Learning Management System

Kai Wang 30002810 T06

Jay Gurjar 30096042 T01

Ali Siddiqi 30092156 T01

Kundai Dziwa 30090173 T06

Abstract

In Kenya, Sir Richard Highschool is looking for a system or software that helps the school manage communication between its students, teachers and admins. So we have developed a learning management system that caters to their needs and solves their problems. The web based solution uses React.js on the frontend, Flask and MySql for the backend.

Table of Contents

Introduction

Despite the growing efforts to spread the capabilities of more education, there is still a pocket of improvement. Schools in Kenya and other developing countries do not have access to affordable learning management systems(LMS). They need a platform where students can view the professor's notes, submit assignments, view their grades and more. The solution is building a web based LMS where students can have access to their course content, view the status of their grades and communicate to teachers or students. The motivation is because building schools in Kenya is expensive so instead, having a LMS where students can learn through the web is a much more efficient option. This report comprehensively shows the entire working project as well as the design of it.

The proposed solution creates a LMS that will allow students to access course content. The project is scalable and more functionality can be added to it in the future. On the frontend React.js is used and Flask used for the api while the mysql was used for the database. The different types of users for the system are students, instructors and admin.The features that are included:

Student View:

- Students can login and logout
- A areas where students can view each of their courses
- A view where students can see their teachers posted documents/notes
- A view where students can upload files
- A classlist section that display all the students and teachers in a course
- A profile section where students can see and edit their personal information such as phone number, email, name, date of birth and request admin approval.
- A search feature that allows students to search for documents
- Students receive a course, teacher and TA evaluation form and can evaluate the components and submit it.
- Students can submit assignments
    - View Feedbacks(Grades) from teacher on assignments.

Teacher View:

- Create new course content
    - Upload files to the course page

    - Remove files from the course page
    - Edit files from the course page
- Can login and log out
- Assign grades to students for assignments
    - Grades can be added
- View the courses that they teach
- View the grades of each student submission
- View a classlist of all students an

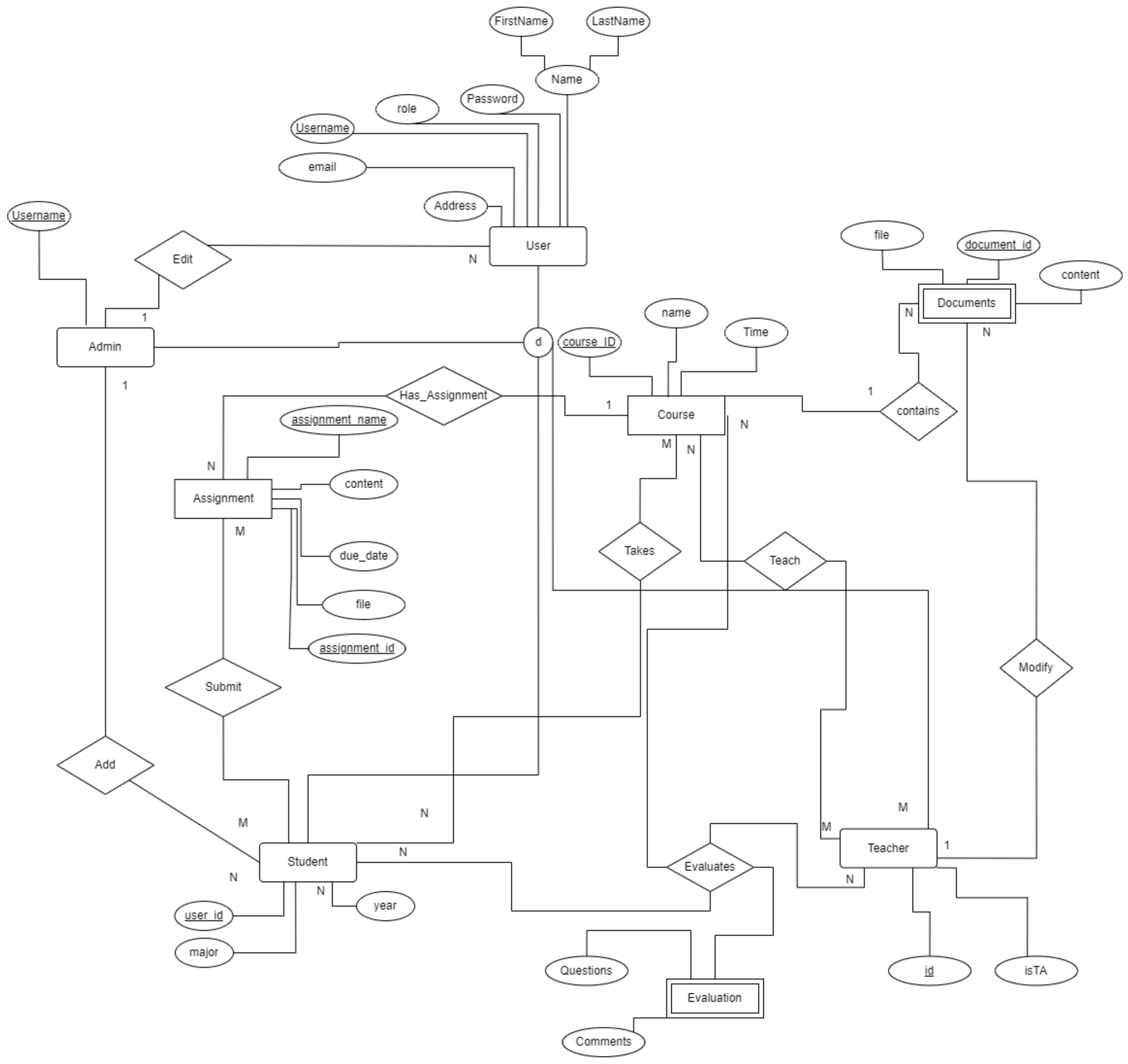
Admin

- Can add students to a Course
    - Remove students from a course
    - Edit students from a course

- Assign instructors to a course
    - Remove instructors from a course
    - Edit instructors from a course
    - Assign new instructor
- Edit the student profiles by student request
    - Edit their personal information such as phone number, email, name, date of birth
- Edit the profiles of teachers and their personal information or approve edits by teacher request.
- Sends/Receives evaluation from students and weighs the feedback using an algorithm.

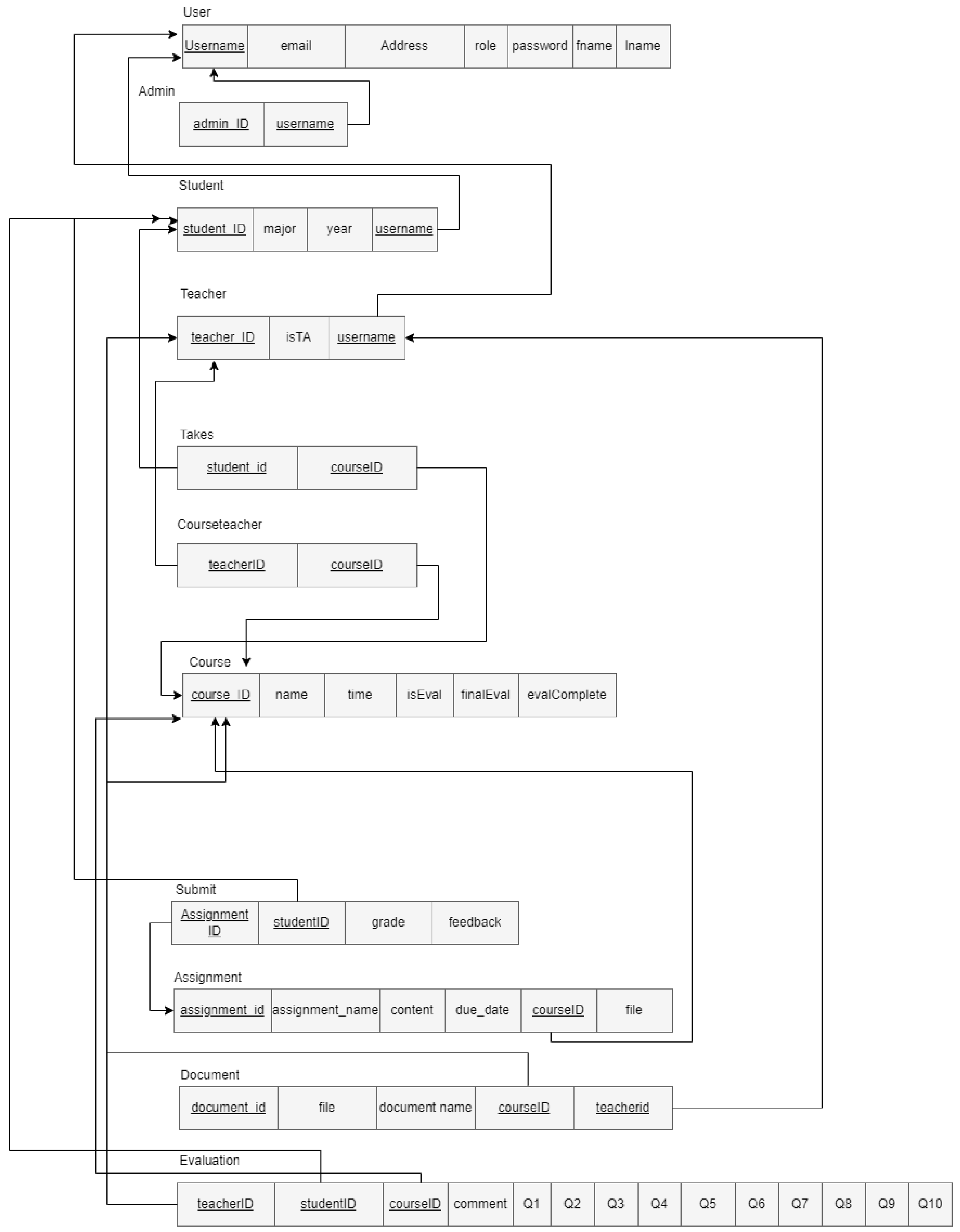
Design

Final EERD Diagram



The EERD diagram labeled above was the final diagram used for the project. There were a few small changes made, for example notifications, discussions, posts were removed with the approval of the professor and TA as they were deemed not too essential for project success. Another entity that was removed was emails. Instead of having emails as its own entity, we decided to store emails in the user entity which makes it a lot easier to access. We also added some extra functionalities such as the ability to upload a file in Assignment and Documents.

Final Relational Model



The relational model above was the final relational model used in order to build our database. The relational model was created following the EERD diagram and was designed accordingly. Many new tables were created such as Courseteacher, Takes, Submit which were used to store many to many relationships. Some unusual decisions that we had to make were deciding how a student can take multiple courses and how a course can handle multiple students. Ultimately we decided that it was better to create a many to many relationship which can store all these different cases. For example, a student can take four different classes such as CPSC 471, ENCM 511, CPSC 457 and CPSC 441. If we directly had the course connected to the student, it would get messy very quickly as the number of students grew. To handle these situations, we created the many to many relationship Takes which uses two foreign keys such as course_id and student_id to identify each individual.

Database Design

MySQL was chosen as the database used for this project. The database was designed closely based on the relational model to make sure that all relationships are met and the database integrity is maintained.

The queries below were used for the transactions implemented. They are very similar to the basic SQL queries however the values being inserted are dynamic and are allocated based on the operations performed in the front end. All of these queries were performed in flask using the MySQL connector.

An example of a query being run inside a function which selects all teachers then displays them on the front end.

```
app.config['MYSQL_HOST'] = 'localhost'
app.config['MYSQL_USER'] = 'root'
app.config['MYSQL_PORT'] = 3306
app.config['MYSQL_PASSWORD'] = ""
app.config['MYSQL_DB'] = "lmsdb"                      #database used
app.config['MYSQL_CURSORCLASS'] = 'DictCursor'

mysql = MySQL(app)

@app.route('/instructors', methods=['GET'])          #once inside the function
def instructors():
    if request.method == 'GET':
       cur = mysql.connection.cursor()                #connect to the database

       #execute the query
       cur.execute("SELECT * FROM user where role=(%s)", ("teacher",))

       instructors = cur.fetchall()                   #store the fetch data
       response = jsonify(instructors)                #respond in front end
       response.status_code = 200                     #success status
       cur.close()                                    #close the connector
       return response                                #return
```

SQL Statements used

The queries below performed the transactions in this project

Registering User:
```
"INSERT INTO
user(username,firstname,lastname,address,role,password,email) VALUES
(%s,%s,%s,%s,%s,%s,%s)" ,
(username,firstname,lastname,address,role,password,email)
```

Registering Student:
```
"INSERT INTO student(username,studentID,major,year)
VALUES(%s,%s,%s,%s)", (username,studentID,major,year)
```

Registering Admin:
```
"INSERT INTO admin(username,adminid) VALUES(%s,%s)",
(username,adminid)
```

Registering Teacher:
```
"INSERT INTO teacher(username,teacherid,isTA) VALUES(%s,%s,%s,)",
(username,teacherid,isTA)
```

Student Login:
```
"select student.studentID from user, student where user.username =
student.username and user.username=(%s) and user.password=(%s)",
(userName, password))
```

Teacher Login:
```
"select teacher.teacherid from user, teacher where user.username =
teacher.username and user.username=(%s) and user.password=(%s)",
(userName, password))
```

Admin Login:
```
"select email from user, admin where user.username=admin.username and
user.email=(%s) and user.password=(%s)", (email, password))
```

Inserting a Course:
```
"INSERT INTO course(courseid, name,time) VALUES(%s,%s,%s)", (courseid,
name,time)
```

See all students:
```
"SELECT * FROM user where role=(%s)", ("student",))
```

See individual student profile
```
"select * from user where username=(%s)", (stuUser,))
```

Update student name
```
"update user set firstname=(%s),lastname=(%s) where username=(%s)",
(firstName, lastName, stuUser))
```

Update student information
```
“update student set major=(%s),year=(%s) where username=(%s)", (major,
year, stuUser))
```

See all teachers
```
"SELECT * FROM user where role=(%s)", ("teacher",))
```

See individual teacher/instructor profile
```
"select * from user where username=(%s)", (insUser,))
```

Update teacher information
```
"update user set firstname=(%s),lastname=(%s) where username=(%s)",
(firstName, lastName, insUser))
```

See course Information
```
("select * from course where courseid=(%s)", (courseID,))
```

Set the teacher for the course:
```
“INSERT INTO courseteacher(courseid,teacherid) VALUES(%s,%s)”,
(courseid,teacherid)
```

Delete teacher from a course
```
"delete from courseteacher where teacherID=(%s) and courseid=(%s) ",
(teacherID, courseid))
```

Enroll a student into a course
```
"INSERT INTO takes(courseid,studentID) values (%s,%s)", (courseid,
studentID))
```

Unenroll a student from a course
```
"delete from takes where studentID=(%s) and courseid=(%s) ",
(studentID, courseid))
```

See the list of classes for a specific student
```
"select course.courseid, course.name, course.time from
takes,course,student,user where takes.courseid=course.courseid and
student.studentid=takes.studentid and student.username=user.username
and student.username=(%s)", (stuUser,))
```

See the list of students for a specific course
```
"select user.firstname, user.lastname,user.email from
takes,course,student,user where takes.courseid=course.courseid and
student.studentid=takes.studentid and student.username=user.username
and course.courseid=(%s)", (courseID, )
```

See the list of teachers for a specific course
```
"select user.firstname,  user.lastname,user.email, teacher.isTA from
courseteacher, course, user, teacher WHERE courseteacher.courseid =
course.courseid AND teacher.teacherid = courseteacher.teacherid AND
teacher.username = user.username AND course.courseid = (%s)",
(courseID, )
```

Admin can update user information
```
"update user set firstname=(%s),lastname=(%s) where username=(%s)",
(firstName, lastName, insUser))
```

See all courses that a specific instructor is teaching
```
("select teacher.teacherid,course.courseid,course.name from
courseteacher, course, teacher, user where
courseteacher.courseid=course.courseid and
teacher.teacherid=courseteacher.teacherid and
teacher.username=user.username and user.username=(%s)", (insUser,))
```

Send evaluations
```
"update course set isEval=(%s), finalEval=(%s) where courseid=(%s)",
(1, 1, courseID))
```

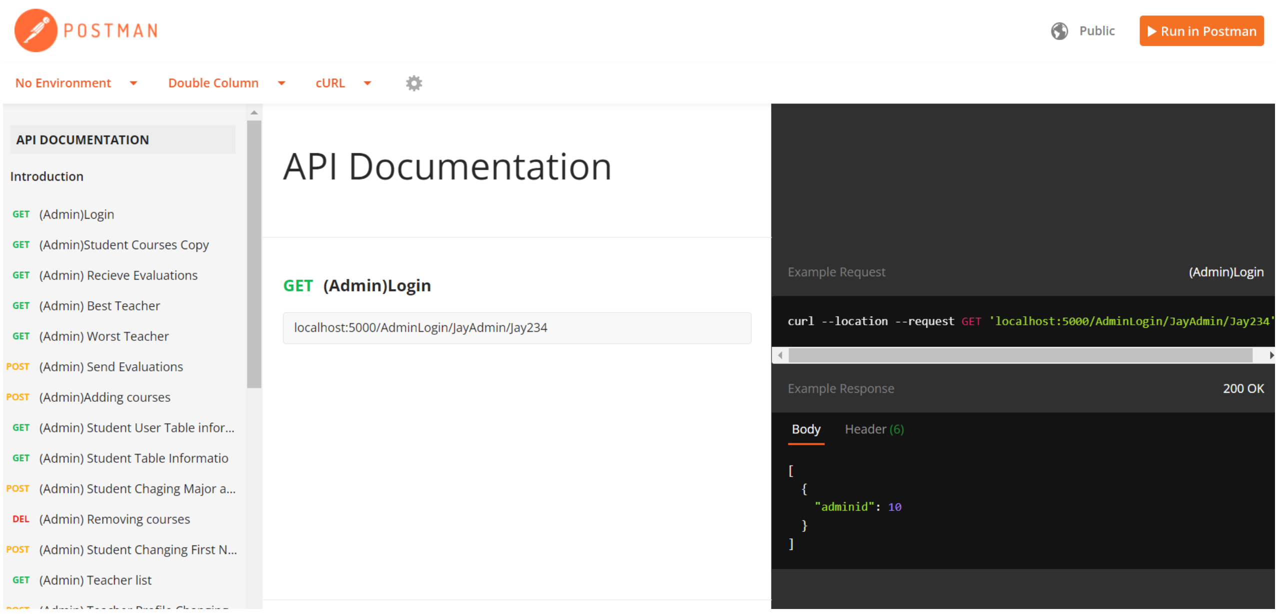
See all course evaluations for a specific course
```
select teacherid,studentid,comment,Q1,Q2,Q3,Q4,Q5,Q6,Q7,Q8,Q9,Q10 from
evaluation where courseid=(%s)", (courseID,))
```

Student submits an evaluation
```
"insert into evaluation(teacherid,Q1,Q2,Q3,Q4,Q5,Q6,Q7,Q8,Q9,Q10,
courseid,studentID,comment) values (%s,%s,%s,%s,%s,%s,%s,%s,%s,%s,
%s,%s,%s,%s)", (teacherID, Q1, Q2, Q3, Q4, Q5, Q6, Q7, Q8, Q9,Q10,
courseID, studentID, comments))
```

Get the assignment grade
```
"select assignment_name, grade from submit,assignment where
submit.assignment_id = assignment.assignment_id and
submit.studentid=(%s) and courseid=(%s)", (studentID, courseID)
```

API DOCUMENTATION

Here is the link to the postman api and sample requests.
https://documenter.getpostman.com/view/18285135/UVJZodkK#3d12851d-0f3f-41dc-938a-160527e97a1a



API DOCUMENTATION

Introduction

- GET (Admin)Login
- GET (Admin)Student Courses Copy
- GET (Admin) Recieve Evaluations
- GET (Admin) Best Teacher
- GET (Admin) Worst Teacher
- POST (Admin) Send Evaluations
- POST (Admin)Adding courses
- GET (Admin) Student User Table infor...
- GET (Admin) Student Table Informatio
- POST (Admin) Student Chaging Major a...
- DEL (Admin) Removing courses
- POST (Admin) Student Changing First N...
- GET (Admin) Teacher list

API Documentation

GET (Admin)Login

localhost:5000/AdminLogin/JayAdmin/Jay234

Example Request (Admin)Login

```
curl --location --request GET 'localhost:5000/AdminLogin/JayAdmin/Jay234'
```

Example Response 200 OK

Body Header (6)

```
[
  {
    "adminid": 10
  }
]
```

Screenshot of postman

USER GUIDE

Installation

Requirements to run the project:

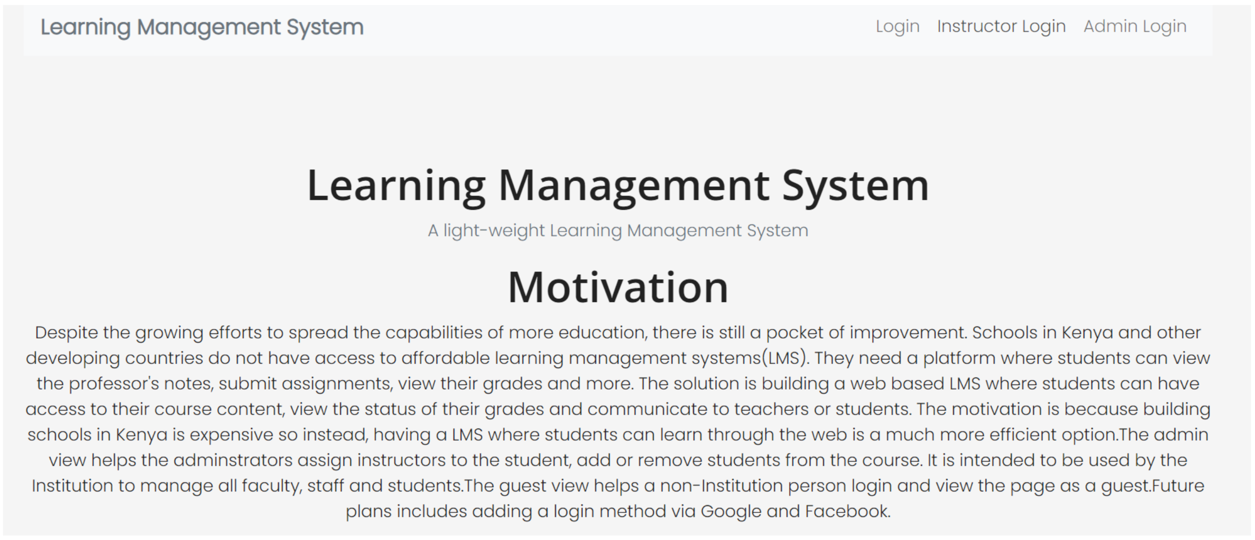
- Python 3.6 above
- MySQL

After the project is cloned, you should be in the Learning-Management System folder.
Install flask: Pip install flask
Create a virtual environment: python3 venv venv
Run: in one terminal: npm install, npm start,
Run in another terminal: ./venv/Scripts/activate

```
pip install flask flask_mysqldb flask_cors
backend/python app.py
```

The project will now open in your default browser.

After the project starts running, you will be greeted with a welcome page shown in the image below.



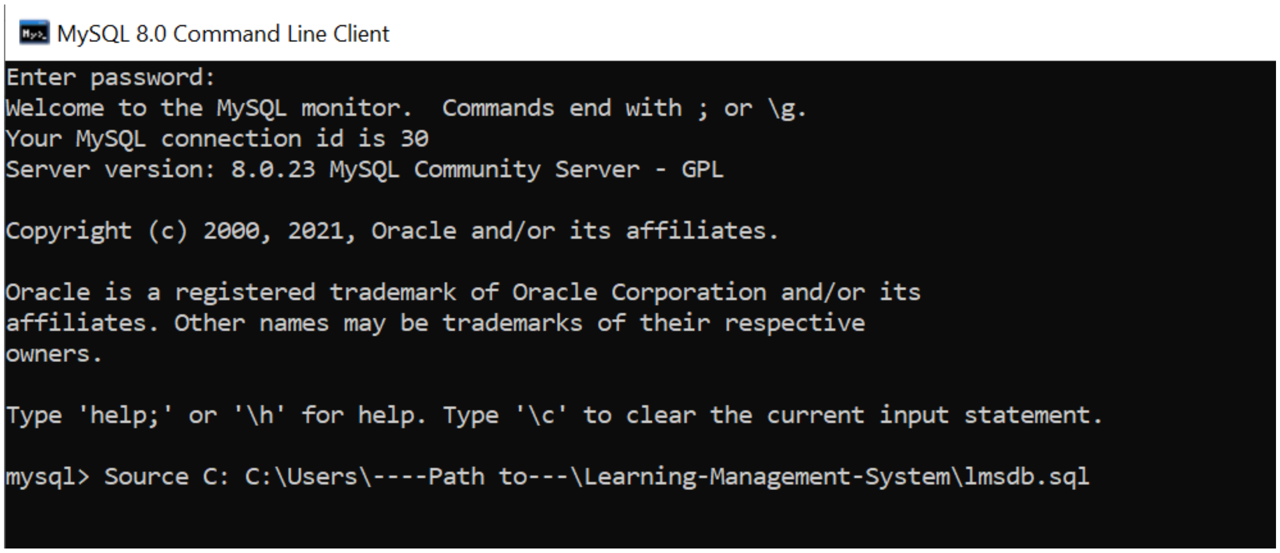
Learning Management System

A light-weight Learning Management System

Motivation

Despite the growing efforts to spread the capabilities of more education, there is still a pocket of improvement. Schools in Kenya and other developing countries do not have access to affordable learning management systems(LMS). They need a platform where students can view the professor's notes, submit assignments, view their grades and more. The solution is building a web based LMS where students can have access to their course content, view the status of their grades and communicate to teachers or students. The motivation is because building schools in Kenya is expensive so instead, having a LMS where students can learn through the web is a much more efficient option.The admin view helps the adminstrators assign instructors to the student, add or remove students from the course. It is intended to be used by the Institution to manage all faculty, staff and students.The guest view helps a non-Institution person login and view the page as a guest.Future plans includes adding a login method via Google and Facebook.

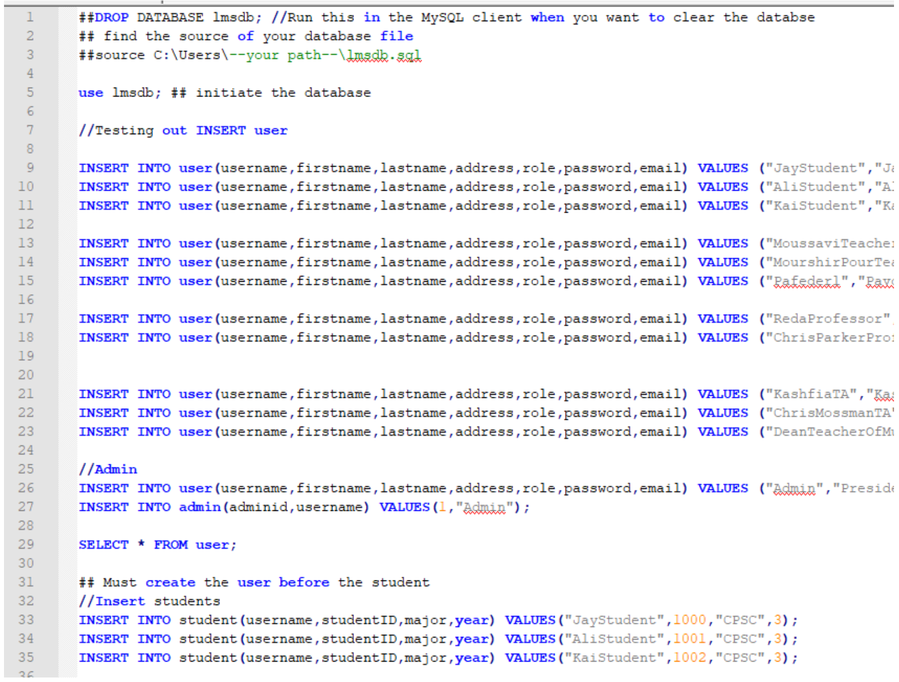
The database "lmsdb.sql" should be used and is sourced by using the command
Source C: C:\Users\----Path to---\Learning-Management-System\lmsdb.sql


MySQL 8.0 Command Line Client

```
Enter password:
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 30
Server version: 8.0.23 MySQL Community Server - GPL

Copyright (c) 2000, 2021, Oracle and/or its affiliates.

Oracle is a registered trademark of Oracle Corporation and/or its
affiliates. Other names may be trademarks of their respective
owners.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

mysql> Source C: C:\Users\----Path to---\Learning-Management-System\lmsdb.sql
```

The user can manually add students/teachers as an admin however if the user wants to quickly populate the database, they can do so using the querytest.txt file provided. The user should select the sections that they want to populate in the database and paste it in the MySQL command line client.

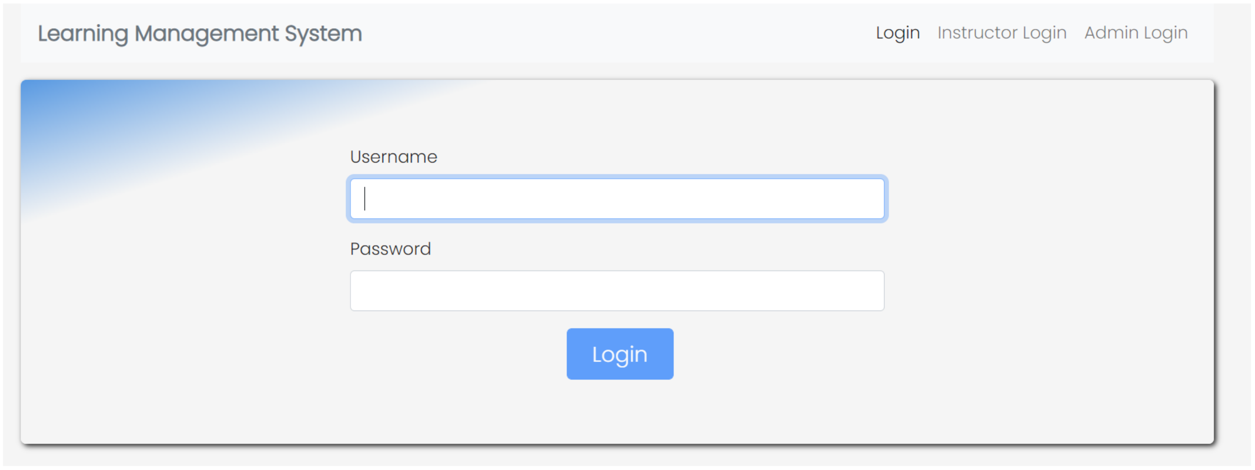
Screenshot of querytest.txt file.



```
##DROP DATABASE lmsdb; //Run this in the MySQL client when you want to clear the databse
## find the source of your database file
##source C:\Users\--your path--\lmsdb.sql

use lmsdb; ## initiate the database

//Testing out INSERT user

INSERT INTO user(username,firstname,lastname,address,role,password,email) VALUES ("JayStudent","Ja
INSERT INTO user(username,firstname,lastname,address,role,password,email) VALUES ("AliStudent","A
INSERT INTO user(username,firstname,lastname,address,role,password,email) VALUES ("KaiStudent","Ka

INSERT INTO user(username,firstname,lastname,address,role,password,email) VALUES ("MoussaviTeache
INSERT INTO user(username,firstname,lastname,address,role,password,email) VALUES ("MourshirPourTe
INSERT INTO user(username,firstname,lastname,address,role,password,email) VALUES ("Pafederl","Pay

INSERT INTO user(username,firstname,lastname,address,role,password,email) VALUES ("RedaProfessor"
INSERT INTO user(username,firstname,lastname,address,role,password,email) VALUES ("ChrisParkerPro


INSERT INTO user(username,firstname,lastname,address,role,password,email) VALUES ("KashfiaTA","Ka
INSERT INTO user(username,firstname,lastname,address,role,password,email) VALUES ("ChrisMossmanTA"
INSERT INTO user(username,firstname,lastname,address,role,password,email) VALUES ("DeanTeacherOfM

//Admin
INSERT INTO user(username,firstname,lastname,address,role,password,email) VALUES ("Admin","Preside
INSERT INTO admin(adminid,username) VALUES(1,"Admin");

SELECT * FROM user;

## Must create the user before the student
//Insert students
INSERT INTO student(username,studentID,major,year) VALUES("JayStudent",1000,"CPSC",3);
INSERT INTO student(username,studentID,major,year) VALUES("AliStudent",1001,"CPSC",3);
INSERT INTO student(username,studentID,major,year) VALUES("KaiStudent",1002,"CPSC",3);
```

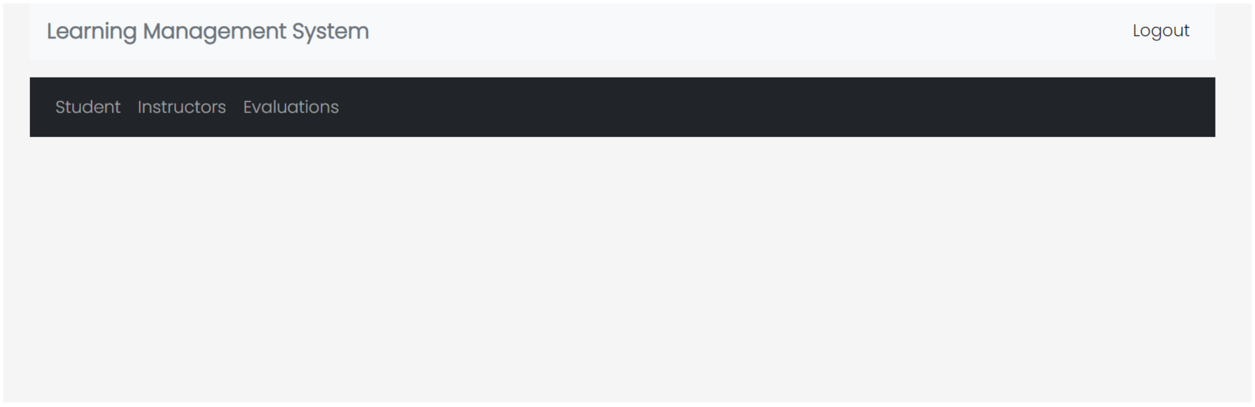
Once the database is populated, the application becomes fully operational.

Login page, log in with your respective role

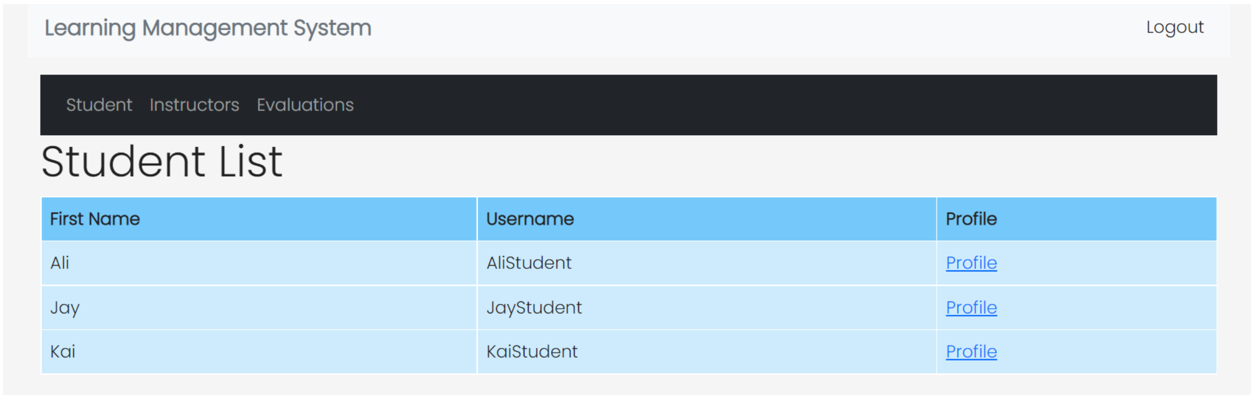


Admin

Once logged in as an admin



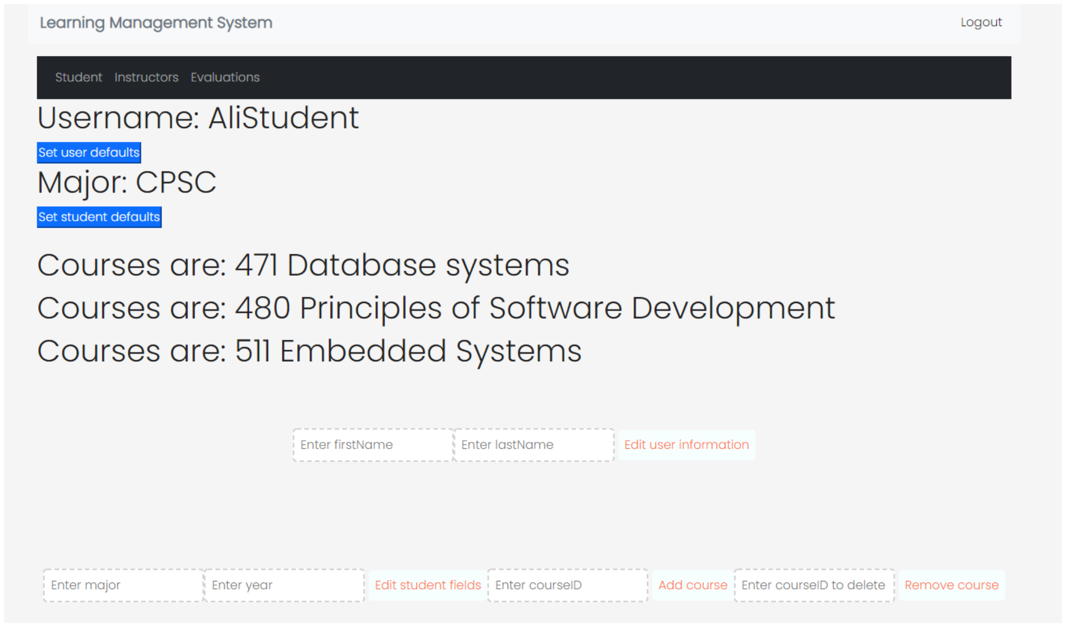
You now have the privilege to modify students, teachers and see evaluations.



Student List

First Name	Username	Profile
Ali	AliStudent	Profile
Jay	JayStudent	Profile
Kai	KaiStudent	Profile

The admin can view the list of students and make changes to their profile.



Username: AliStudent

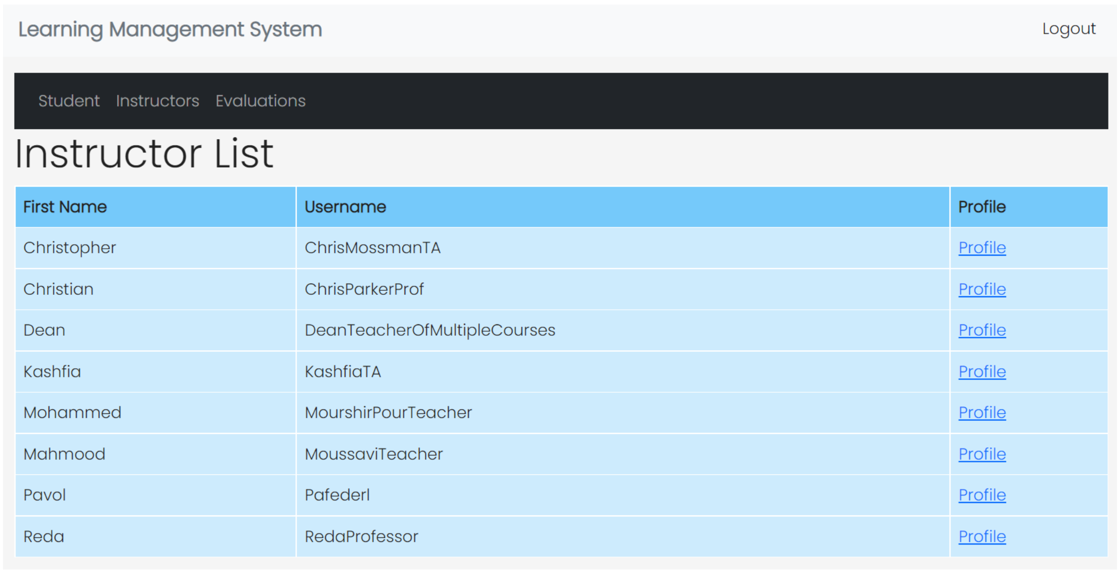
Major: CPSC

Courses are: 471 Database systems

Courses are: 480 Principles of Software Development

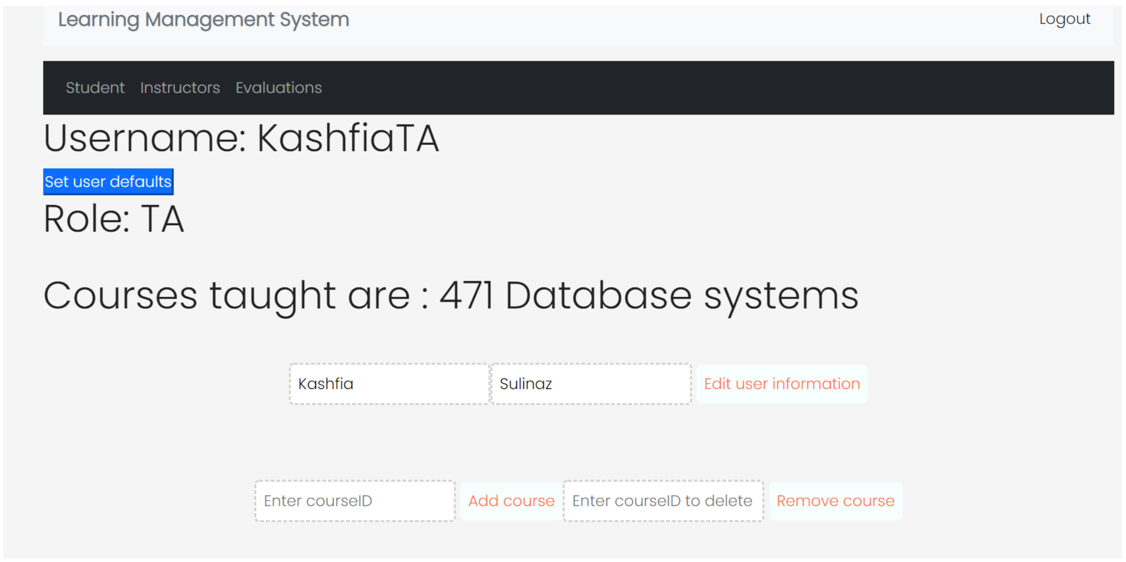
Courses are: 511 Embedded Systems

The admin can also view the list of all instructors and make changes to their profile as well.



Instructor List

First Name	Username	Profile
Christopher	ChrisMossmanTA	Profile
Christian	ChrisParkerProf	Profile
Dean	DeanTeacherOfMultipleCourses	Profile
Kashfia	KashfiaTA	Profile
Mohammed	MourshirPourTeacher	Profile
Mahmood	MoussaviTeacher	Profile
Pavol	Pafederl	Profile
Reda	RedaProfessor	Profile

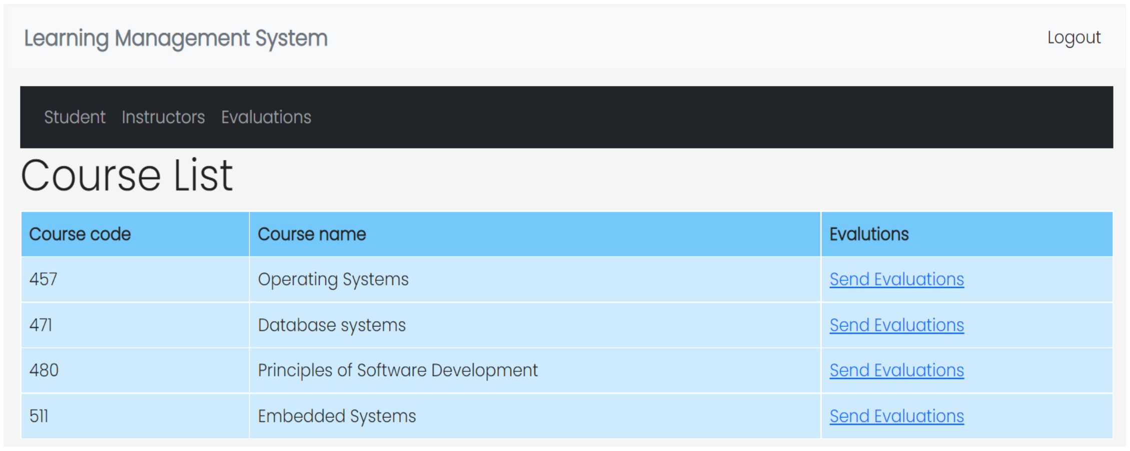


Username: KashfiaTA

Role: TA

Courses taught are : 471 Database systems

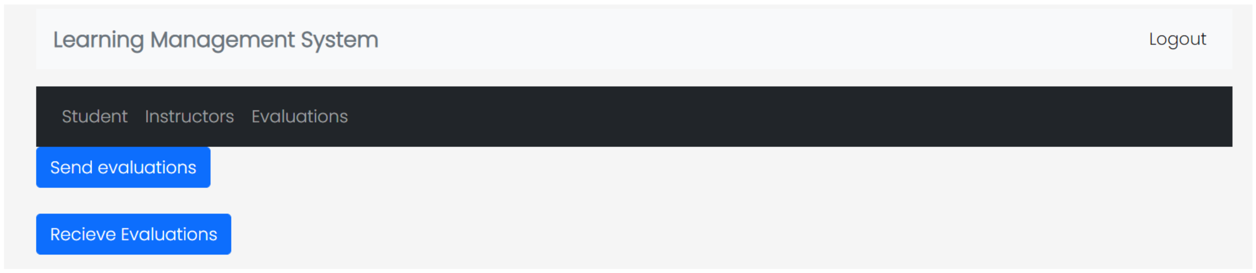
The admin can also view all courses and their evaluations and choose the action performed.



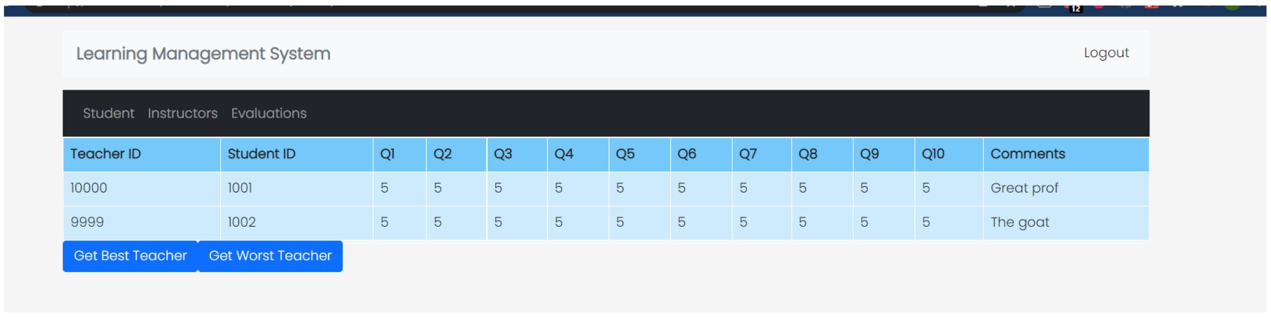
Course List

Course code	Course name	Evalutions
457	Operating Systems	Send Evaluations
471	Database systems	Send Evaluations
480	Principles of Software Development	Send Evaluations
511	Embedded Systems	Send Evaluations

After clicking send evaluations, there are two options to send all students a evaluation or receive the evaluations



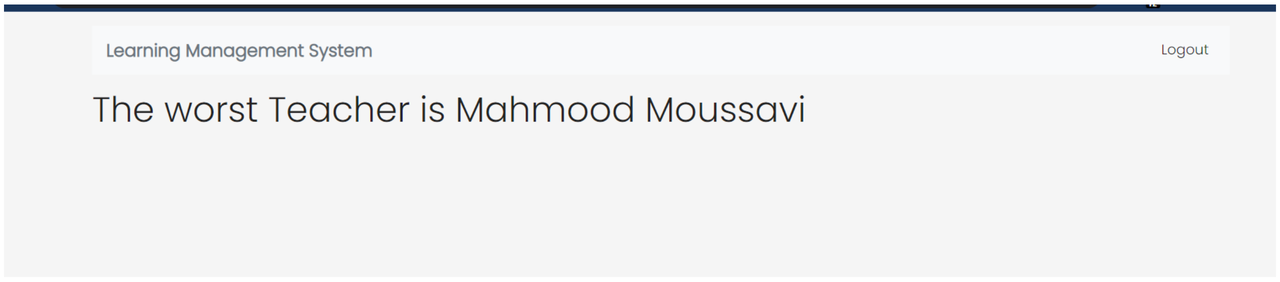
After clicking receive evaluation


Learning Management System Logout

Student Instructors Evaluations

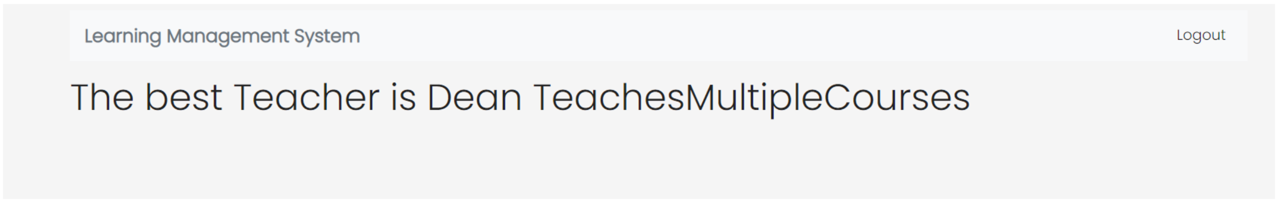
Teacher ID	Student ID	Q1	Q2	Q3	Q4	Q5	Q6	Q7	Q8	Q9	Q10	Comments
10000	1001	5	5	5	5	5	5	5	5	5	5	Great prof
9999	1002	5	5	5	5	5	5	5	5	5	5	The goat

After clicking Get best teacher button: finds the best teacher based on evaluation marks and an algorithm component as well.


Learning Management System Logout

The worst Teacher is Mahmood Moussavi

After clicking Get worst teacher button: finds the worst teacher based on evaluation marks and an algorithm component as well

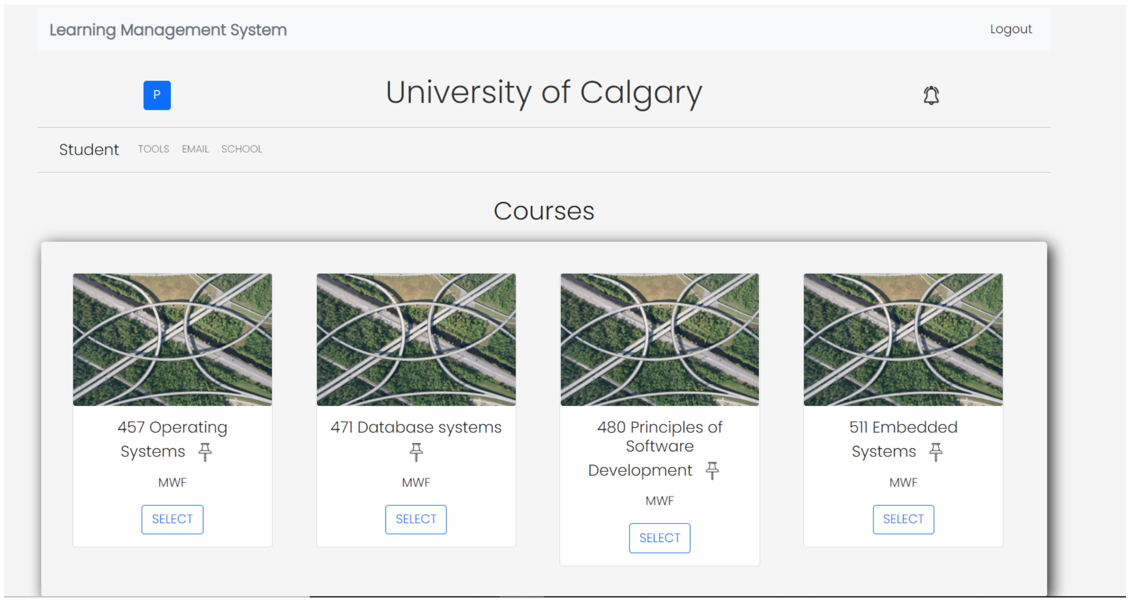

Learning Management System Logout

The best Teacher is Dean TeachesMultipleCourses

Student

The student login page will look like the image below, with many options such as courses, tools, email and school.The student can click on the highlighted P and see his own profile.

Courses page:



Learning Management System Logout

University of Calgary

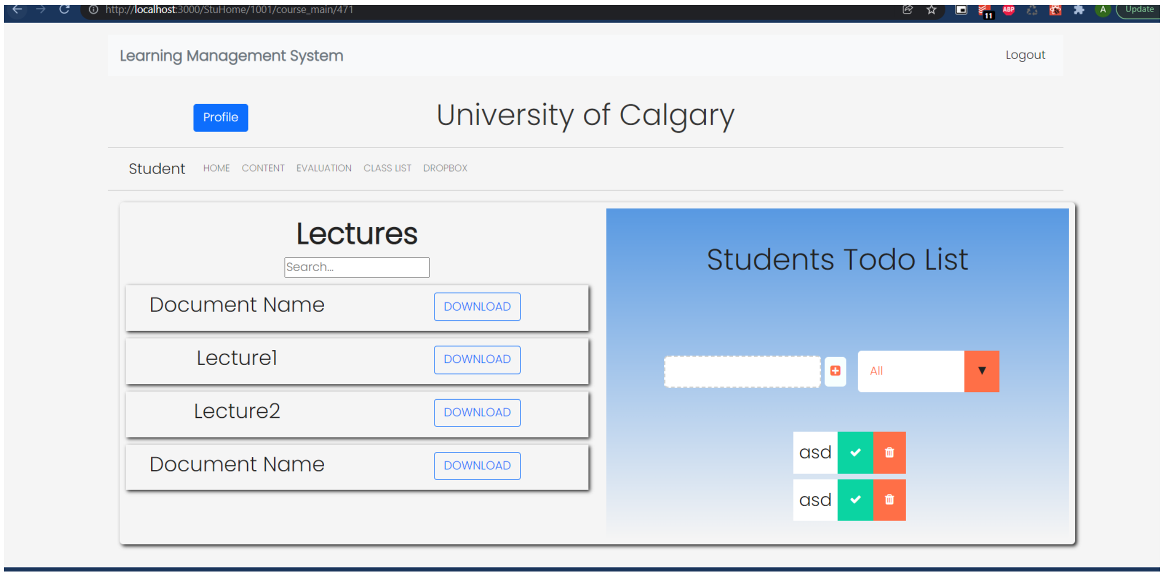
Student TOOLS EMAIL SCHOOL

Courses

457 Operating Systems	471 Database systems	480 Principles of Software Development	511 Embedded Systems
MWF	MWF	MWF	MWF
SELECT	SELECT	SELECT	SELECT

Once the student clicks on a specific course, they will be able to see the contents, emails, dropbox and evaluation for that class.

Home page: shows lecture and todolist



http://localhost:3000/StuHome/1001/course_main/471

Learning Management System Logout

Profile University of Calgary

Student HOME CONTENT EVALUATION CLASS LIST DROPBOX

Lectures

Search...

Document Name	DOWNLOAD
Lecture1	DOWNLOAD
Lecture2	DOWNLOAD
Document Name	DOWNLOAD

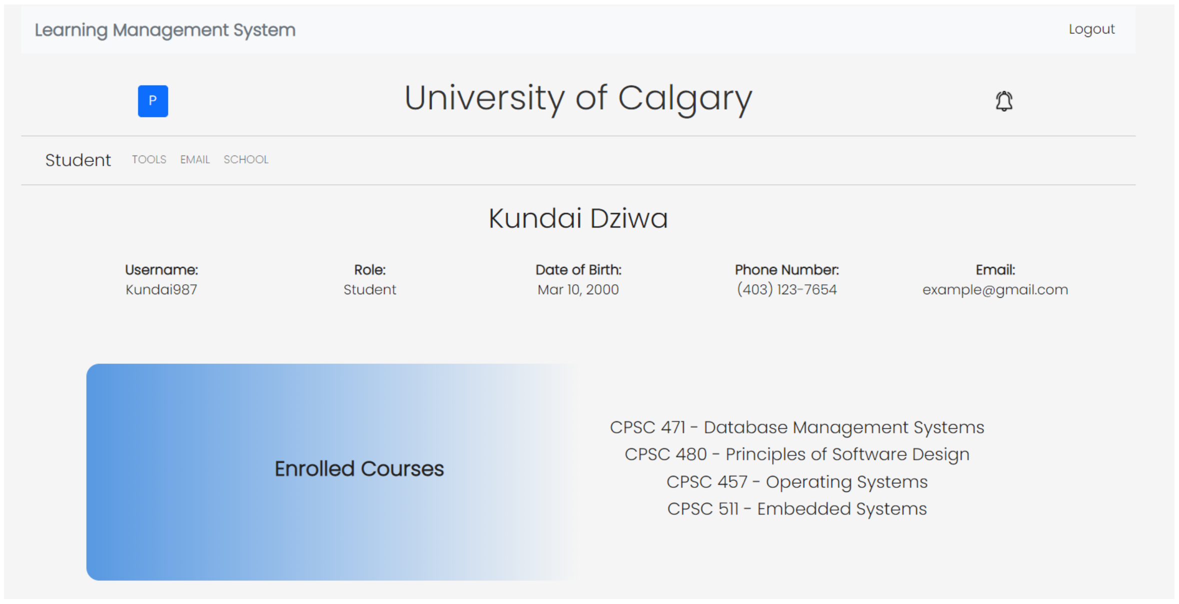
Students Todo List

All

asd

asd

Student Profile page: shows info about student such as name and phone number


Learning Management System Logout

P University of Calgary

Student TOOLS EMAIL SCHOOL

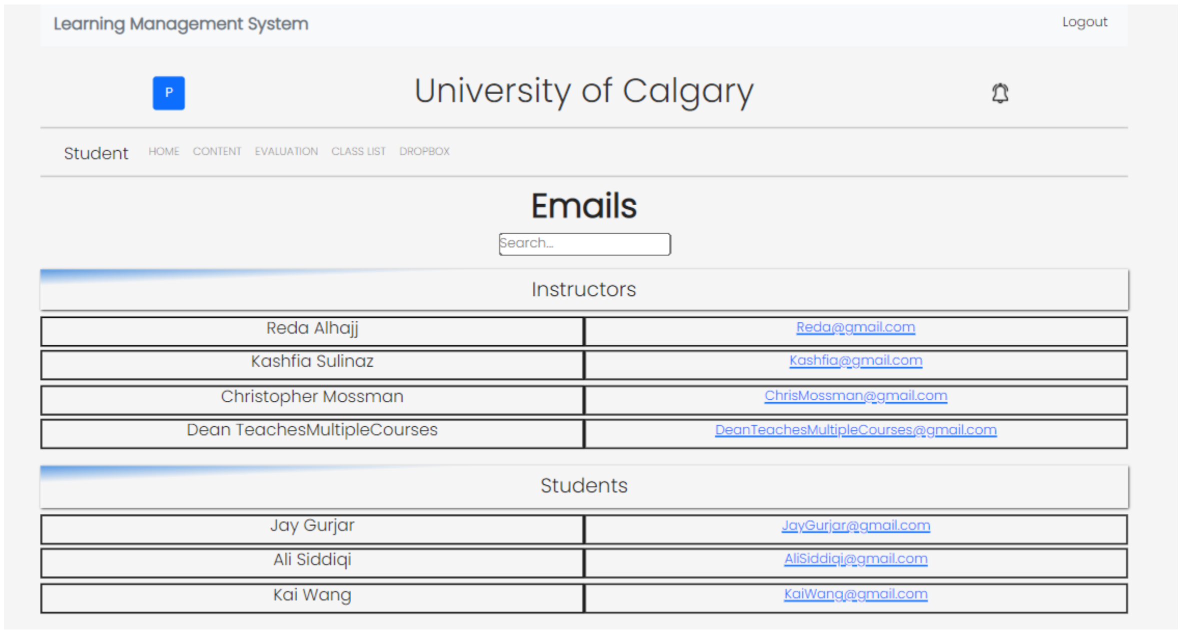
Kundai Dziwa

Username:	Role:	Date of Birth:	Phone Number:	Email:
Kundai987	Student	Mar 10, 2000	(403) 123-7654	example@gmail.com

Enrolled Courses

CPSC 471 - Database Management Systems
CPSC 480 - Principles of Software Design
CPSC 457 - Operating Systems
CPSC 511 - Embedded Systems

Class List page: shows teacher and students in a course


Learning Management System Logout

P University of Calgary

Student HOME CONTENT EVALUATION CLASS LIST DROPBOX

Emails

Search...

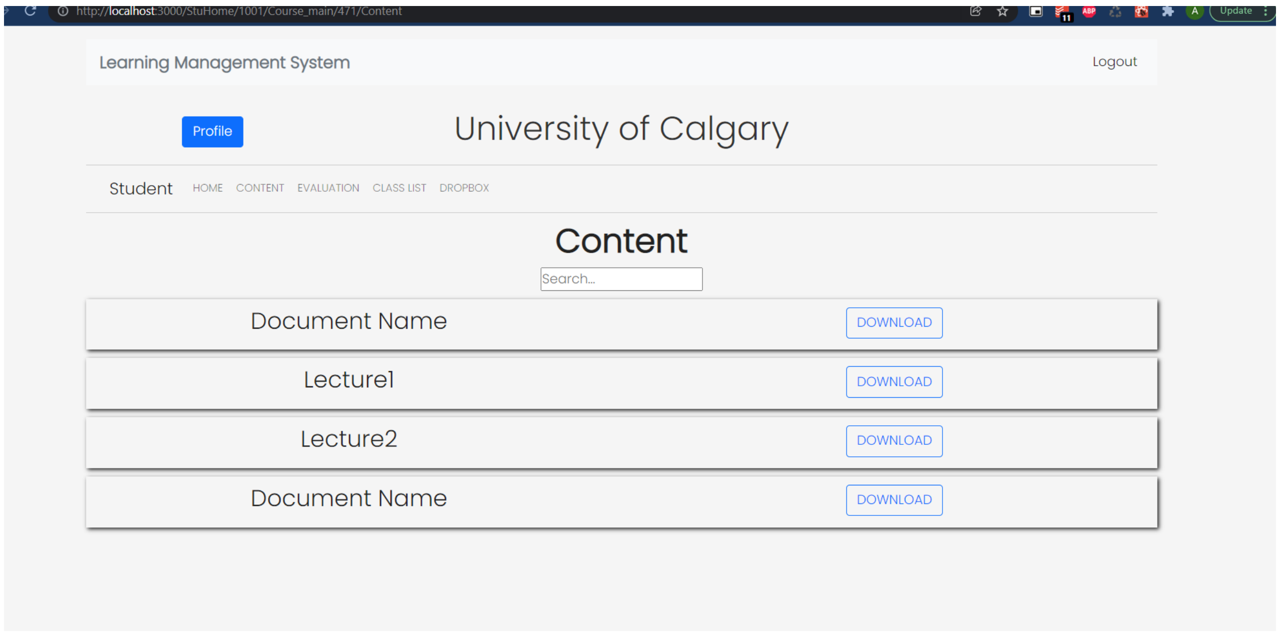
Instructors

Reda Alhajj	Reda@gmail.com
Kashfia Sulinaz	Kashfia@gmail.com
Christopher Mossman	ChrisMossman@gmail.com
Dean TeachesMultipleCourses	DeanTeachesMultipleCourses@gmail.com

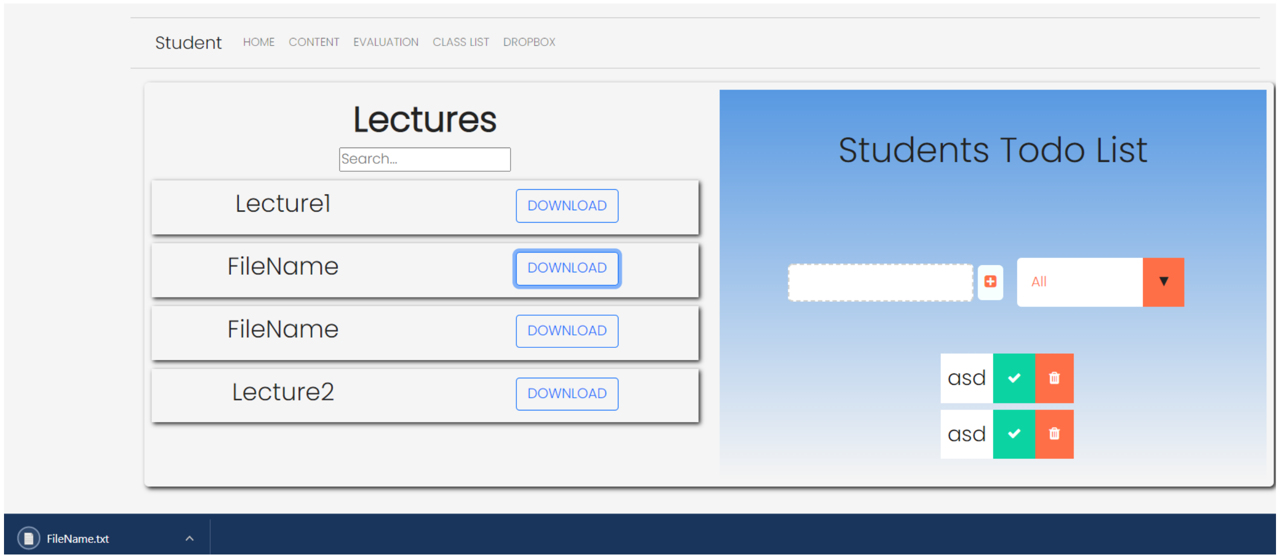
Students

Jay Gurjar	JayGurjar@gmail.com
Ali Siddiqi	AliSiddiqi@gmail.com
Kai Wang	KaiWang@gmail.com

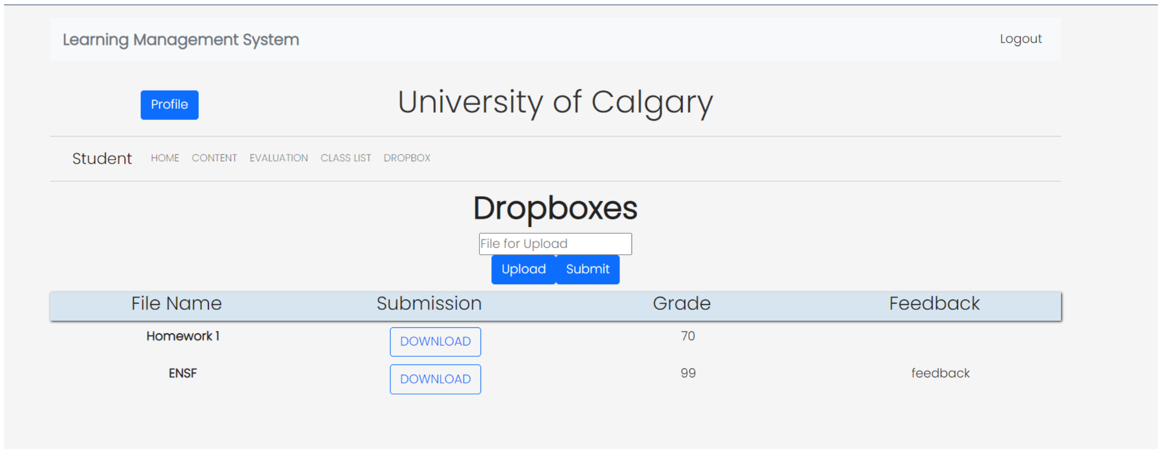
Content page: shows the lectures the teacher uploaded



Clicking on the download button downloads the file:



Dropbox page: allows submission of assignments and sends it all to the teacher


Learning Management System

Logout

Profile

University of Calgary

Student HOME CONTENT EVALUATION CLASS LIST DROPBOX

Dropboxes

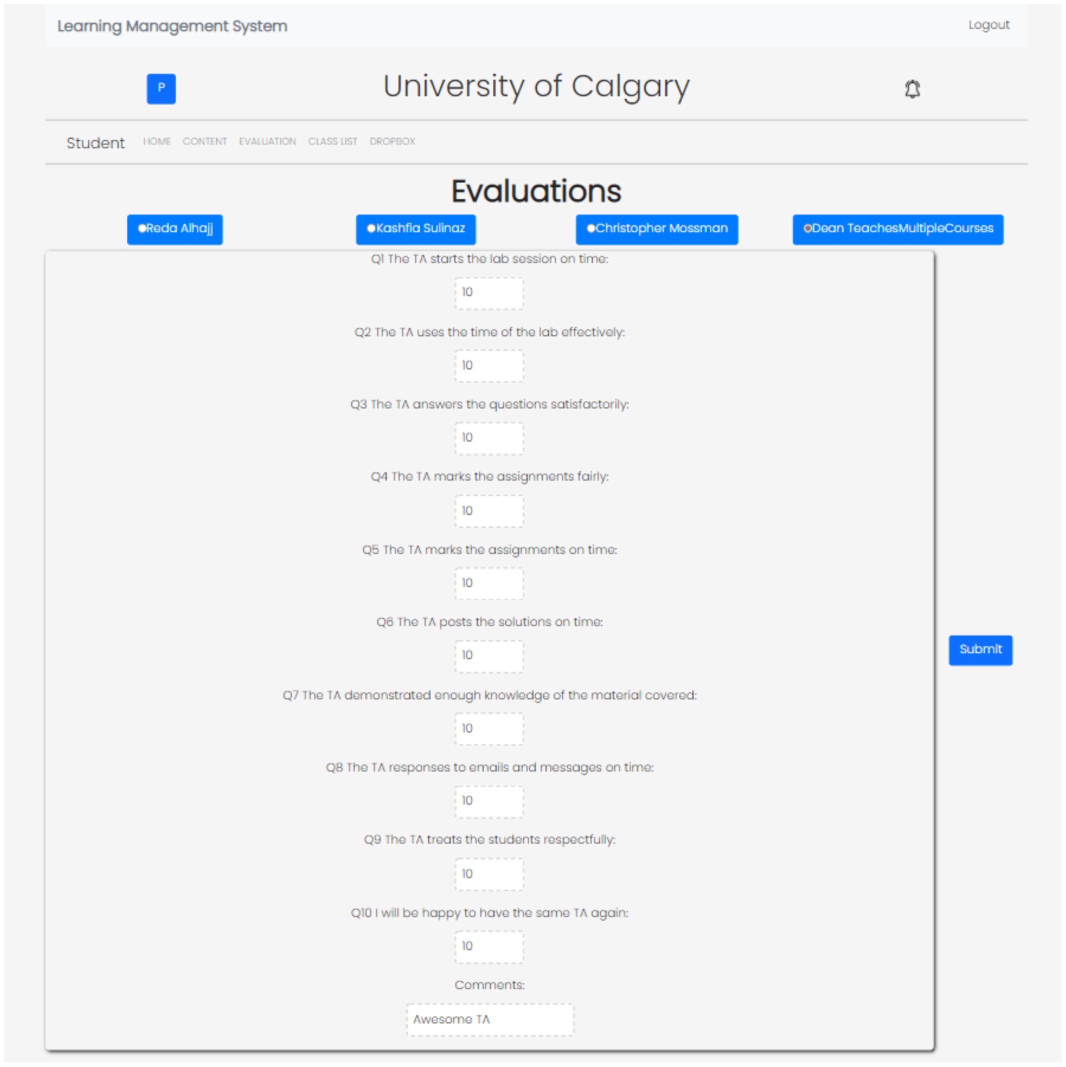
File for Upload

Upload Submit

File Name	Submission	Grade	Feedback
Homework 1	DOWNLOAD	70	
ENSF	DOWNLOAD	99	feedback

The student can submit an evaluation form for the professor and TA for each class after the Admin has pressed "send evaluations"

Evaluation page: allows students to evaluate teachers and TA's


Learning Management System

Logout

P

University of Calgary

Student HOME CONTENT EVALUATION CLASS LIST DROPBOX

Evaluations

●Reda Alhajj ●Kashfia Sulnaz ●Christopher Mossman ○Dean TeachesMultipleCourses

Q1 The TA starts the lab session on time:

10

Q2 The TA uses the time of the lab effectively:

10

Q3 The TA answers the questions satisfactorily:

10

Q4 The TA marks the assignments fairly:

10

Q5 The TA marks the assignments on time:

10

Q6 The TA posts the solutions on time:

10

Q7 The TA demonstrated enough knowledge of the material covered:

10

Q8 The TA responses to emails and messages on time:

10

Q9 The TA treats the students respectfully:

10

Q10 I will be happy to have the same TA again:

10

Comments:

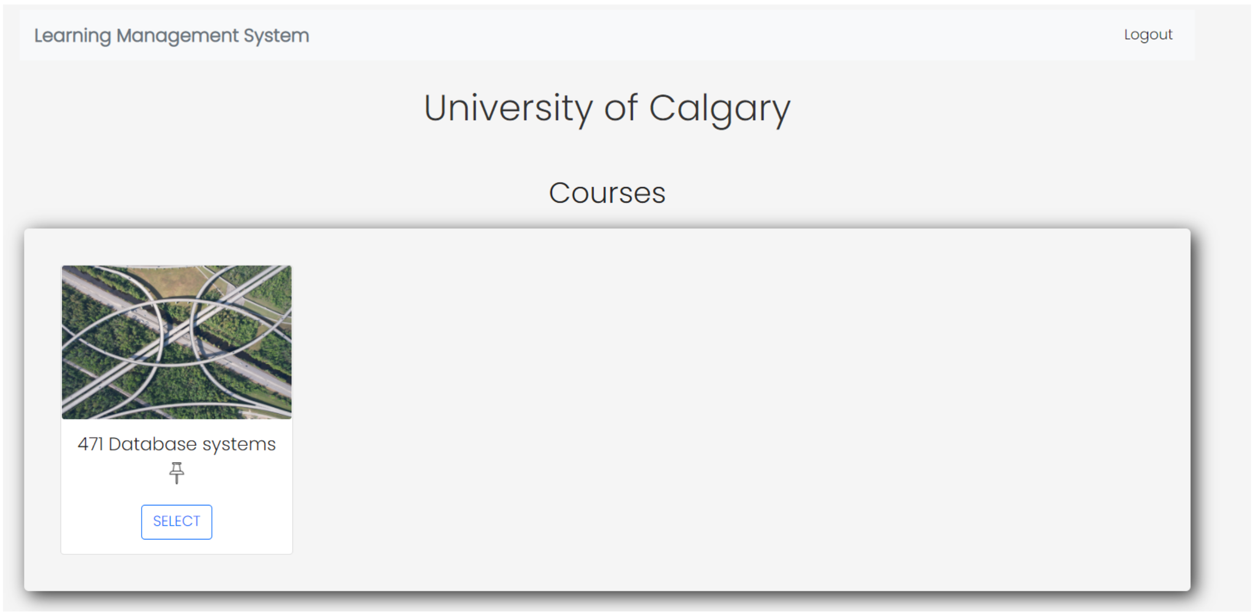
Awesome TA

Submit

Instructor

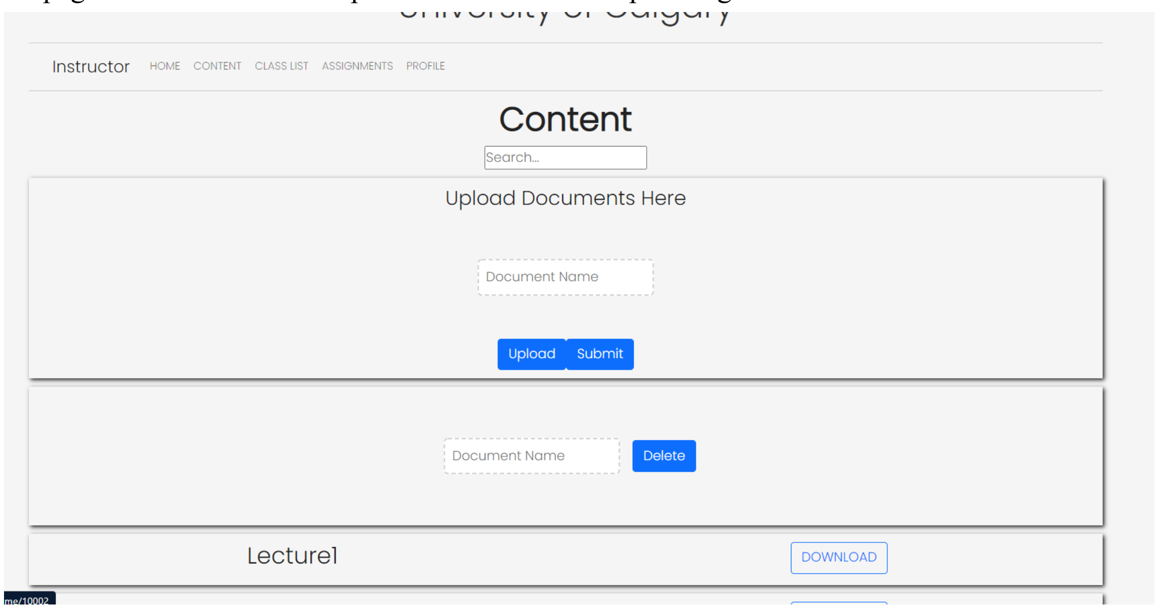
Instructor main login page. They will only be able to see the courses that they are teaching in.

Courses page: displays all the courses a teacher teaches



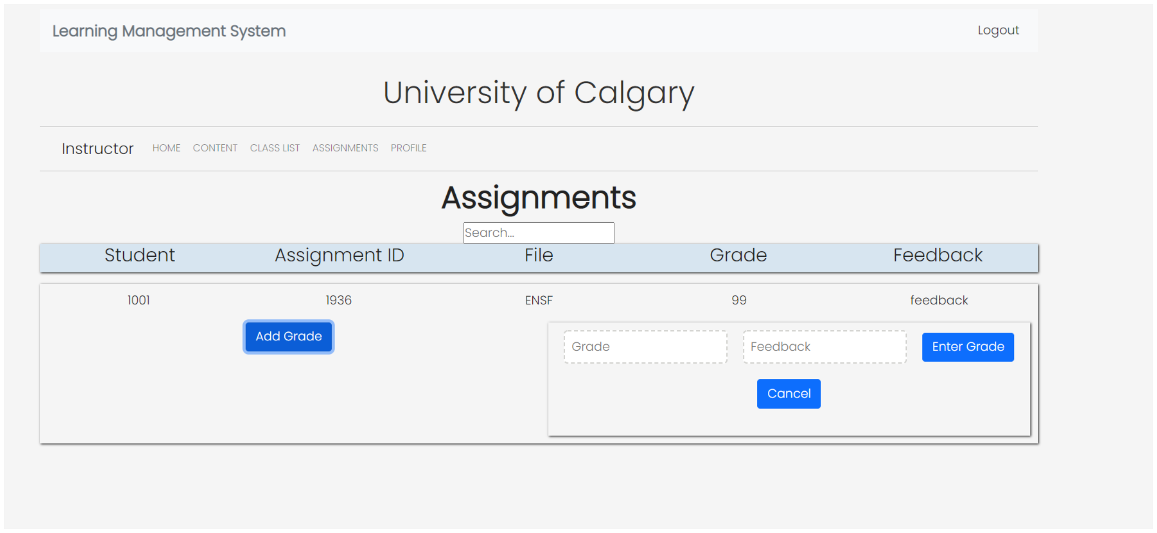
The instructor can upload files, set the name and the students in that course will be able to see the file uploaded.

Content page: shows the lectures uploaded and allow for uploading new lectures



The professor can see each student who submitted an assignment. The instructor can then give a mark and feedback.

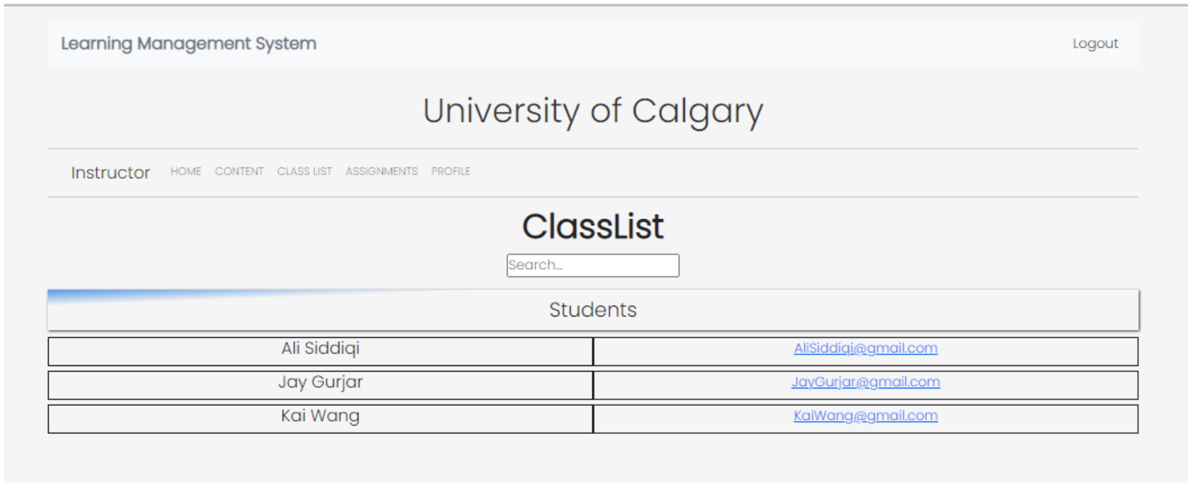
Assignments: shows the student assignment submissions and also allows for grading and adding feedback to the assignment



Student	Assignment ID	File	Grade	Feedback
1001	1936	ENSF	99	feedback

The professor can see the list of students within the course that the instructor is teaching.

ClassList: shows teachers and students in the course



Students	
Ali Siddiqi	AliSiddiqi@gmail.com
Jay Gurjar	JayGurjar@gmail.com
Kai Wang	KaiWang@gmail.com

File upload/download same as student user

Conclusion

The learning management system project allows a student to quickly communicate with an instructor and enhance their ability to learn. The admin can quickly make any changes to the student or instructors and see feedback from the students.