

# **CSE300 Software Engineering**

# **Learning Management System**

**Group 8**

# **Design Document**

## 

[**Introduction**](#_ac4p35k8n4nz) **4**

[Purpose of this document](#_8f80k9qhfles) 4

[Document Overview](#_4pe2wgpimkqe) 4

[Identification](#_4pe2wgpimkqe) 4

[Scope](#_4pe2wgpimkqe) 4

[References](#_4pe2wgpimkqe) 4

[Tools and techniques](#_4pe2wgpimkqe) 4

[Key Stakeholders](#_4pe2wgpimkqe) 5

[**Design Overview**](#_e0tepljlgatm) **6**

[Background information](#_pktngbdz0c07) 6

[System Evolution Description](#_6td1n6aq6xgl) 6

[Constraints](#_6td1n6aq6xgl) 6

[Risk Analysis](#_6td1n6aq6xgl) 6

[Issues](#_6td1n6aq6xgl) 9

[Assumptions](#_6td1n6aq6xgl) 10

[Dependencies](#_6td1n6aq6xgl) 10

[**Scope of Work**](#_kpn8gsaf7yen) **11**

[System Functions](#_g78z8hm2yp6l) 11

[Similar System](#_g78z8hm2yp6l) 11

[User Problem Statement](#_g78z8hm2yp6l) 11

[User objectives](#_g78z8hm2yp6l) 11

[**Database Design**](#_83ipi9epex1) **12**

[Physical ER diagram](#_p8jmypfw4r1y) 12

[Modules and Interaction](#_v5lht3p3d5pf) 13

[Student and Student\_Course](#_utmdvu3o5vjj) 13

[Faculty and Faculty\_Course](#_cvpsrcf1cqve) 13

[Course and Faculty\_Course](#_fndbeogcphp3) 13

[Course and Student\_Course](#_2lvvpzi0zvqc) 13

[Faculty\_Course and Faculty\_Assignment](#_w5lohg46kcnt) 13

[Student\_Course and Student\_Assignment](#_d2cvsp384dqi) 14

[Student\_Assignment and Student\_Grade](#_wfr0k7xot2td) 14

[Faculty\_Assignment and Resource](#_dviaf33ijs7a) 14

[Data Dictionary](#_1lgx12ejm4f7) 15

[Table 1: Faculty](#_qd7b19lrwrb0) 15

[Table 2: Student](#_nzyaxae0dpsn) 15

[Table 3: Course](#_y83zpavuz4dn) 16

[Table 4: Student\_Course](#_ur1kzyqjturg) 16

[Table 7: Faculty\_Course](#_81hwfmb5gmfr) 17

[Table 8: Faculty\_Assignment](#_fh3qyyqoc0pn) 17

[Table 5: Student\_Assignment](#_wgjjyjrtdi9o) 18

[Table 9: Resource](#_vsfcx9dcvozd) 20

[**Software Architecture and Architecture design**](#_zergqv49ogy5) **21**

[Architecture Design](#_s0dq0umw89k) 21

[Topology Diagram](#_7nulcgd1r5oi) 21

[Website Architecture Diagram](#_abq3w8k9uime) 22

[Software Architecture](#_jmfmk3budacz) 23

[Software Architecture Diagram](#_3cwx9kgumm61) 23

[Software element](#_rm03c629bohi) 24

[**User interface design**](#_gp6v2i25qqtt) **25**

[User interface](#_eiw596gfaknz) 25

[Screen Images](#_359l54t7i3yk) 25

[Login Screen](#_k72mxmxyu41) 25

[Faculty-Dashboard](#_nw4ji2v12koa) 27

[Faculty-courses](#_9q58xhc5m8ft) 28

[Faculty-View Assignments](#_7id1evb7rmza) 30

[Faculty-Enter Grades](#_nsdimlcq9e73) 31

[Faculty-View Grading Status](#_xwqk6v9nf3hd) 32

[Faculty-Create/Edit Assignment](#_gcybnotkx4gr) 33

[Faculty-Grading Assignment](#_8i6cpp8kl8ww) 34

[Student-Dashboard](#_e6yz3amz5rq5) 35

[Student-Course](#_39whasvdwr74) 36

[Student-View Assignment](#_4h77xppj1pg1) 37

[Student-Upload/Edit Assignment](#_mmuoajjqguur) 38

[Student-View Grades](#_o752gz105ge) 39

[Objects and Actions](#_n6d37iu0x8jh) 44

[Interface design rules](#_i99wvvtxi7tn) 47

[Components available](#_zctjvuixuo63) 47

[User Interface Development Description](#_po4adr57mnhg) 49

## 

# Introduction

## Purpose of this document

The purpose of this Software Design Document is to provide a description of the design of a system fully enough so that to allow for software development to proceed with an understanding of what is to be built and how it is expected to be built.

## Document Overview

Chapter 1: Introduction of the design document

Chapter 2: Design Overview

Chapter 3: Scope of Work

Chapter 4: System Design

Chapter 5: Detailed Design

Chapter 6: Interface design

Chapter 7: User Interface design

## Identification

* + 1. Title: Learning Management System
    2. Version: 1.0.0
    3. Release: 1

## Scope

This document aims to provide the design details of the software Learning Management System. It can be used as documentation that will guide developers in understanding the software.

## References

*Software Engineering: A Practitioner’s Approach, 7/e* (McGraw-Hill 2009)

## Tools and techniques

Tools

* Miro Online whiteboard
* Lucid chart
* Adobe XD
* Github

Languages

* Python
* JavaScript
* SQLite

## Key Stakeholders

The key stakeholders for this software are:

1. Students (Primary User)

* Students can use the software to submit assignments and get their grades on the same platform. They can also get the resource that the faculty will share.

1. Faculty of the school (Primary user)

* Faculties can create assignment submission posts and grade students based on their submissions. They can also share resources with students.

1. IT Administrator

* The IT administrator will add users in the software. He/she will add classes too and assign them to individual users.

# Design Overview

## Background information

* There are various software like Moodle, Canvas, Gradescope, etc that are used in universities today share resources and submit assignments and provide quizzes
* However, most of them are based on older technologies like PHP and generally do not scale up to higher server load
* The Learning Management System is discussion is built in Python using Django framework and is extremely scalable in nature

## System Evolution Description

As this project is on a small scale for the sake of demonstration. The system can be scaled up to a deployment server and the database can be moved to a different server.

In terms of functionalities, a quiz option can be added to the system that would enable users to attempt graded quizzes inside the software itself.

## Constraints

Google OAuth for login is not provided which is the primary method used for authentication in our university. Also, the course registration is not done on the platform and needs to be provided from outside the application into the database

## Risk Analysis

* Inappropriate Project Manager - It results in problems at the very beginning, mainly in implementation and enforcement of the work. That leads to incorrect change management, risk management and scope management.
* Requirements Analysis
  + The requirements are unclear if they are not understandable by analysts and developers.
  + The requirements are incomplete if they are missing some of the user needs, constraints and other requirements.
  + The requirement is inconsistent if it contradicts any other requirement in the project.
  + Usually analysts and developers focus on what the system should do and ignore how the system should be (i.e. usability, maintainability, scalability, testability, etc.). Non-functional requirements are essential to project success as much as the functional requirements.
  + If the requirements are clear, verifiable, accurate, consistent,complete and feasible then they are realistic to be put in the requirements document and then implemented.
* Design Activity
  + At the time of design verifying, it might be discovered that many alternatives to the same design may exist. Which one to choose depends on the system and its nature.
  + When the design is verified, it might be found that the design does not match some of the requirements.
  + If the system was not decomposed correctly and the components were not defined well, then developers may face difficulties in assigning functions to each component and defining its objectives. It also may threaten the allocation activity since the components’ functionalities are derived from the functional requirements in the requirements document.
  + Sometimes, the passing data is not used (tramp data); it passes only to be passed to another component to be used there. If this data was not managed carefully, it can reduce readability and lead to confusion.
* Construction Activity
  + In the architectural design method, if a wrong choice was made then the system implementation will not be completed successfully and problems in the integration may arise later.
  + The choice of the architectural design method may affect the choice of the programming language which must be considered or else the developers may choose a language that does not support the architectural design method in use.
  + If the system was too complex, and the developers do not have the enough skills and experience to manage this complexity, then they will create a complicated not understandable design which will, while being implemented, suffer from different difficulties.
  + An inaccurate estimate about the available reusable components was made in the analysis phase can result in the components having to be developed from scratch again. Thus time schedule and budget may be under-estimated and will cause project delay and budget over-run.
* Documentation Activity
  + While documenting the requirements, maintainability and feasibility is not considered in mind, which makes it difficult to modify the data without rewriting it, in the requirement document.
  + Developers must be involved in the requirements analysis and definition phase, so that they can understand the requirement document at the time of execution. Thus, the risk of developing a design for a system other than the intended one decreases.
  + The design document must be detailed enough to allow the programmers to work independently. If the components in the design document are not clearly defined; their inputs, outputs, functions and relationships were not stated properly.
  + Moreover, if the design document was written in an uncommon natural language, then it is unclear and might be non readable by developers which results in an increase of risk.
* Coding Activity
  + The improper choice of the programming language can affect the development process in many different aspects.The wrong choice of programming language may not support the applied architectural design method . It may reduce the system’s maintainability and portability.
  + If the coding document was incomplete, then programmers will not be able to work independently since they have to make their own decisions to fill the gaps in the design document, which may affect product outcome.
  + Development team usually has more than one programmer. These programmers may work on different components, and each may follow his own way of thinking and coding which will lead to inconsistent, complex and ambiguous code. Moreover, if they work on the same component, then it might result in different versions for the same component.
  + Developing the correct user interface requires a good understanding of user needs and detailed specification in the design.
* Testing Activity
  + If the testing process were not automated, then manual testing will be monotonous and boring and will continue to fail to produce results.
  + Test cases have to be documented automatically while doing the testing so that it can be used in future for similar cases.
  + In unit testing, each unit is tested individually. The module being tested might need data from another module or send it to another module; this is solved by coding drivers and stubs.

## Issues

|  |  |
| --- | --- |
| Ref. | Issue |
| 1 | Faculty able to submit the assignment in his own course |

## Assumptions

|  |  |
| --- | --- |
| Ref. | Assumption |
| 1 | Students and faculty database can be imported from other companion software |
| 2 | Course Registration can be done from other companion software |

## Dependencies

The dependencies are as follows:

* django-livereload-server
* django-session-timeout

# Scope of Work

## System Functions

Student side

* Upload assignments
* View Grades

Faculty side

* Share resources with users
* Grade assignments
* Create posts where students can submit assignments

## Similar System

* Piazza ([Link](https://piazza.com/))
* AU LMS ([Link](https://lms.ahduni.edu.in/login/index.php))
* Google classroom ([Link](https://classroom.google.com/))
* Gradescope ([Link](https://www.gradescope.com/))

## User Problem Statement

The traditional learning management system has few problems:

* Complicated user interface
* Sluggish performance
* No notifications for new assignments
* Improper integration of resource sharing

## User objectives

* To give access to many resources
* To eliminate the traditional barriers
* To make learning effective
* To effortlessly blend in technology
* To give the privilege of accessing content anytime
* To make the evaluation and feedback an easy process

# Database Design

## Physical ER diagram

## Modules and Interaction

### Student and Student\_Course

Relationship between Student and student\_Course is one to many i.e one student can be enrolled in one or many courses. Here, student\_id is primary key which is referred as foriegn key in Student\_Course. In the Student\_Course table, course\_id, student\_id and course\_enrollment\_year is taken as a combinational primary key.

### Faculty and Faculty\_Course

Relationship between Faculty and Faculty\_Course is one to many i.e one faculty can take one or many courses. Here, faculty\_id is primary key which is referred as foriegn key in Faculty\_Course. In the Faculty\_Course table, course\_id, faculty\_id and course\_enrollment\_year is taken as a combinational primary key.

### Course and Faculty\_Course

Relationship between Course and Faculty\_Course is one to one i.e one course is taken by only one faculty. Here, course\_id is primary key in Course table which is referred as foriegn key in Faculty\_Course. In Faculty\_Course table, course\_id, faculty\_id and course\_enrollment\_year is taken as a combinational primary key.

### Course and Student\_Course

Relationship between Course and Student\_Course is one to many i.e one course has one or many students. Here, course\_id is primary key in Course table which is referred as foriegn key in Student\_Course. In the Student\_Course table, course\_id, faculty\_id and course\_enrollment\_year is taken as a combinational primary key.

### Faculty\_Course and Faculty\_Assignment

Relationship between Faculty\_Course and Faculty\_Assignment is one to many i.e each course can have many assignments. Faculty\_Course contains the information about the course taken by the respective faculties. Faculty\_Assignment contains details of the assignment given by the faculty in each course. The faculty\_id, course\_id and course enrollment year are the primary keys of Faculty\_Course and it is referred as foriegn key in Faculty\_Assignment table.

### Student\_Course and Student\_Assignment

Relationship between Student\_Course and Student\_Assignment is one to many i.e each course can have many assignments. Student\_Course contains the id of courses enrolled by students. Student\_Assignment contains details of the assignment given by the faculty in each course. It consists of the solution submitted by each student. The student\_id, course\_id and course enrolment year are the primary keys of Student\_Course and it is referred as foriegn key in Student\_Assignment table.

### Student\_Assignment and Student\_Grade

Relationship between Student\_assignment and Student\_Garde is one to one i.e one student assignment will have one grade. Here, assign\_id, student\_id and course\_id is primary key in Student\_assignment table which is referred as foriegn key in Student\_Grade. In the Student\_Grade table, course\_id, assign\_id and student\_id is taken as a combinational primary key.

### Faculty\_Assignment and Resource

Relationship between Resource and Faculty\_Course is one to many i.e one Faculty\_course can have zero or many resources. Here, course\_id, faculty\_id and course\_enrollment\_id is primary key in Faculty\_Course table which is referred as foriegn key in Resources. In the Resources table, course\_id, resource\_id and course\_enrollment\_year is taken as a combinational primary key.

## Data Dictionary

### Table 1: Faculty

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Element | Description | Data type | Required? | Constraints |
| faculty\_id | Unique id given to every user(faculty) belonging to the institution | varchar(10) | y | Primary Key |
| first\_name | The first name of the faculty | varchar(20) | y | Not null |
| last\_name | The last name of the faculty | varchar(20) | y | Not null |
| email\_id | Email id of the faculty provided by the institution | varchar(20) | y | Not null |

### Table 2: Student

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Element | Description | Data type | Required? | Constraints |
| student\_id | Unique id given to every user (student) belonging to the institution | varchar(10) | y | Primary Key |
| first\_name | The first name of the student | varchar(20) | y | Not null |
| last\_name | The last name of the student | varchar(20) | y | Not null |
| email\_id | Email id of the student provided by the institution | varchar(20) | y | Not null |

### Table 3: Course

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Element | Description | Data type | Required? | Constraints |
| course\_id | The unique course id of the course | varchar(10) | y | Primary Key |
| course\_name | The name of the course | varchar(20) | y | Not null |
| course\_details | Details regarding the course that the faculty deems necessary to add | varchar(20) | n | can be null |
| course\_credits | The number of credits the course will have | Integer | y | Not null |

### Table 4: Student\_Course

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Element | Description | Data type | Required? | Constraints |
| course\_id (PK,FK) | The unique course id of the course | varchar(10) | y | Primary key, Foreign Key (referred from Course table ) |
| student\_id (PK,FK) | The enrollment number of the student | varchar(10) | y | Primary key, Foreign Key (referred from Student table ) |
| Course\_enrollment\_year (PK) | The year the student enrolled in the course | datetime | y | Primary key |

### Table 7: Faculty\_Course

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Element | Description | Data type | Required? | Constraints |
| course\_id | The unique course id of the course | varchar(10) | y | Primary key, Foreign Key (referred from Course table ) |
| faculty\_id | Unique id given to every user belonging to the institution | varchar(10) | y | Primary key, Foreign Key (referred from Faculty table ) |
| course\_enrollment\_year | The year the faculty took the course | datetime | y | Primary Key |

### 

### Table 8: Faculty\_Assignment

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Element | Description | Data type | Required? | Constraints |
| assign\_id | The unique id of the assignment | varchar(10) | y | Primary Key |
| course\_id | The unique course id of the course | varchar(10) | y | Primary key,  Foriegn Key (referred from Faculty\_course table ) |
| faculty\_id | The unique id of faculty | varchar(10) | y | Primary key,  Foriegn Key (referred from Faculty\_Course table ) |
| course\_enrollment\_year | The year in which faculty has enrolled the course | datetime | y | Foriegn Key (referred from Faculty\_Course table ) |
| PDF | File name of the assignment | varchar(20) | y | n |
| marks | Marks assignment by the faculty to the student for the assignment | Integer | y | n |
| deadline | The last date allowed by the faculty to submit assignment | datetime | y | n |

### Table 5: Student\_Assignment

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Element | Description | Data type | Required? | Constraints |
| assign\_id | The unique id of the assignment | varchar(10) | y | Primary Key,  Foreign key (referred from Faculty\_Assignment table) |
| course\_id | The unique course id of the course | varchar(10) | y | Primary key,  Foriegn Key (referred from Student\_Course table ) |
| student\_id | The enrollment number of the student | varchar(10) | y | Primary key,  Foriegn Key (referred from Student\_Course table ) |
| course\_enrollment\_year | The year in which student enrolled course | datetime | y | Foriegn Key (referred from Student\_Course table ) |
| PDF | File name of the assignment | varchar(20) | y | n |
| time\_of\_submission | The time at which the assignment was submitted | datetime | y | n |

* + 1. Table 6: Student\_Grade

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Element | Description | Data type | Required? | Accepts null values? |
| assign\_id | The unique id of the assignment | varchar(10) | y | Primary Key,  Foreign key (referred from Student\_Assignment table) |
| student\_id | The enrollment number of the student | varchar(10) | y | Primary Key,  Foreign key (referred from Student\_Assignment table) |
| course\_id | The unique course id of the course | varchar(10) |  | Foreign key (referred from Student\_Assignment table) |
| comments | Notes by the faculty if any | varchar(20) | n | can be null |
| marks | Marks assignment by the faculty to the student for the assignment | Integer | y | n |

### Table 9: Resource

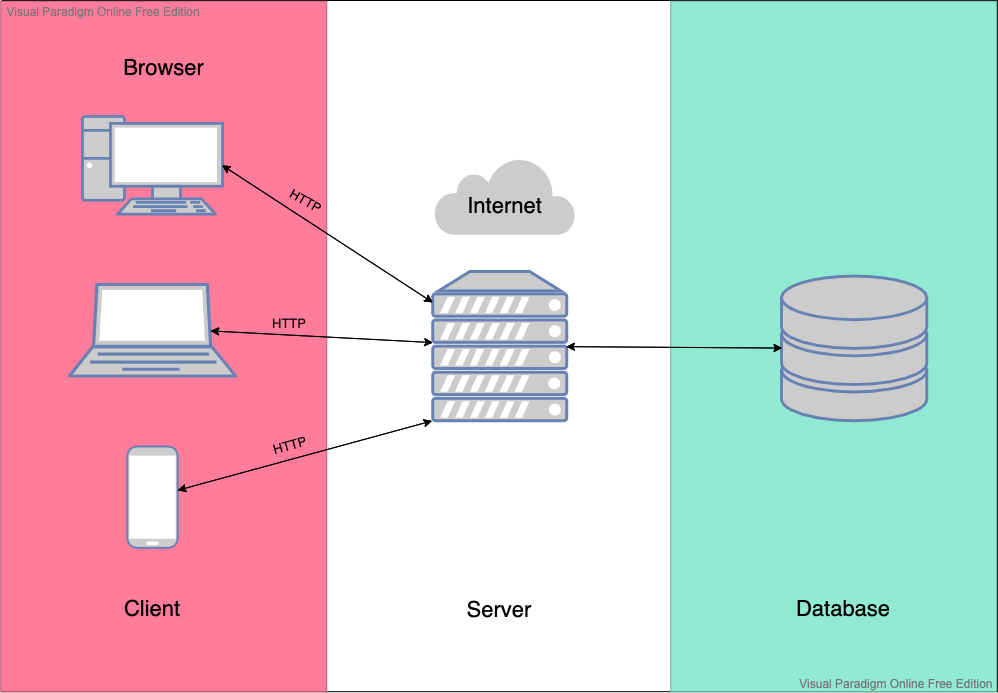
|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Element | Description | Data type | Required? | Constraints |
| resource\_id | The unique id of the resource | varchar(10) | y | Primary Key |
| course\_id | The unique id of the course for which the resource was provided | varchar(20) | y | Primary key,  Foriegn Key (referred from Faculty course table ) |
| course\_enrollment\_year | Year in which faculty enrolled course | datetime | y | Primary key,  Foriegn Key (referred from Faculty course table ) |
| faculty\_id | The unique id of the faculty | varchar(10) | y | Foriegn Key (referred from Faculty course table ) |
| resource\_material | The filename of the resource document | varchar(20) | y | n |

# Software Architecture and Architecture design

## Architecture Design

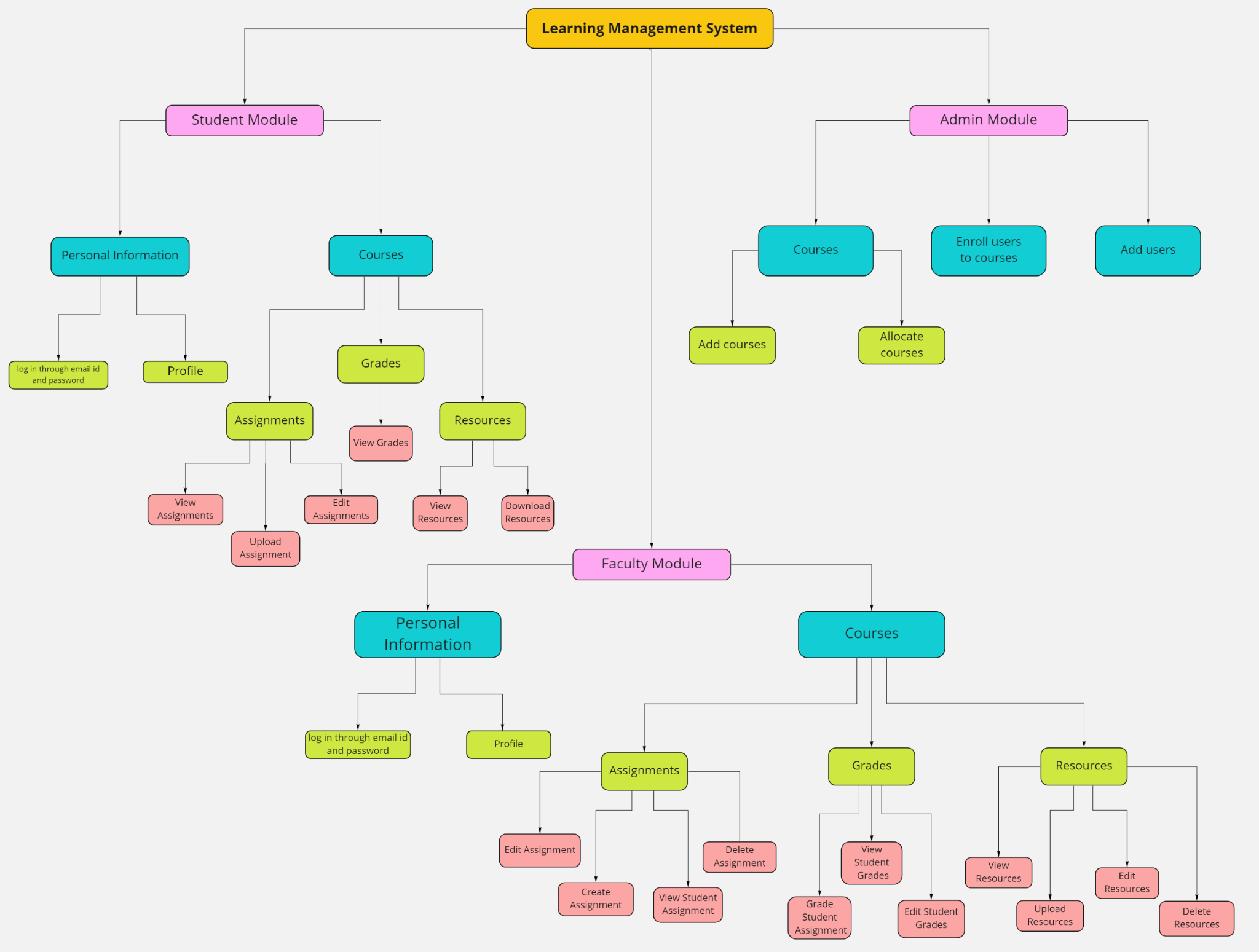
### Topology Diagram

A network topology diagram helps visualize the communicating devices, which are modeled as nodes, and the connections between the devices, which are modeled as links between the nodes.



### Website Architecture Diagram

Website architecture diagram is used to plan a web site's purpose, navigation and its overall organization. It is an important tool as it provides an overall view of the physical deployment of the software system and its evolution roadmap.

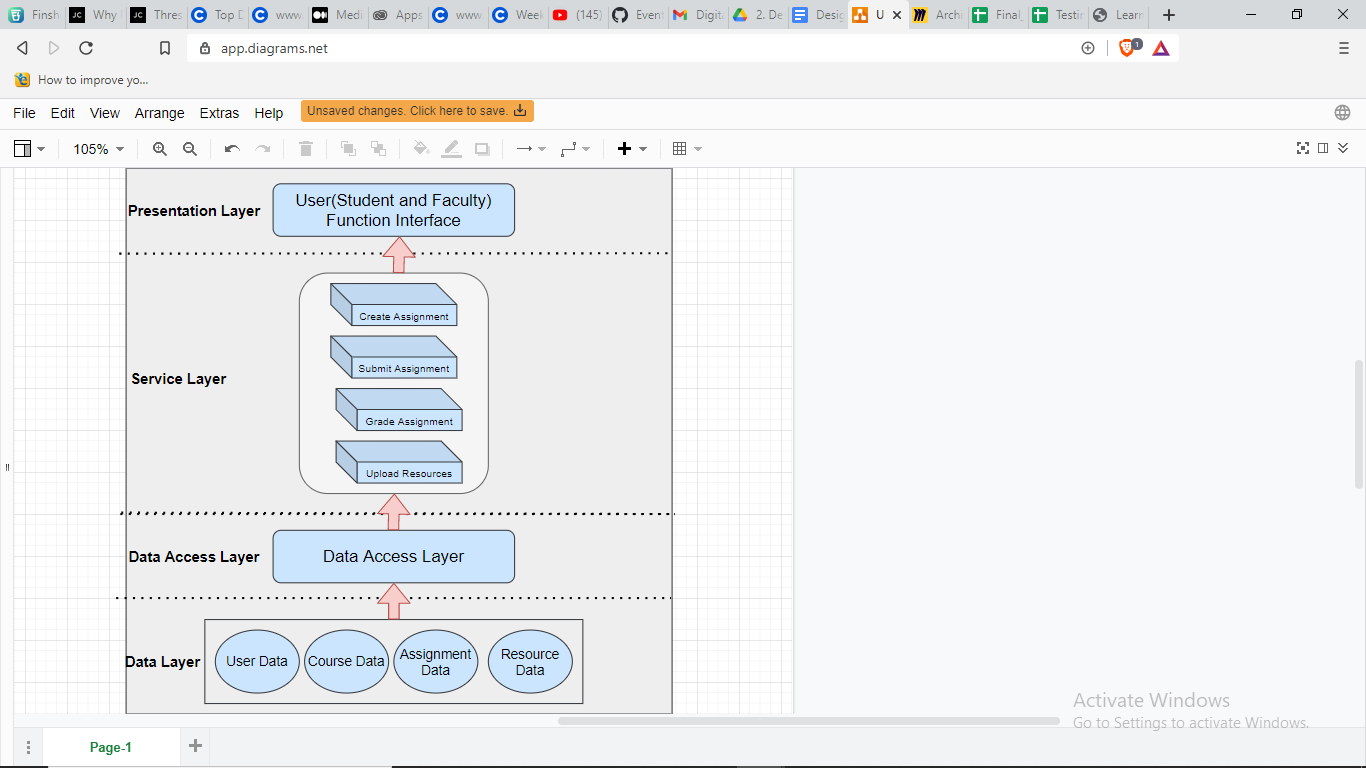


## 

## Software Architecture

### Software Architecture Diagram

Architecture diagrams can help system designers and developers visualize the high-level, overall structure of their system or application for the purpose of ensuring the system meets their users' needs. You can also use architecture diagrams to describe patterns that are used throughout the design.



### Software element

|  |  |  |
| --- | --- | --- |
|  | System | Note |
| Programming Language | Python | Python is an interpreted, high-level and general-purpose programming language. Python's design philosophy emphasizes code readability with its notable use of significant indentation. |
| Frontend/Backend Framework | Django | Django is a collection of Python libs allowing you to quickly and efficiently create a quality Web application, and is suitable for both frontend and backend |
|
| Database | SQLite | SQLite is a C-language library that implements a small, fast, self-contained, high-reliability, full-featured, SQL database engine. SQLite is the most used database engine in the world |

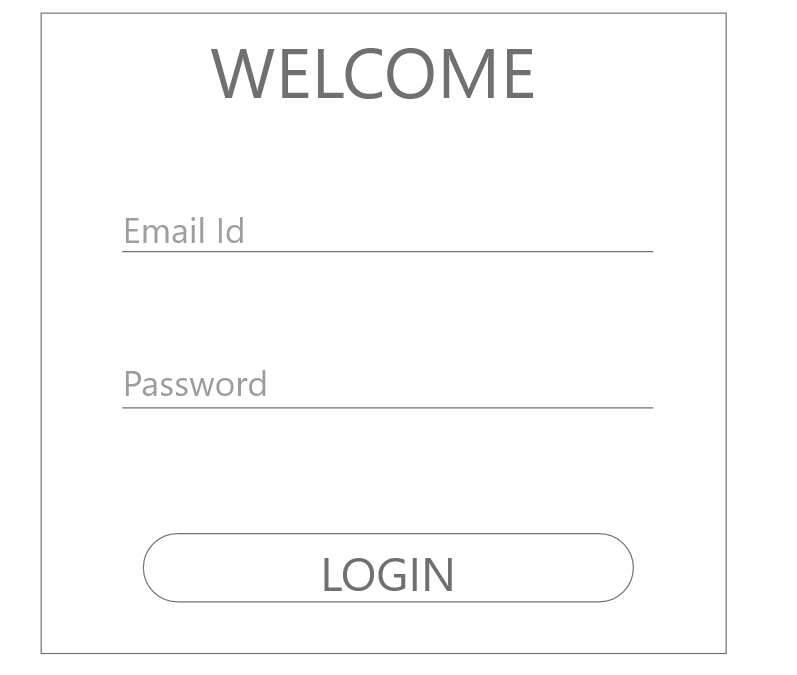
# User interface design

## User interface

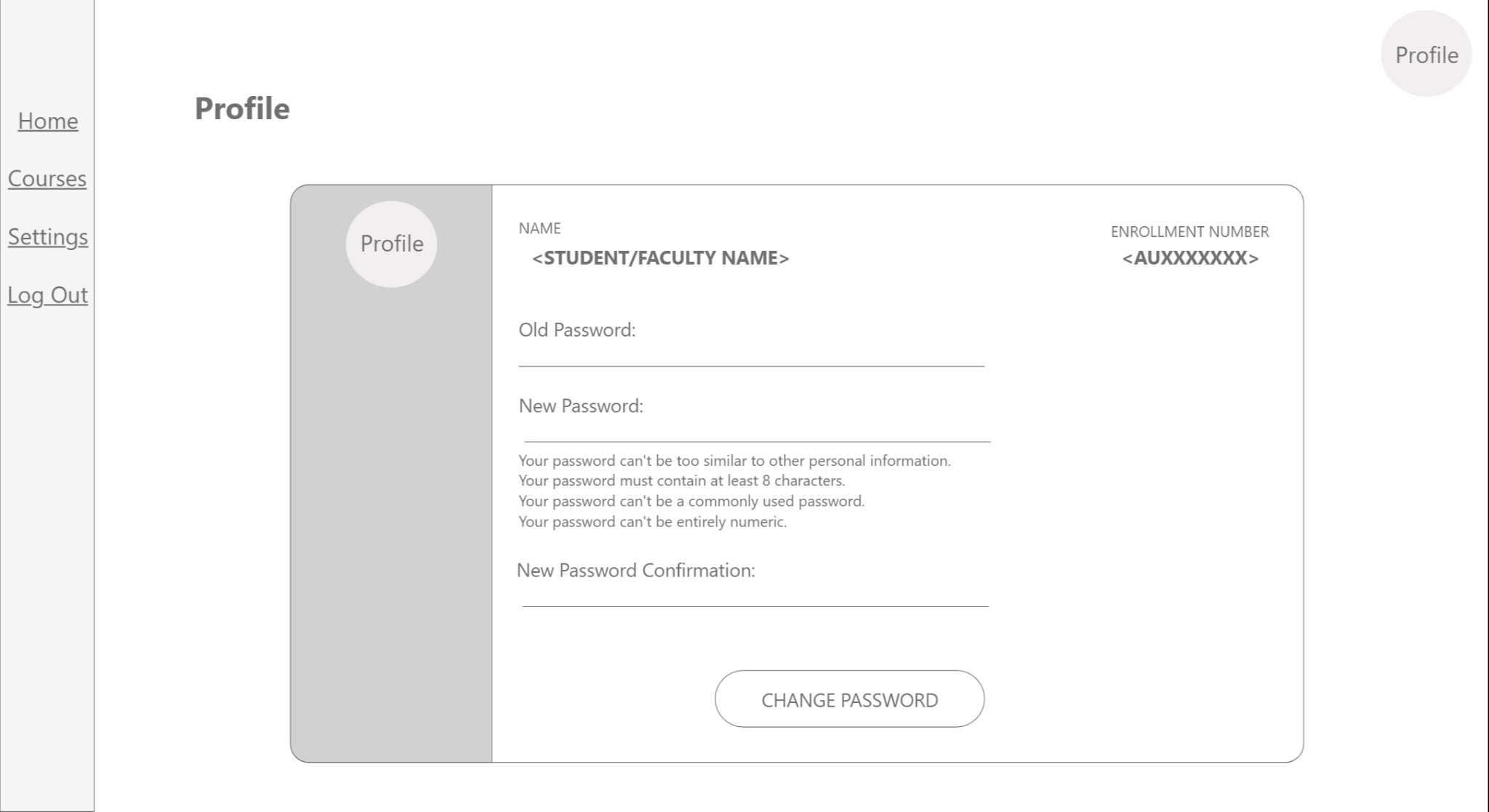
### Screen Images

The screen images for the Learning Management system are displayed below

#### Login Screen

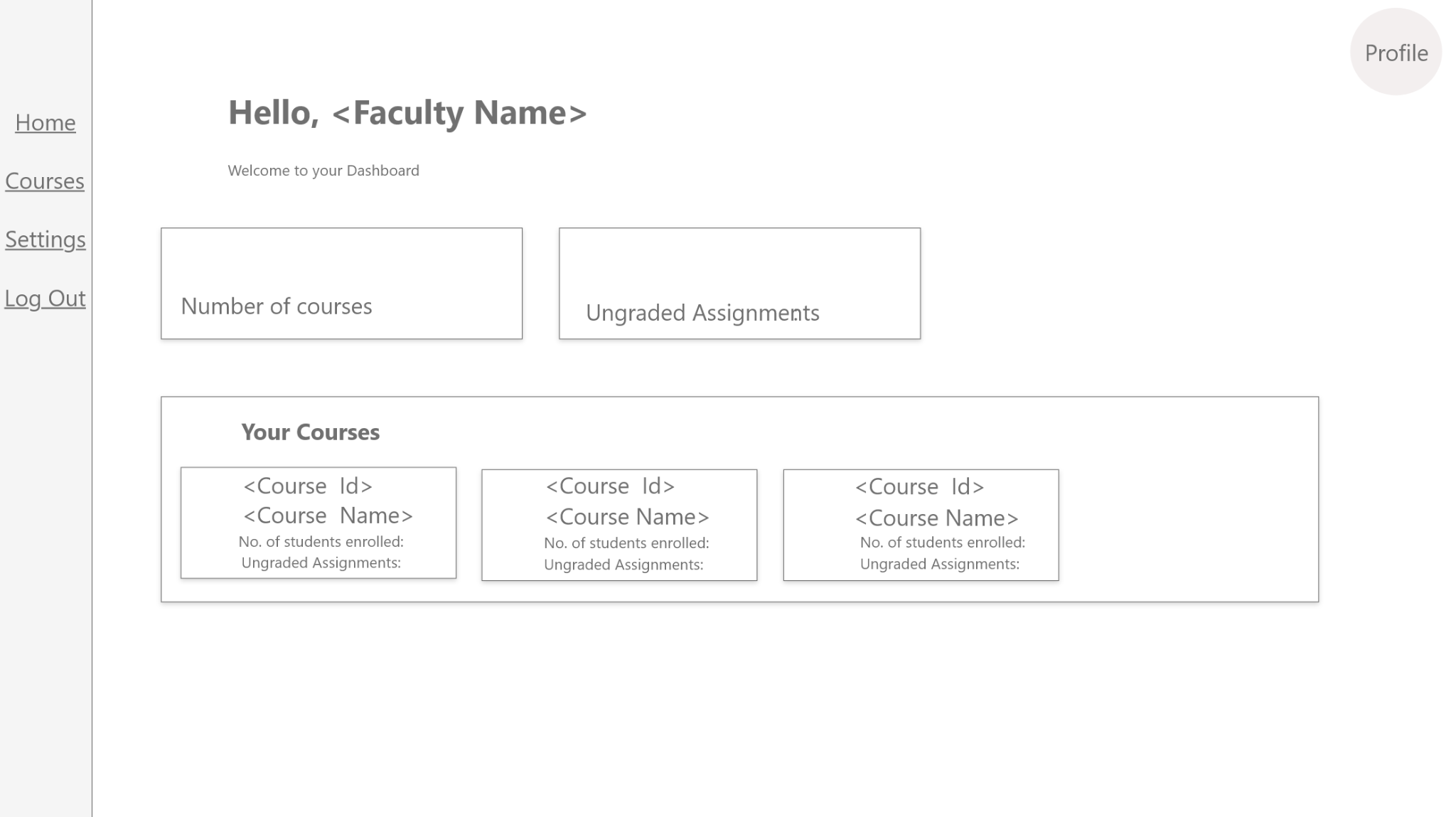


* All students, faculty members and administrative staff can login using their university ‘Email address’ and ‘password’. If the Email address is not registered or the password is incorrect, an error message will pop up or else the user will login.
  + - 1. Student/Faculty-View/Edit Profile



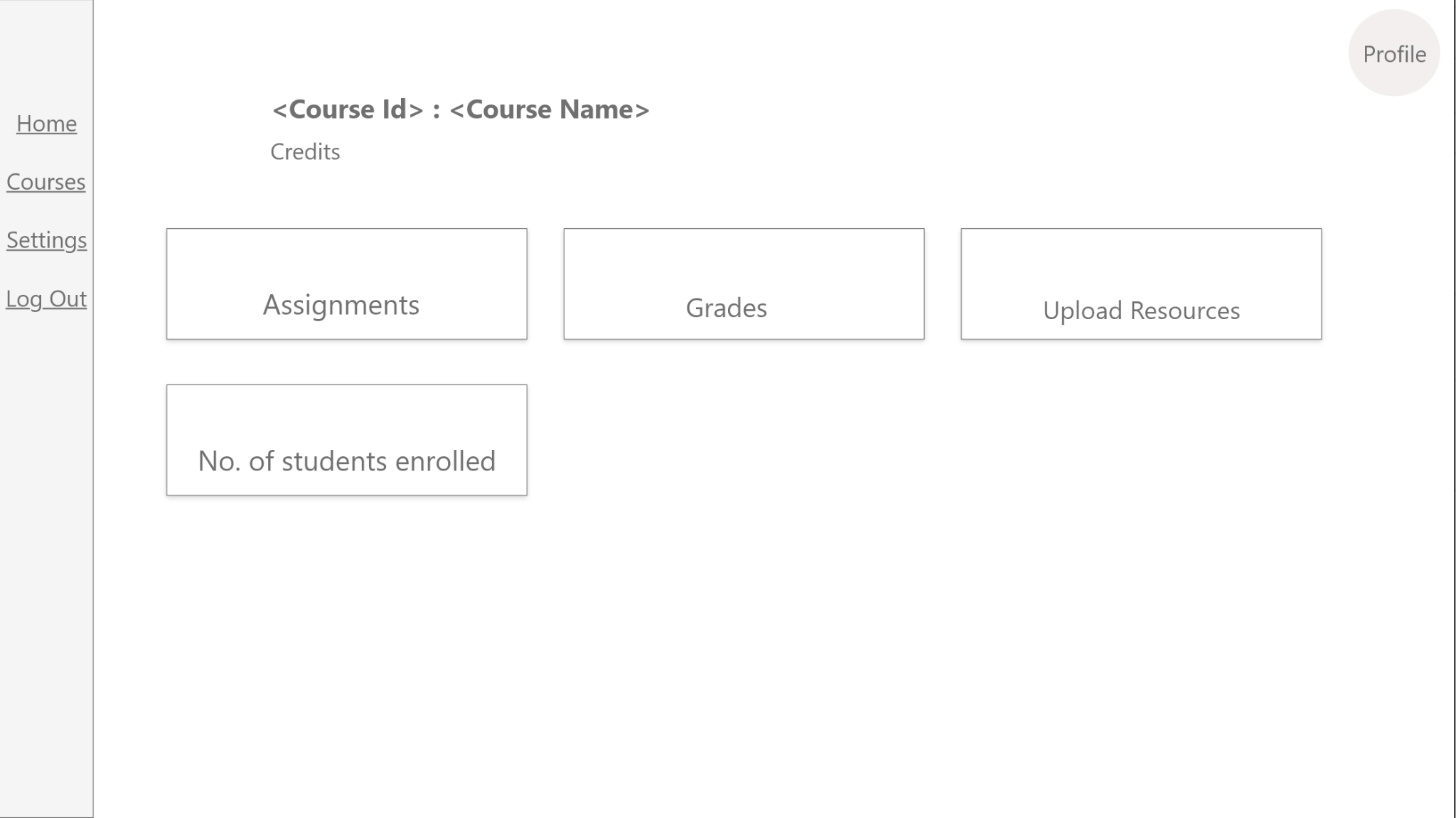
* Students and faculty members can change their password on this page. For doing so, they need to enter their existing password, the new password they want to set and confirm the new password.

#### Faculty-Dashboard

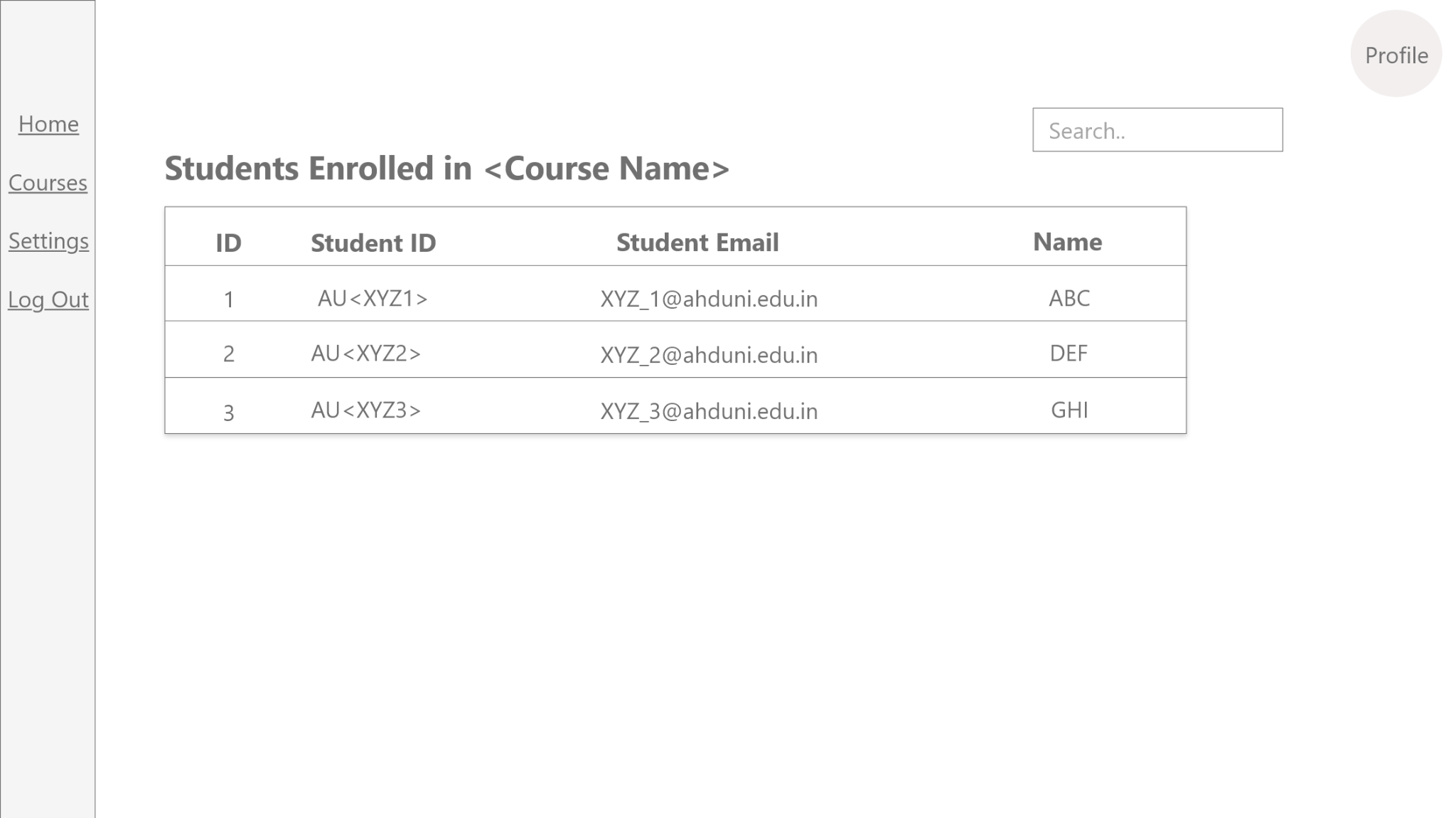


* This is the Faculty Dashboard. Once a faculty member logs in, he/she will be able to see this screen. Faculty members will immediately be able to view the current courses they are teaching or assisting, number of students enrolled in them, the number of ungraded assignments and upcoming quizzes.

#### Faculty-courses

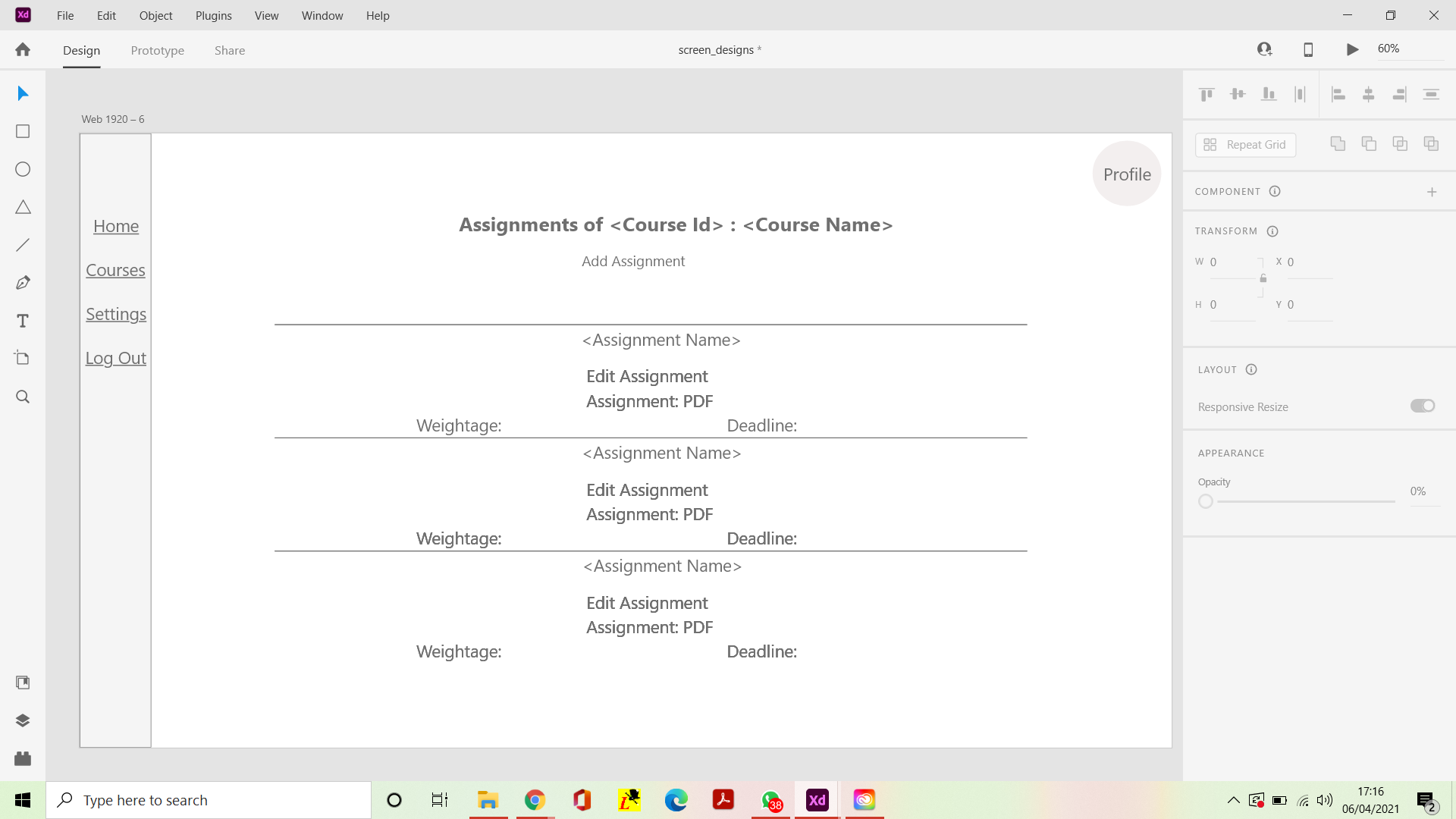


* Once a faculty member goes to a particular course, he/she would get directions to add/edit or delete assignments and grades and to upload resources relevant to the course.
  + - 1. Faculty- View Enrolled Students



* This page enables the faculty of the course to view the number of students enrolled that have enrolled for the course and their details like Enrolment ID, Email ID and Name.

#### Faculty-View Assignments



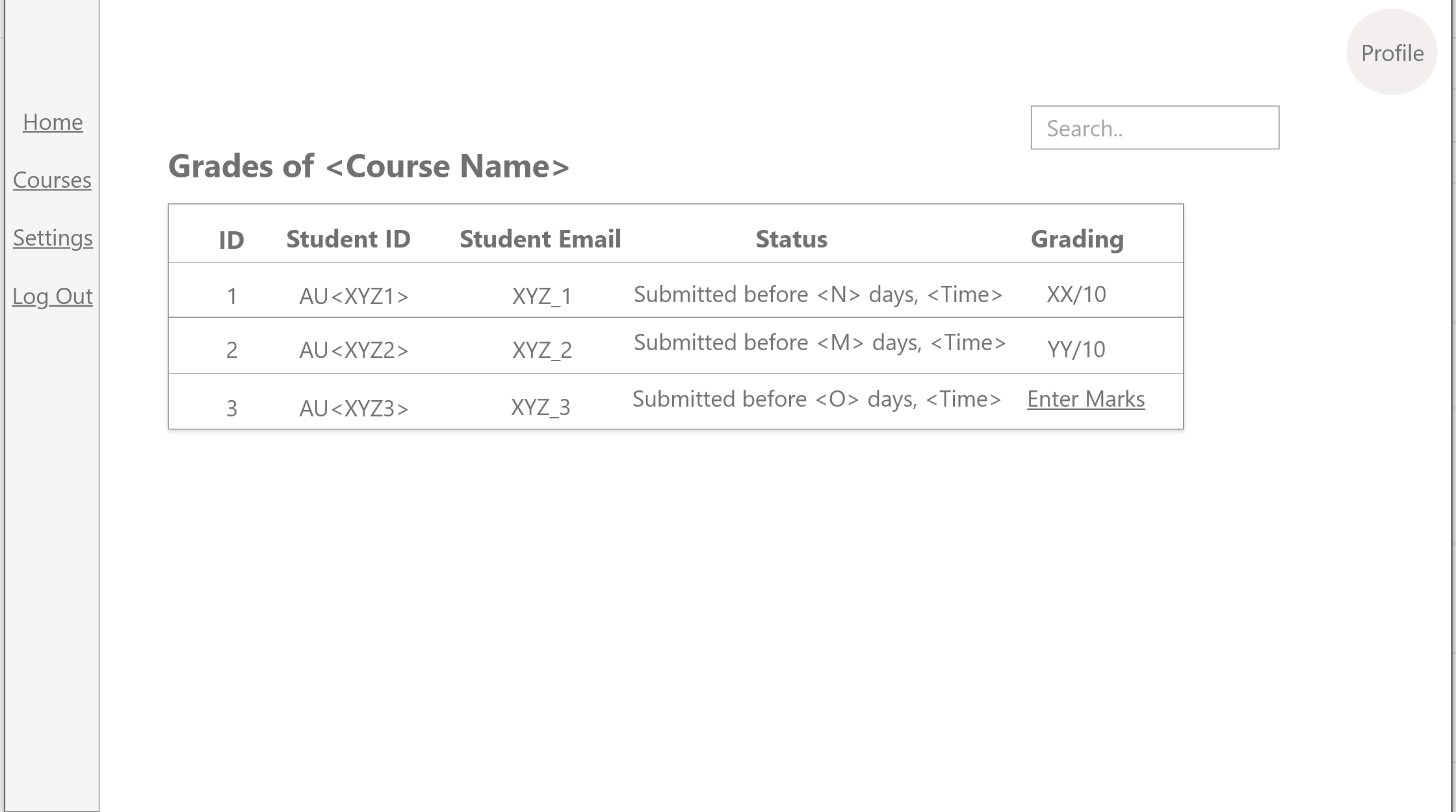
* The assignments can be added/updated by faculty members. While creating an assignment, the faculty can upload documents, assign weightage and deadline to the assignment.

#### Faculty-Enter Grades



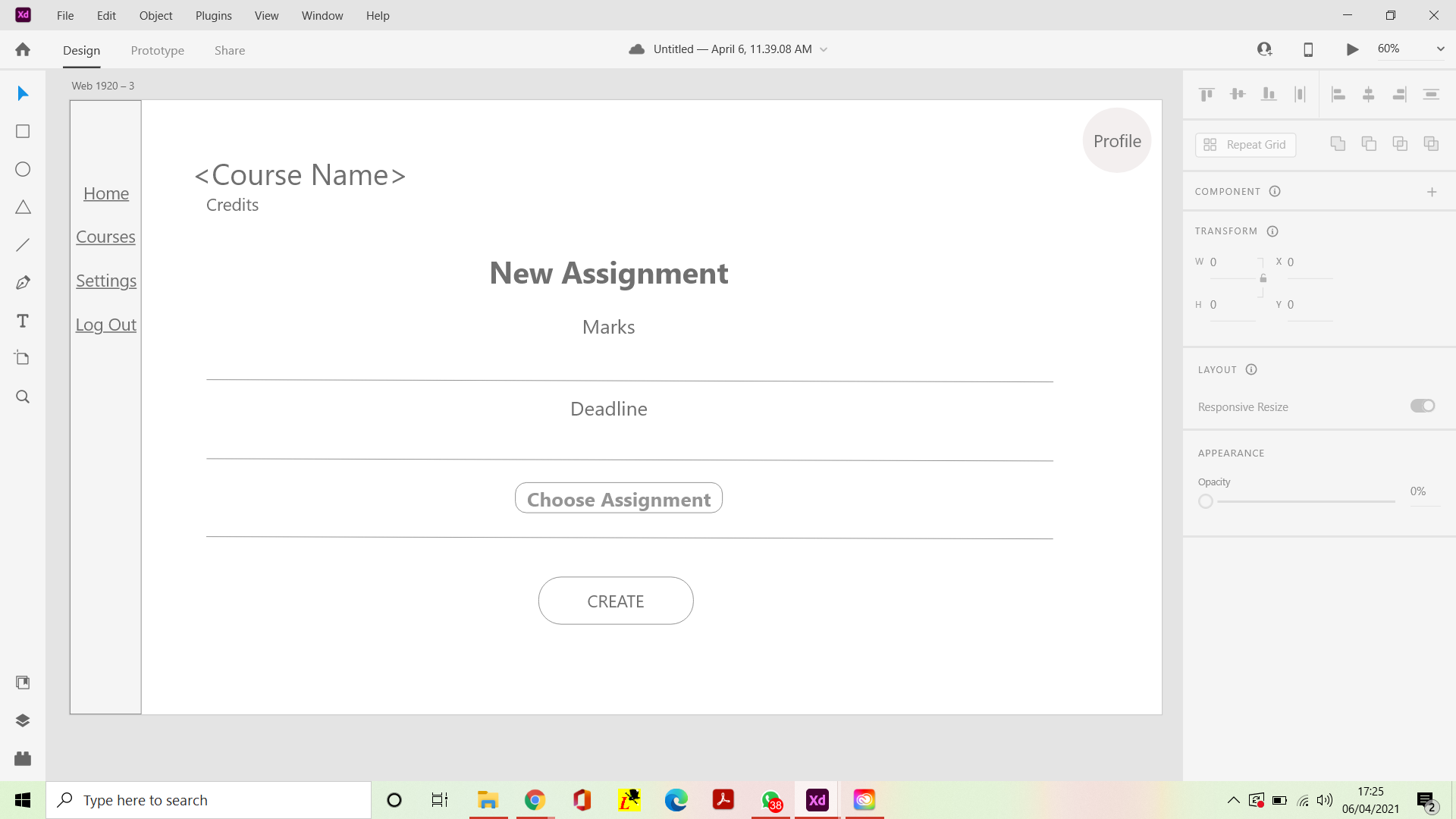
* Faculty members can view the status, submissions and deadline of the assignments. They can enter the marks of ungraded assignments.

#### Faculty-View Grading Status



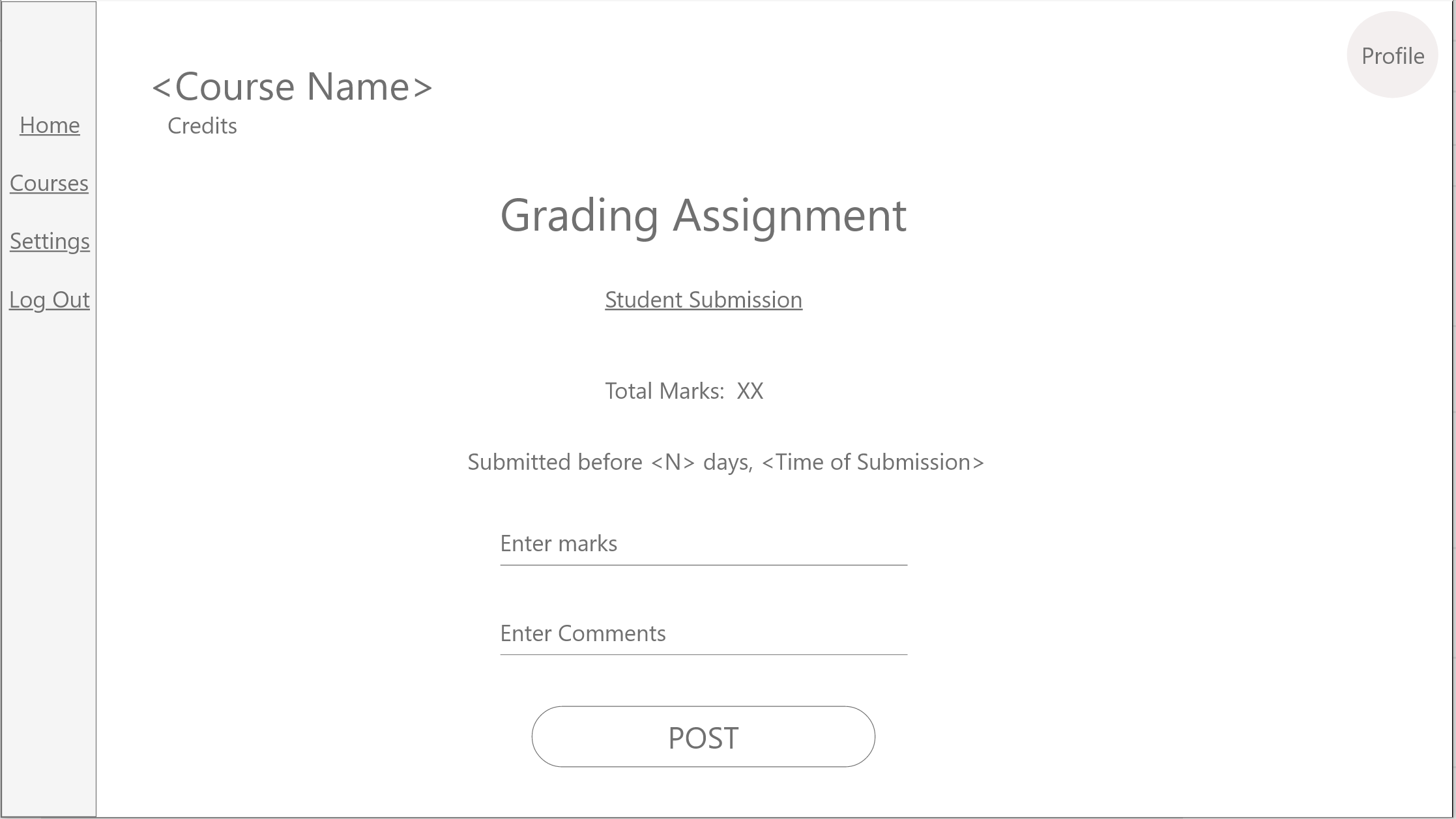
* The information about the grading status of assignments can be viewed in detail by the faculty members. They can view the submission date and time, marks of individual students.

#### Faculty-Create/Edit Assignment



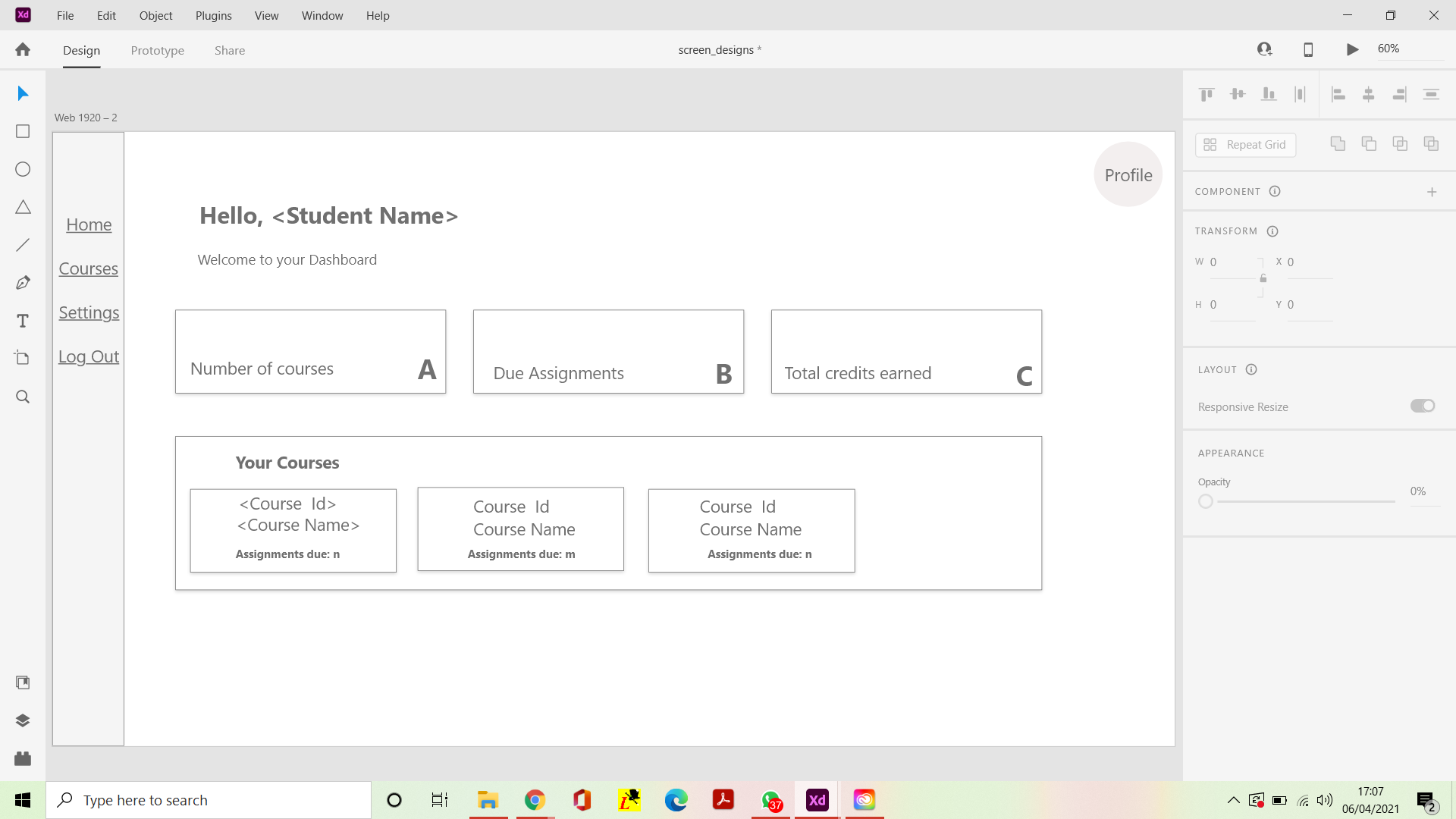
* To create a new assignment, the faculty can choose a document from his/her computer as the assignment. He/She can also set the weightage and deadline for the assignment.

#### Faculty-Grading Assignment



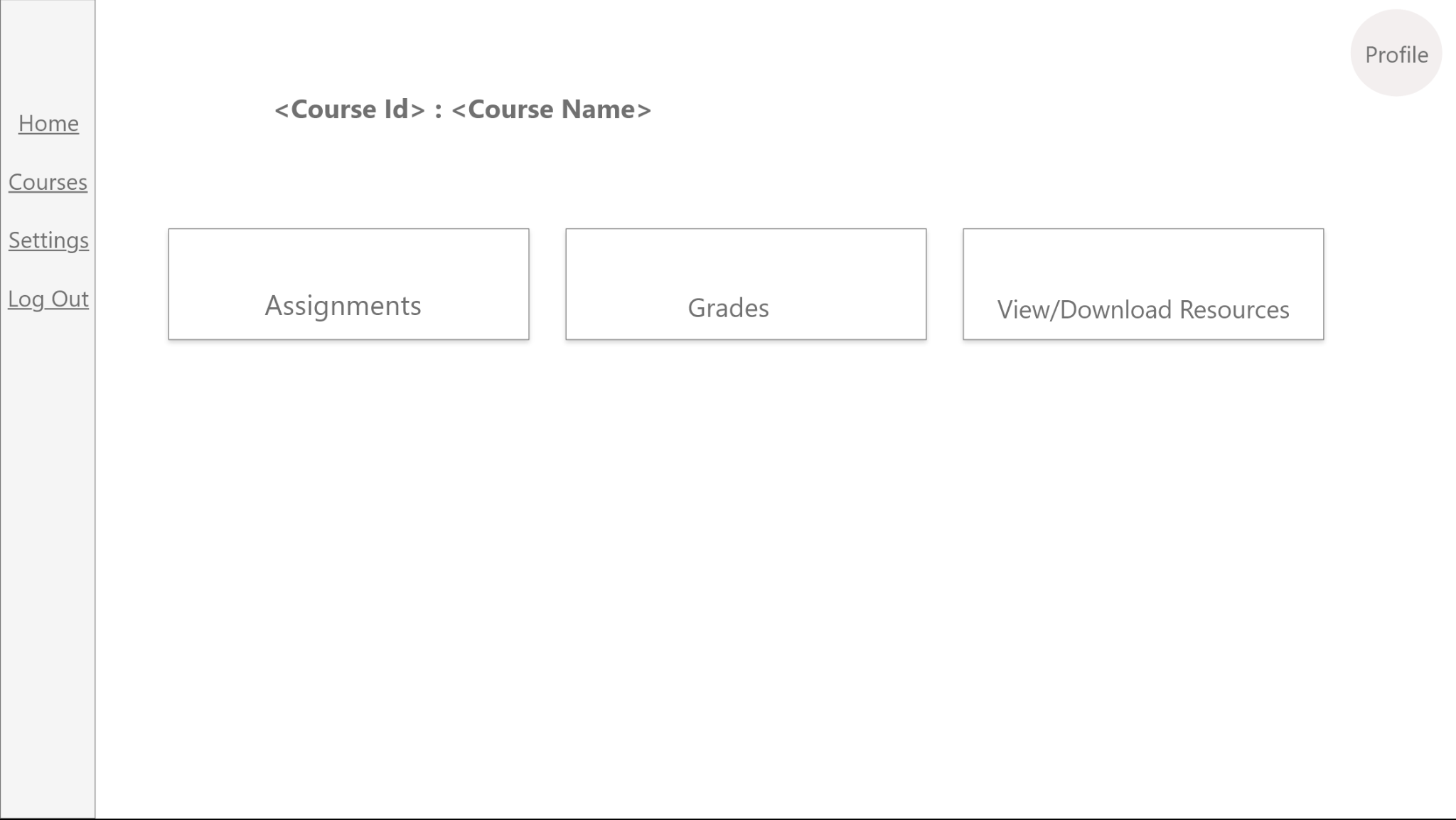
* The assignment submission for individual students can be viewed/downloaded by the faculty and they can also assign grades and provide comments about the assignment.

#### Student-Dashboard



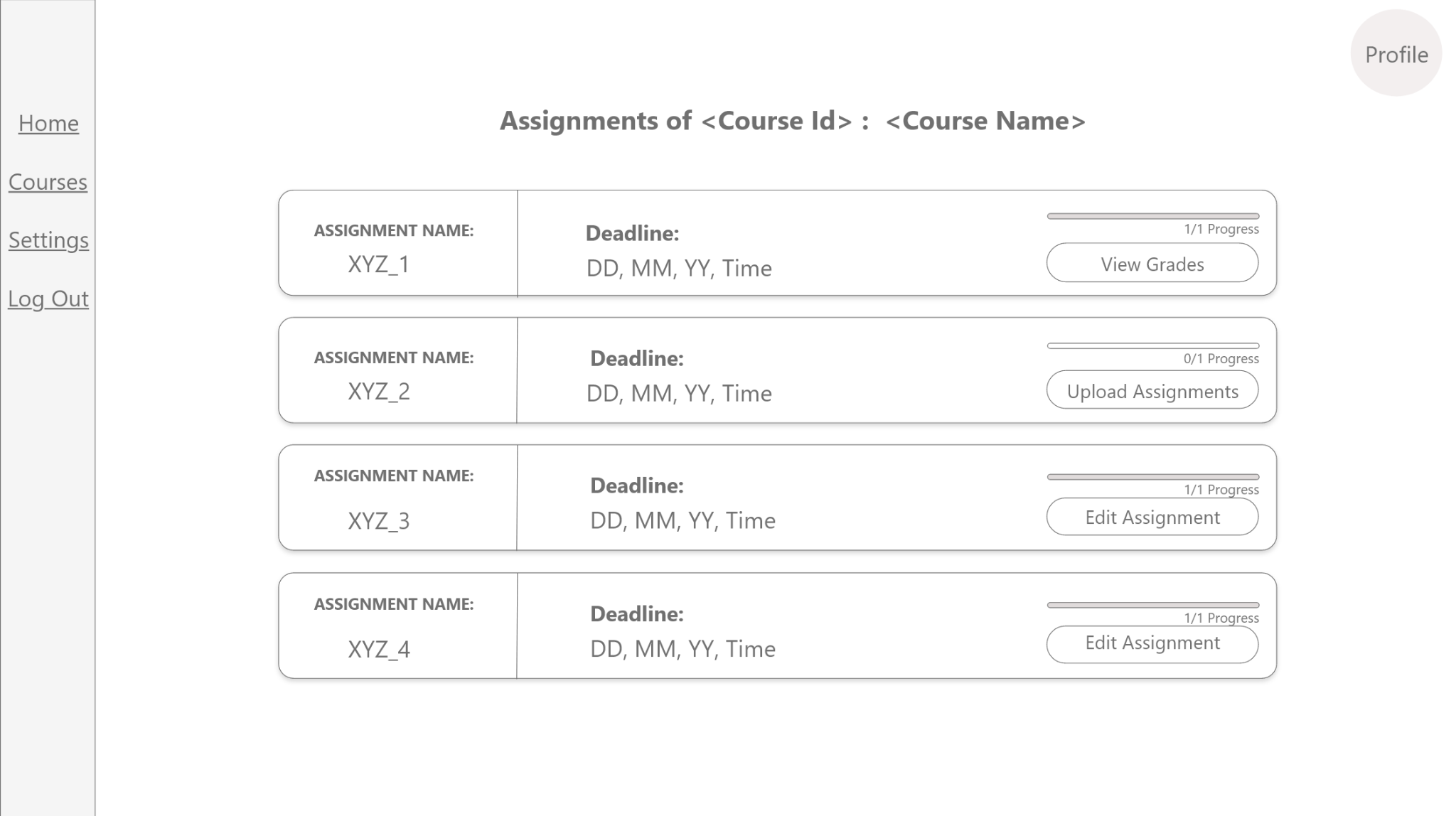
* This is the student Dashboard. Once a student logs in, he/she will be able to see this screen. Students will immediately be able to view the number of courses they are registered for, current courses they are enrolled in, the number of assignments due and the total number of credits they have earned.

#### Student-Course



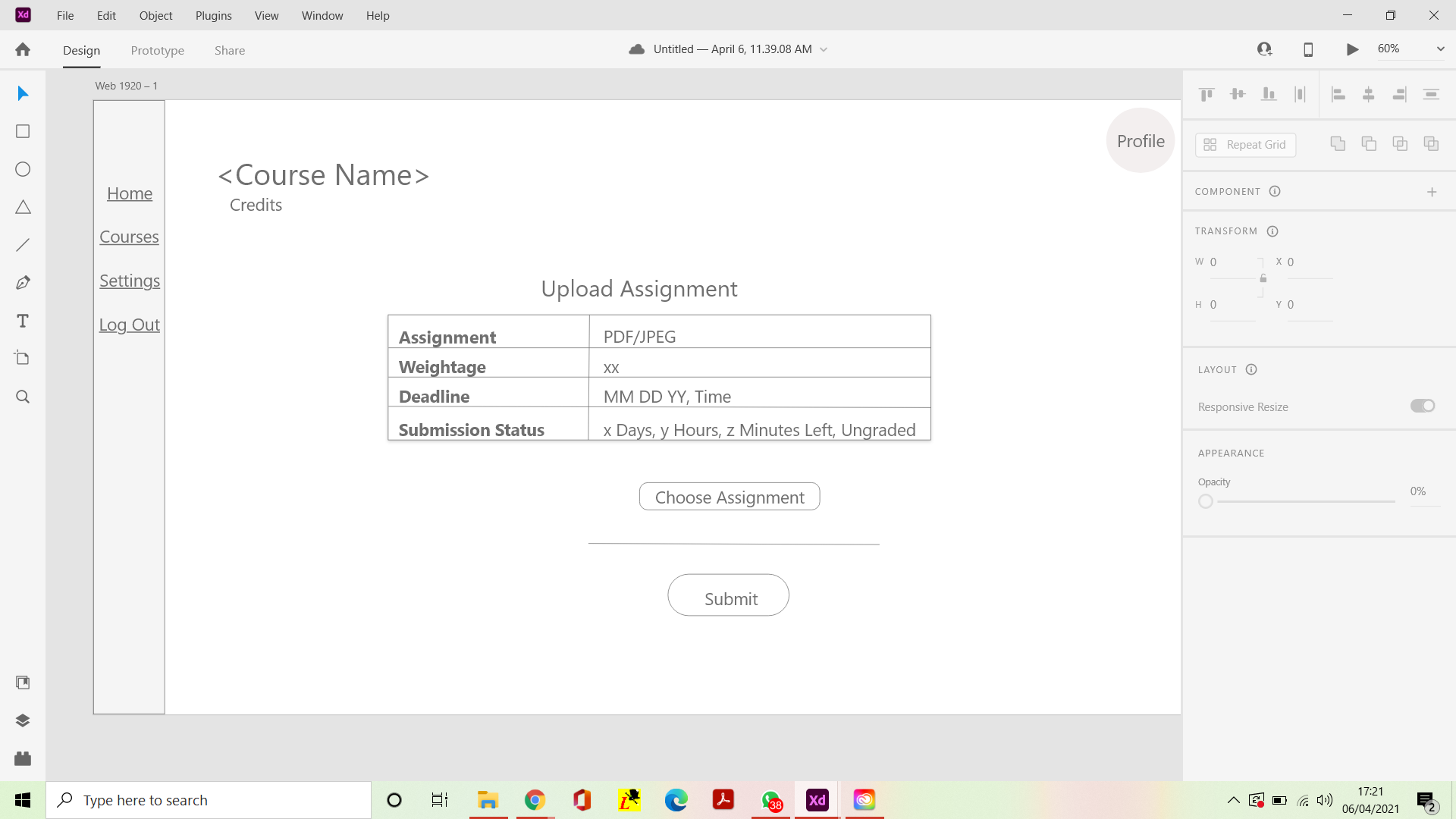
* Once the student clicks on a particular course, he/she would get directions to view the assignments, grades received and download the provided resources relevant to the particular course.

#### Student-View Assignment



* The assignments assigned by the faculty member can be viewed by the student with the deadline to submit the assignment. Students will be able to upload the assignment, edit the submitted assignment or view the grades received on the assignment based on the current status of the assignment.

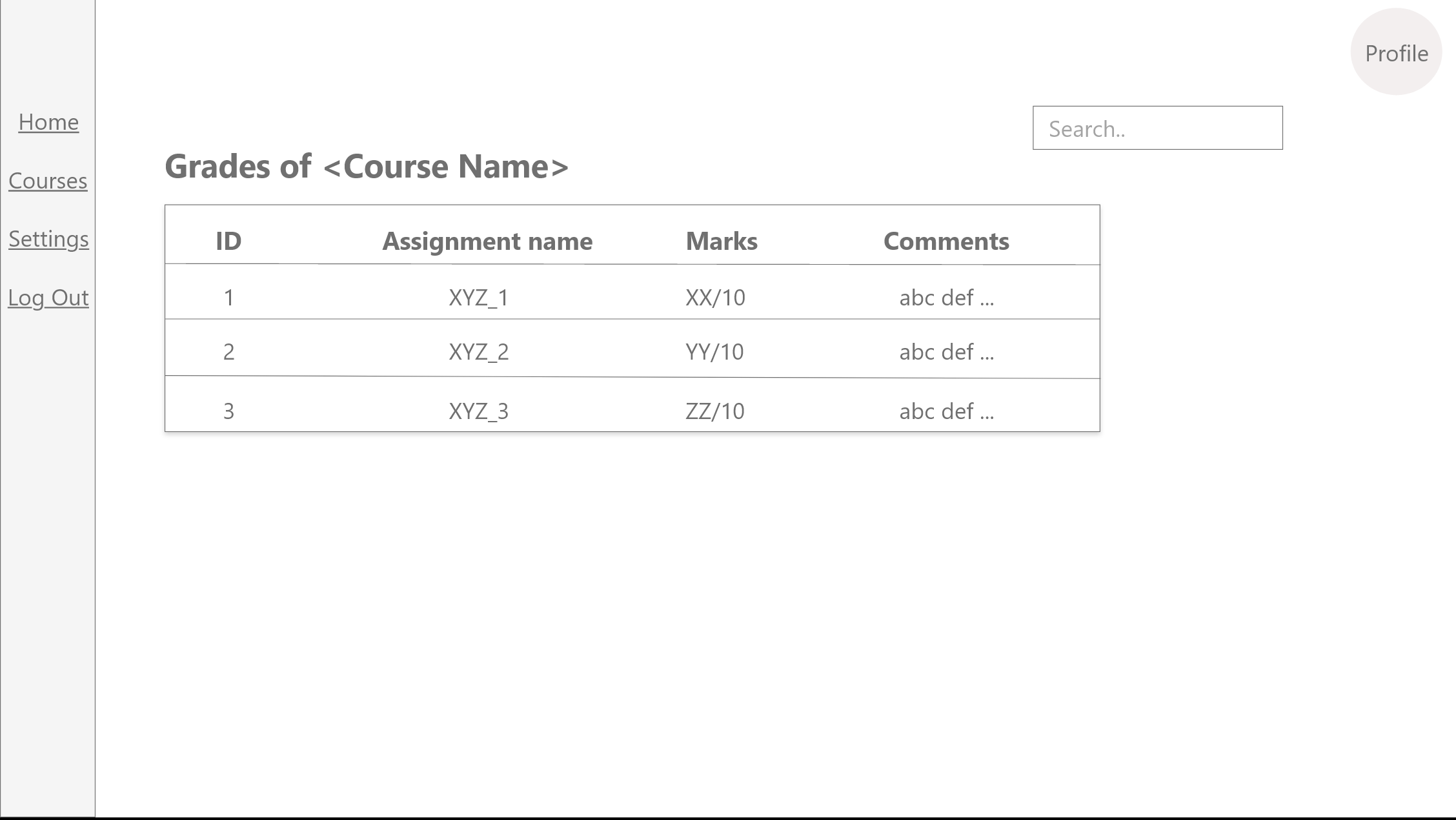
#### Student-Upload/Edit Assignment



* Students are provided a link where they can select documents from their computer

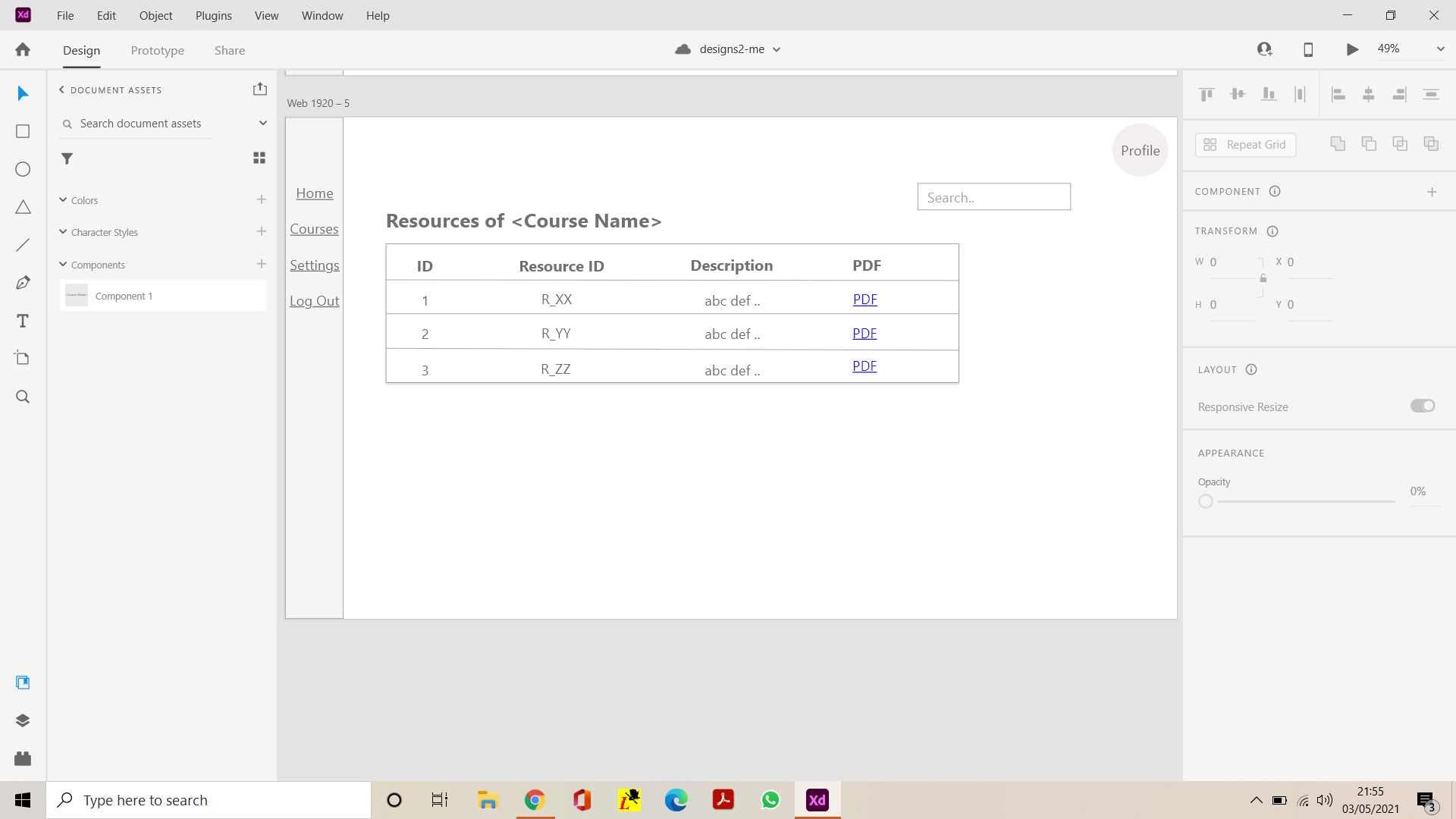
and upload them as submission to the assignment. The status of their submission can also be seen with the weightage and deadline of the assignment.

#### Student-View Grades



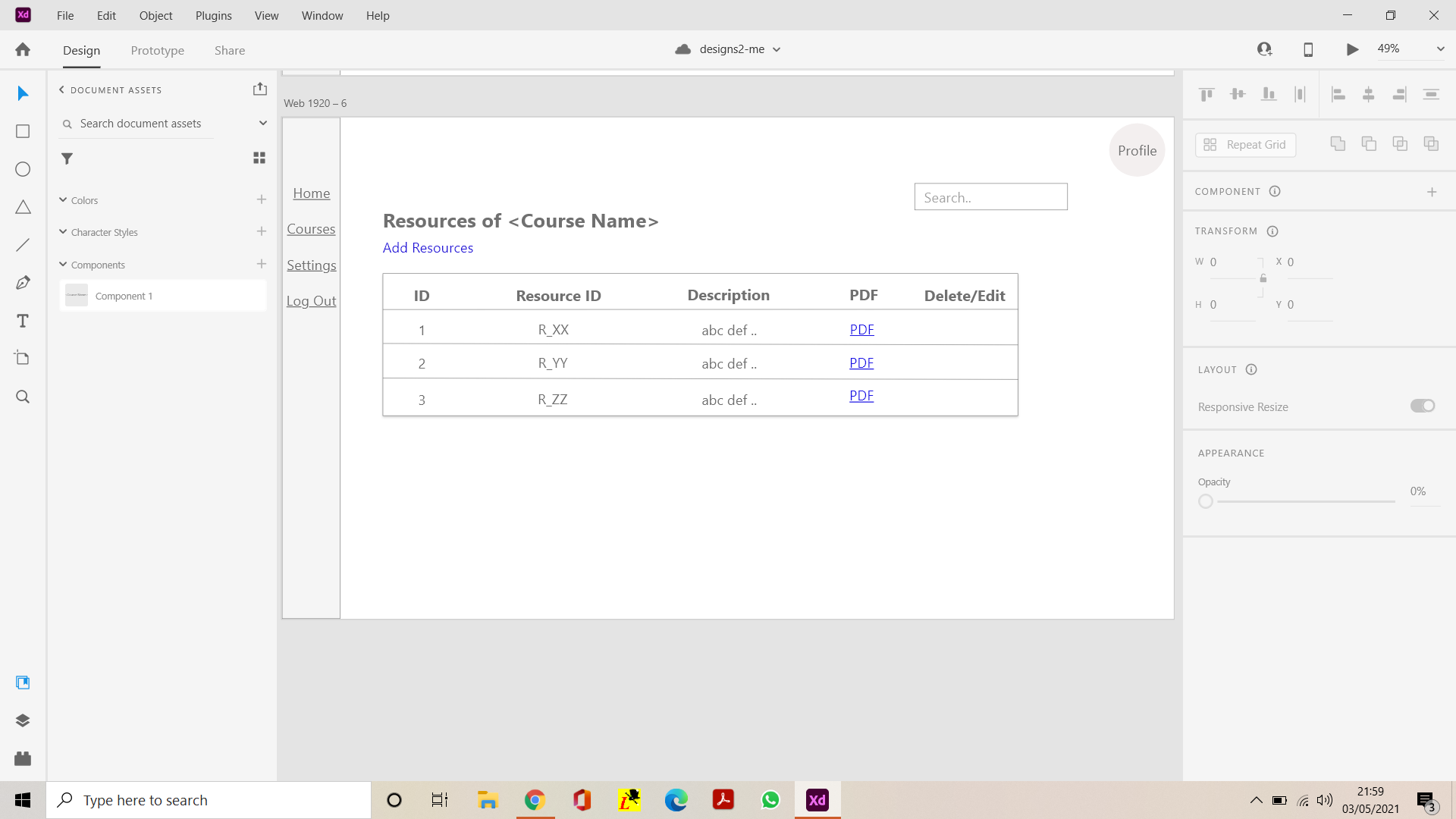
* The grades assigned by the faculty members for individual assignments can be viewed by the students with the comments provided by the faculty members.

6.1.1.16 Student View/Download Resource



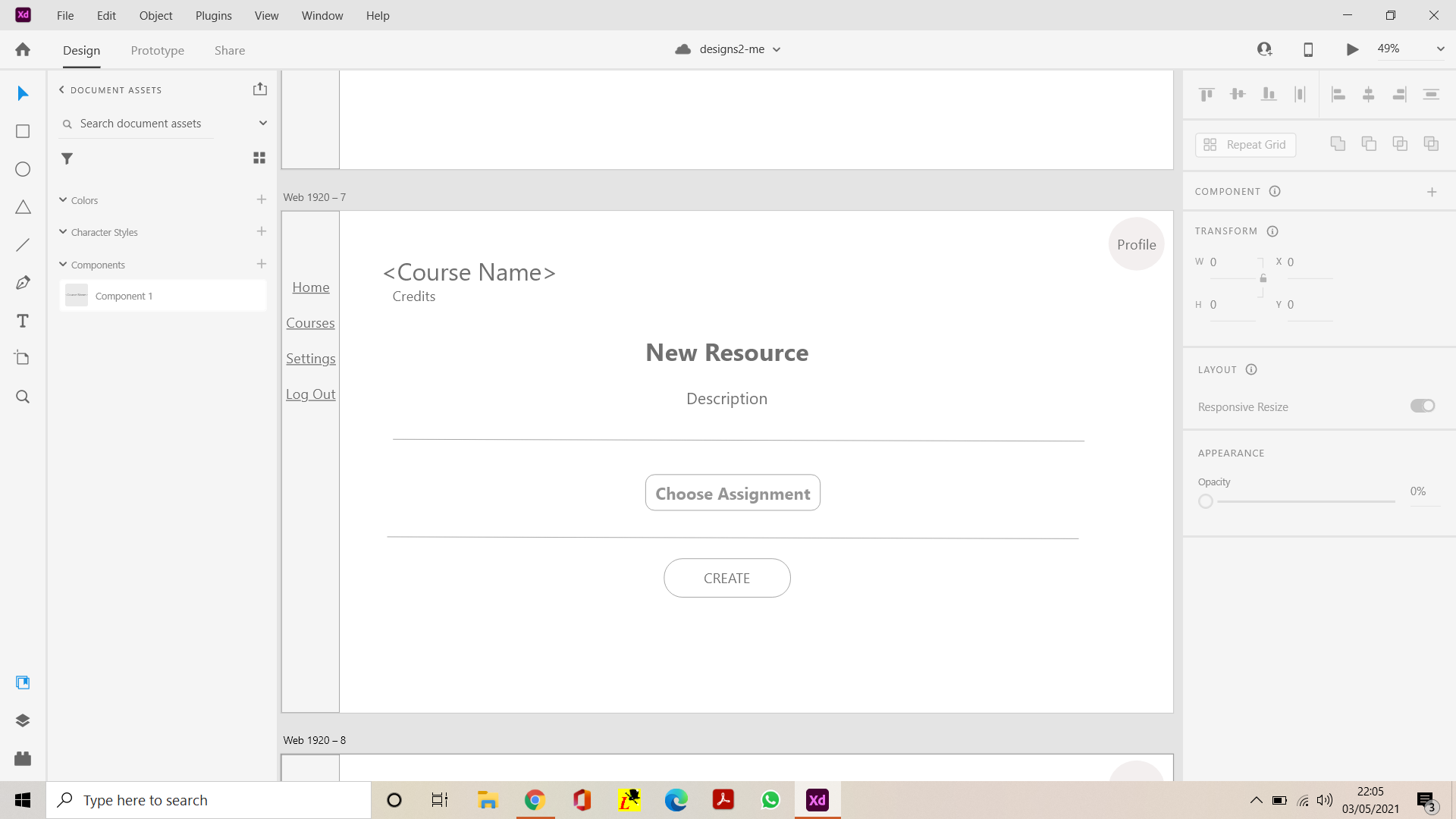
* The Student can find all the resources added by the faculty on this page. Students can also click on the PDF link button and can view or download the resource material.

6.1.1.17 Faculty - Upload Resources



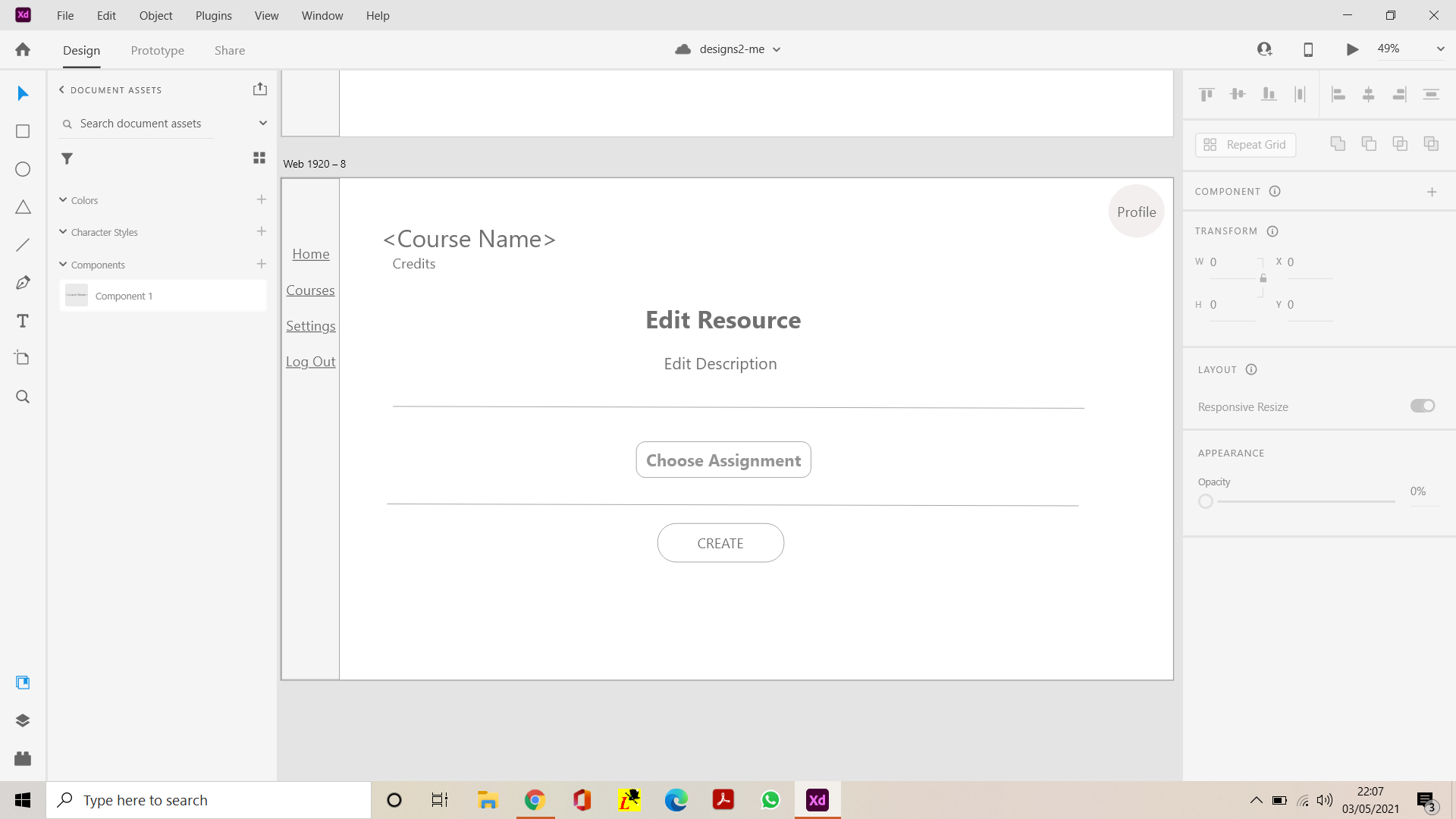
* Apart from viewing the resources that the faculty has uploaded, the faculty can also add, edit and delete any resource.

6.1.1.18 Faculty - Add Resource



* The Faculty can add a new resource. The faculty can choose an assignment from his/her local library and also mention any description if necessary.

6.1.1.19 Faculty - Edit Resource



* The faculty can edit any existing resource and can change the assignment from his/her local library. Moreover faculty can also edit the description of the Resource.

### 

### Objects and Actions

|  |  |
| --- | --- |
| **Navigation Buttons/Links** | **Activity performed** |
|  | This button will redirect to the user dashboard if the credentials are correct. |
|  | This link in the navigation bar will navigate the user to the dashboard from any page of the website. |
|  | This link in the navigation bar will navigate the user to the specific course clicked on in the dropdown list, |
|  | This link in the navigation bar will navigate the user to the settings page of the website. |
|  | This link in the navigation bar will log the user out. |
|  | This link will redirect the user to his/her profile page. |
|  | This button is used to submit the changed password. |
|  | This button is for the students to view the grades they have received. |
|  | This button is for students to edit the assignment submitted before the deadline in case they want to make changes and resubmit the assignment. |
|  | This button is for students to upload an assignment for submission. |
|  | This button is for students and faculties to choose a document to upload from their computer. |
|  | This button is for the students to submit their assignment for grading. |
|  | This button is for the faculty members to create a new assignment for the students. |
|  | This button enables the faculty members to enter marks for a specific assignment. |
|  | Faculty members can post the marks and comments of individual students on clicking this button. |
|  | This link on the home page will direct the user to a specific course. |
|  | This link on the courses page will direct the users to the Assignments page for the specific course. |
|  | This link on the courses page will direct the users to the Grades page for the specific course. |
|  | This link on the courses page will direct the user to upload course relevant resources. |
|  | This link on the courses page will direct the user to the enrolled students page. |
|  | This link directs the faculty members to the Create Assignment page. |
|  | This link directs the faculty members to the Edit Assignment page. |
|  | This link is provided to the faculty members to review the assignment uploaded. |
|  | On clicking on this icon, users will be displayed a date and time picker. |
|  | This link directs the faculty members to the Grading status page. |
|  | This link is provided to the faculty members to view the assignment submitted by students. |
|  | This link is provided to the students to view the assignment uploaded by faculty members. |

### Interface design rules

* Strive for consistency.
* Enable Frequent Users to Use Shortcuts.
* Offer informative feedback.
* Design dialogs to yield closure.
* Offer Simple Error Handling.
* Permit easy reversal of actions.
* Keep users in control.
* Reduce short-term memory load.
* Visibility
* Affordance

## Components available

|  |  |  |
| --- | --- | --- |
|  | Component | Examples for the website |
| Input Controls | Buttons |  |
| Dropdown lists |  |
| Date and Time picker |  |
| Text fields |  |
| Navigational Components | Search Field |  |
| Icons |  |
| Navigation bar |  |
| Information Components | Progress bar |  |
| Containers | Accordion |  |

## User Interface Development Description

The following principles are kept in mind while developing the user interface:

The following principles are kept in mind while developing the user interface:

**Consistency**

* Similar operations to achieve similar tasks.
* Makes things easy to learn and use
* Identical technology in prompts, menus, help.
* Color, capitalization, layout, fonts
* Consistency between updates of the website

**Offer Informative Feedback**

* For every user action, there should be system feedback.
* Feedback: Sends back information to the user that what action has been done and what has been accomplished. This allows the user to continue with activities.

**Design Dialogues to Yield Closure**

* Sequences of actions should be organized into groups with beginning, middle and end.

**Prevent Errors and Constraints**

* Design system such that the user can not make serious errors.
* Gray out menu that are not appropriate –
* Do not allow alphabetic characters in numeric entry fields Constraints
* Restricting users to actions that can take place at a given time.
* Prevents unintentional error.

**Permit Easy Reversal of Actions**

* As much as possible, all actions should be reversible.
* This feature relieves anxiety in the user and makes the user comfortable as the user knows that his/her action can be reverted back.

**Reduce Short Term Memory Load**

* Humans have limited capacity to store/process information in short term memory.
* Website locations should remain visible.
* Multiple page display should be consolidated
* Sufficient training time should be given for difficult sequence of actions

**Visibility**

* More visible functions are, more user know what to do next
* Car control board

**Affordance**

* Refers to an attribute of an object that allows people to know how to use it. • Eg: Mouse button invites pushing and by doing so activating clicking.
* When affordance of physical objects are perceptually obvious, it is easy to know how to interact with it.