SE-I COURSE PROJECT (PHASE 2 COVER SHEET)

Discussions Scheduled for Week 11 or 12 *(Specific dates TBA later).*

- Print 1 copy of this cover sheet and attach it to a printed copy of the documentation *(SRS, … etc.)*. You must submit softcopies of all your documents *(as PDFs)*; details will be announced later.
- Please write all your names in Arabic. ○ Please make sure that your students' IDs are correct.
- Handwritten Signatures for the attendance of all team members should be filled in before the discussion.
- Please attend the discussion on time *(announced separately)*, late teams will lose 3 grades.

Project Name: _____

Team Information *(typed not handwritten, except for the attendance signature)*:

	ID [Ordered by ID]	Full Name [In Arabic]	Attendance [Handwritten Signature]	Final Grade
1				
2				
3				
4				
5				

Grading Criteria:

Items		Grade	Notes
System Architecture – *including applied Architectural Pattern(s)*	2		
Sequence Diagram(s) *including* ***System Sequence Diagrams (SSDs)***	3		
Collaboration/Communication Diagram(s)	2		

Class Diagram *(3 versions)* **1) An initial version based on the requirements and UseCase/Activity diagrams.** **2) An intermediate version based on the interaction diagrams.** **3) A final version, after applying the design pattern(s) and any other modifications.**	**4.5**		
Package Diagram(s)	**2**		
3 Mandatory Design Patterns Applied *(Including a typed description)*	**4.5**		
Object Diagrams *(Including object diagrams that illustrate the preconditions and the post-conditions of selected functions)*	**2**		

N.B. .. **You must update and resubmit the initial part of the documentation submitted in phase 1** (including the Functional / Non-Functional requirements, Use-case Diagrams & Descriptions, Activity Diagrams, .. etc.).

Teaching-Assistant's Signature: _____

_____

20

1. INTRODUCTION

1.1 PURPOSE

The purpose of this document is to make use of the resources that will help the student to gain all the information that a Student needs from the portal in the form of reports and simple to use. Our project is based on a database, which stores and maintains the information of different modules within the system.

1.2 DOCUMENT CONVENTIONS

- Database
- Entity Relationship

1.3 INTENDED AUDIENCE AND READING SUGGESTIONS

This project is a prototype for the Campus Portal system This project is useful for the Admin and for the students. this SRS contains the admin who has the ability to add information to the system & contains the student who has the ability to view their information.

1.4 PROJECT SCOPE

The purpose of the Campus Portal system is to help the student to gain all the information that a Student needs from the portal and to create a convenient and easy-to-use application for Students.

The system is based on a relational database with Students & admin and system functions. We will have a database server supporting hundreds of Students names and admins.

Above all, we hope to provide a comfortable user experience for the student. This project is based on a web application that is have information on different sides for the campus like (courses, students, timetables, etc.).

Admin can add information to the system that is stored in the database through web application because the admin has access to the system.

This application works in multiple PC's installed on multiple Computers by sharing same database by which users of different departments can use it sitting at different locations simultaneously

1.5 REFERENCES

- Software Engineering (9th Edition) for Ian Sommerville

- UML @ Classroom - An Introduction to Object-Oriented Modeling (2015) by Seidlet al.

2. OVERALL DESCRIPTION

2.1 PRODUCT PERSPECTIVE

A distributed Campus database system stores the following information.

- **student details:**
  which includes student name , the semester , his weekly marks,PT1&PT2 , timetable , courses, booklets …..

- **admin details:**
  which includes admin name and the information about the student.

2.2 PRODUCT FEATURES

The major features of Campus database system as shown in entity–relationship model (ER model)

The diagram shows the layout of Campus database system – entity–relationship model

2.3 USER CLASS and CHARACTERISTICS

Users of the system should be able to View student information

- The Student does not have to go personally to college office for the enquiry.
- The Student has to Login and then the portal is Available to him keeping the data secure.
- The application enables the students to be updated with college cultural activities.
- If application saves time for the student as well as teaching and non-teaching staffs.

The **student** should be able to do the following functions:

- View Timetable
- View Booklet
- View Test Solutions
- View Video Links
- View Weekly Marks
- View PT1/PT2
- University Link

The **admin** should be able to do the following functions:

- Add Student
- Add Course
- Add Timetable
- Add Booklet

- Add Test Solutions
- Add Video Links
- Add Weekly Marks
- Add PT1/PT2

2.4 OPERATING ENVIRONMENT

- distributed database
- client/server system
- Operating system: Windows.
- database: MySQL database
- platform: HTML /CSS/PHP

2.5 DESIGN and IMPLEMENTATION CONSTRAINTS

There are no Design and Implementation Constraints

2.6 User Documentation

- krazytech.com
- drawsql.app
- Visual paradigm online
- Wondershare EdrawMax
- Microsoft Visio Pro 2013

3.System Features

Functional Requirements

REQ-1: There is a common Log-in for both admins and students.
REQ-2. After log-in, credentials are checked, and is remembered once logged in
REQ-3. Admin shall add students
REQ-4. Password shall be generated by the system & is sent to the student's mail ID.
REQ-5. Admin shall add courses & its subjects shall be semester wise
REQ-6. Admin shall add Timetable in the form of .JPG & they must be semester wise.
REQ-7. Admin shall add booklets and they must be .pdf files
REQ-8. Admin shall add test solutions and they must be .pdf files
REQ-9. Admin shall add video links and they must be in the form of URL.
REQ-10. Admin shall add weekly marks, they must not be subject wise and out of 25.
REQ-11. Admin shall add PT1/PT2, they must be subject wise and out of 25.
REQ-12. Student shall login into the app with password sent to his/her E-mail ID.
REQ-13. Student can check timetable which is an image, the image can be pinch zoomed.
REQ-14. Student can see a list of booklets which are viewed by default by Google Docs.
REQ-15. Student can see a list of test solutions which are viewed by default by Google Docs.
REQ-16. Student can download booklets and test solutions.
REQ-17. Student can check out video links which are directed to the dedicated web link.
REQ-18. Student can see his weekly marks, which are displayed as a Bar report.
REQ-19. Student can see his marks in the form of 2 reports, which are Line chart and Pie chart.

REQ-20. Line Chart is divided into 3 fragments (Highest, Average and Students Marks) to help the student with his progress and rank.

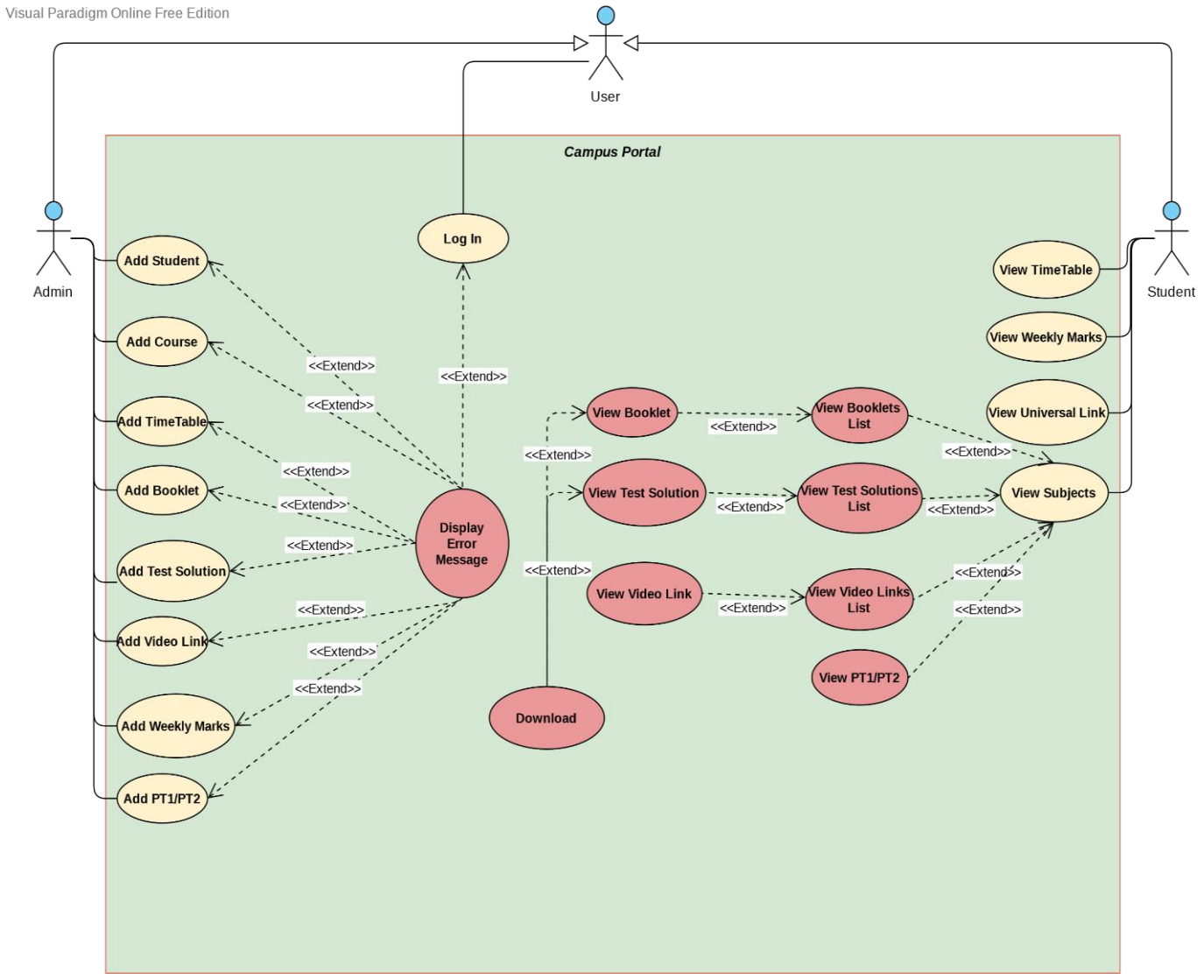
REQ-21. Pie Chart shows only the student's marks.

REQ-22. Student can see university link which is redirected to the web.

4.Other Nonfunctional Requirements:

1. The Student has to Login and then the portal is Available to him keeping the data secure. (security)
2. The App makes it very easy to use and makes it simple to navigate between different things on portal (Performance)
3. It provides the means to enquire online through the App without having to go to college and enables the students to be updated with college cultural activities. (Performance)
4. Application saves time for the student as well as teaching and non-teaching staffs. (Software Quality Attributes)
5. Student can only check timetable limited to his/her course and semester. (Business Rules)
6. Student can only check list of booklets limited to his/her course and semester. (Business Rules)
7. Student can only check list of test solutions limited to his/her course and semester. (Business Rules)

Use case diagram



Use-Case Description

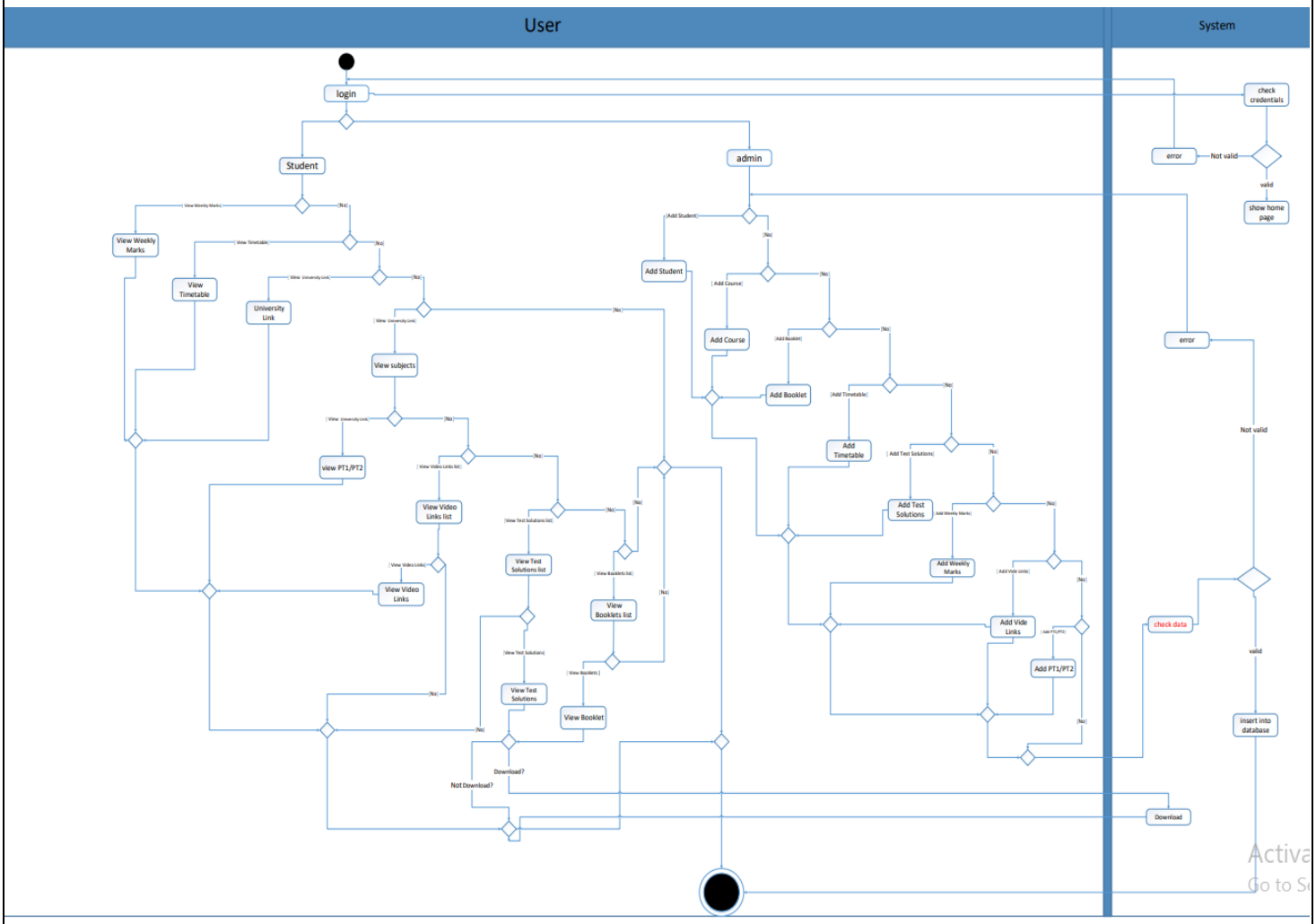
Identifier	UC1
Name	Campus Portal Login
Initiator	User
Pre-conditions	User must have an already existing account.
Post-Conditions	The system displays the relevant homepage according to the user's role, whether he is a student or an admin.
Main success scenario	The user enters his credentials into the login form, the system confirms that the credentials are valid and according to his role, it displays either the student homepage or the admin homepage.
Goal	The user accesses the Campus portal to be able to access the relevant functions according to their role.

Identifier	UC2
Name	Admin Homepage
Initiator	Admin
Pre-Conditions	The admin must have successfully logged in, and the homepage shown must be the admins' homepage.
Post-Conditions	Any addition of data by the admin is validated then stored in the database.
Main Success Scenario	1. After the admin logs in successfully, the admins' homepage is displayed showing all the relevant functions of the admin. 2. The admin clicks on a certain function from the available functions (Add Student/Course/Timetable /Booklet/Test Solution/Video Link/ Weekly Marks/ PT1 or PT2). 3. The System confirms that the added data by the admin is valid and adds it into the database.
Goal	The admin successfully adds any chosen data.

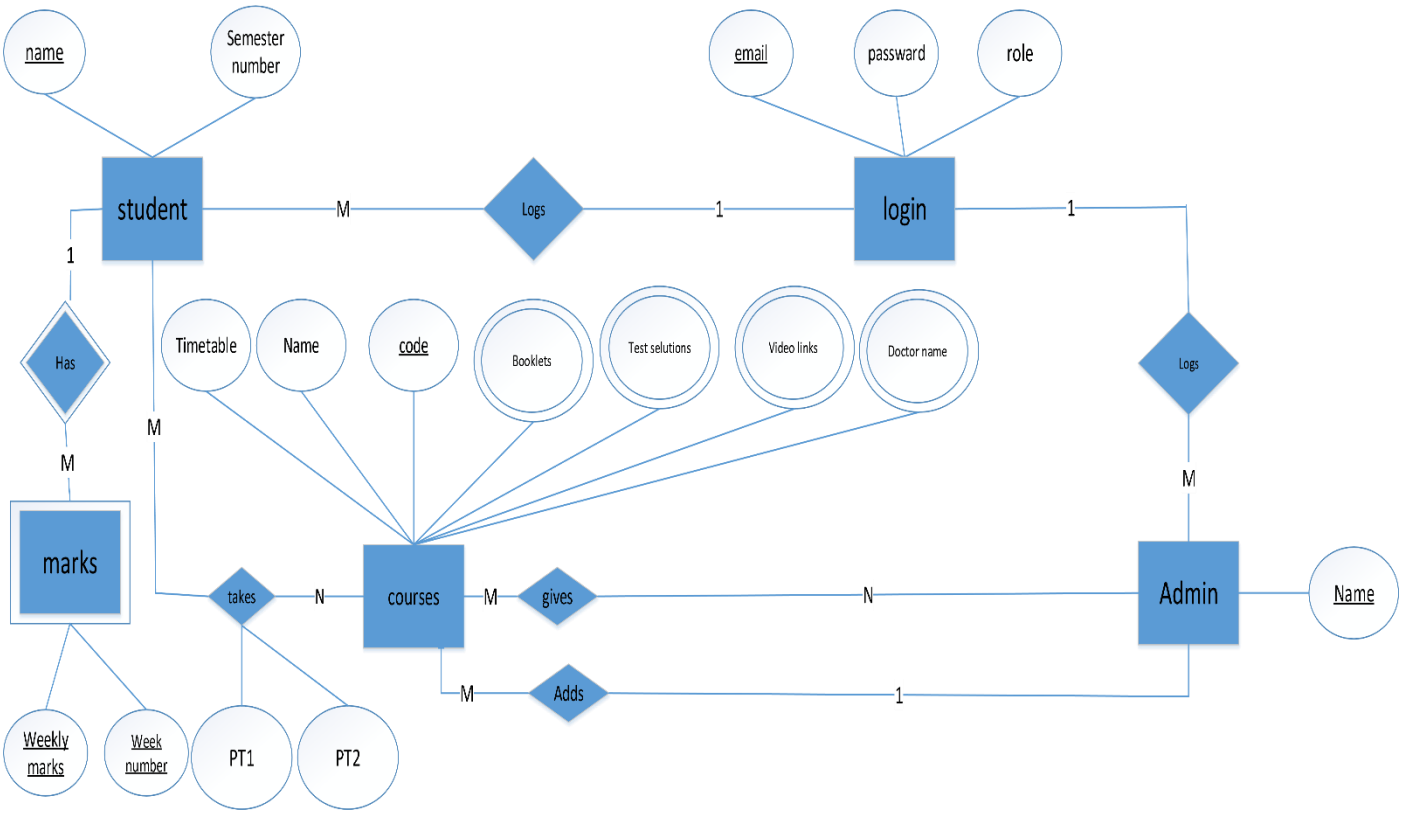
Identifier	UC3
Name	Student Homepage

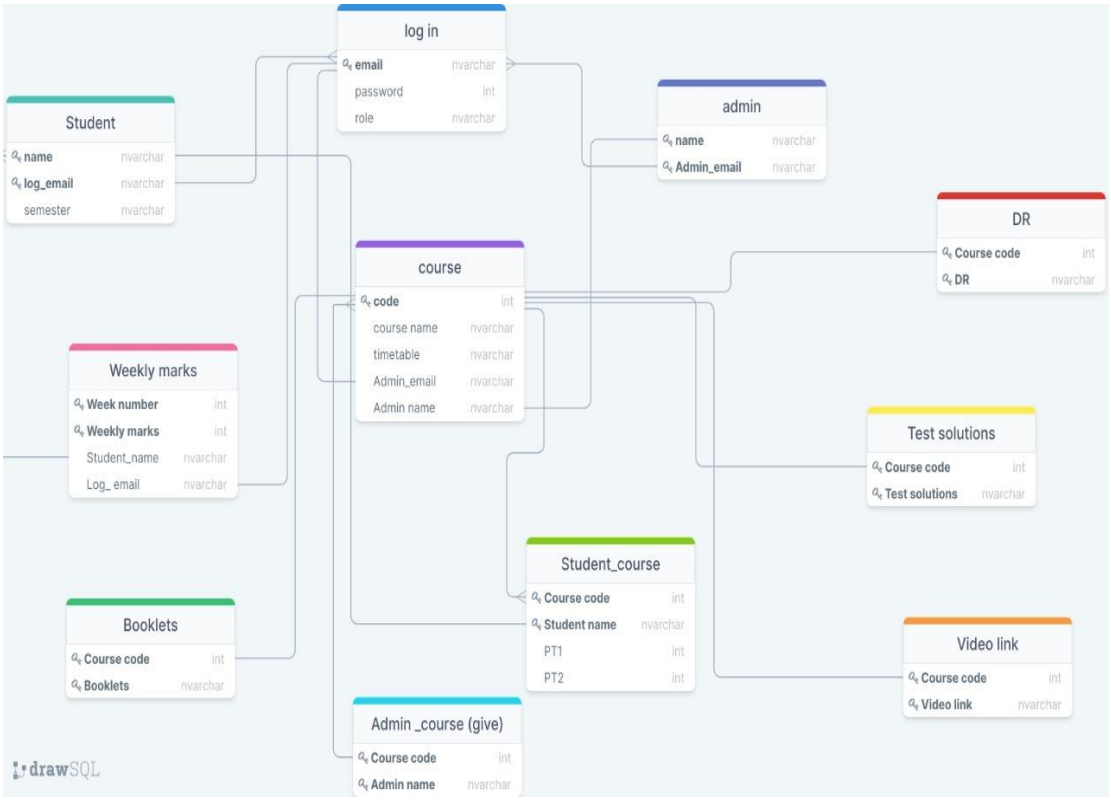
Initiator	Student
Pre-Conditions	The Student must have successfully logged in, and the homepage shown must be the students' homepage.
Post-Conditions	The student can only view his own data and is not able to see data of other students.
Main Success Scenario	1. After the student logs in successfully, the students' homepage is displayed showing all the relevant functions of the student. 2. The student clicks on a certain function from the available functions: • View his own Timetable • View his own weekly marks. • View Universal Link • View Subjects' List > View Booklets List > View Booklet > Download • View Subjects' List > View Test Solutions List> View Test Solution > Download • View Subjects' List > View Video Links List > View Video Link • View Subjects' > View PT1/PT2.
Goal	The student successfully views and possibly downloads any chosen data.

Activity diagram

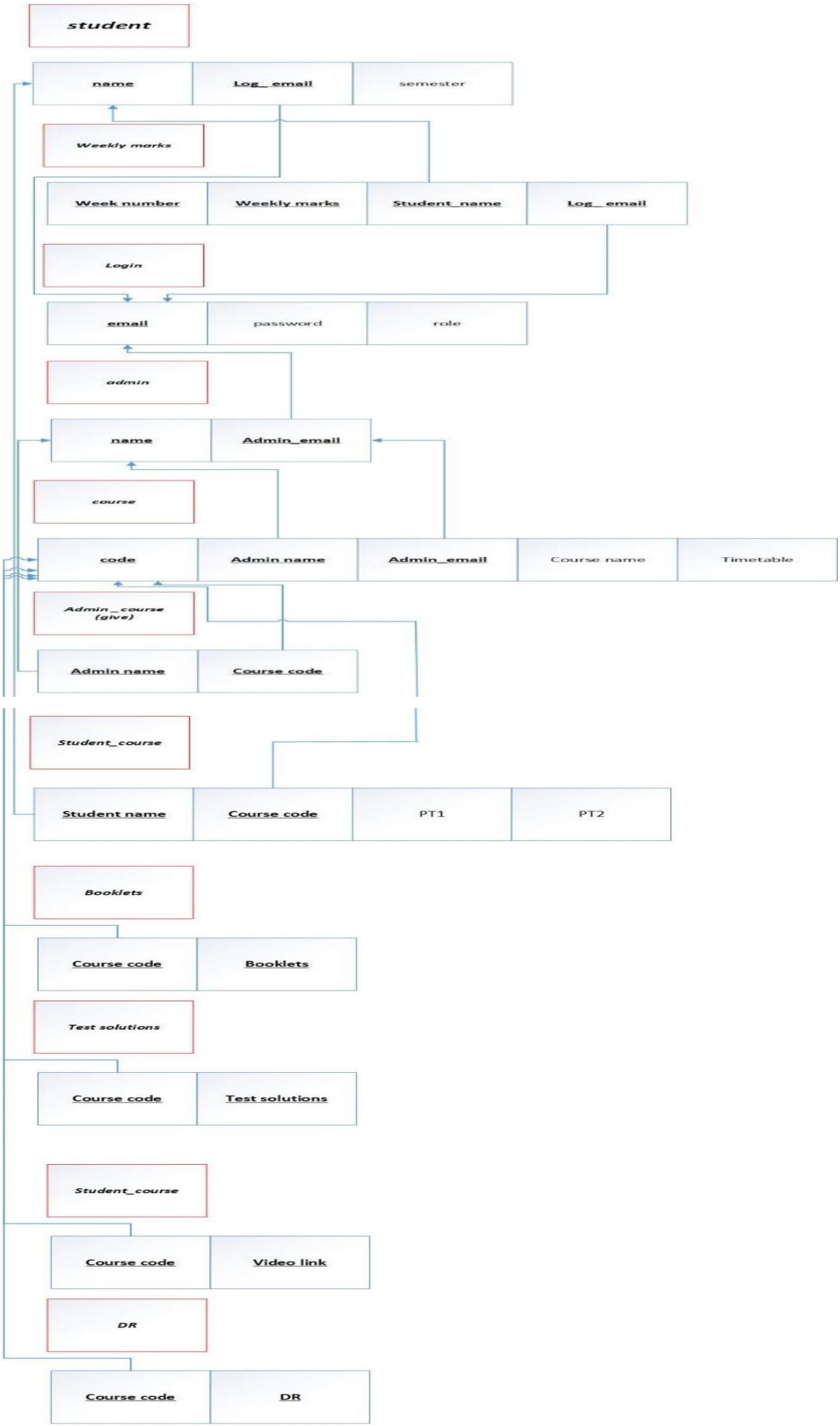


ERD diagram

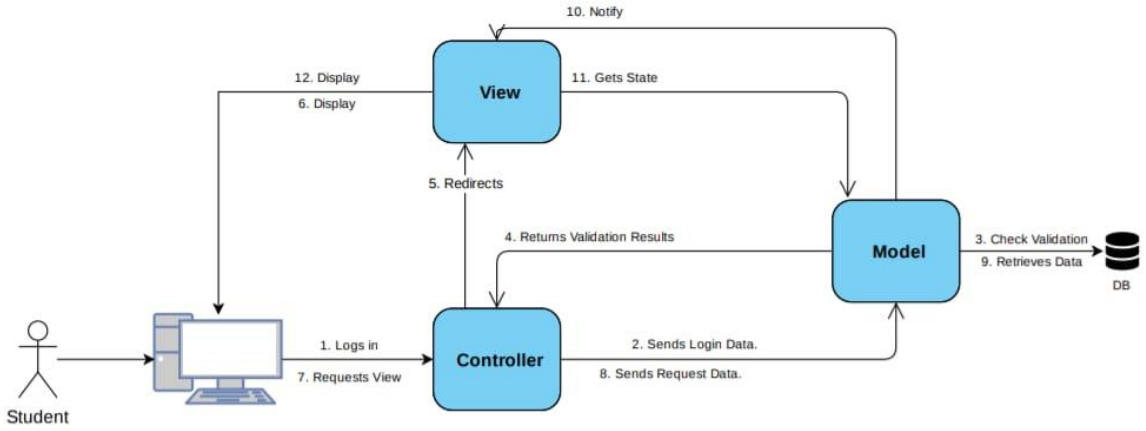

name
Semester number
email
password
role
student
M
Logs
1
login
1
1
Has
Timetable
Name
code
Booklets
Test solutions
Video links
Doctor name
Logs
M
M
M
marks
takes
N
courses
M
gives
N
Admin
Name
M
Adds
1
Weekly marks
Week number
PT1
PT2

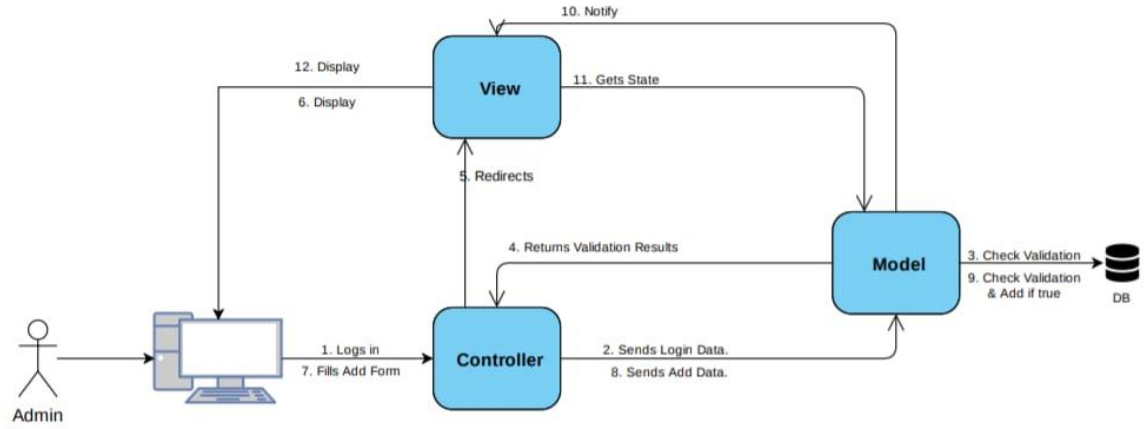

log in
email nvarchar
password int
role nvarchar
admin
name nvarchar
Admin_email nvarchar
Student
name nvarchar
log_email nvarchar
semester nvarchar
DR
Course code int
DR nvarchar
course
code int
course name nvarchar
timetable nvarchar
Admin_email nvarchar
Admin name nvarchar
Weekly marks
Week number int
Weekly marks int
Student_name nvarchar
Log_ email nvarchar
Test solutions
Course code int
Test solutions nvarchar
Student_course
Course code int
Student name nvarchar
PT1 int
PT2 int
Booklets
Course code int
Booklets nvarchar
Video link
Course code int
Video link nvarchar
Admin _course (give)
Course code int
Admin name nvarchar
drawSQL

Mapping

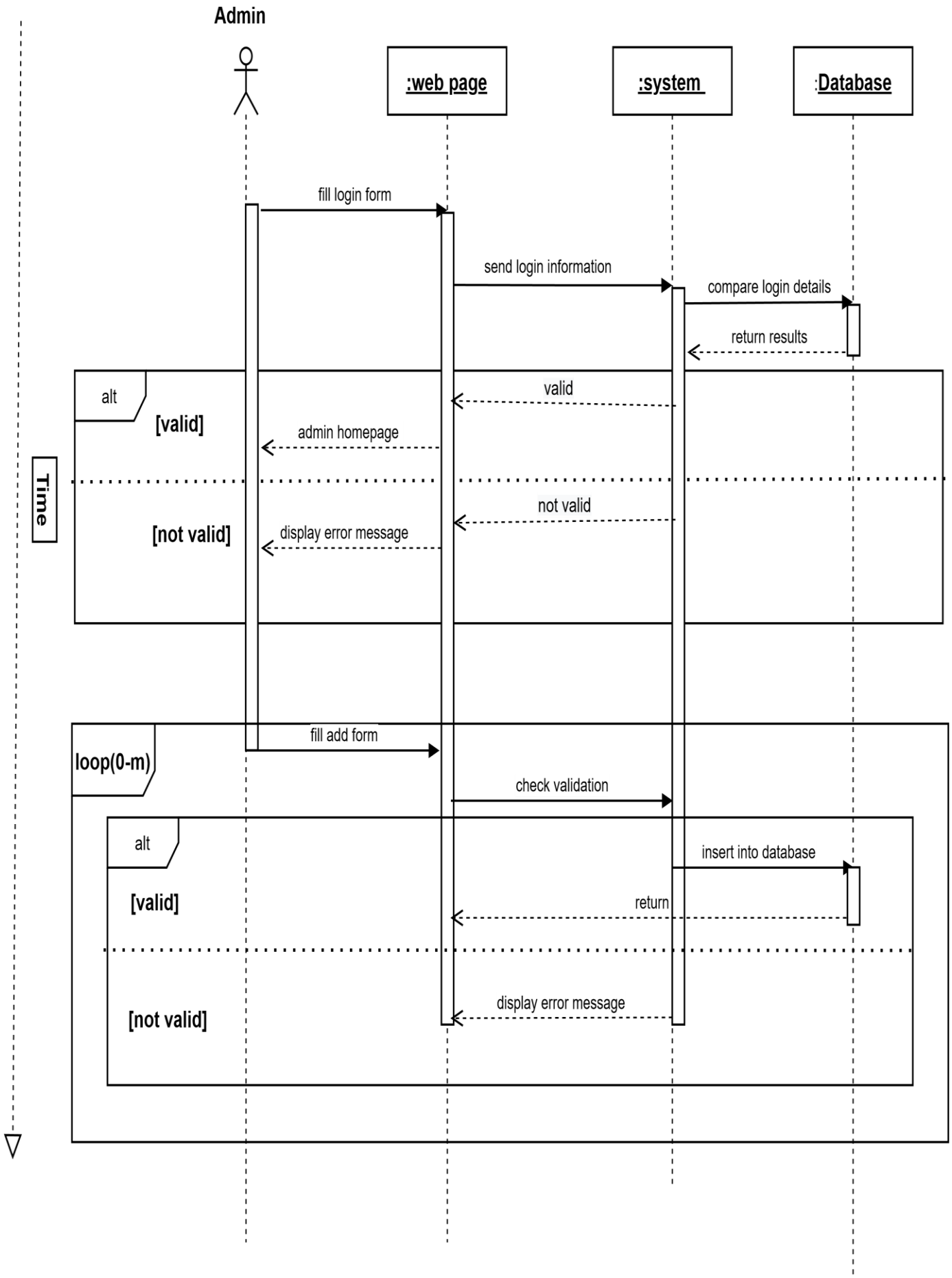

student
name
Log_ email
semester
Weekly marks
Week number
Weekly marks
Student_name
Log_ email
Login
email
password
role
admin
name
Admin_email
course
code
Admin name
Admin_email
Course name
Timetable
Admin _course (give)
Admin name
Course code
Student_course
Student name
Course code
PT1
PT2
Booklets
Course code
Booklets
Test solutions
Course code
Test solutions
Student_course
Course code
Video link
DR
Course code
DR

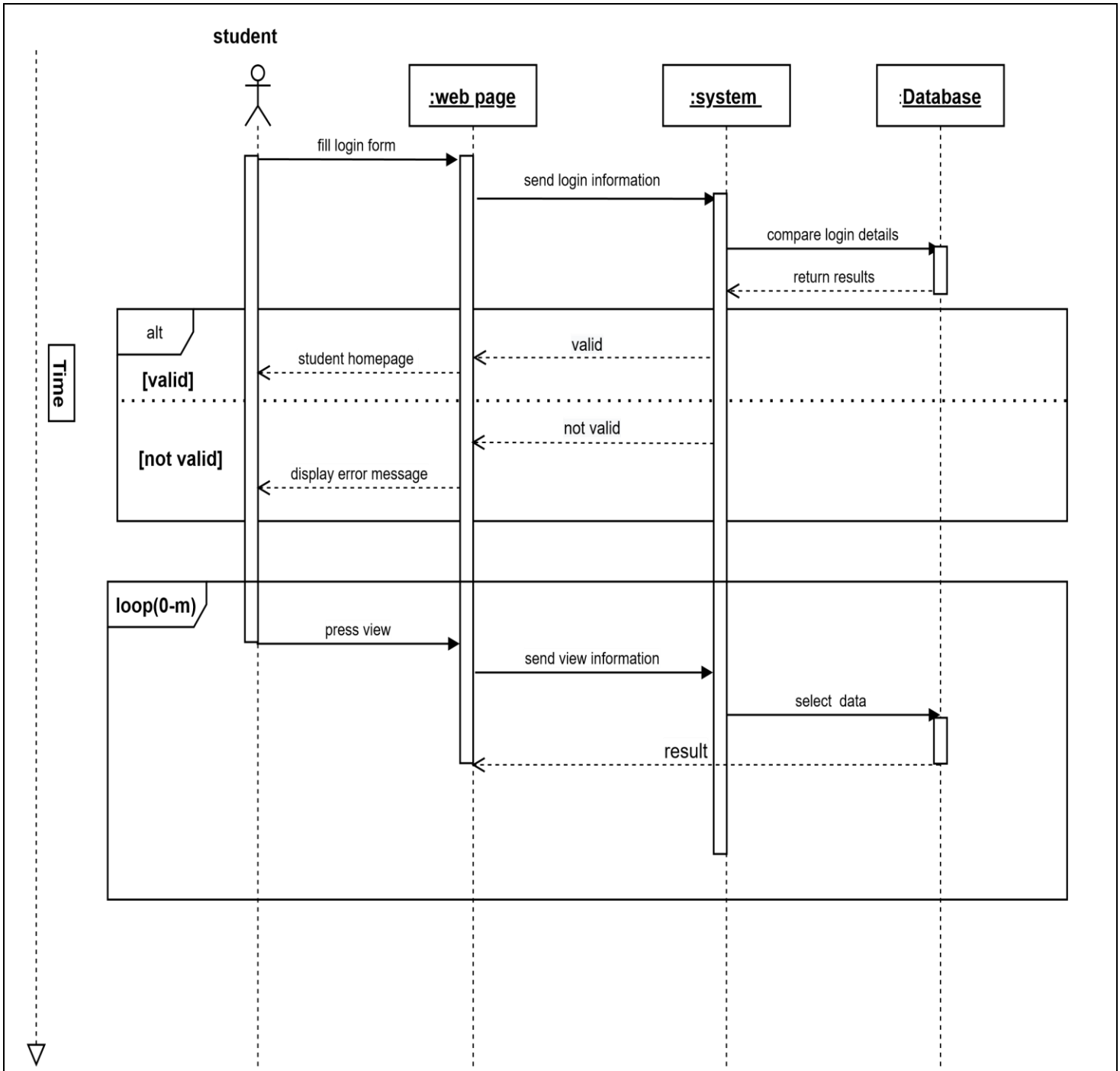
System Architecture

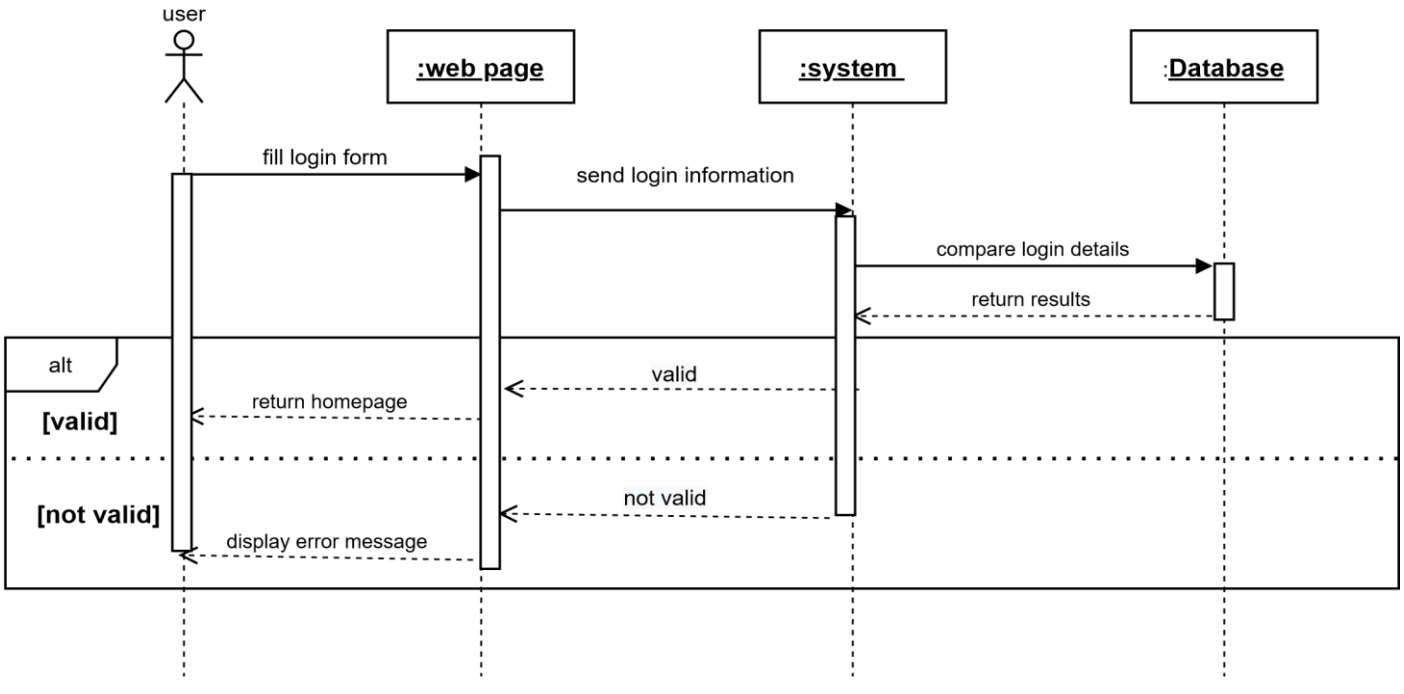




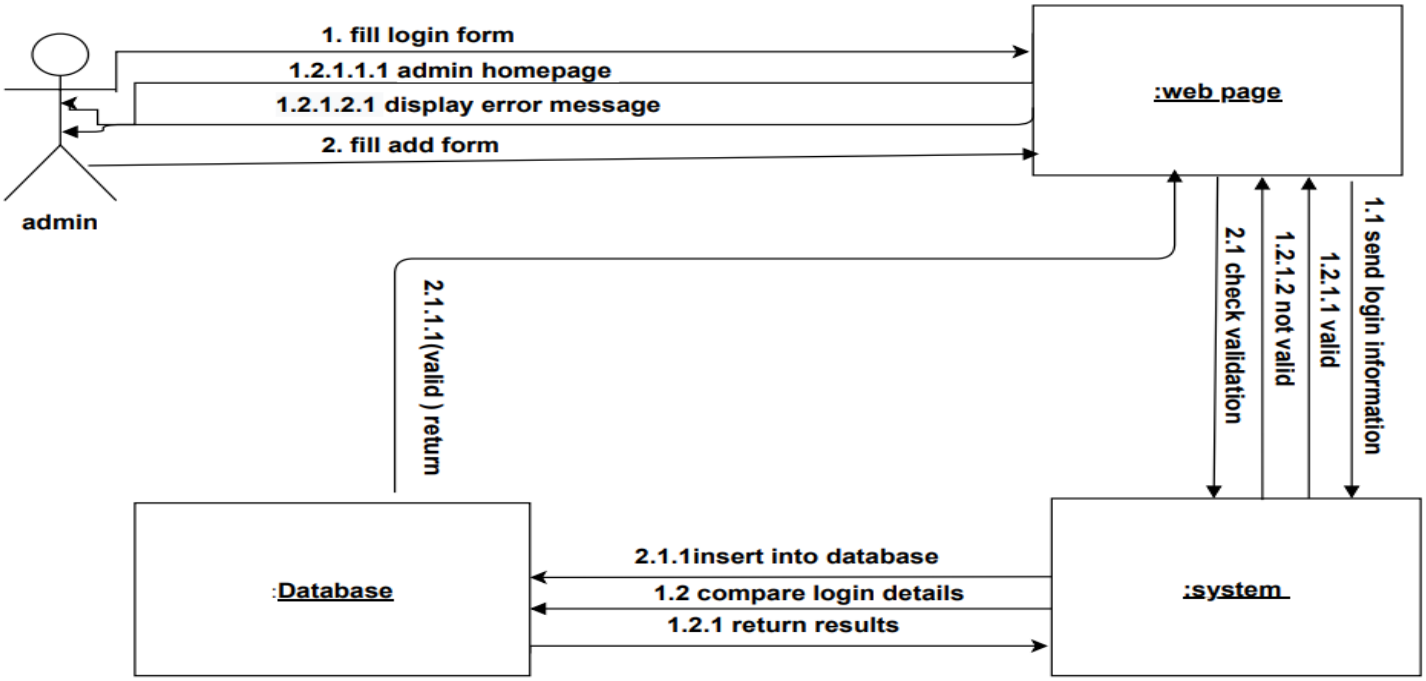
Sequence Diagram(s)

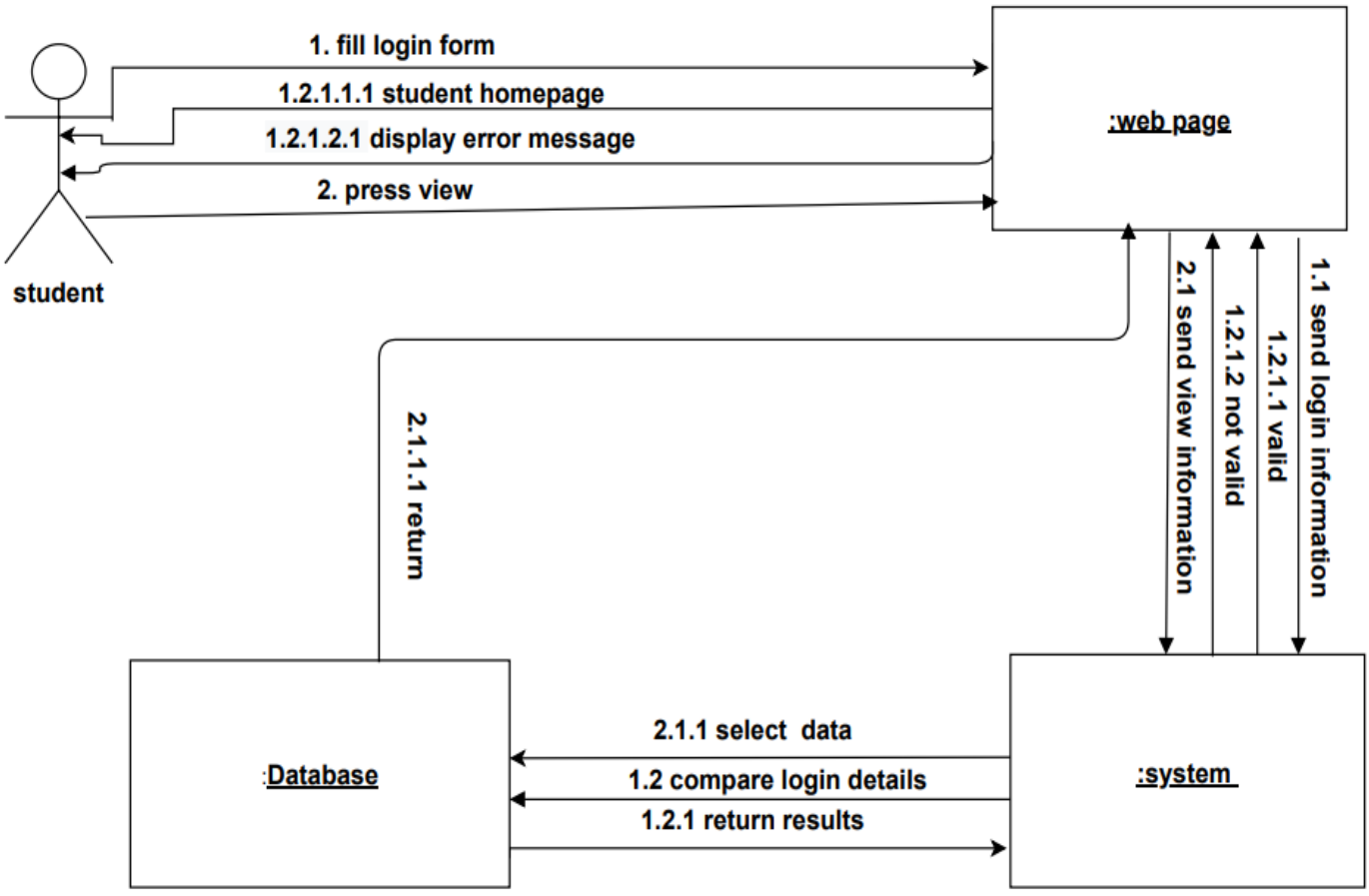



student
:web page
:system
:Database
fill login form
send login information
compare login details
return results
alt
[valid]
[not valid]
valid
student homepage
not valid
display error message
loop(0-m)
press view
send view information
select data
result
Time


user
:web page
:system
:Database
fill login form
send login information
compare login details
return results
alt
[valid]
[not valid]
valid
return homepage
not valid
display error message

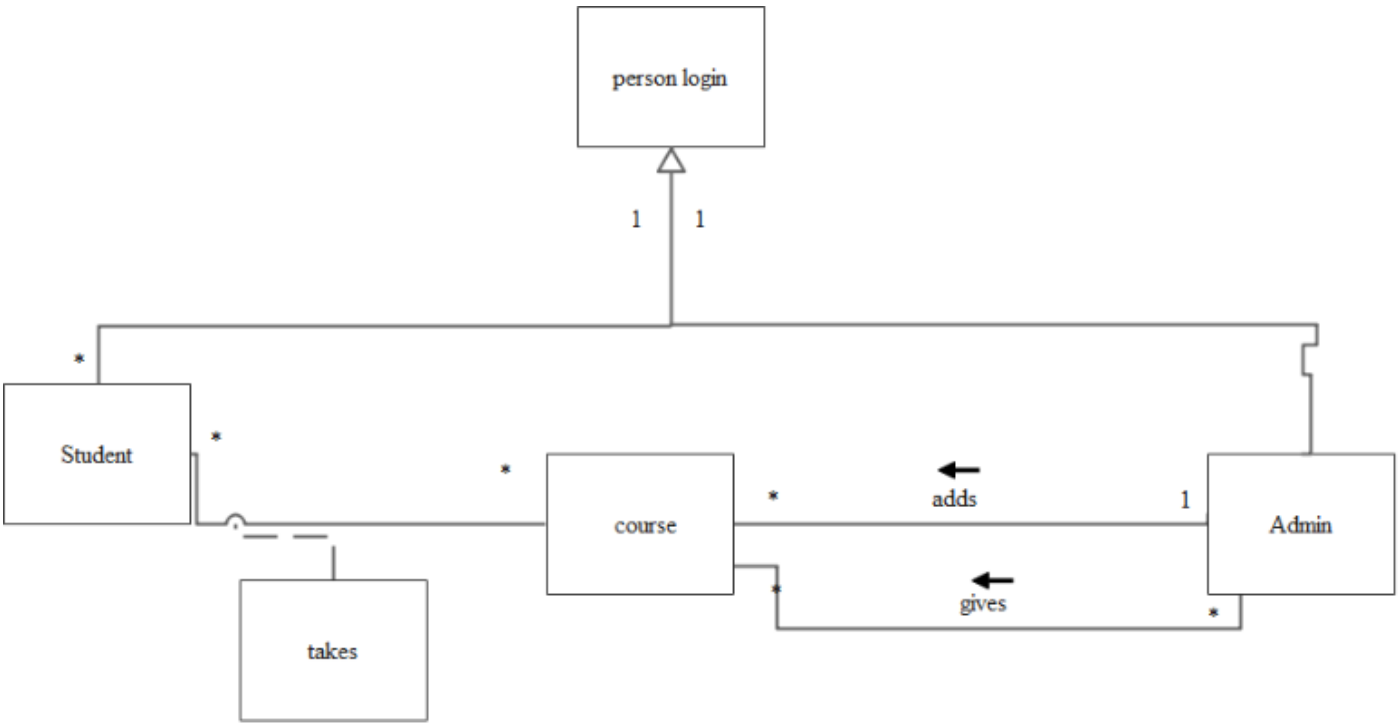
Collaboration/Communication Diagram(s)



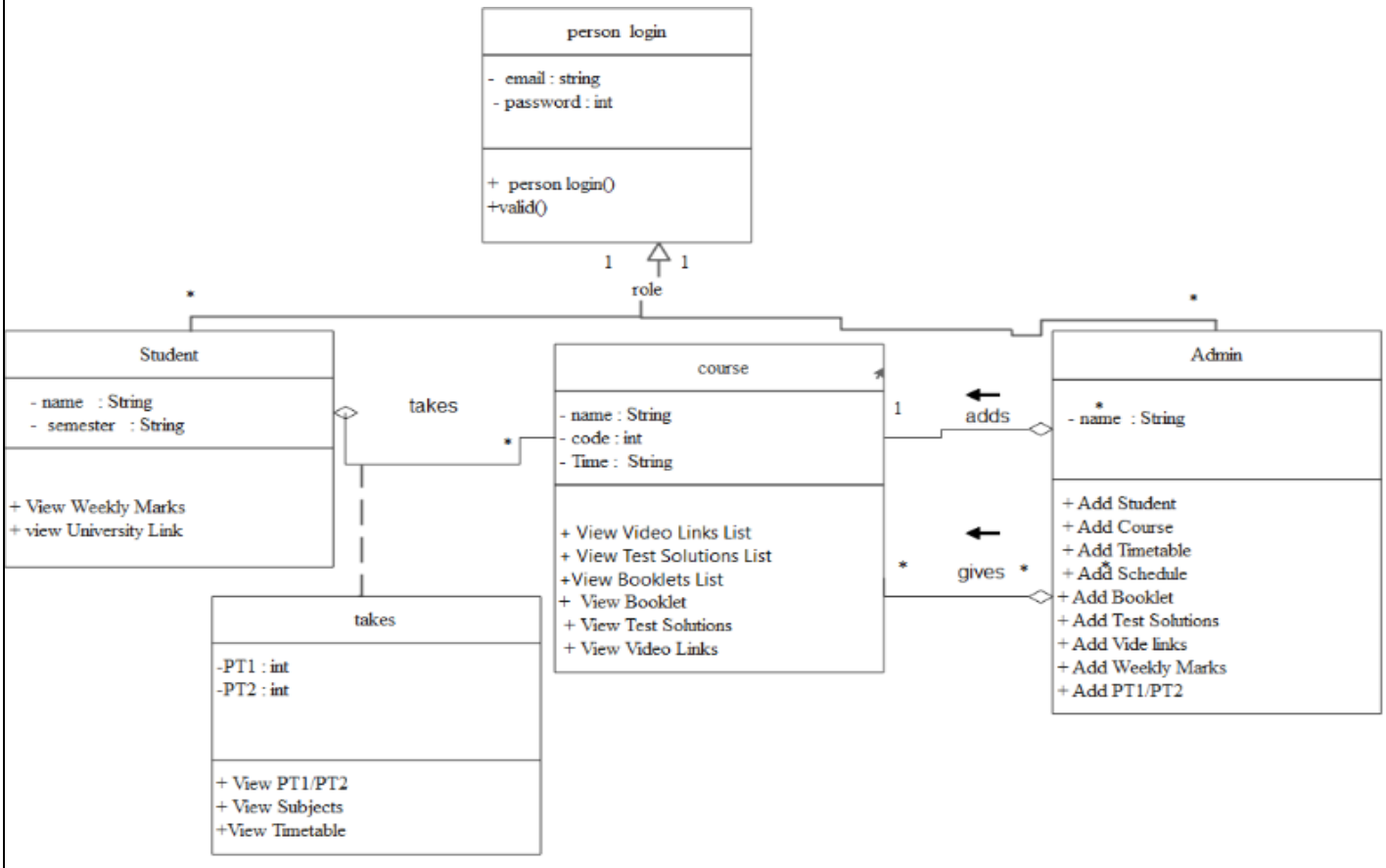


Class Diagram (3 versions)

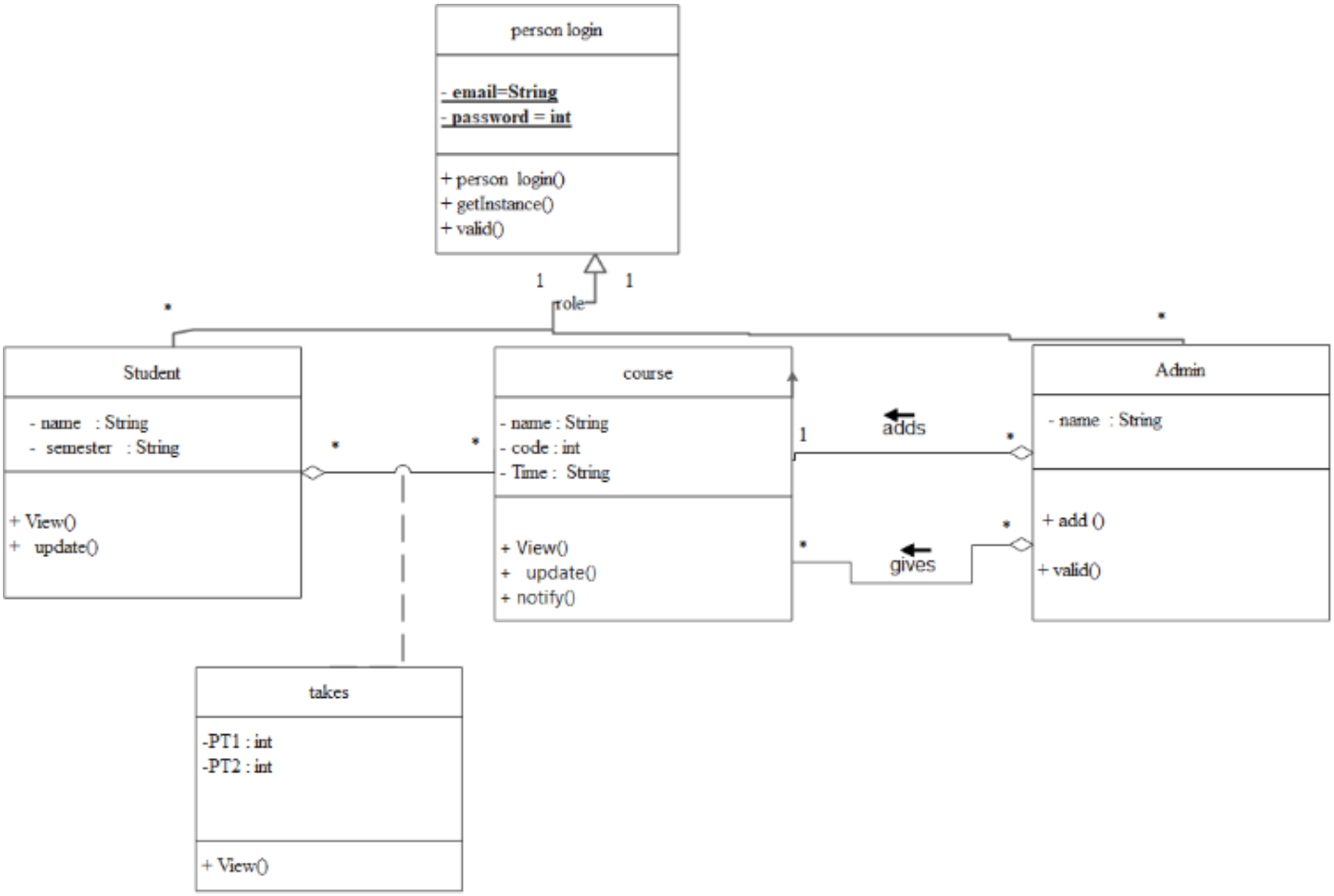
1) An initial version based on the requirements and Use Case/Activity diagrams.



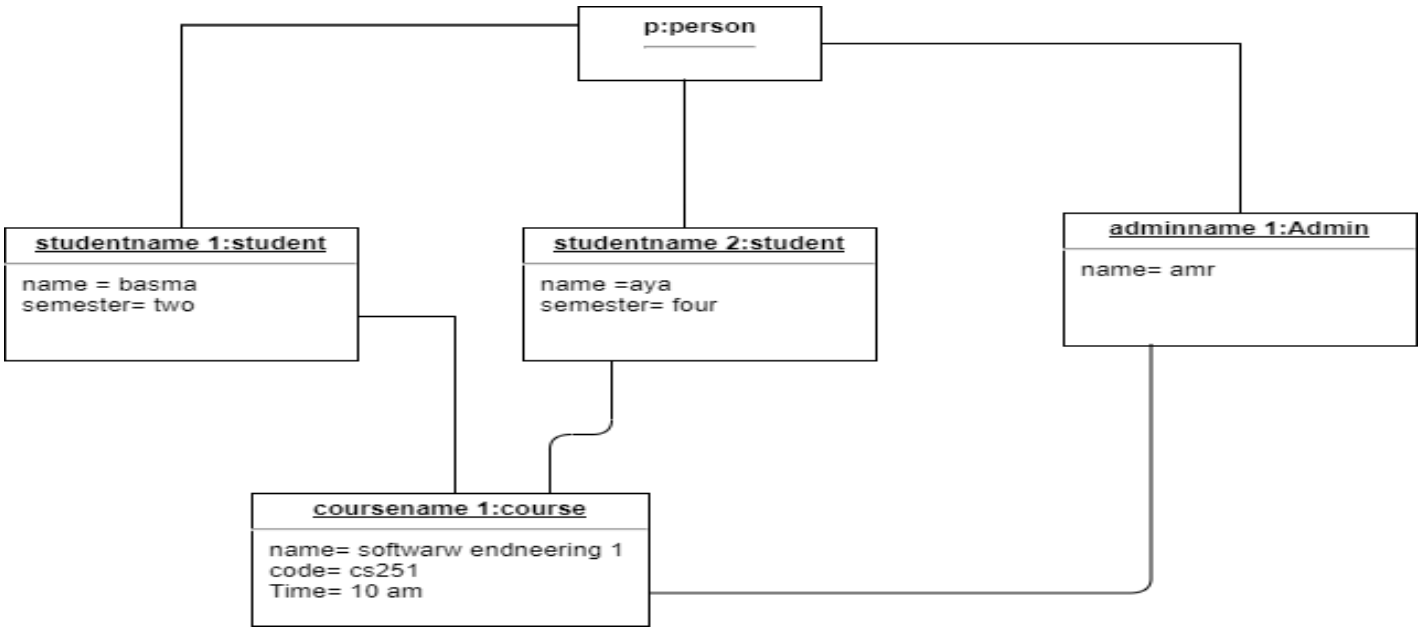
2) An intermediate version based on the interaction diagrams

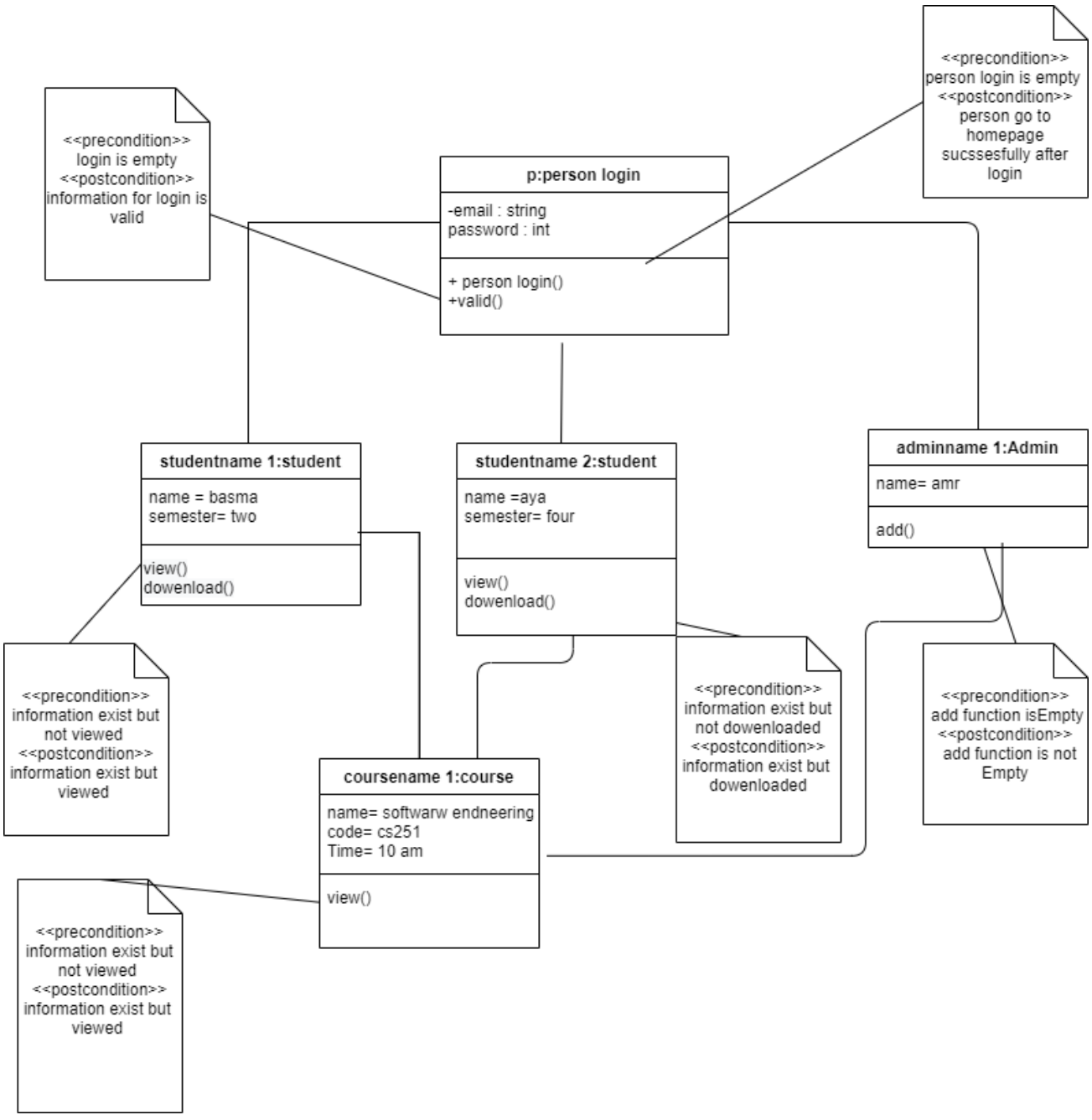


3) A final version, after applying the design pattern(s) and any other modifications.

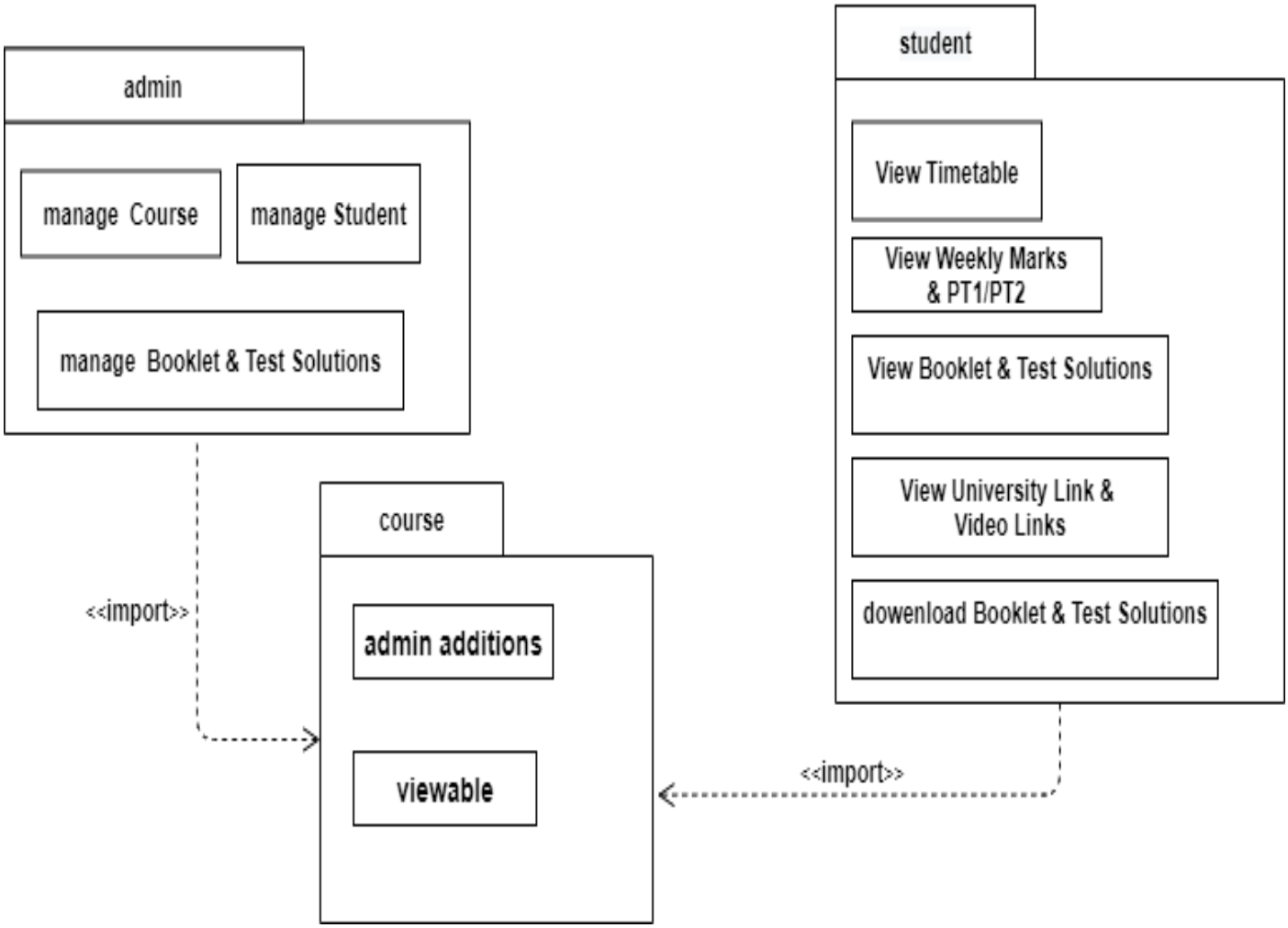


Object Diagrams



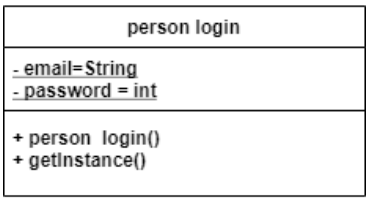


Package Diagram



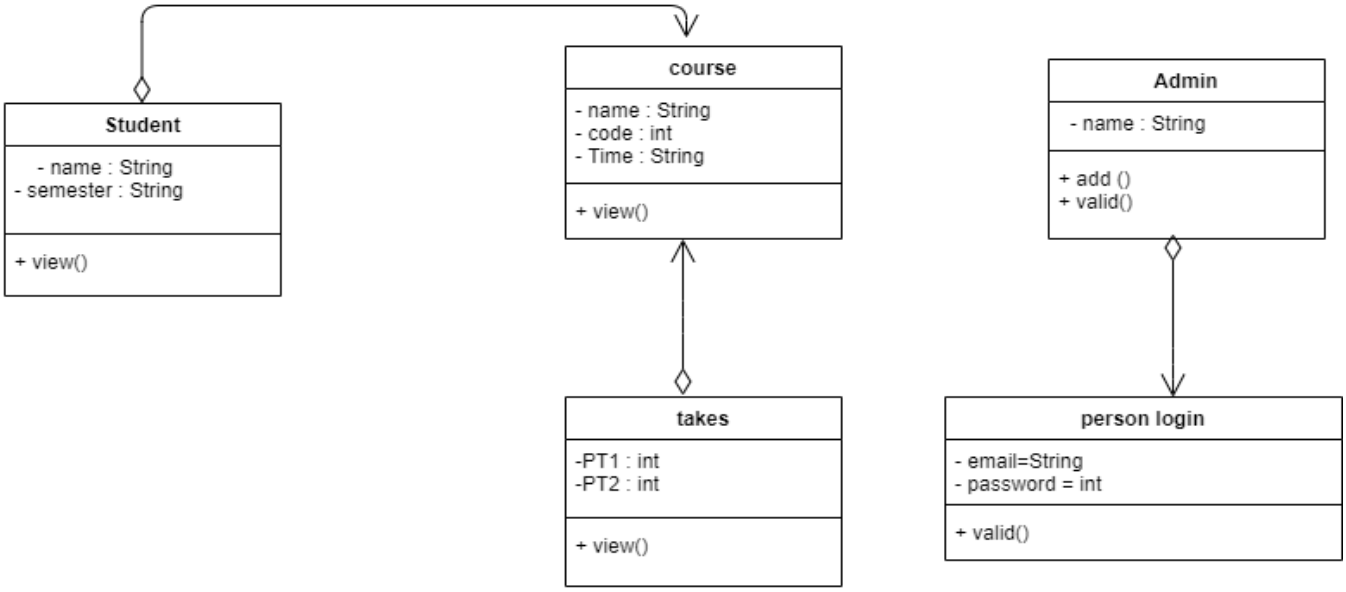
Mandatory Design Patterns

creational: Immutable pattern



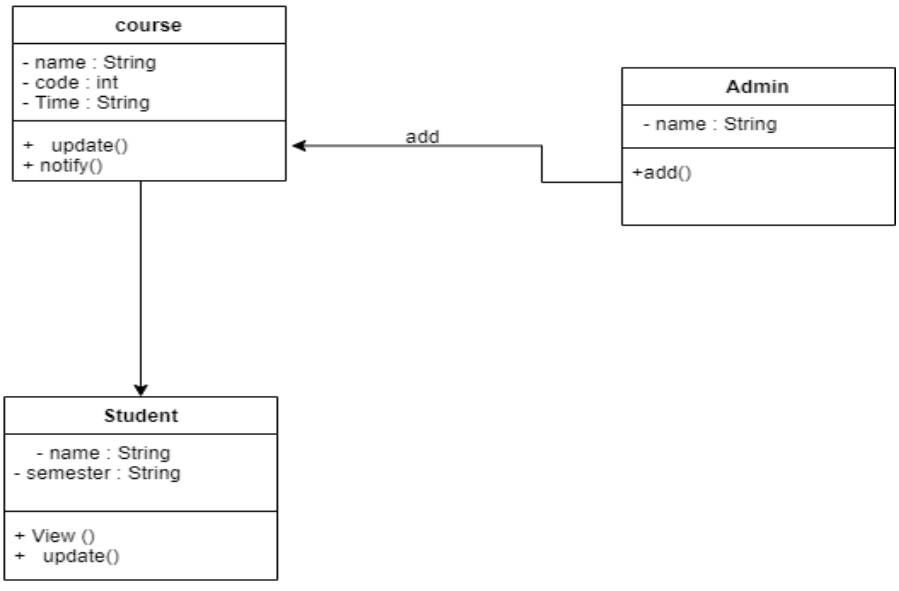
structural: delegation pattern

2.1 STRUCTURAL DELEGATION



Behavior: Observer / Publish – Subscribe

3.1 BEHAVIOURAL



Mandatory Design Patterns Applied (typed description)

Creational	
Name:	Immutable
Context:	An immutable object is an object that has a state that never changes after creation. User logs in with his e-mail and password. So the e-mail and password are data that have to remain constant.
Problem:	How to render the e-mail and password unchangeable?
Forces:	There must be no methods to change the log-in credentials after their values have been set.
Solution:	Data fields of person login were made private. Constructor of person login is the only place to set their values. One access method that gets the values of the data fields. No methods can change the values of the private data fields.

Structural	
Name:	Delegation
Context:	Some methods share the same exact implementation in the system. There should be one method in one already existent class only and all other classes invoke that method. Inheritance is not appropriate.
Problem:	How to not repeat the same methods in multiple classes without using inheritance?
Forces:	You want to minimize development cost by reusing methods. You want to improve efficiency
Solution:	1. **Delegate Class:** person login **Delegating Class:** Admin **Delegate Method: +valid()** Class person login includes **+valid()** which validates the user input. This method is also required whenever the admin enters any input. The **+valid()** method in class Admin calls for the **+valid()** method in person login. 2. **Delegate Class:** course **Delegating Class:** Student and takes **Delegate Method: +view()** Class course includes **+view()** which views certain data from database. This method is also required whenever the classes Student and takes have to view something. The **+view()** method in classes Student and takes call for the **+view()** method in course.

Behavioral	
Name:	Observer

Context:	An observer registers to receive notifications whenever the state of an observable (a.k.a. subject) changes. When partitioning a system into individual classes you want the coupling between then to be loose so you have the flexibility to vary them independently.
Problem:	A mechanism is needed to ensure that when the state of the class Course changes Students are updated to keep them in step.
Forces:	The different parts of a system have to kept in step with one another without being too tightly coupled.
Solution:	**This diagram depends on pull model. Observer:** Student **Observable:** Course. The Admin adds a new course by the **+add()** method. The constructor of the class Course is called. The constructor contains invocation of the **+notify()** method in the same class. The **+notify()** method invokes the **+update()** method in class Student.