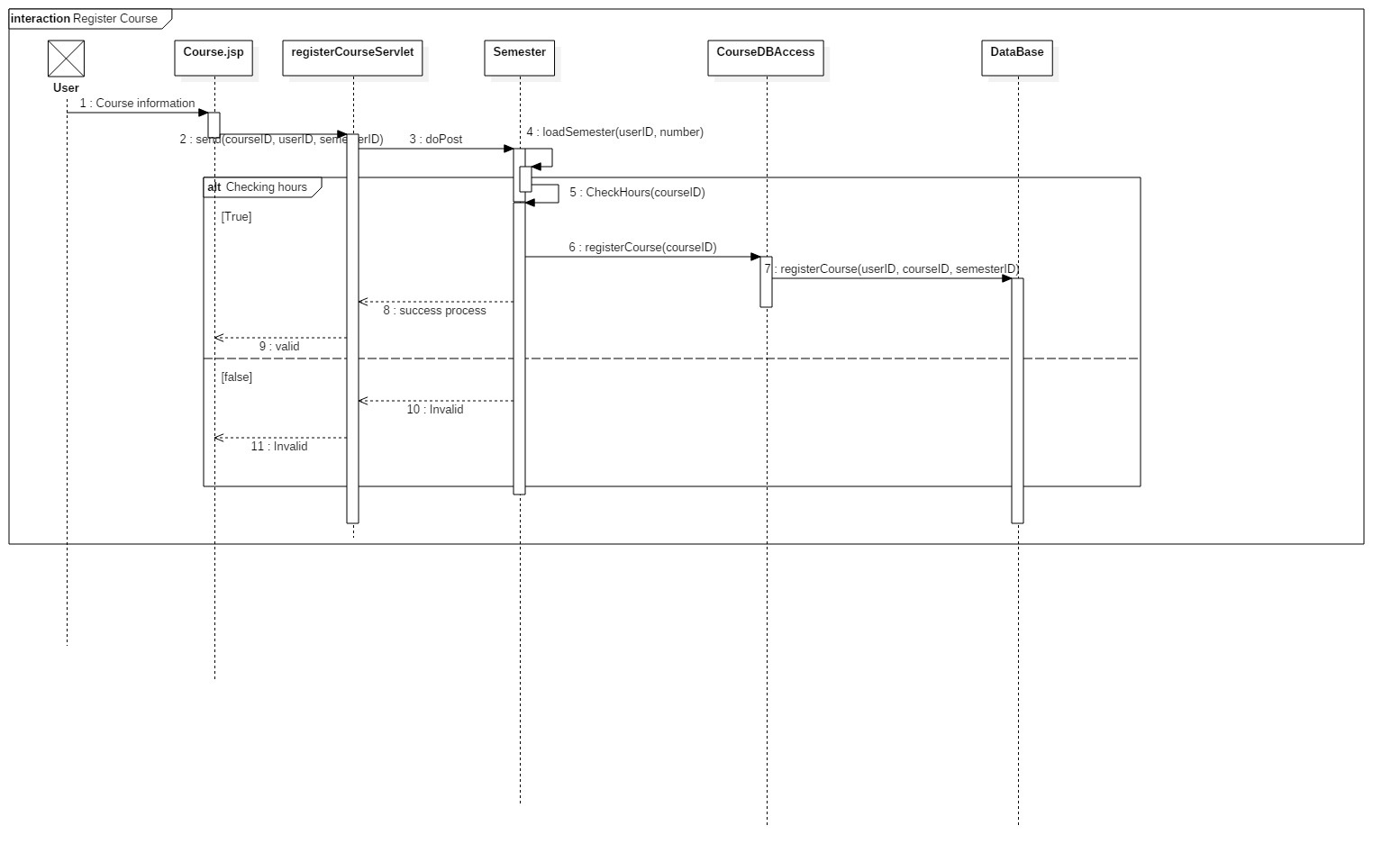
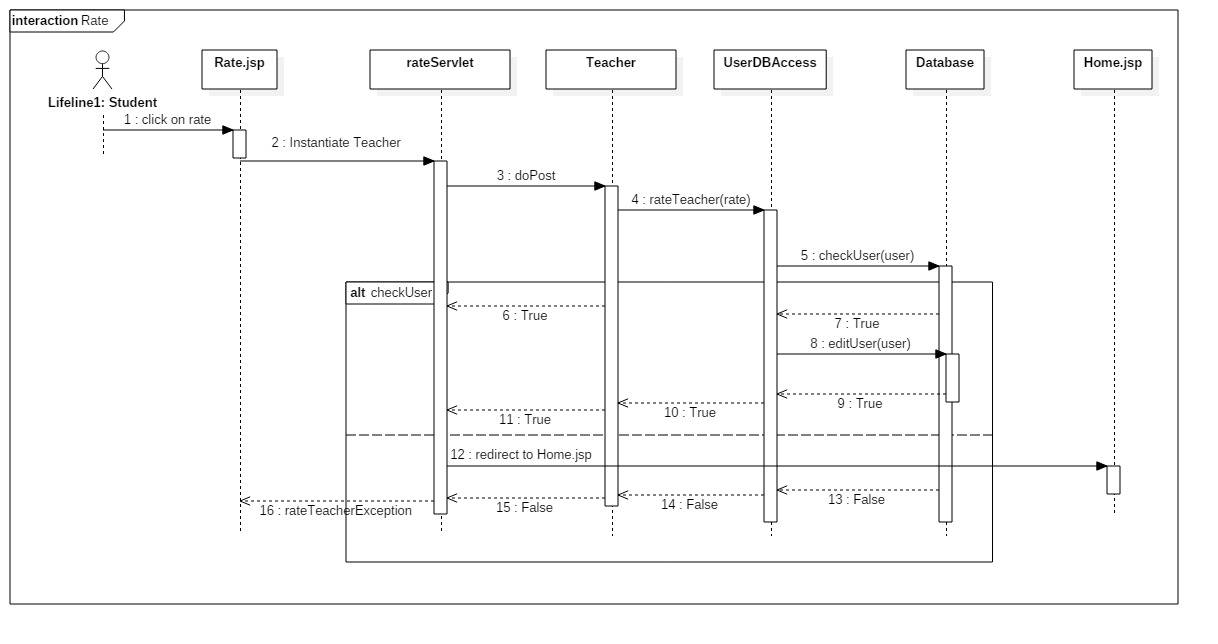
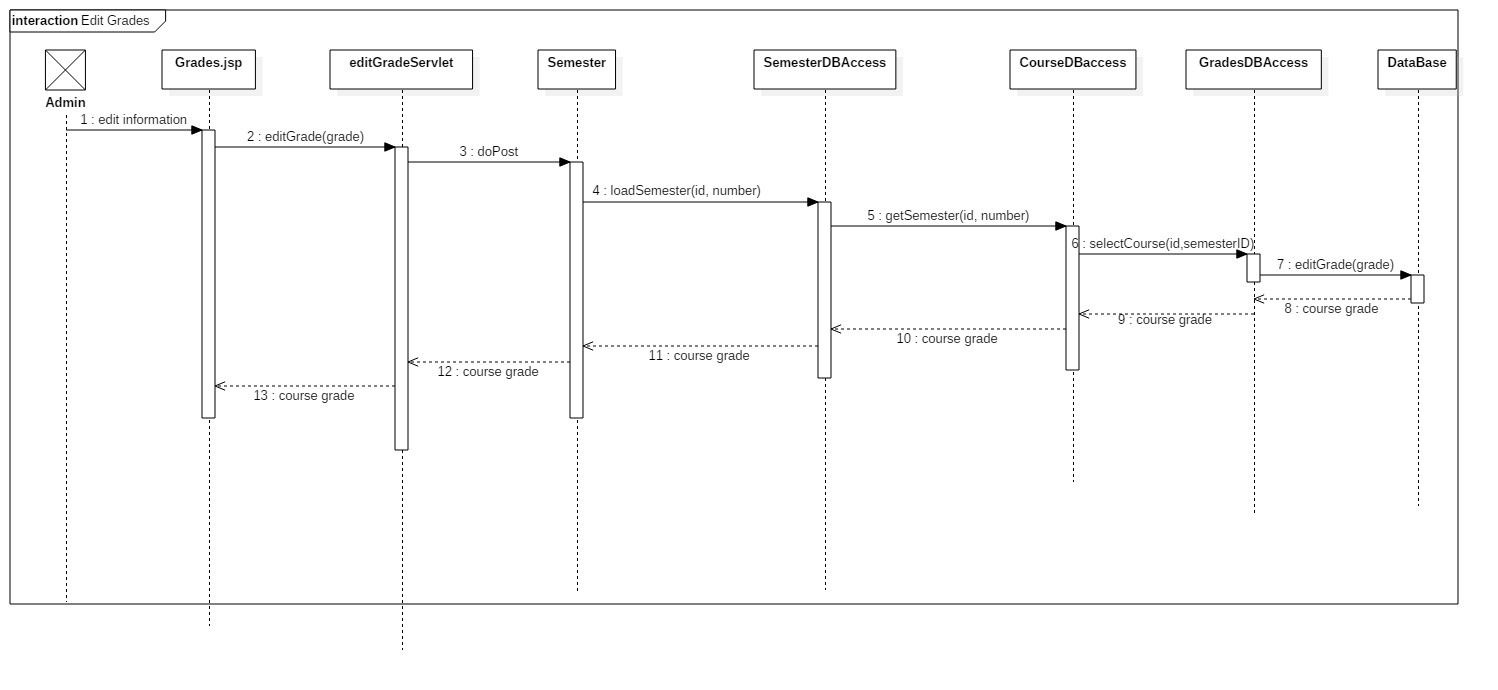
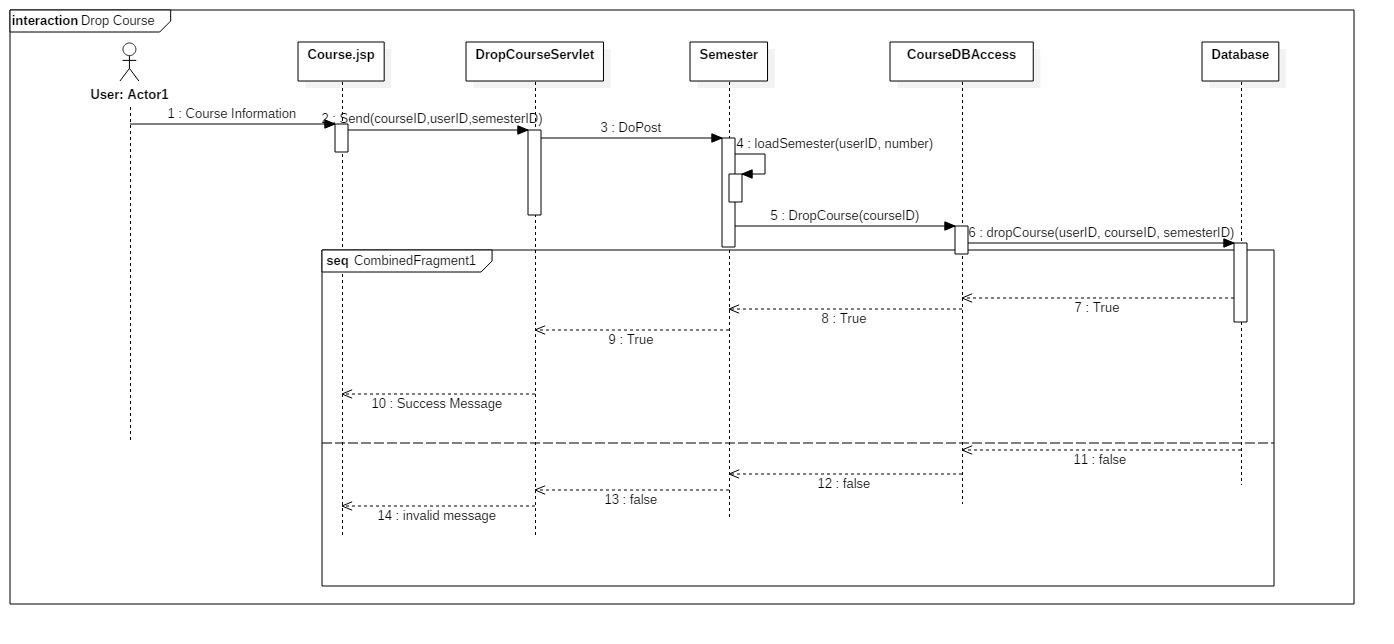
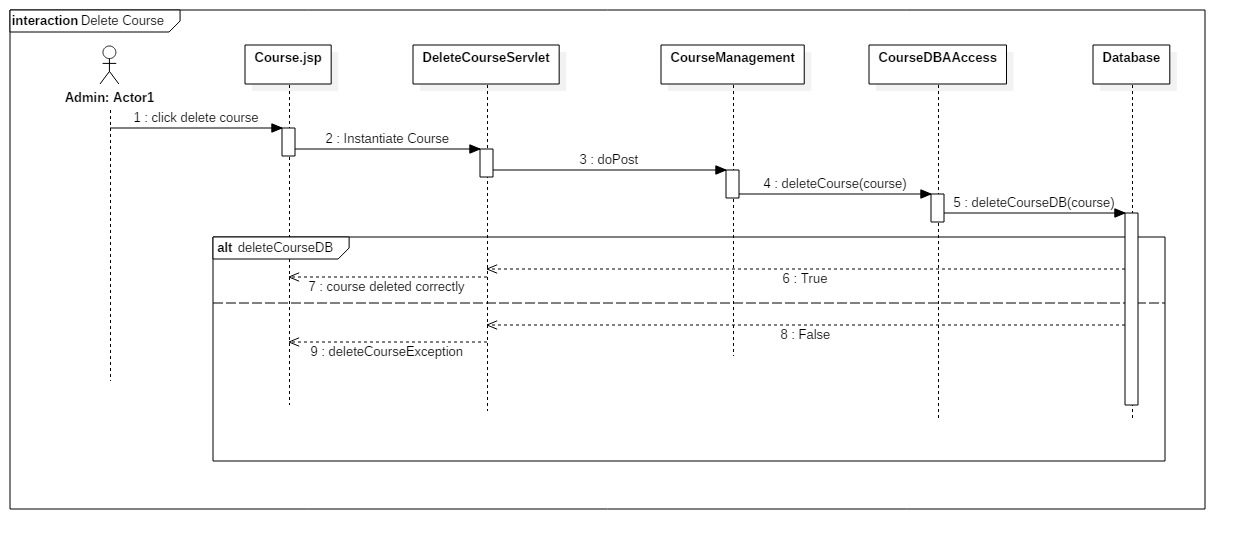
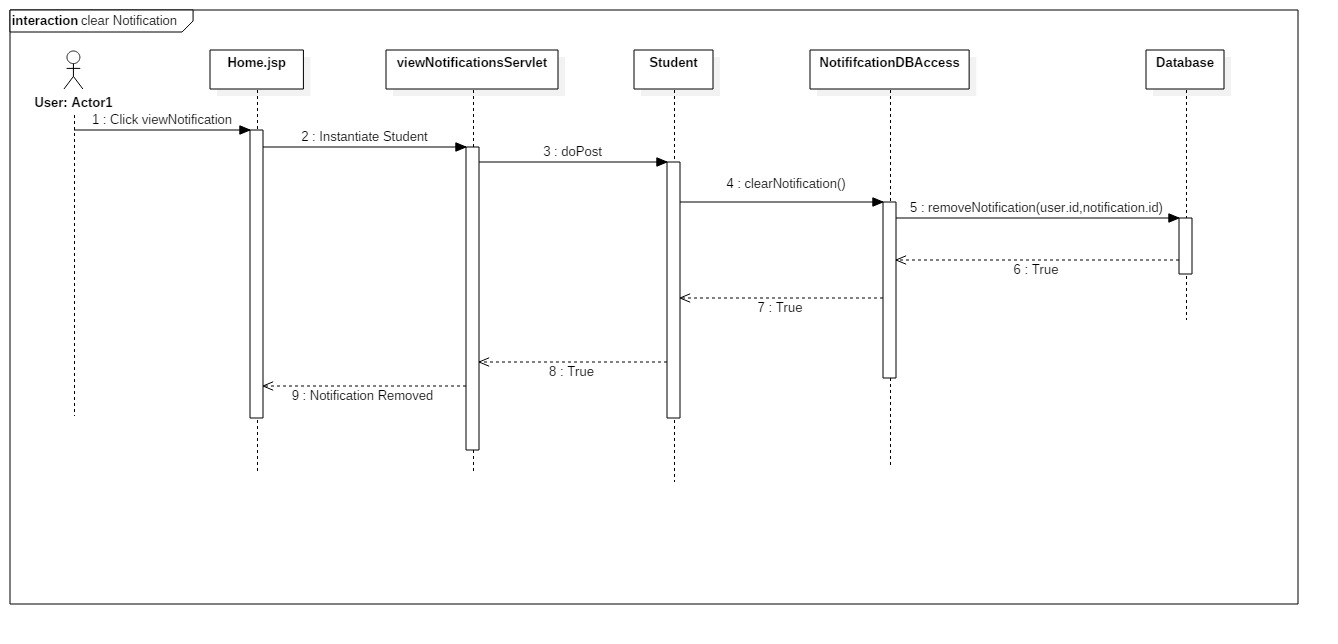
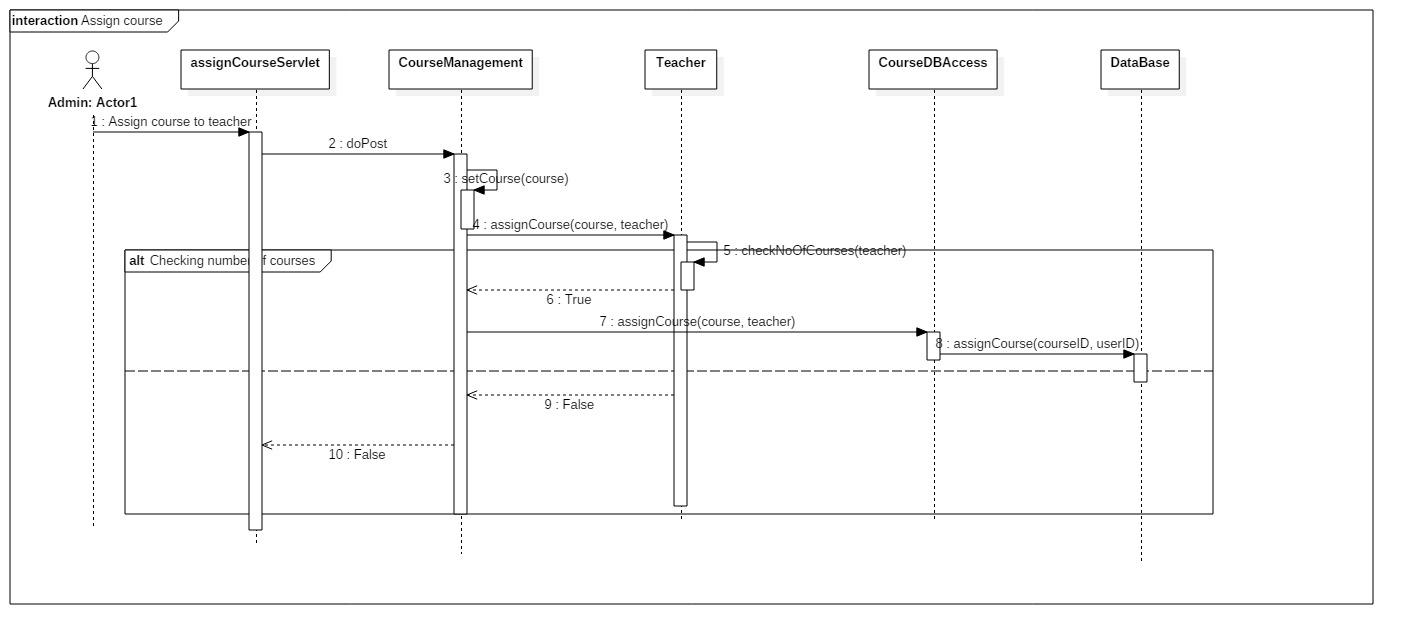
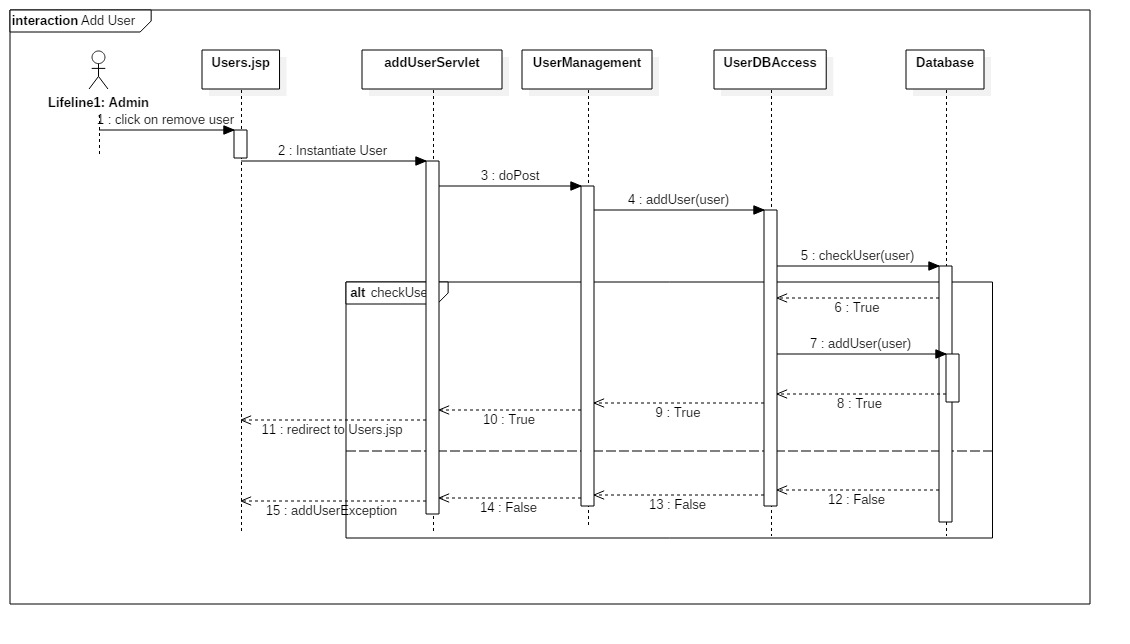
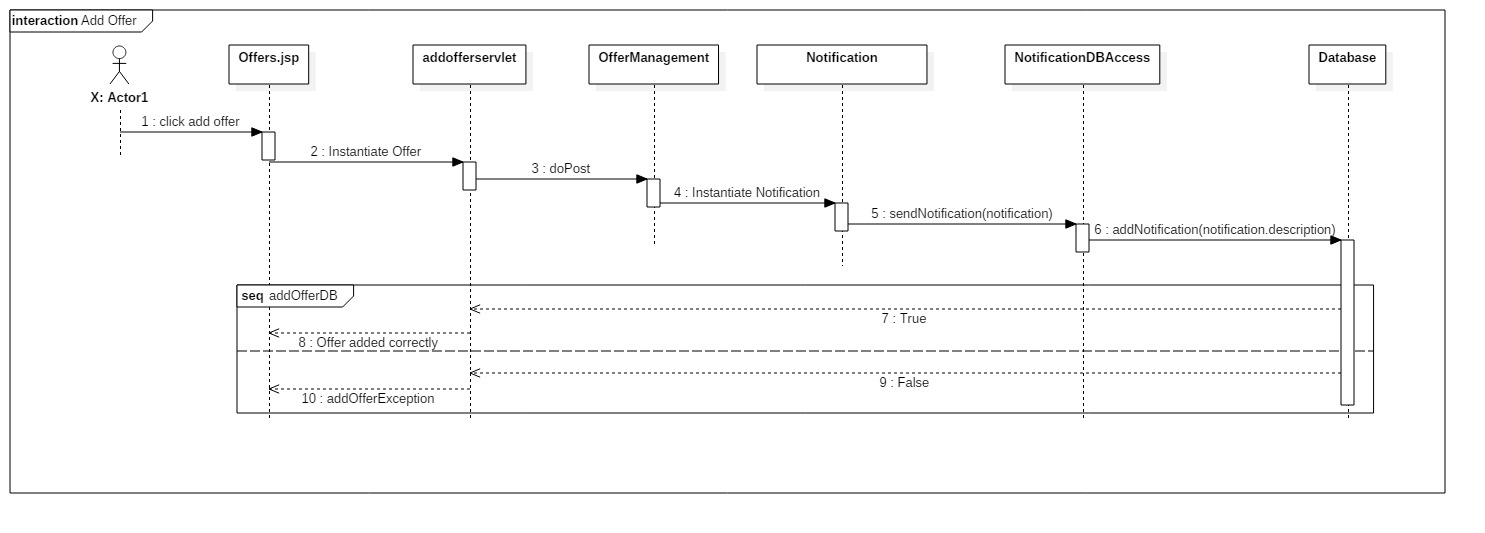
* **If any method in a class is implementing an algorithm (complex enough), then you should describe it here.**

## 

## III. Sequence diagrams

**0****1**

2223



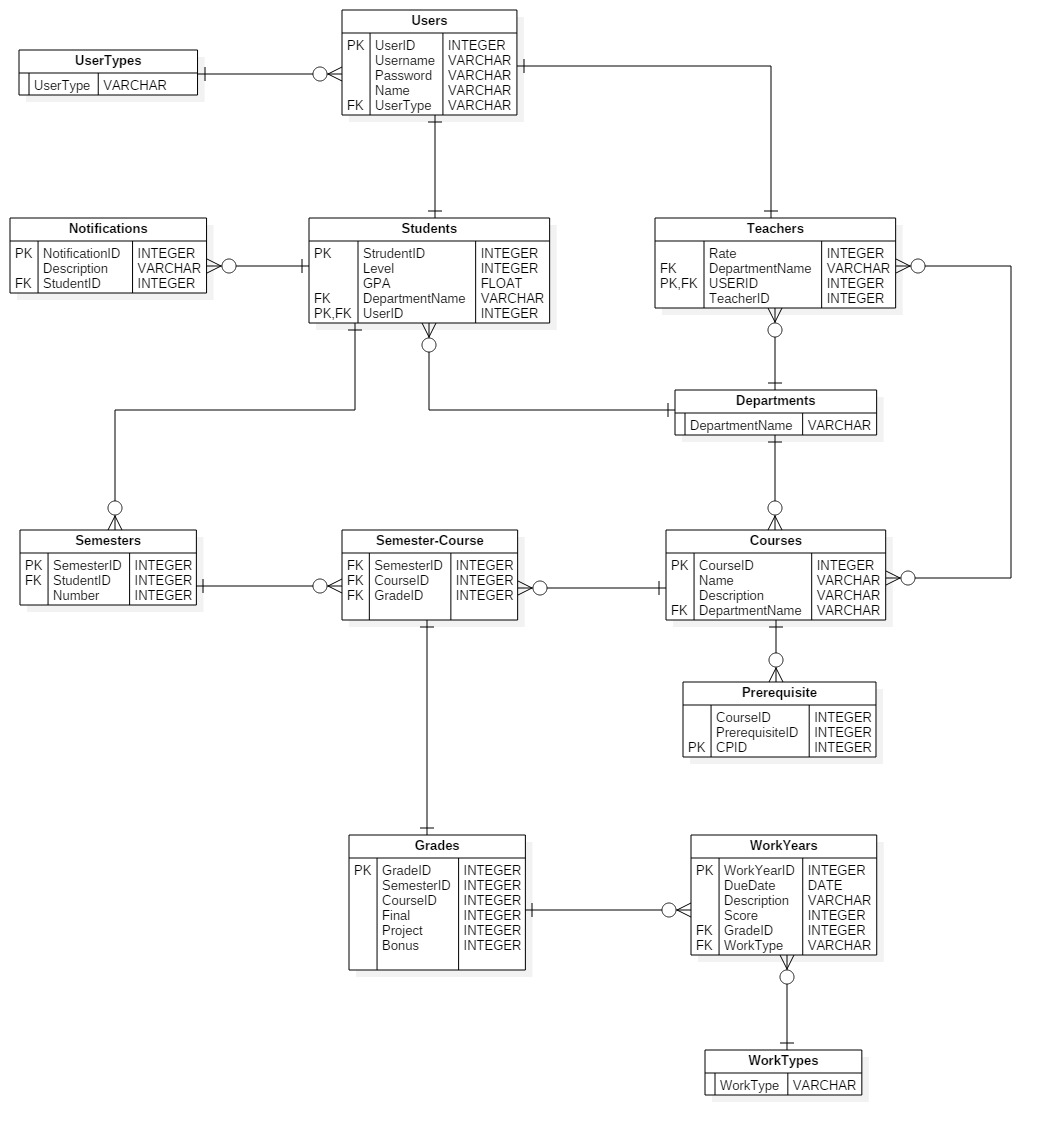
### Class - Sequence Usage Table

* **In this table, we will list EVERY class in class diagram and which sequences used this class diagram. This helps in avoiding either unused classes or extra classes appears in sequence diagrams. In "Overall used methods" section, put all functions appeared in all sequences. If this table was built in ignorance of actual class / sequence diagrams = REJECTED for whole document.**

| **Class Name** | **Sequence Diagrams** | **Overall used methods** |
| --- | --- | --- |
| UserDBAccess | addUser, removeUser, rateUser,sign\_in | checkUser, addUser, editUser |
| NotificationDBAccess | viewNotification, clearNotification, rate, addOffer | removeNotifiction, getNotification, addNotification |
| GradeDBAccess | editGrades, viewGrades, addGrades | addGrade, getGrade |
| SemesterDBAccess | addGrade, editGrade, viewGrade, viewCourses | getSemester |
| CourseDBAccess | addCourse, addGrade, assignCourse, deleteCourse, dropCourse, viewGrades | selectCourse, registerCourse, dropCourse, deleteCourse, addCourse, assignCourse |
| Offer | addOffer |  |
| User | All | All |
| Admin | addGrade, addCourse, addUser, deleteCourse, assignCourse, removeCourse |  |
| Teacher | assignCourse, rate | checkHours, rateTeacher |
| Student | clearNotfication, viewNotifications | clearNotification, viewNotification |
| Notificataion | addOffer, sendNotification |  |
| Grade |  |  |
| Semester | dropCourse, addCourse, editCourse, registerCourse, viewCourses, viewGrades | loadSemester, checkHours, registerCourse |
| Course |  |  |
| UserManagement | addUser | addUser, removeUser |
| CourseManagement | addCourse, assignCourse, deleteCourse | deleteCourse, assignCourse, addCourse |
|  |  |  |

## 

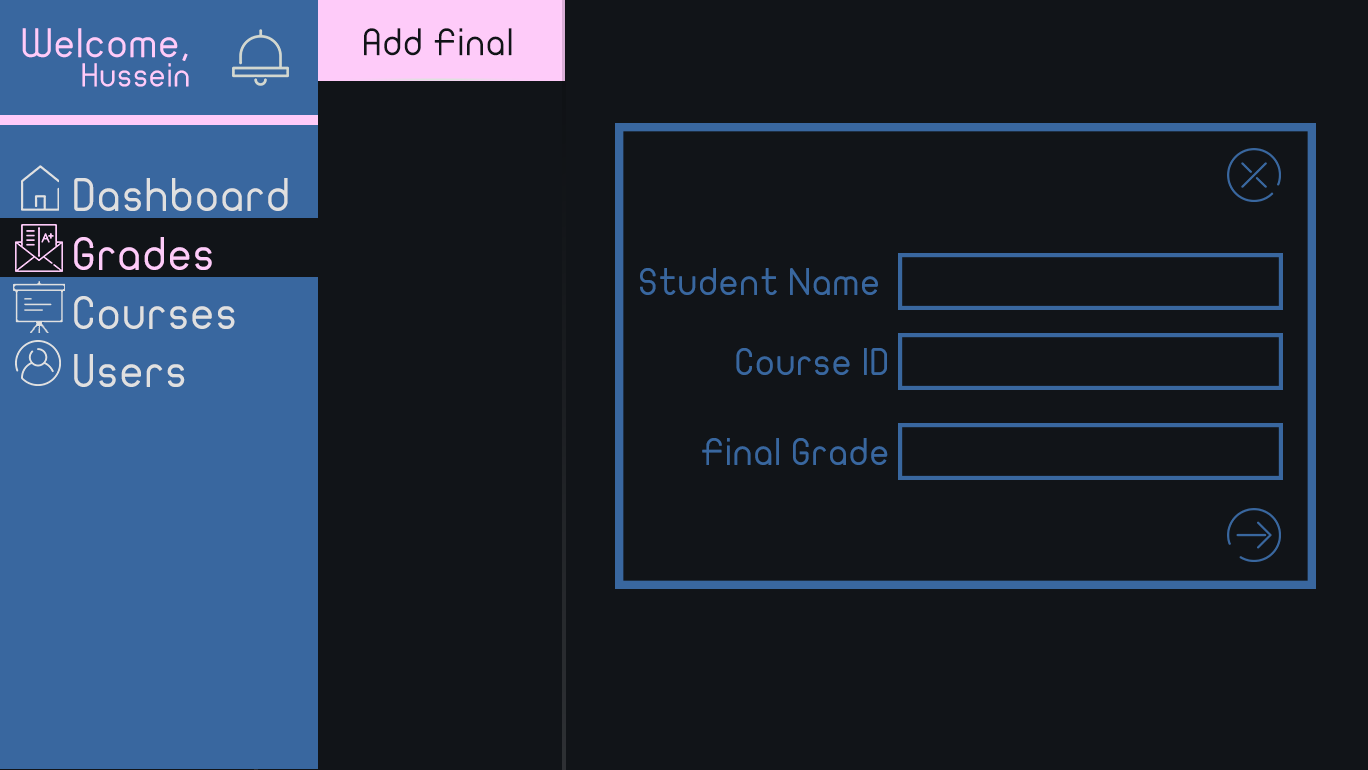
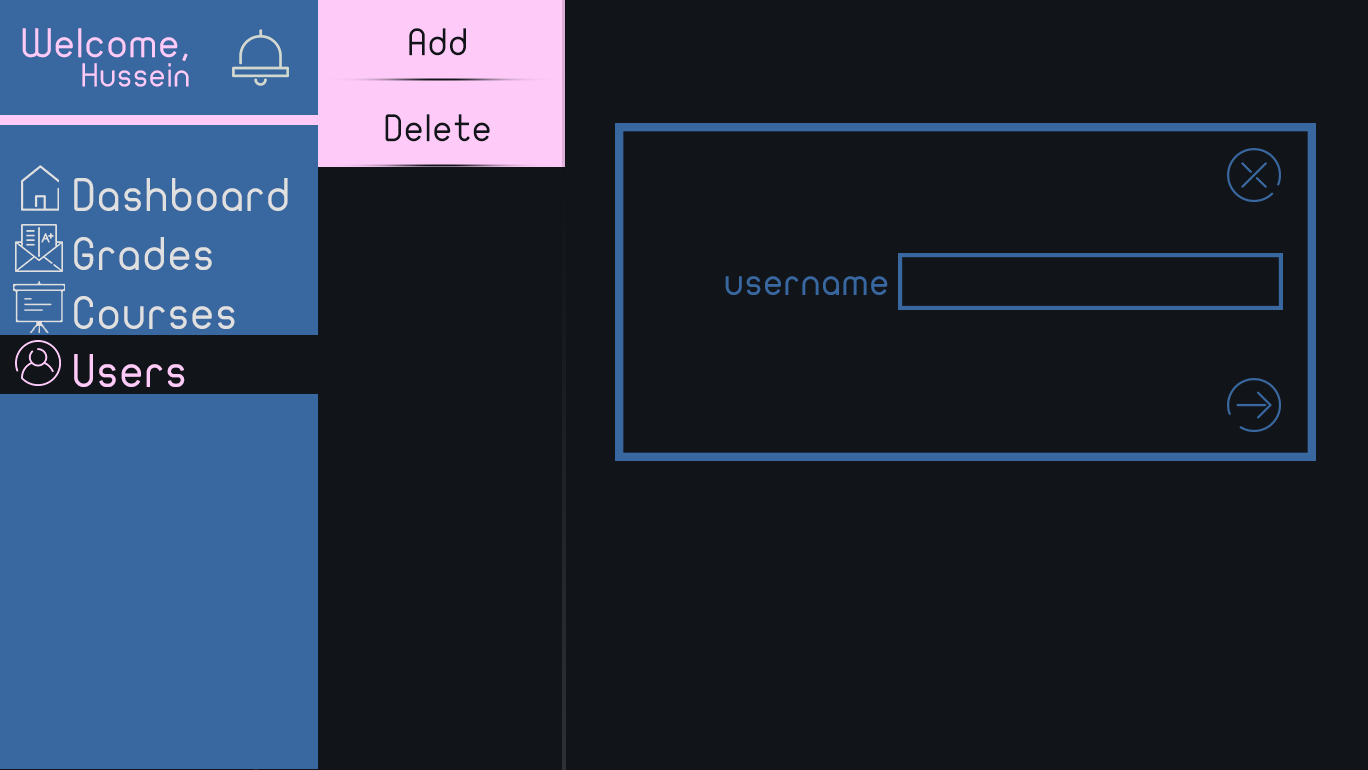
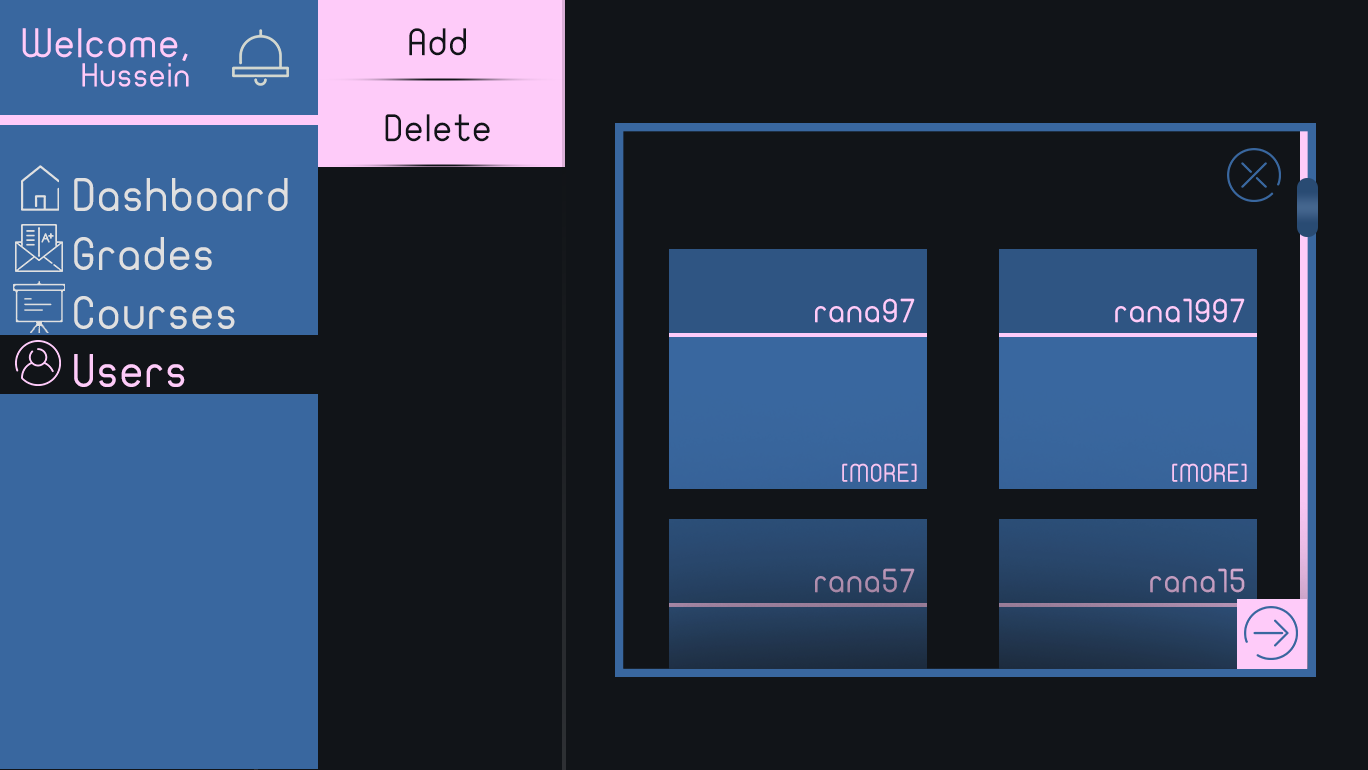
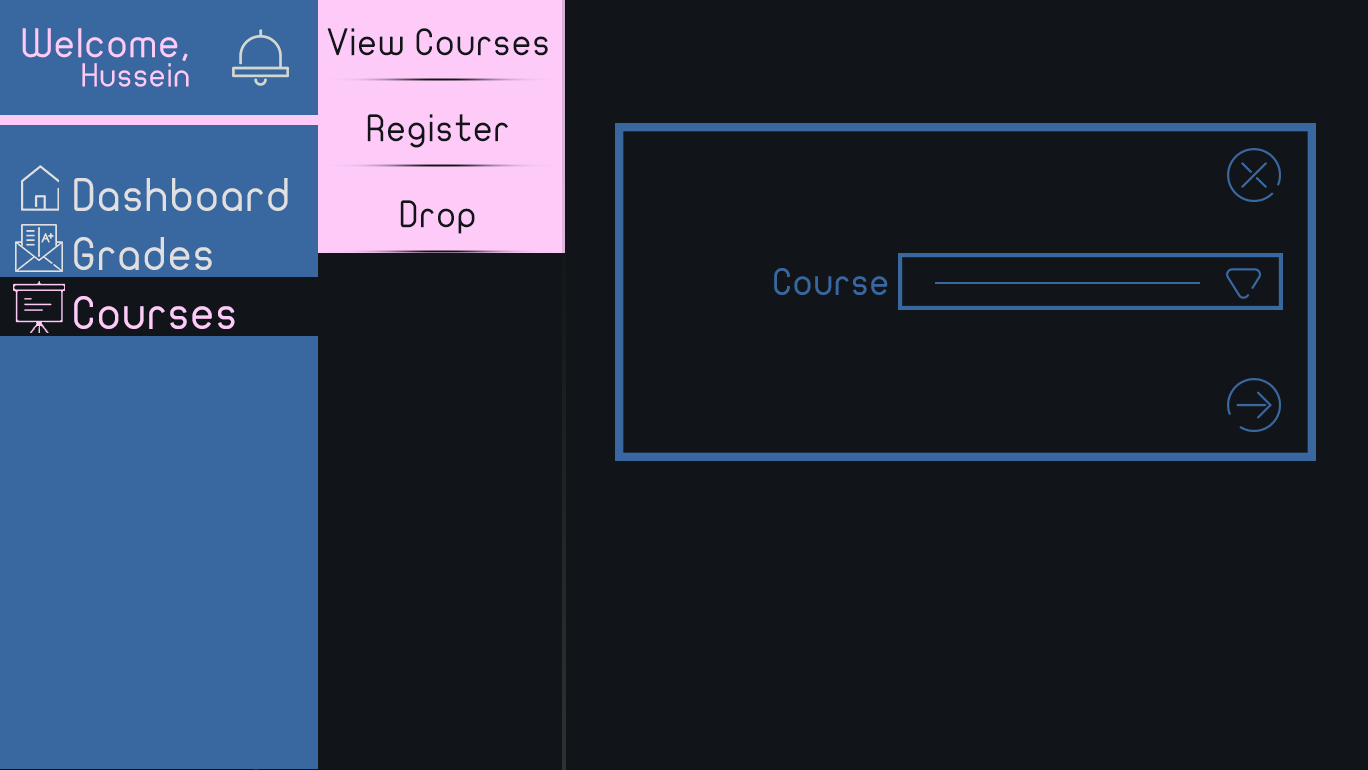
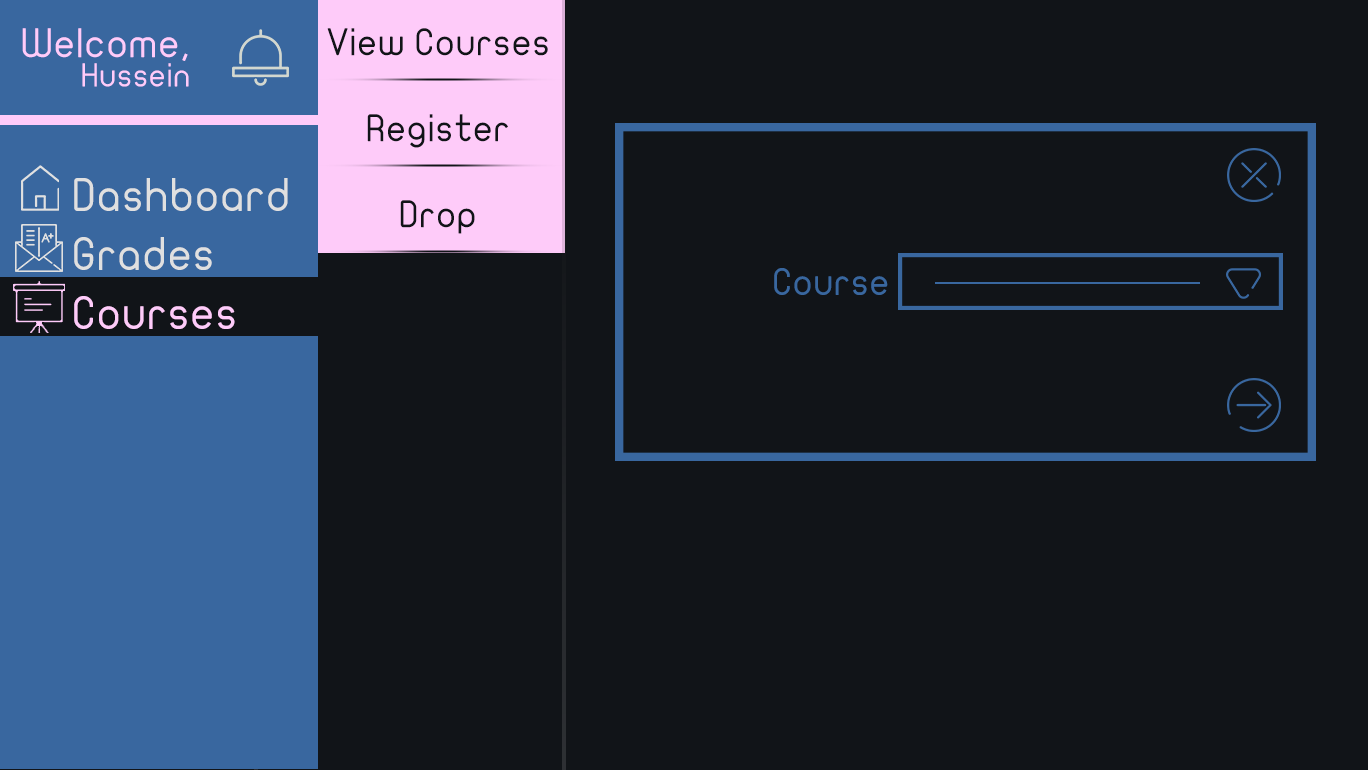
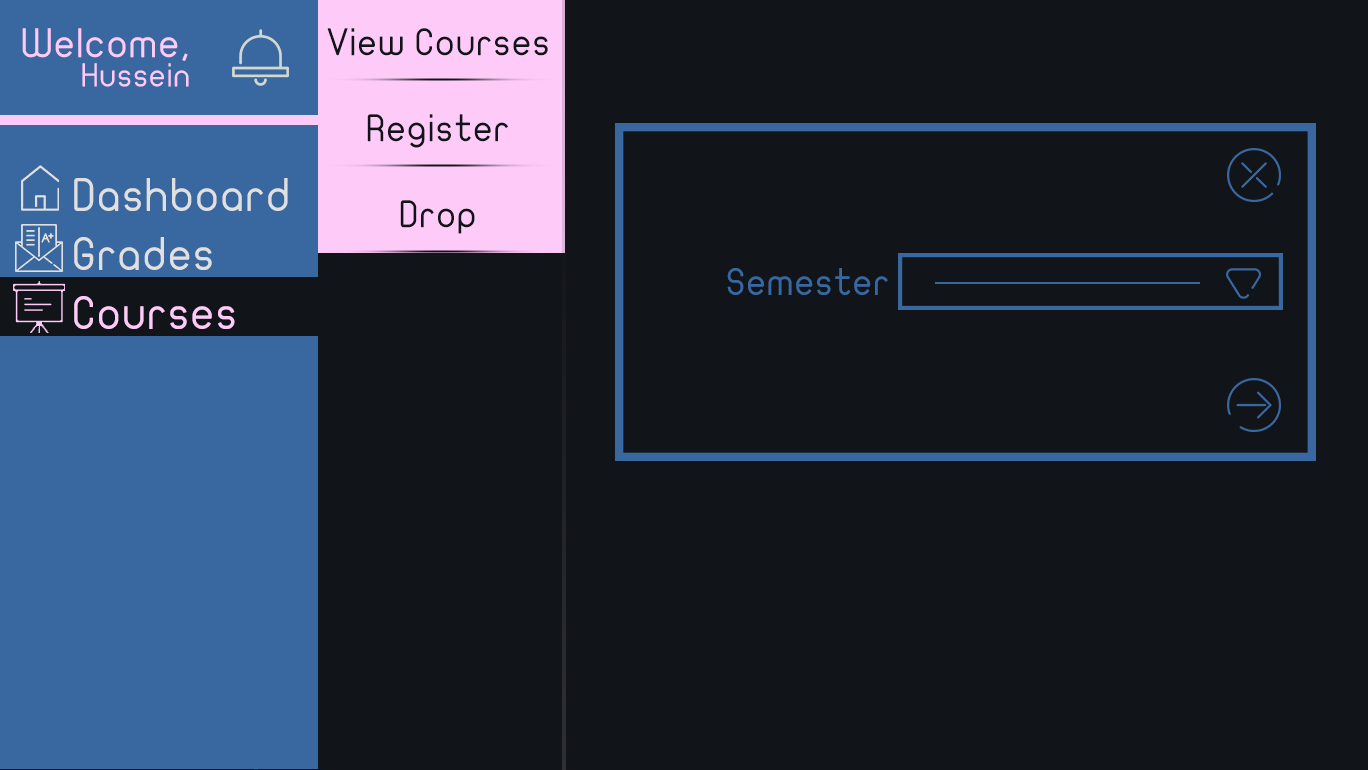
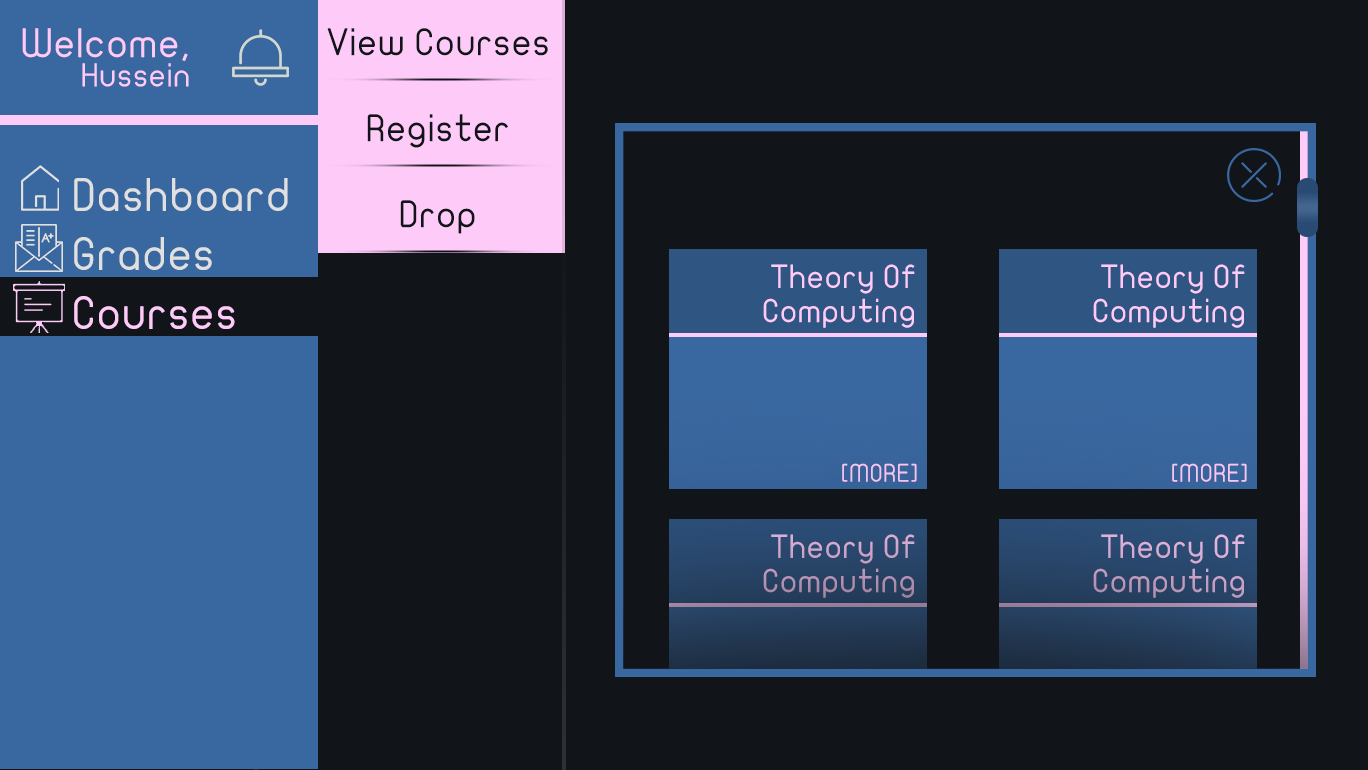
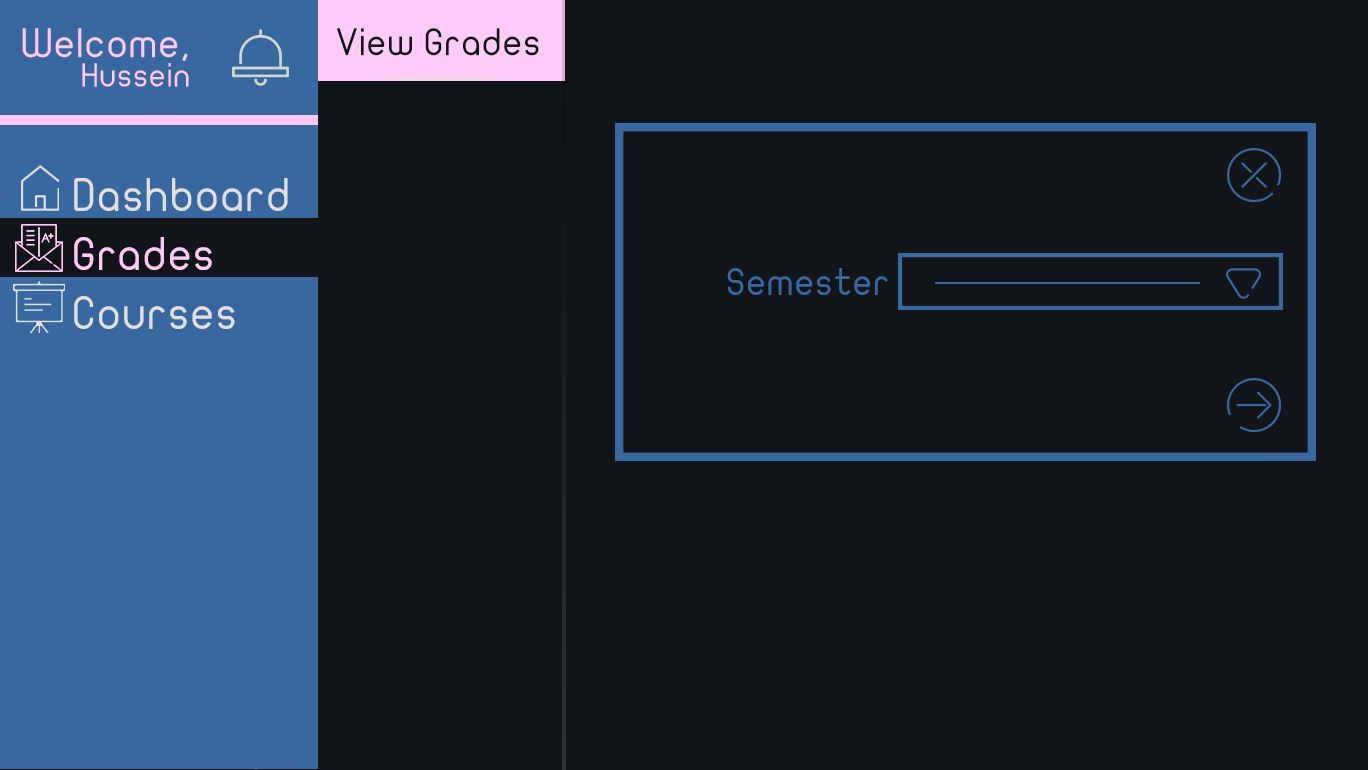
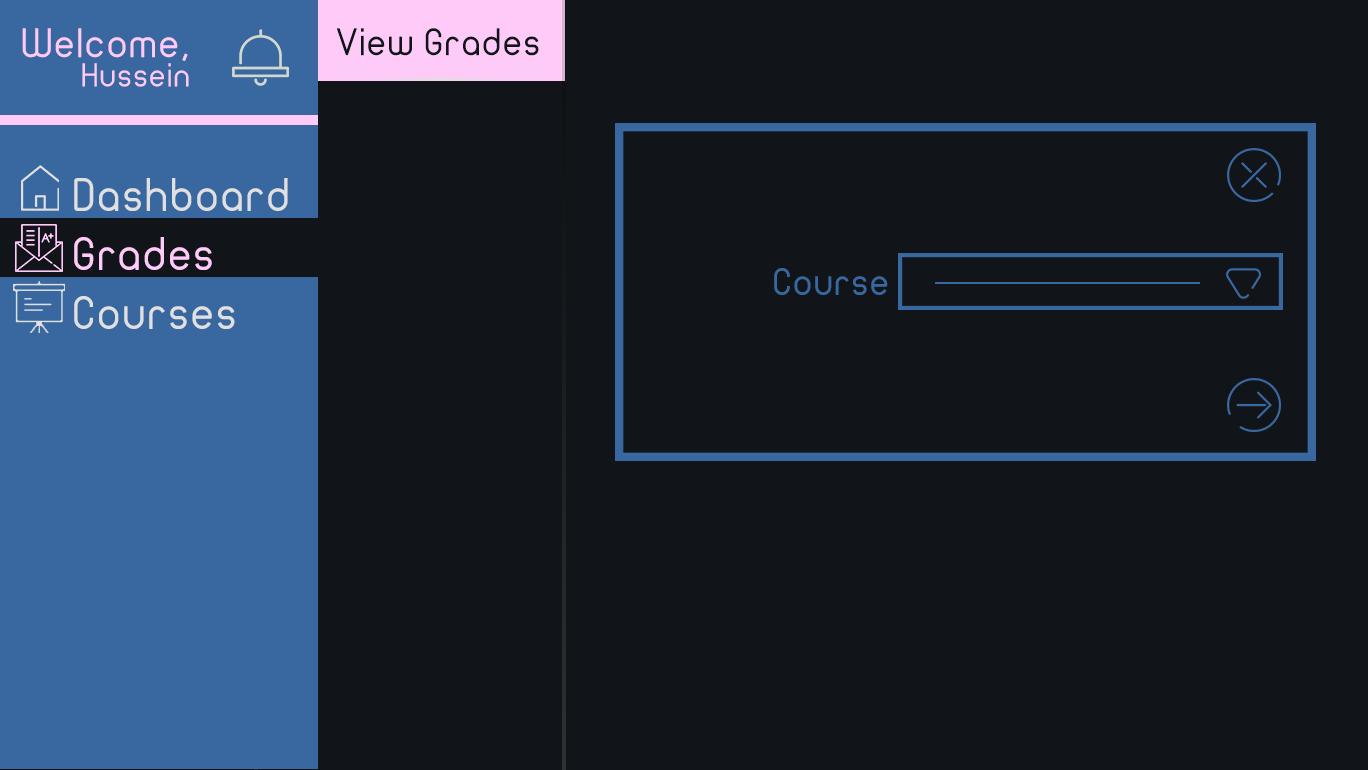
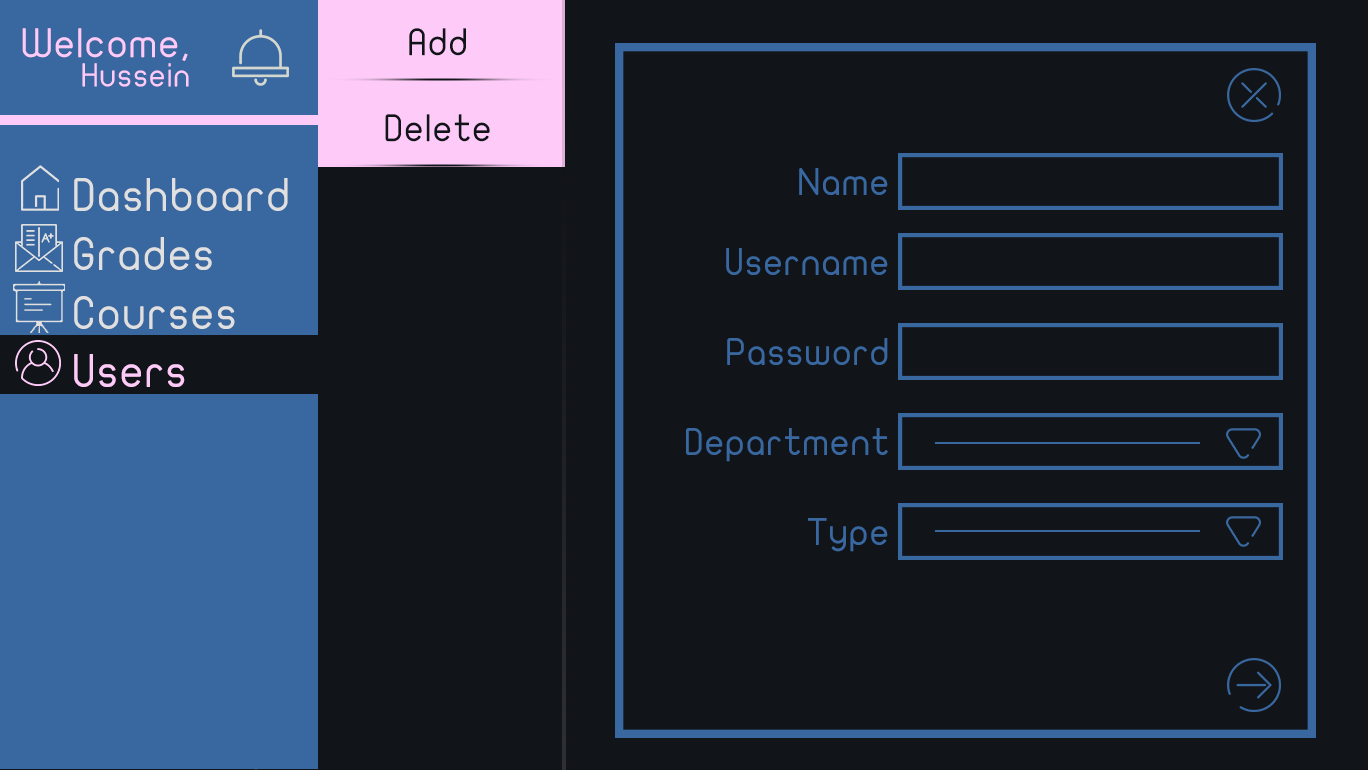
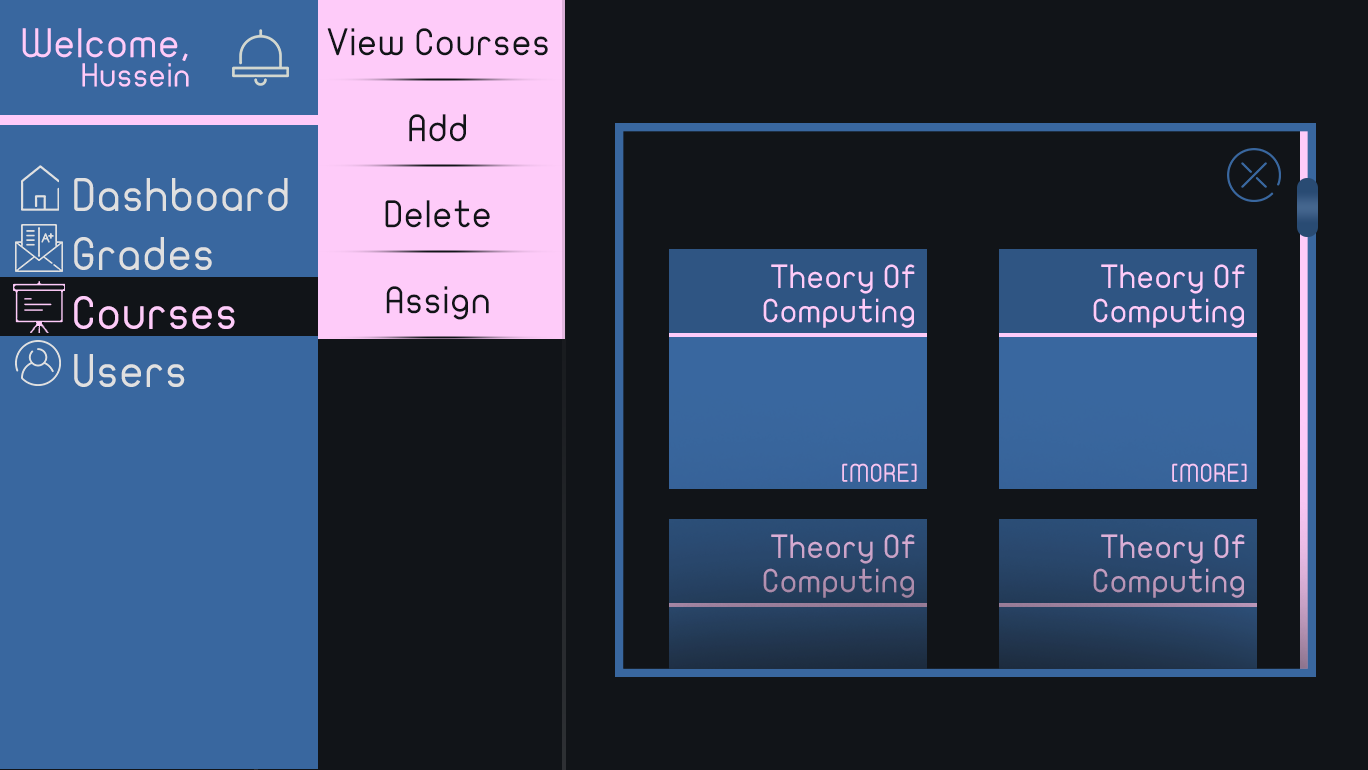
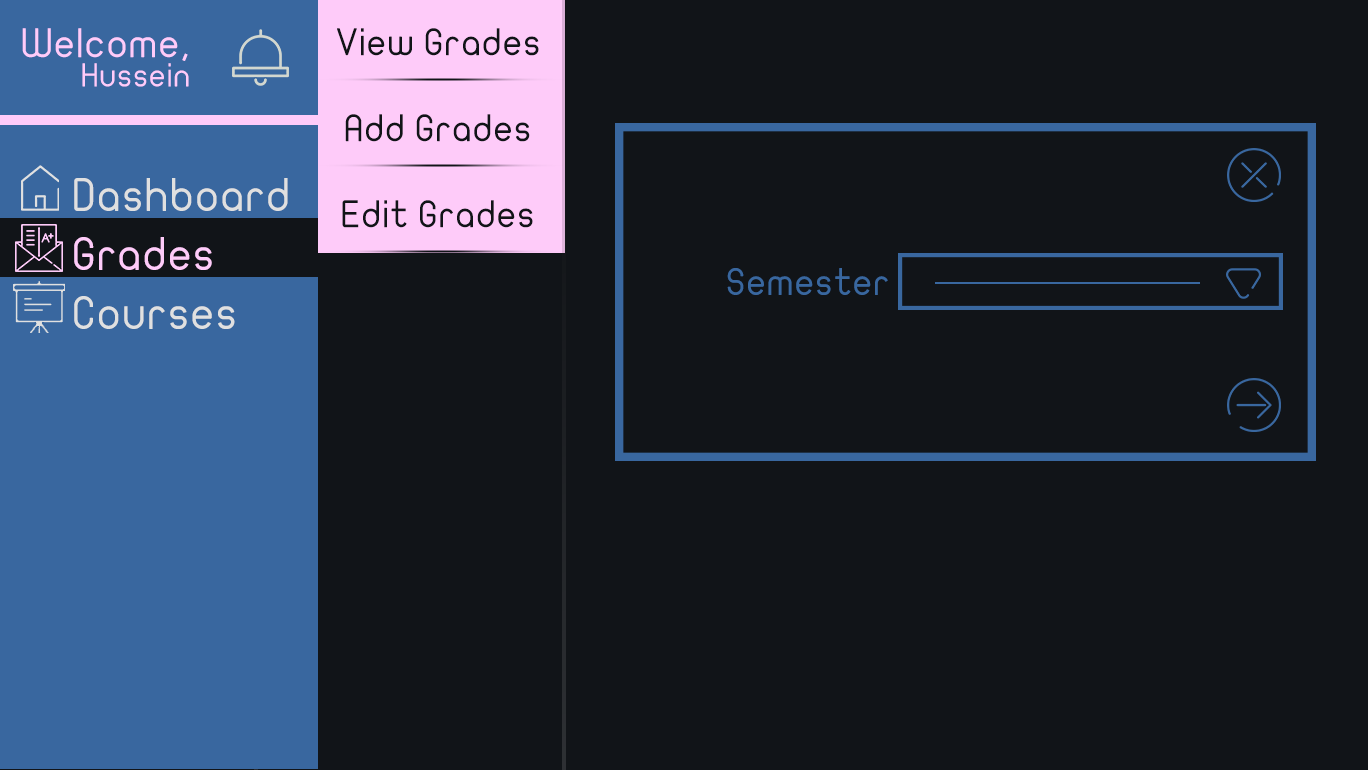
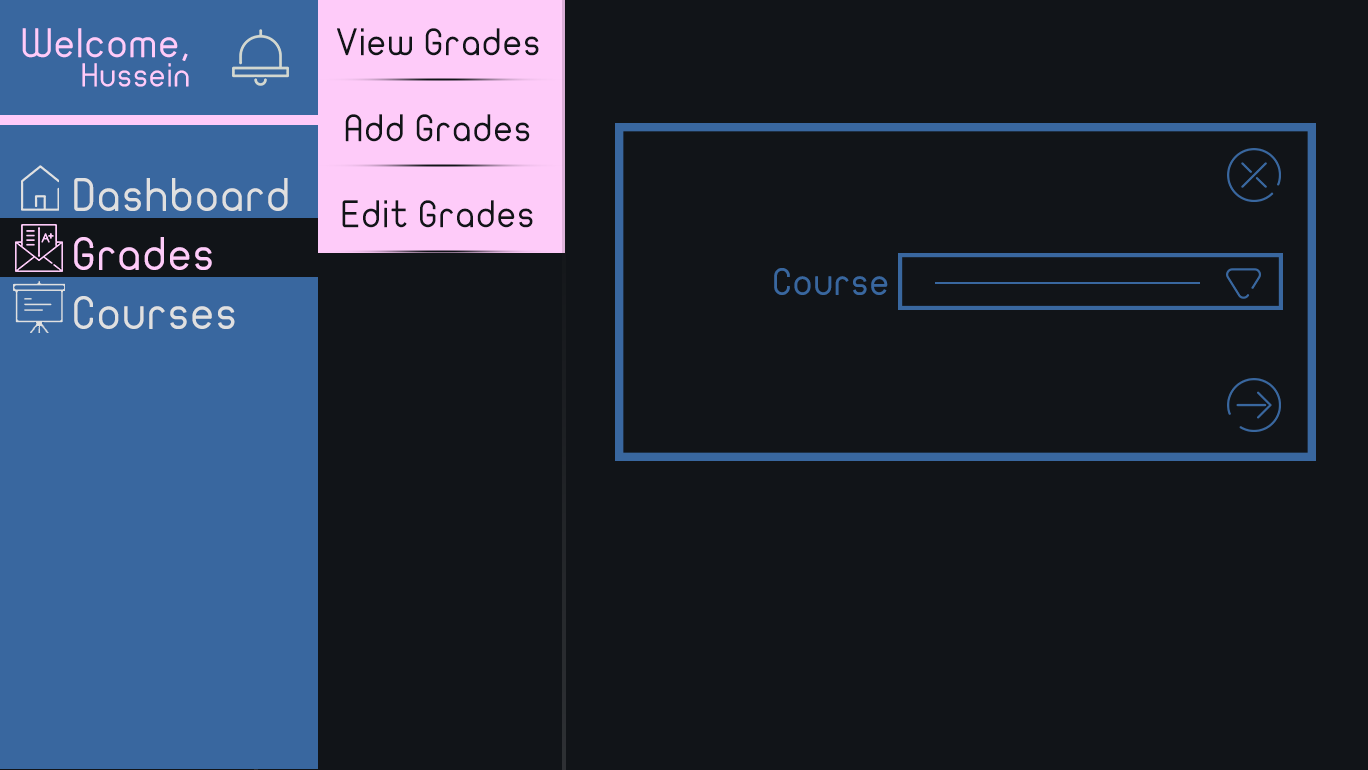
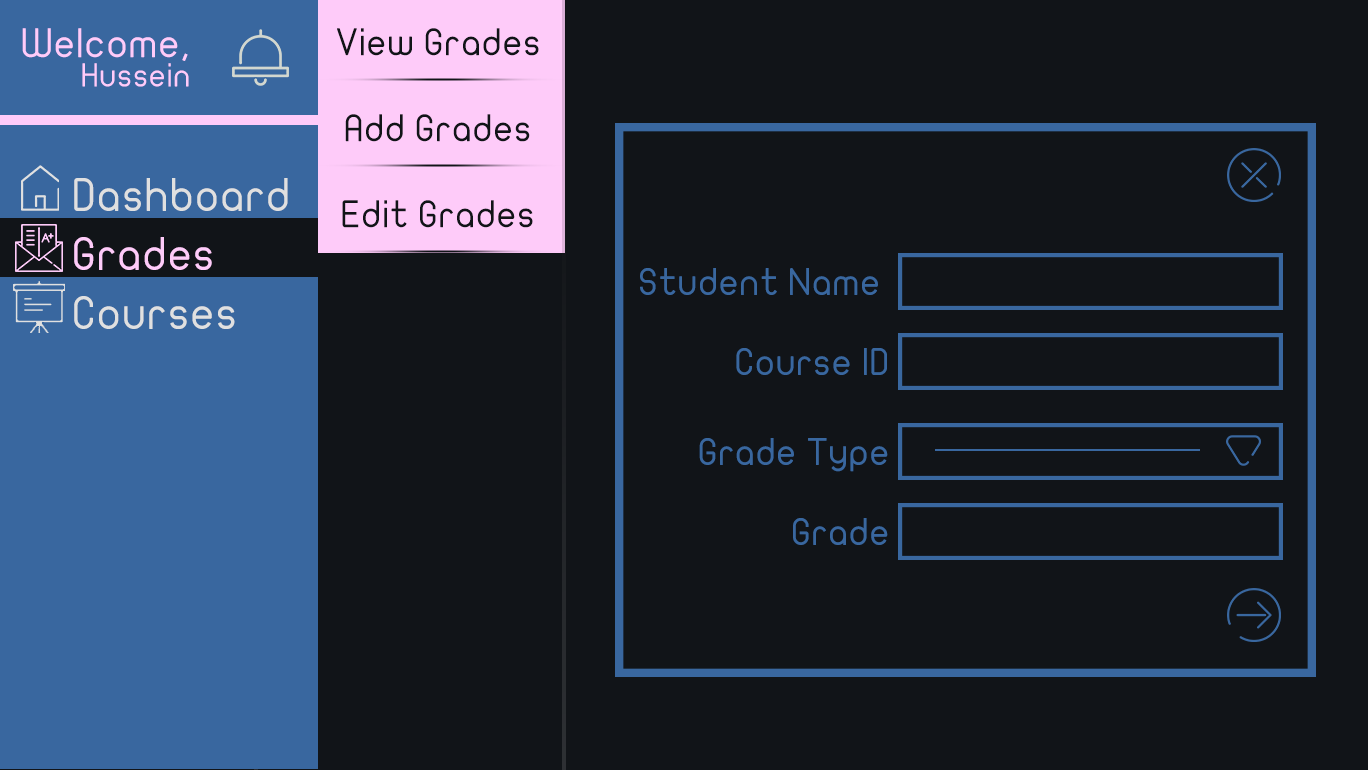
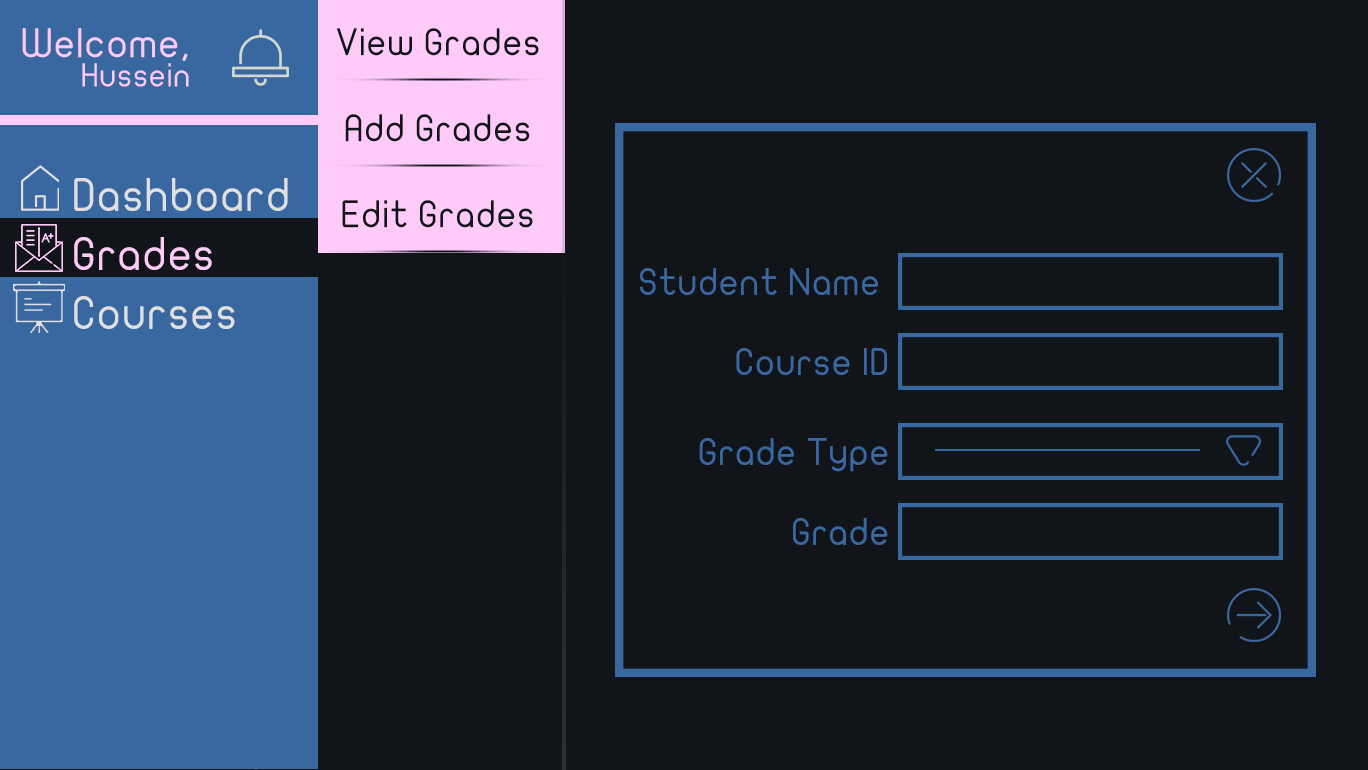
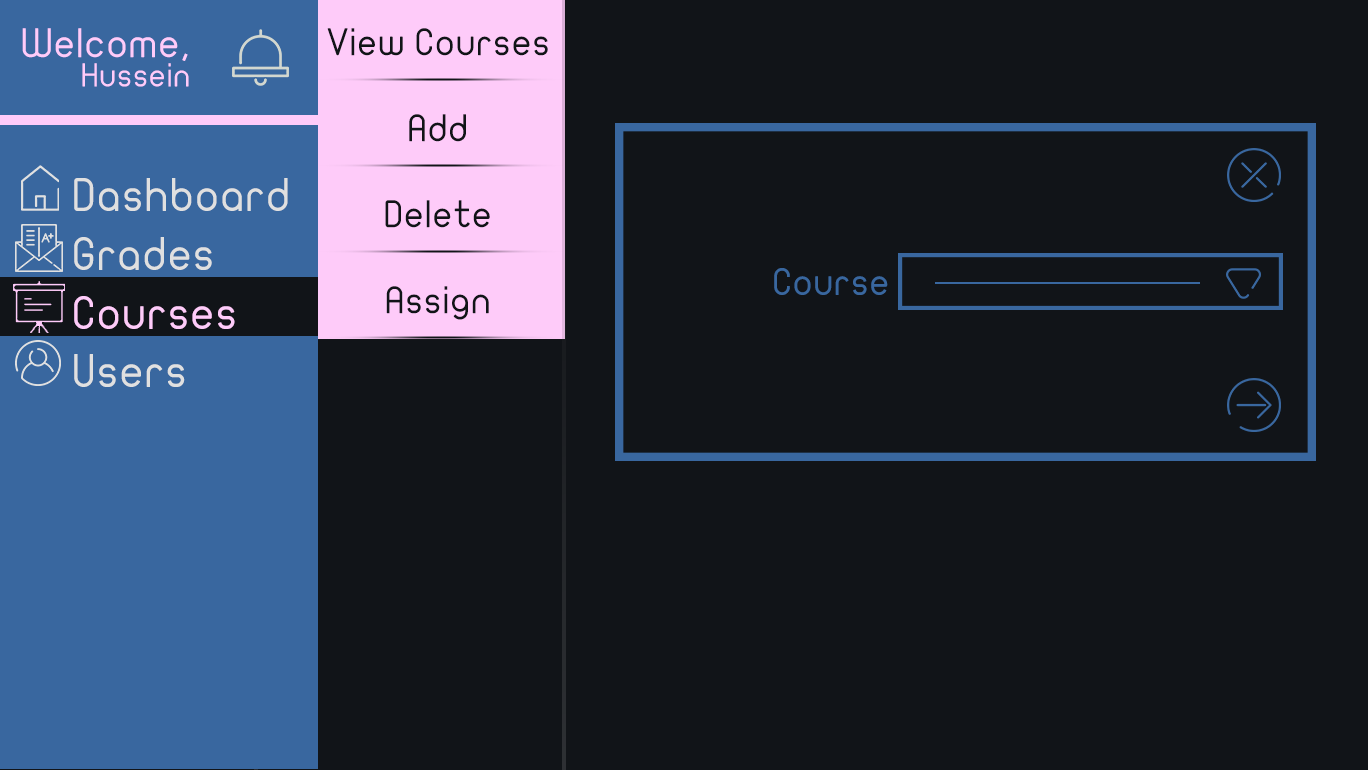
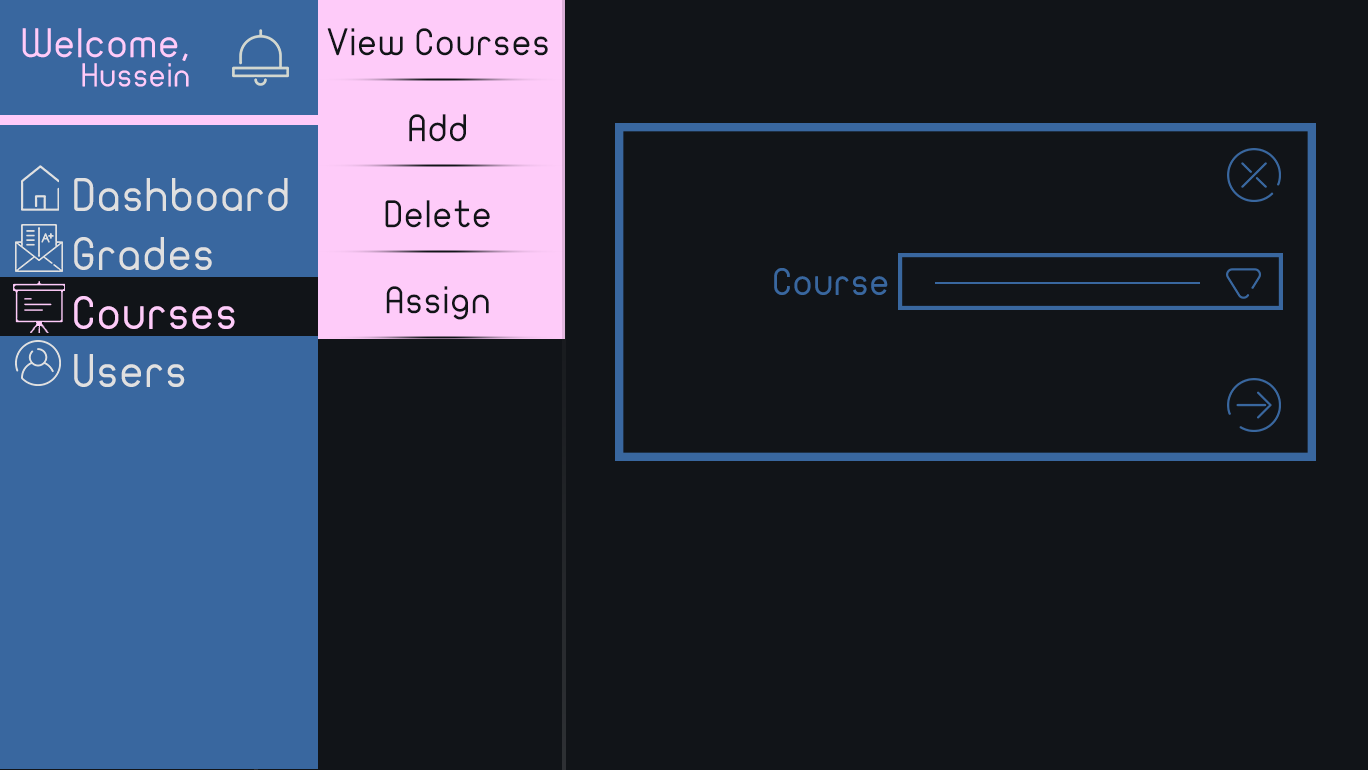
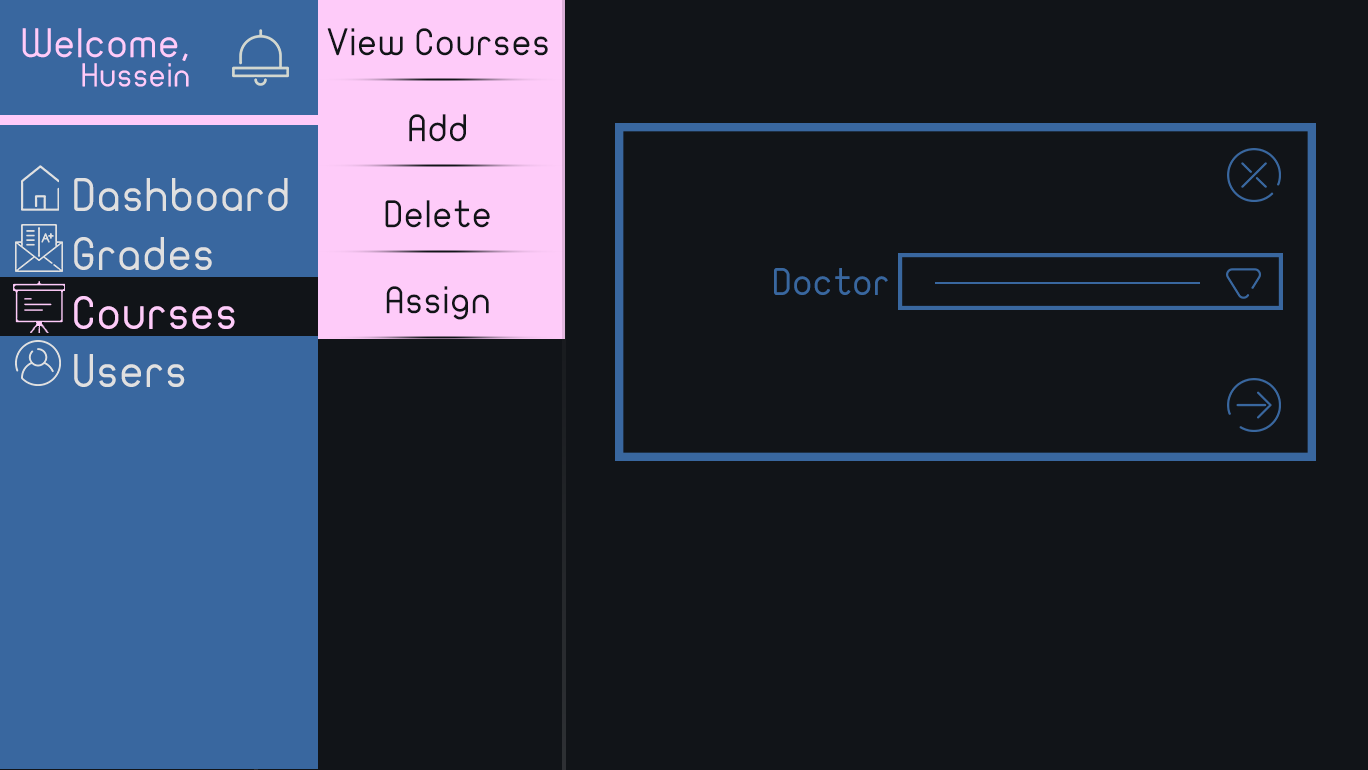
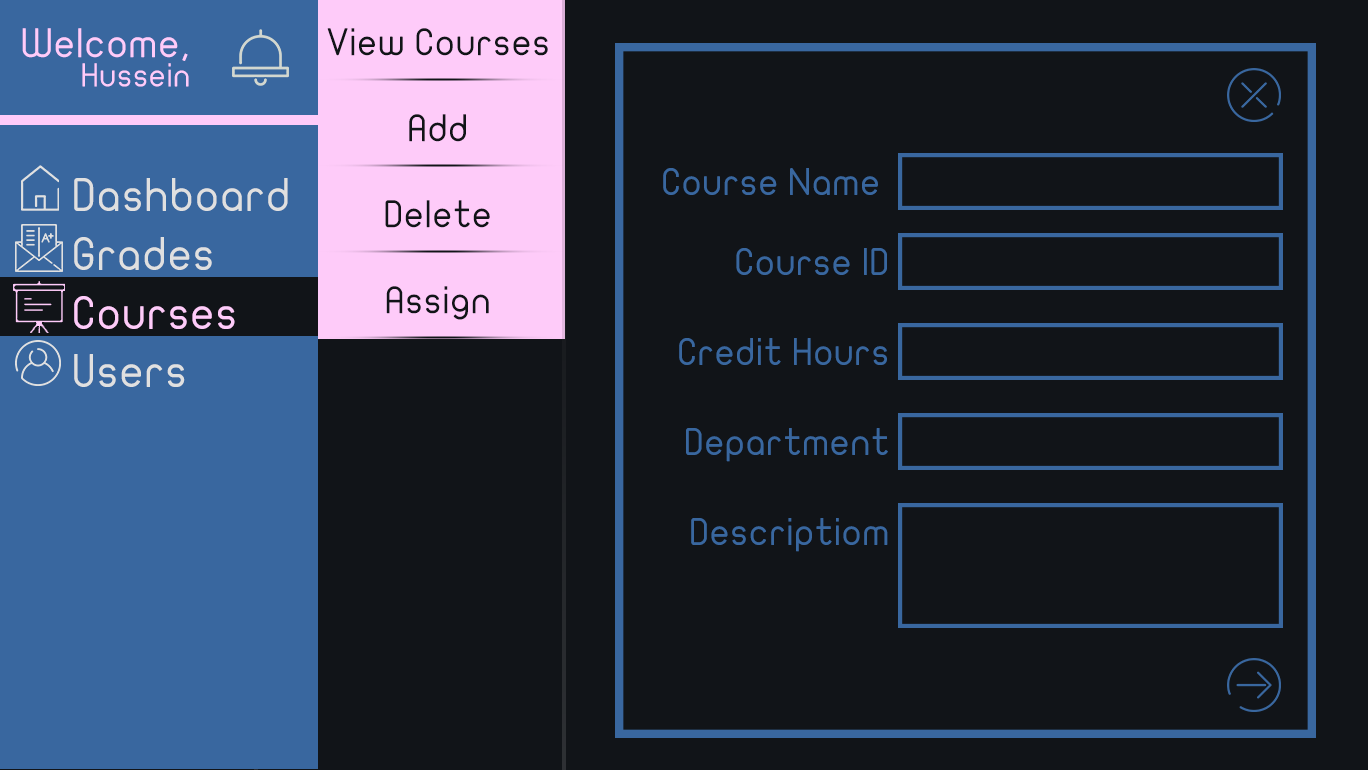
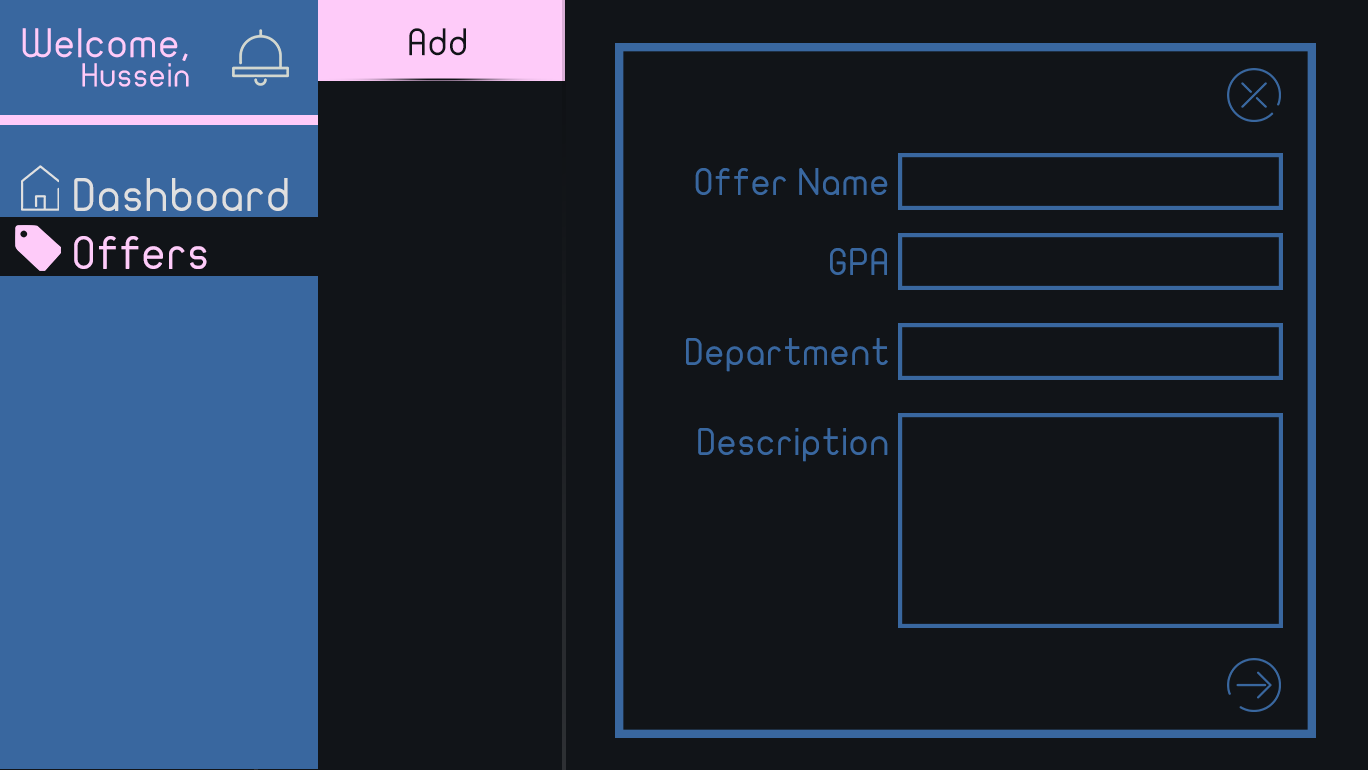
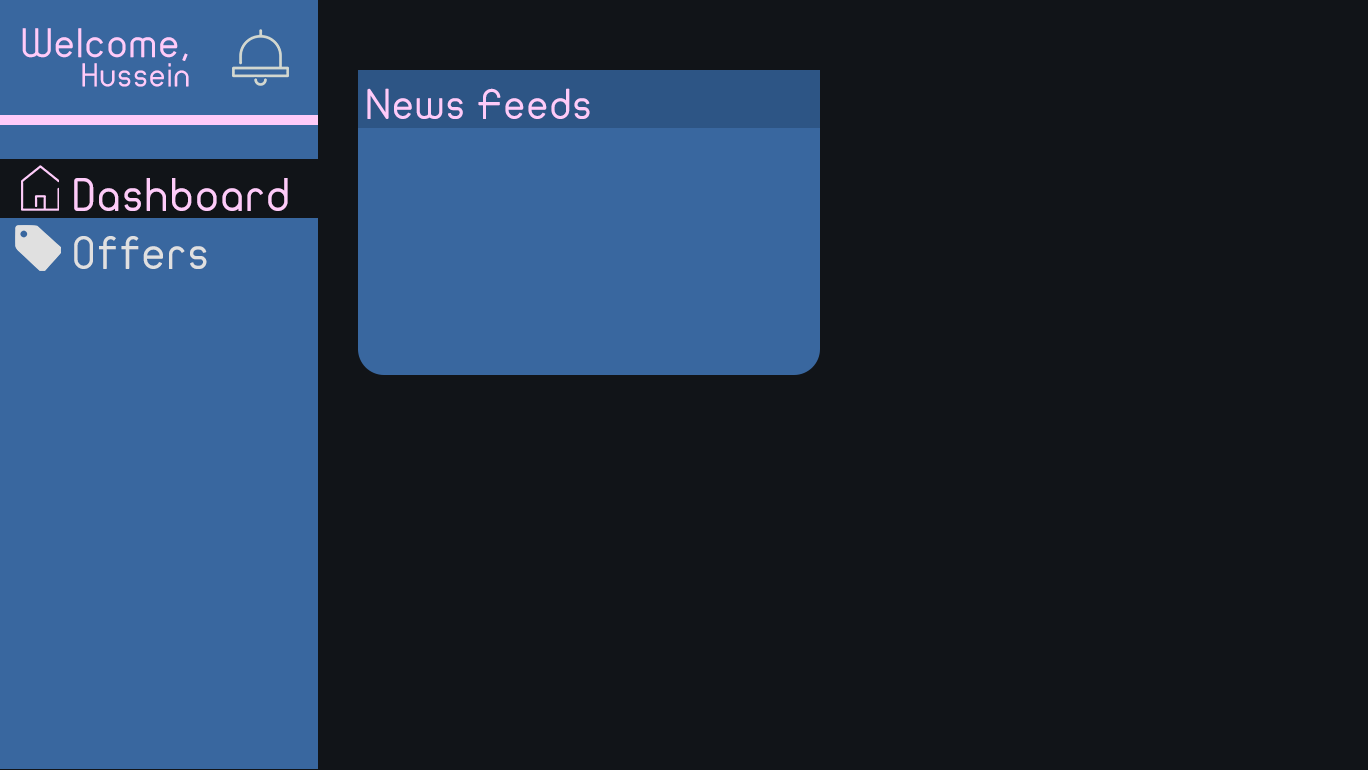
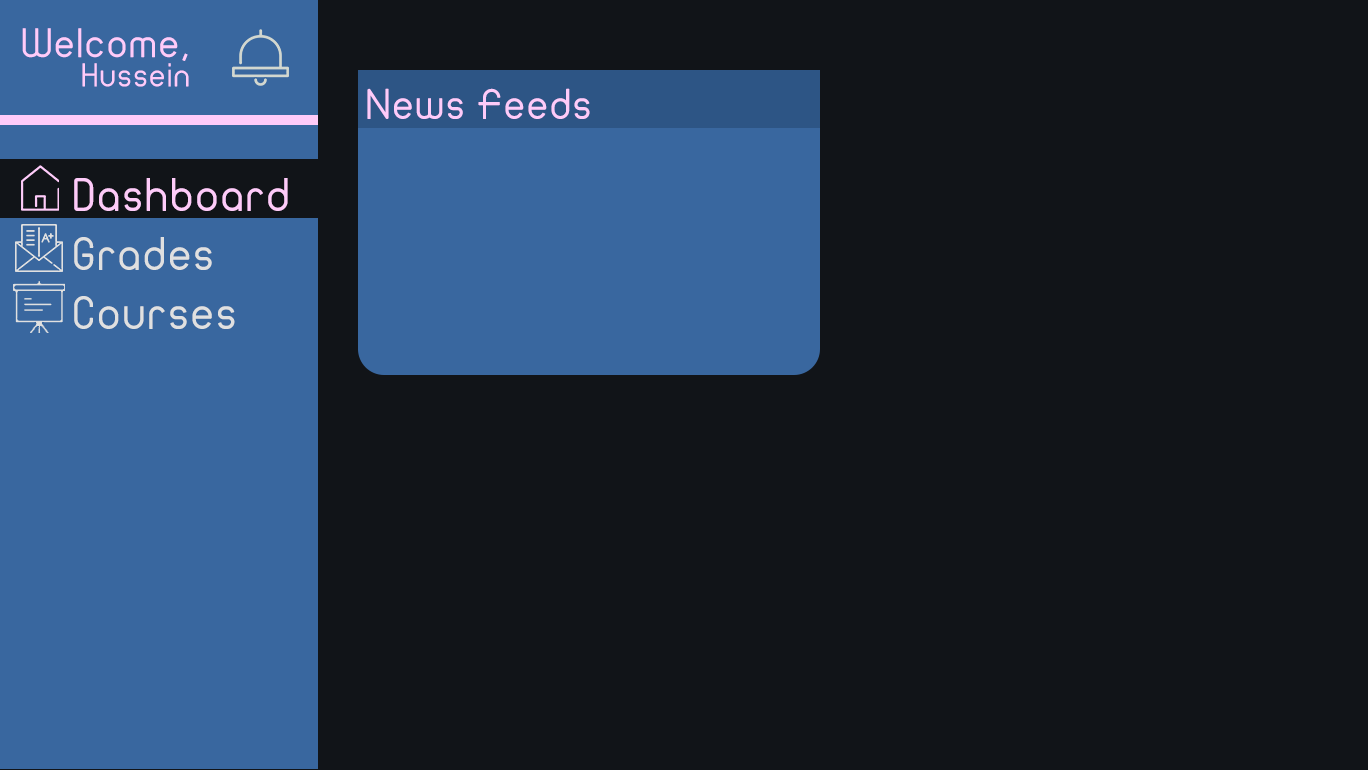
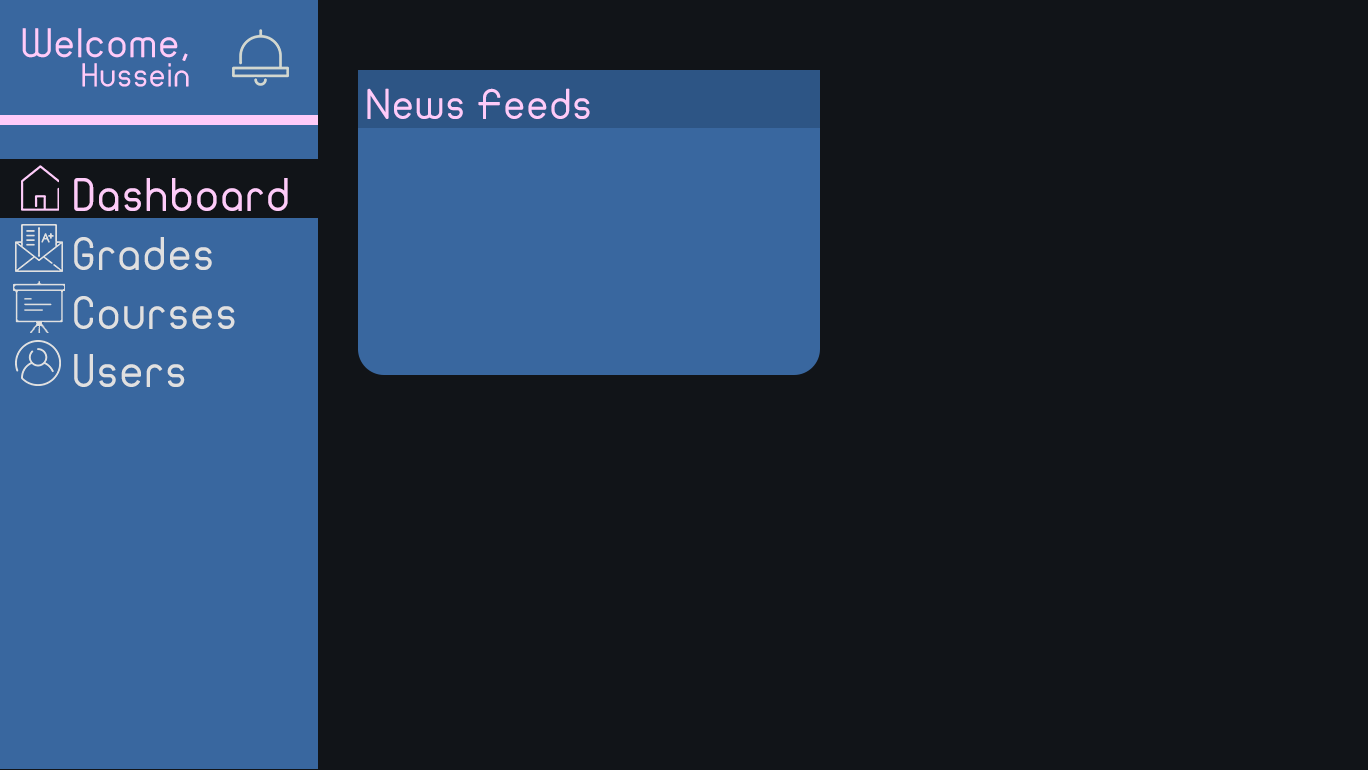
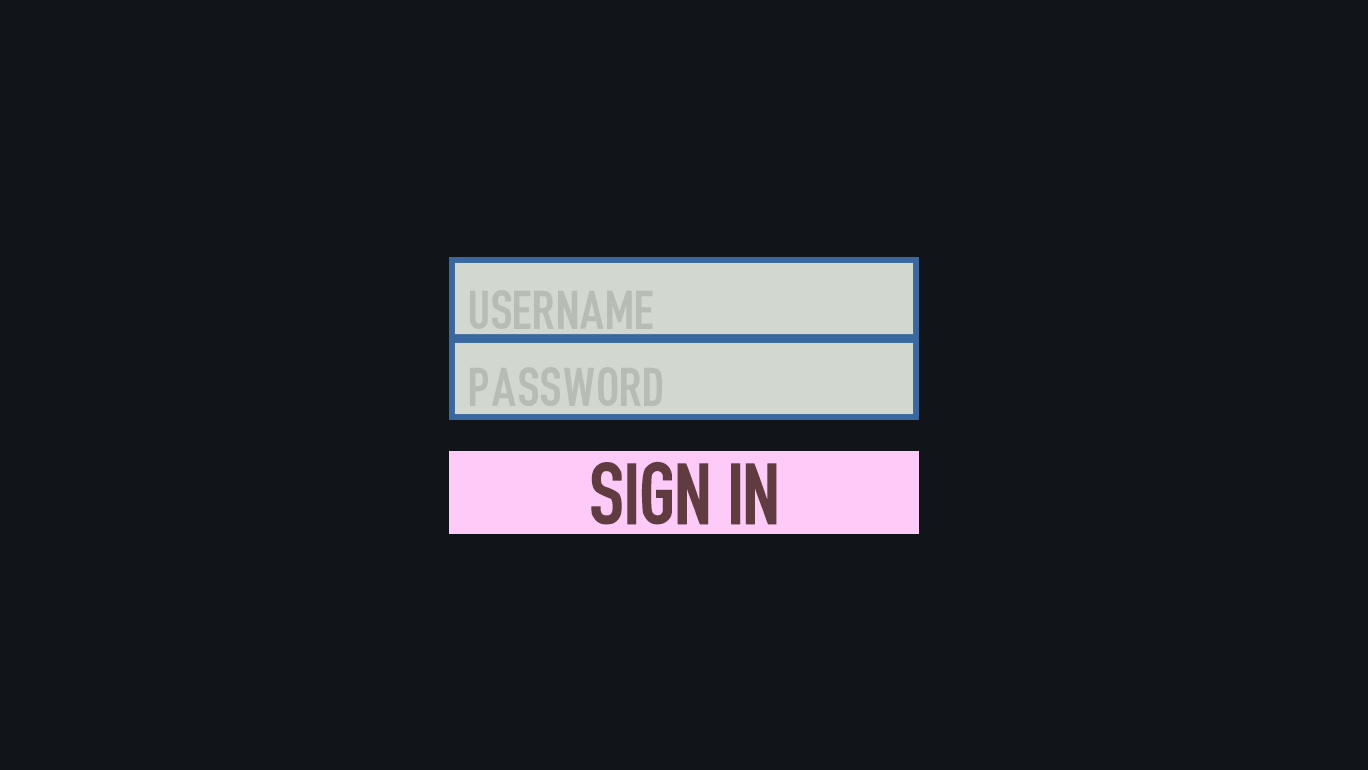
## IV. Physical Entity-Relationship Diagram



## 

## V. User Interface Design

## 



## VI. Bonus: State Diagram and OCL

* **Choose a non-trivial entity that has different states and develop a state diagram for it.**
* **Choose 5 functional requirements that cannot be represented by class and sequence diagrams and develop OCL constraints for them.**

# Ownership Report

* **Remove the following notes and any red notes**
* **For every item in this document, write the owners. If someone is owner of something, s/he understands it 100.%**
* **Team leader must verify the table with the team members.**

|  |  |
| --- | --- |
| **Item** | **Owners** |
|  |  |
|  |  |

# References

* http://www.mhhe.com/engcs/compsci/pressman/graphics/Pressman5sepa/common/cs1/design.pdf

# Authors

* Mostafa Saad and Mohammad El-Ramly