Carleton University

COMP 4905 - Honours Project

Personalized Learning Website

Name: Tingxun Dai

Supervisor: Dr. L.D.Nel

Date: April 25th, 2022

Table of Contents

Abstract

A web application built using the nodejs engine, express application framework and sqlite3 to build a database. Help professors and students operate their own courses and various details more easily and clearly.

Acknowledgments

I use bootstrap as the main design style of my application. I am not using any copyrighted material to make my application. The express application framework and many middlewares of node.js (like the passport) are used. All the rest of the code is done by myself.

Introduction

Learning sites similar to brightspace or culearn. Courses are created by professors, and students can manage their own courses. Students can categorize the courses they are taking and the courses they have completed. Make it easier and faster for students to find the knowledge they need. At the same time, students can discuss under the courseware, videos, assignments or quiz released by the professor. (I think this can be more targeted to help students reduce the time to find the information they need) If logging in as a professor or TA, can give some hints about the assignments and the quiz is related to which lectures or additional material like It's like the quiz of the previous semester. And you can choose delete the student's comment.

Motivation

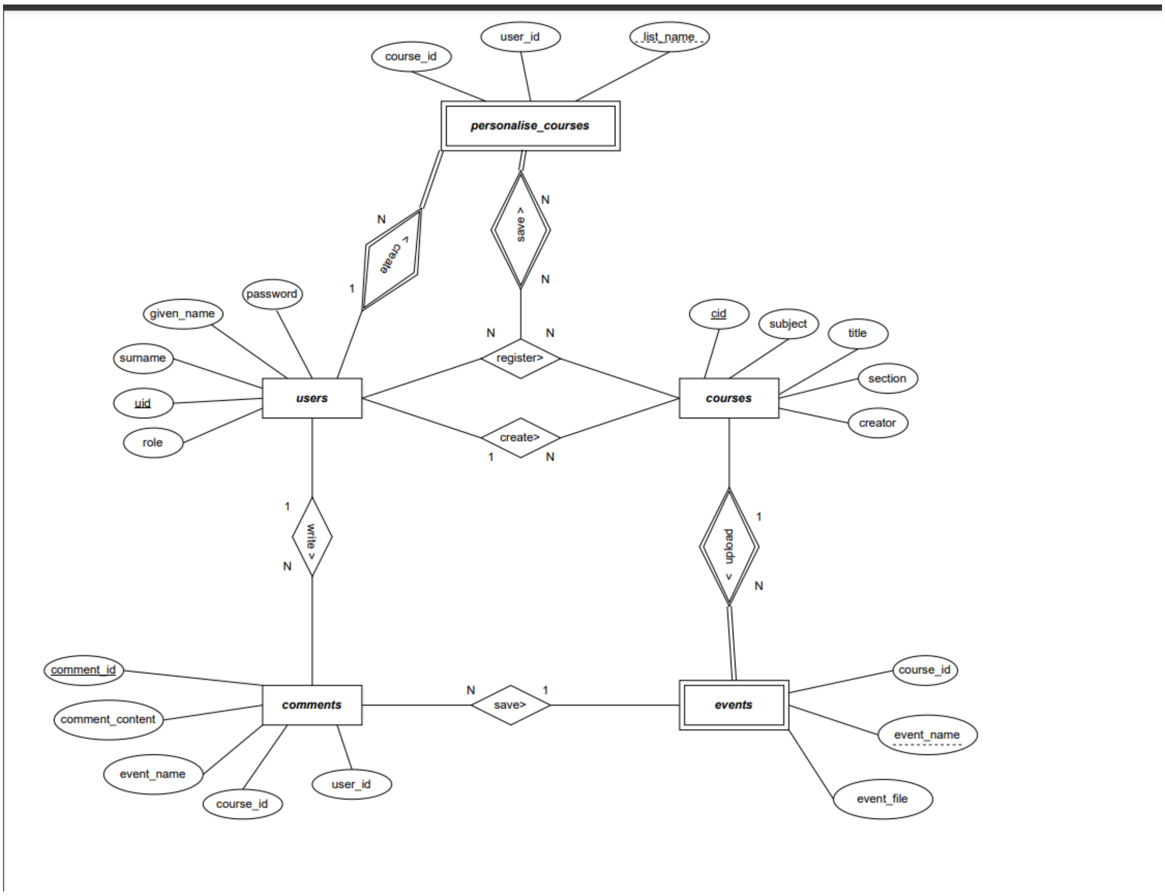
Whether it is culearn in the past or brightspace we use now, it is a website that professors use to publish various information to help students. So I wonder if I can make a website with higher operability for students themselves. In my case, I usually take 2 or 3 major courses in each semester, and the remaining credits are for elective courses. This leads to that when I need to look for the lectures of the computer courses I have taken in the past, I need to click on the list of each semester to find it, so I think that if I can improve the operability of the courses for students, it's easier to find what you need.

Methodology:

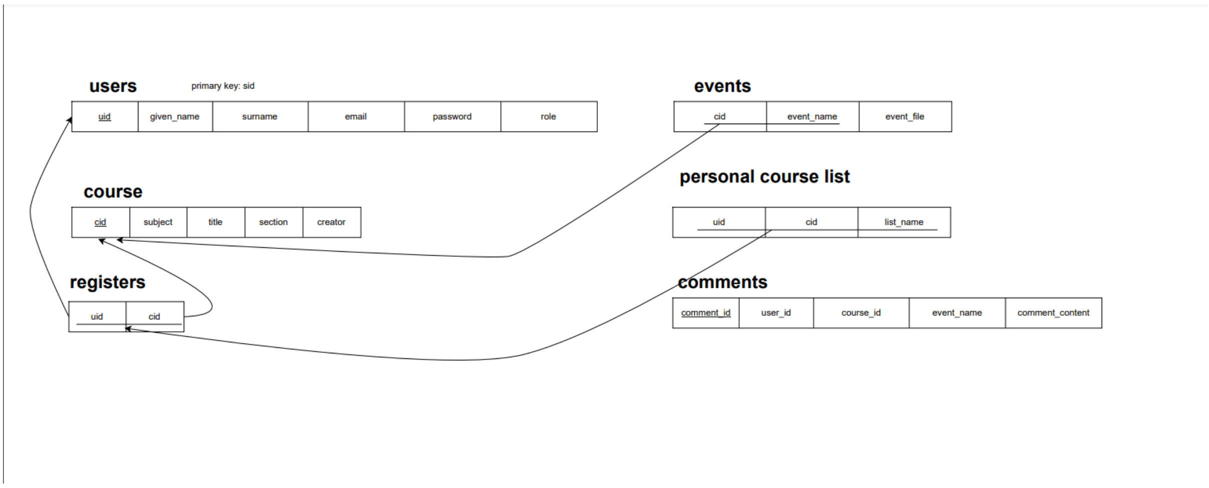
ER moduling -

According to the knowledge learned from COMP 3005 Database Management System, build ER-moduling and schema to help me understand the relationship between each table more clearly when building the database.

ER moduling

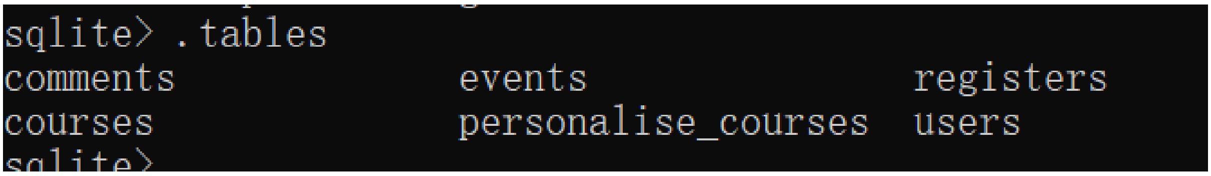


Schema:



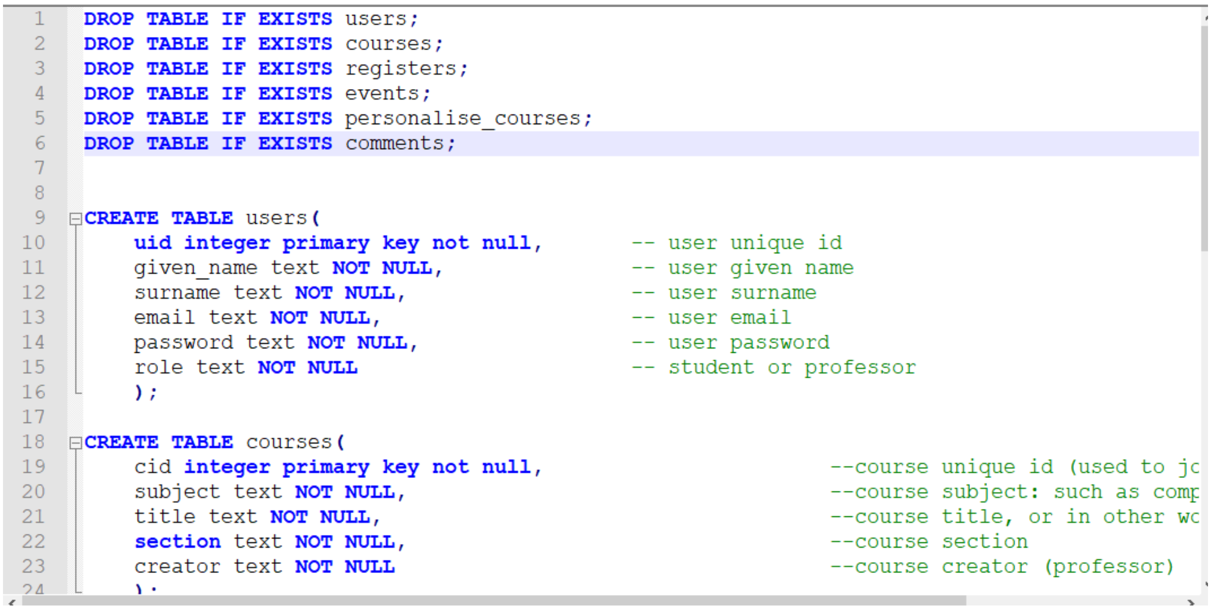
Database Setup -

Build a sqlite3 database based on what I learned about sqlite3 from COMP 3005 Database Management System. The tables in the database are:



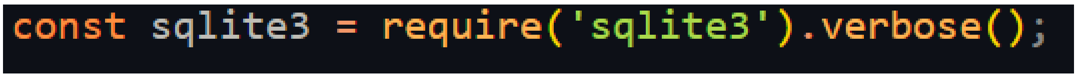
```
sqlite> .tables
comments              events                registers
courses               personalise_courses   users
sqlite>
```

Build the database_insert.sql file to update the database at any time. Here is one snapshot of the database_insert.sql



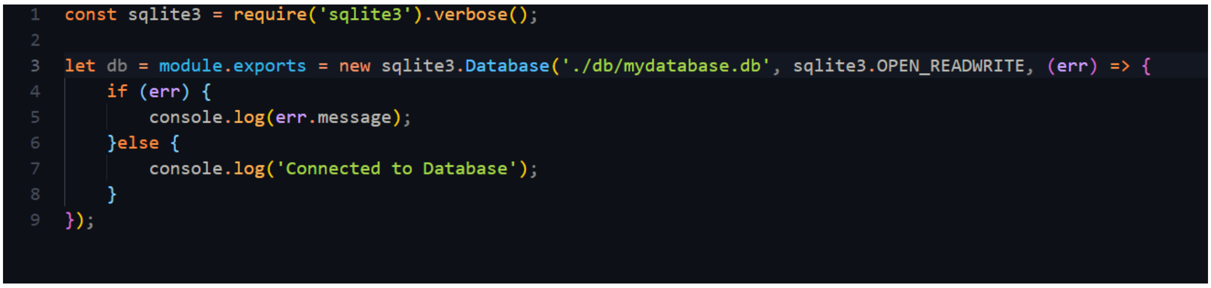
```
DROP TABLE IF EXISTS users;
DROP TABLE IF EXISTS courses;
DROP TABLE IF EXISTS registers;
DROP TABLE IF EXISTS events;
DROP TABLE IF EXISTS personalise_courses;
DROP TABLE IF EXISTS comments;


CREATE TABLE users(
    uid integer primary key not null,        -- user unique id
    given_name text NOT NULL,                -- user given name
    surname text NOT NULL,                   -- user surname
    email text NOT NULL,                     -- user email
    password text NOT NULL,                  -- user password
    role text NOT NULL                       -- student or professor
    );

CREATE TABLE courses(
    cid integer primary key not null,                --course unique id (used to jc
    subject text NOT NULL,                           --course subject: such as comp
    title text NOT NULL,                             --course title, or in other wc
    section text NOT NULL,                           --course section
    creator text NOT NULL                            --course creator (professor)
    );
```

Use nodejs middleware



```
const sqlite3 = require('sqlite3').verbose();
```

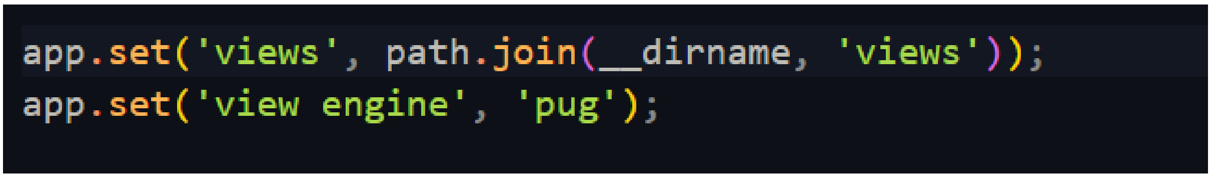
to help us connect web application and database



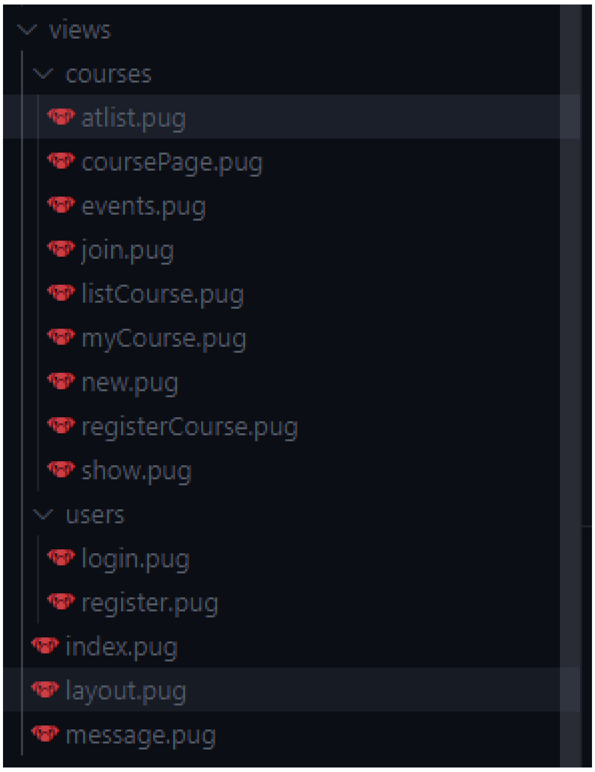
```
const sqlite3 = require('sqlite3').verbose();

let db = module.exports = new sqlite3.Database('./db/mydatabase.db', sqlite3.OPEN_READWRITE, (err) => {
    if (err) {
        console.log(err.message);
    }else {
        console.log('Connected to Database');
    }
});
```

Render Page -

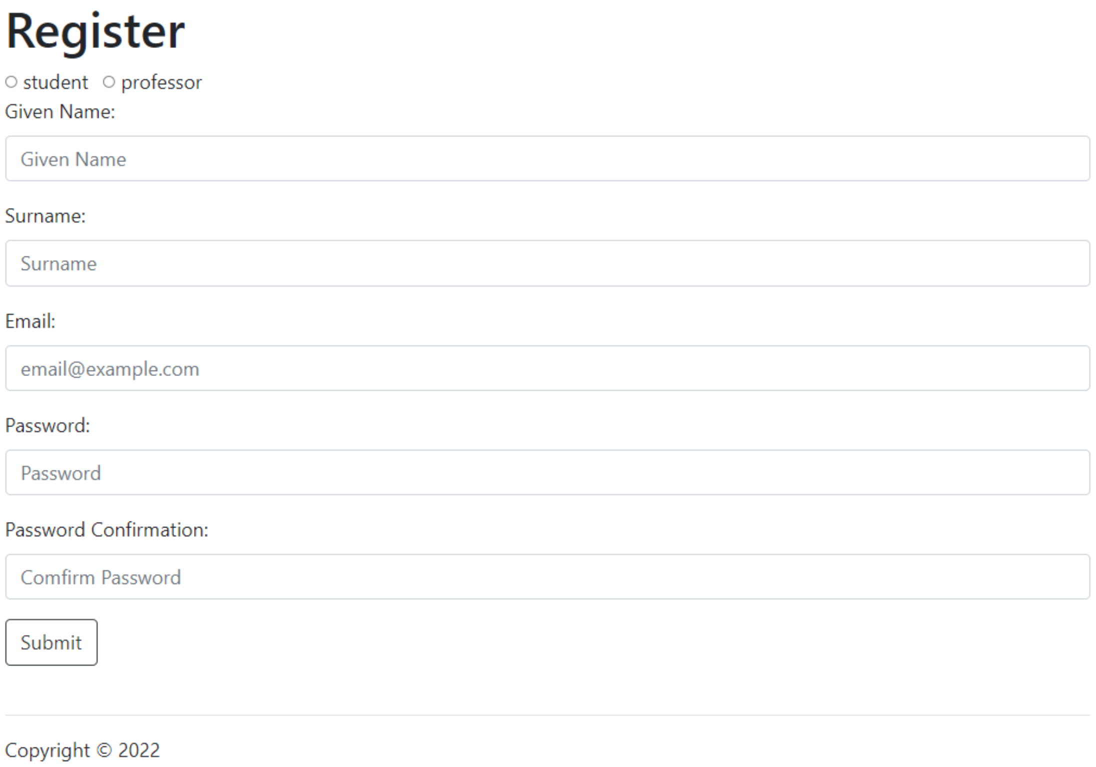
Use pug template to render html page



```
app.set('views', path.join(__dirname, 'views'));
app.set('view engine', 'pug');
```



```
views
  courses
    atlist.pug
    coursePage.pug
    events.pug
    join.pug
    listCourse.pug
    myCourse.pug
    new.pug
    registerCourse.pug
    show.pug
  users
    login.pug
    register.pug
  index.pug
  layout.pug
  message.pug
```



Register

○ student ○ professor

Given Name:

Given Name

Surname:

Surname

Email:

email@example.com

Password:

Password

Password Confirmation:

Comfirm Password

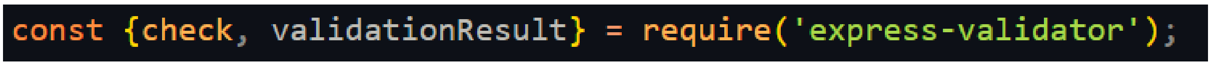
Submit

Copyright © 2022

User system -

In the user system I wrote, there are two important middlewares that need to be introduced:

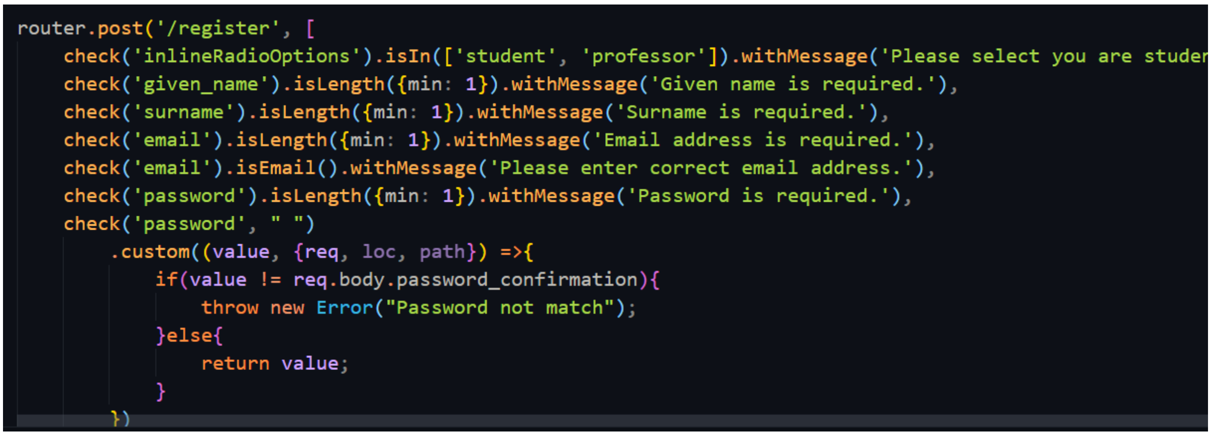
one is express-validator



```
const {check, validationResult} = require('express-validator');
```

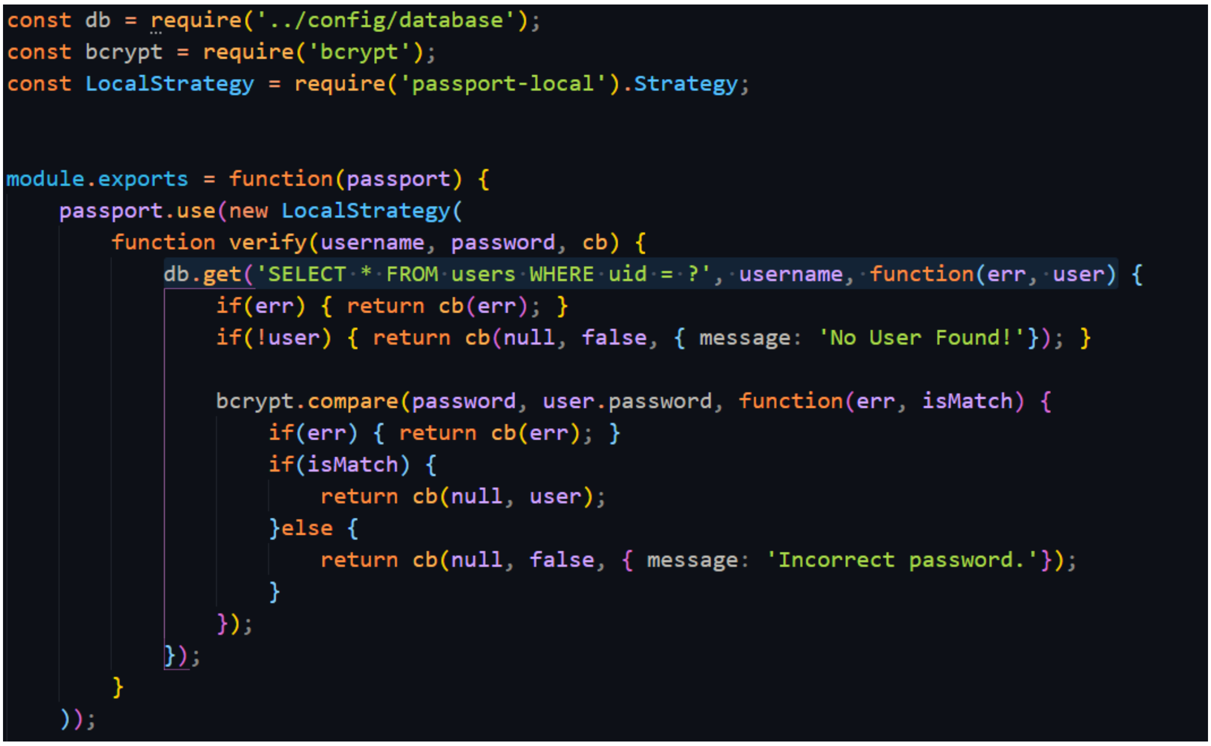
This middleware helps me to check that all input form users have filled in the information

correctly.（Here is check for register:)



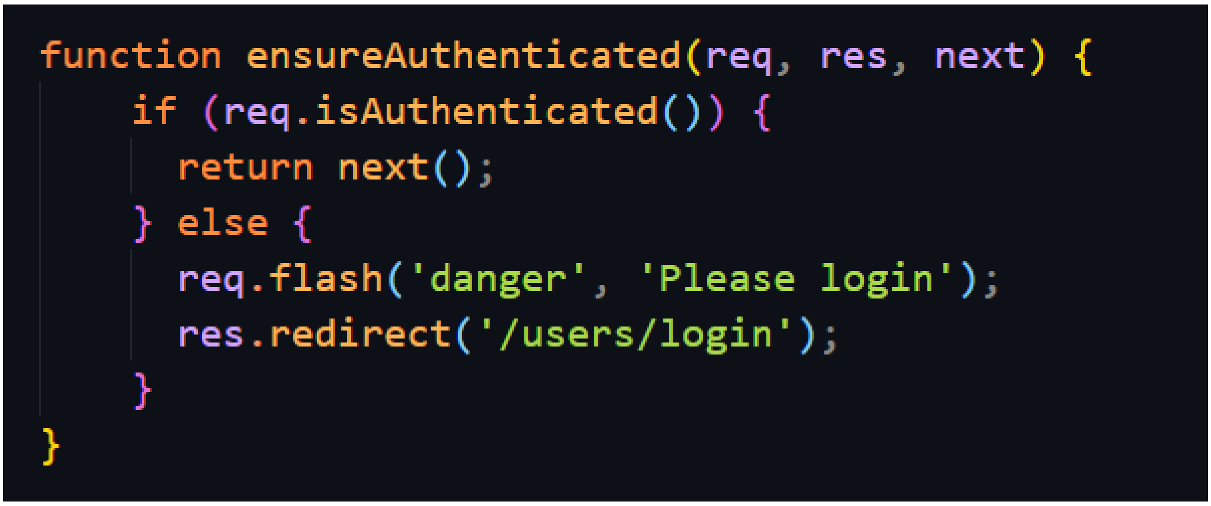
```
router.post('/register', [
    check('inlineRadioOptions').isIn(['student', 'professor']).withMessage('Please select you are studen
    check('given_name').isLength({min: 1}).withMessage('Given name is required.'),
    check('surname').isLength({min: 1}).withMessage('Surname is required.'),
    check('email').isLength({min: 1}).withMessage('Email address is required.'),
    check('email').isEmail().withMessage('Please enter correct email address.'),
    check('password').isLength({min: 1}).withMessage('Password is required.'),
    check('password', " ")
        .custom((value, {req, loc, path}) =>{
            if(value != req.body.password_confirmation){
                throw new Error("Password not match");
            }else{
                return value;
            }
        })
```

Another important middleware is passport:

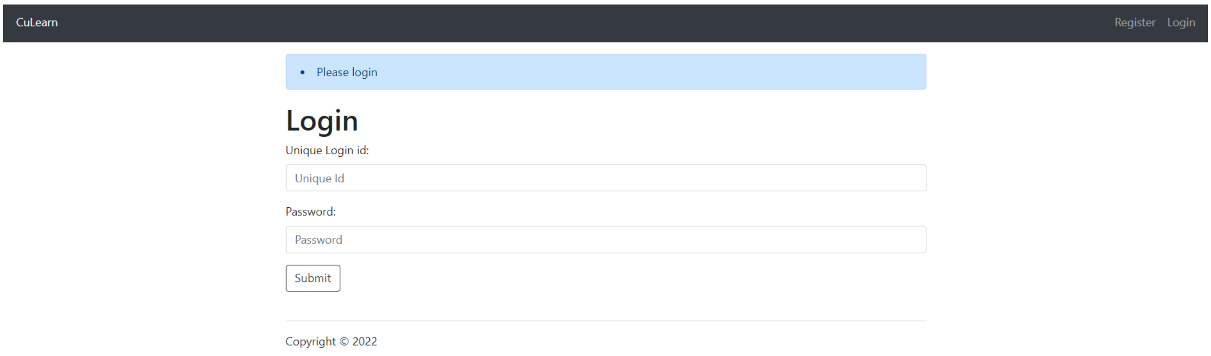


```
const db = require('../config/database');
const bcrypt = require('bcrypt');
const LocalStrategy = require('passport-local').Strategy;


module.exports = function(passport) {
    passport.use(new LocalStrategy(
        function verify(username, password, cb) {
            db.get('SELECT * FROM users WHERE uid = ?', username, function(err, user) {
                if(err) { return cb(err); }
                if(!user) { return cb(null, false, { message: 'No User Found!'}); }

                bcrypt.compare(password, user.password, function(err, isMatch) {
                    if(err) { return cb(err); }
                    if(isMatch) {
                        return cb(null, user);
                    }else {
                        return cb(null, false, { message: 'Incorrect password.'});
                    }
                });
            });
        }
    ));
```

its authentication function is very powerful.



```
function ensureAuthenticated(req, res, next) {
    if (req.isAuthenticated()) {
      return next();
    } else {
      req.flash('danger', 'Please login');
      res.redirect('/users/login');
    }
}
```



- Please login

Login

Unique Login id:

Unique Id

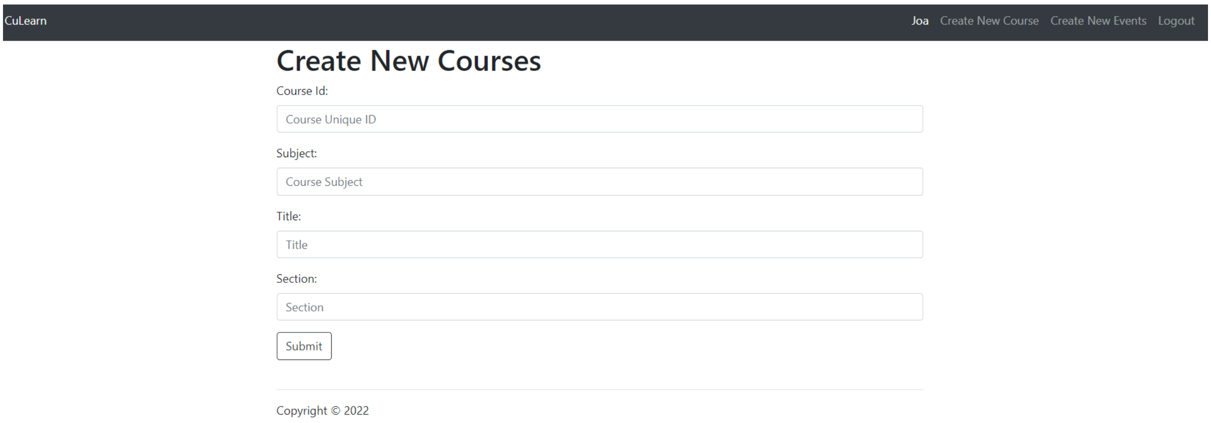
Password:

Password

Copyright © 2022

Course System -

When the user is logged in as a professor, he can create a new course through <Create New Course> in the navigation bar.



CuLearn Joa Create New Course Create New Events Logout

Create New Courses

Course Id:

Course Unique ID

Subject:

Course Subject

Title:

Title

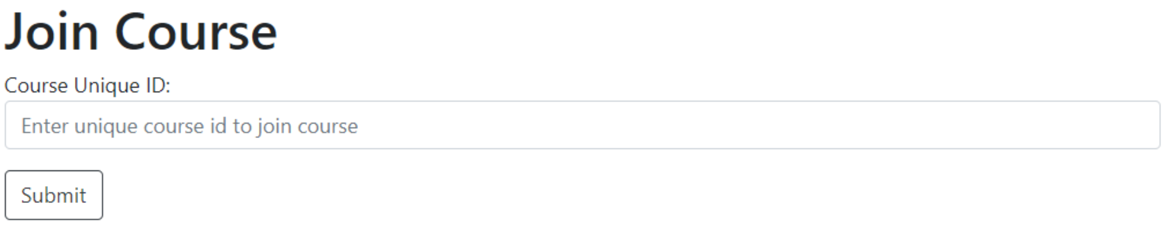
Section:

Section

Submit

Copyright © 2022

When the user is logged in as a student, he can join a new course through <Join New Course> in the navigation bar.



Join Course

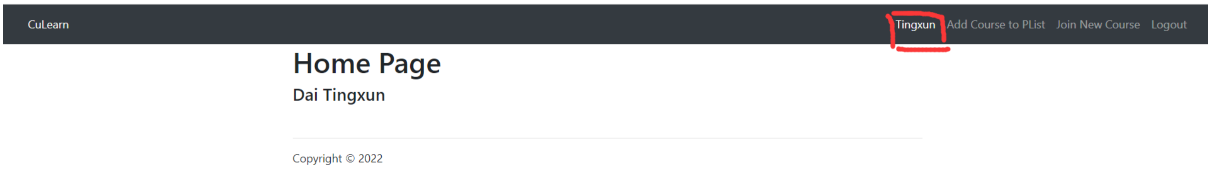
Course Unique ID:

Enter unique course id to join course

Submit

It is crucial to remember the course's unique id.

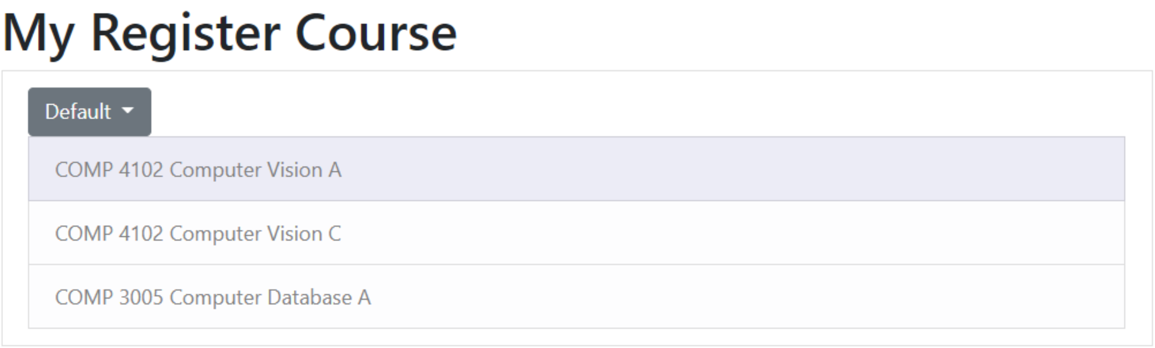
After logging in, click your given_name in the navigation bar to view the courses you have registered or created.



Home Page

Dai Tingxun

Copyright © 2022

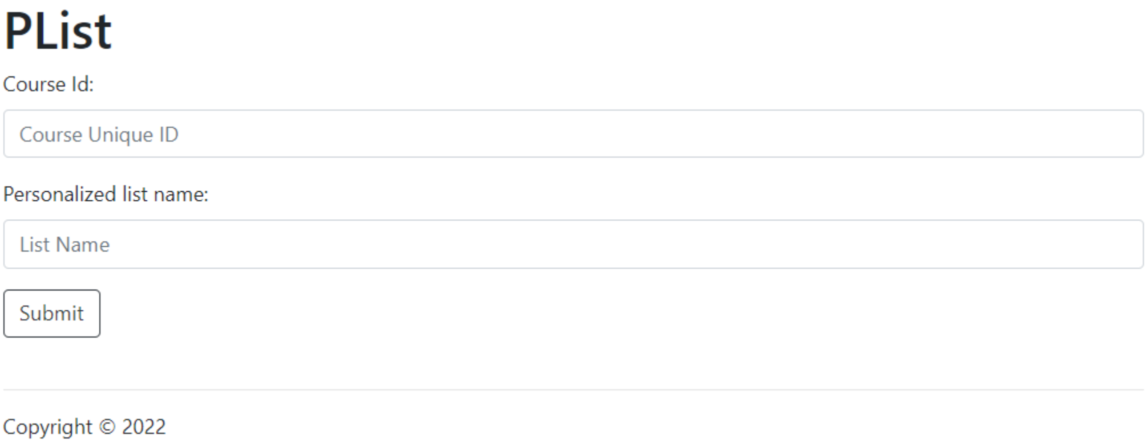

My Register Course

COMP 4102 Computer Vision A

COMP 4102 Computer Vision C

COMP 3005 Computer Database A

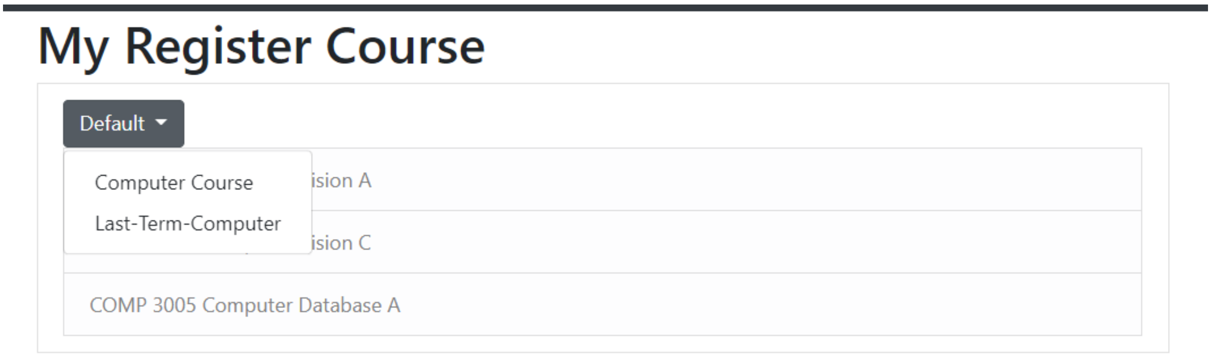
Students can add courses to the new personalized list by entering the course number of the registered courses through "Add Course to PList".


PList

Course Id: Course Unique ID

Personalized list name: List Name

Copyright © 2022



My Register Course

Default

Computer Course

Last-Term-Computer

ision A

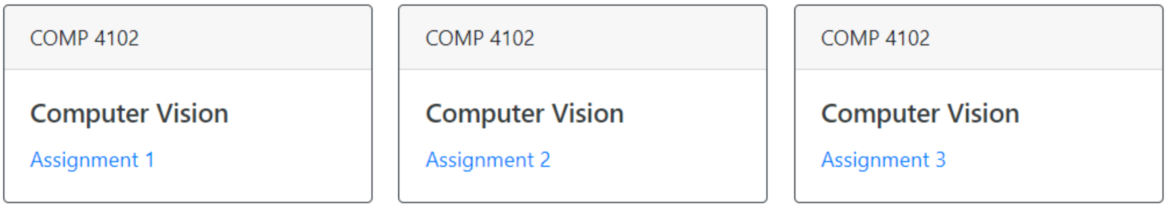
ision C

COMP 3005 Computer Database A

Click different PList names in the drop-down box to view the courses saved in different lists. If you want to view the default list, click the given_name of the user in the navigation bar again.

Event System -

The event system is used by professors to upload different assignments or lectures or quiz to different courses (file acceptance type is limited to pdf files)



COMP 4102

Computer Vision

Assignment 1

COMP 4102

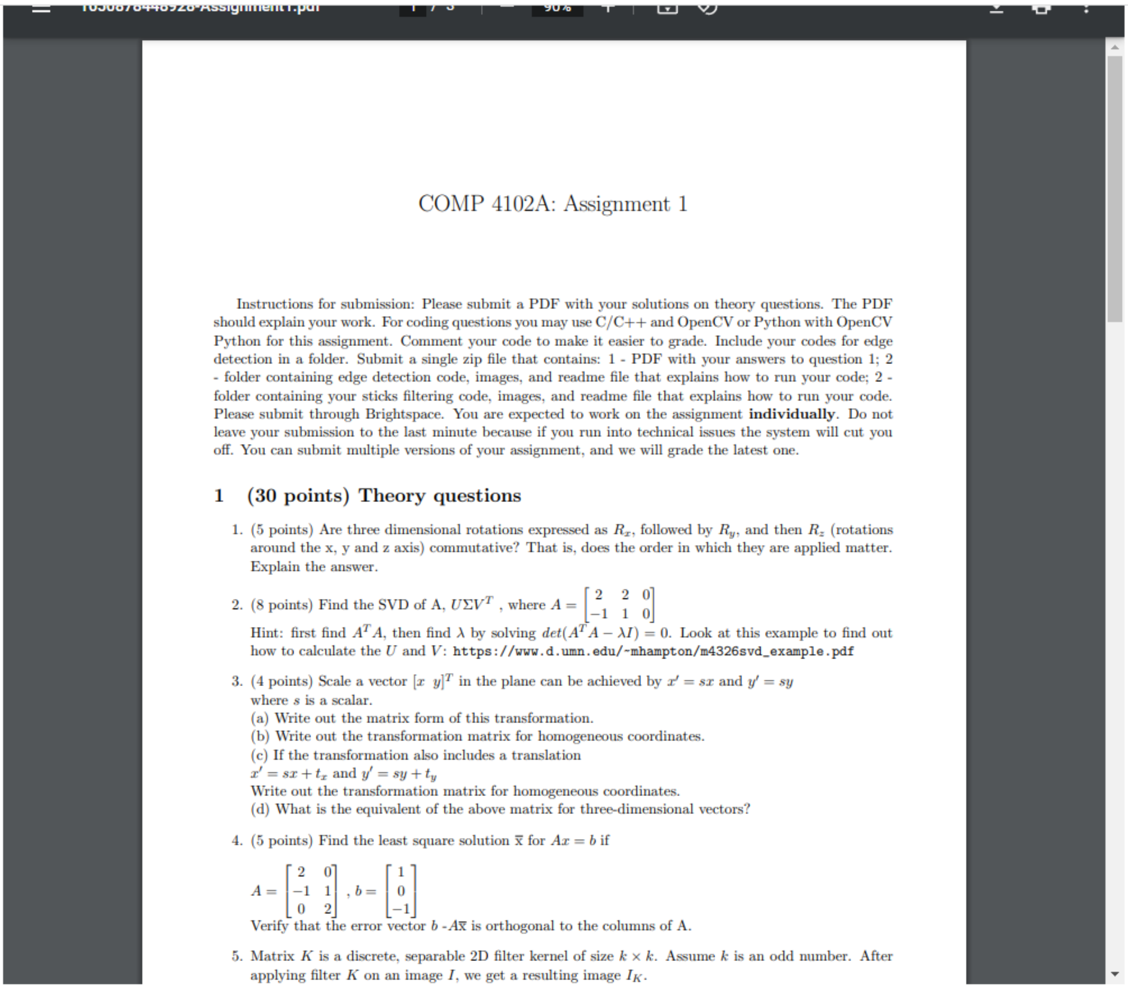
Computer Vision

Assignment 2

COMP 4102

Computer Vision

Assignment 3



COMP 4102A: Assignment 1

Instructions for submission: Please submit a PDF with your solutions on theory questions. The PDF should explain your work. For coding questions you may use C/C++ and OpenCV or Python with OpenCV Python for this assignment. Comment your code to make it easier to grade. Include your codes for edge detection in a folder. Submit a single zip file that contains: 1 - PDF with your answers to question 1; 2 - folder containing edge detection code, images, and readme file that explains how to run your code; 2 - folder containing your sticks filtering code, images, and readme file that explains how to run your code. Please submit through Brightspace. You are expected to work on the assignment **individually**. Do not leave your submission to the last minute because if you run into technical issues the system will cut you off. You can submit multiple versions of your assignment, and we will grade the latest one.

1 (30 points) Theory questions

1. (5 points) Are three dimensional rotations expressed as R_x, followed by R_y, and then R_z (rotations around the x, y and z axis) commutative? That is, does the order in which they are applied matter. Explain the answer.

2. (8 points) Find the SVD of A, $U\Sigma V^T$, where $A = \begin{bmatrix} 2 & 2 & 0 \\ -1 & 1 & 0 \end{bmatrix}$
   Hint: first find A^TA, then find λ by solving $det(A^TA - \lambda I) = 0$. Look at this example to find out how to calculate the U and V: `https://www.d.umn.edu/~mhampton/m4326svd_example.pdf`

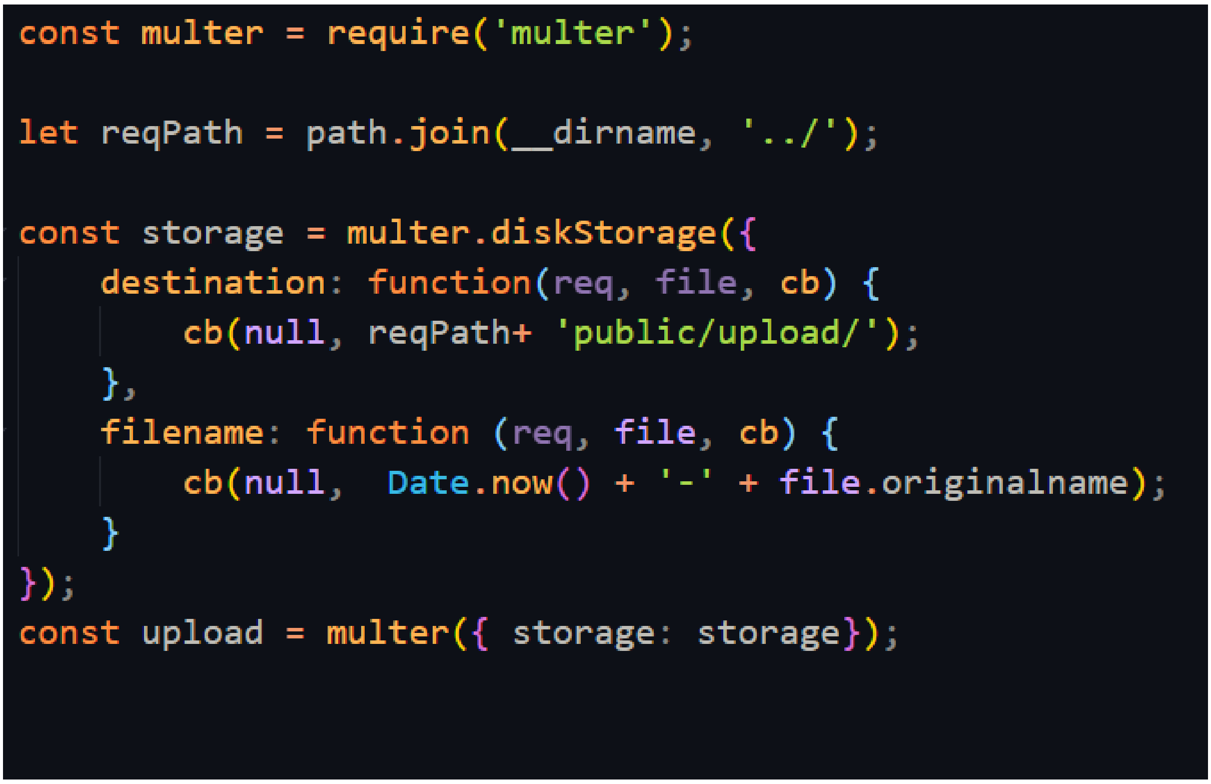
3. (4 points) Scale a vector $[x \;\; y]^T$ in the plane can be achieved by $x' = sx$ and $y' = sy$ where s is a scalar.
   (a) Write out the matrix form of this transformation.
   (b) Write out the transformation matrix for homogeneous coordinates.
   (c) If the transformation also includes a translation
   $x' = sx + t_x$ and $y' = sy + t_y$
   Write out the transformation matrix for homogeneous coordinates.
   (d) What is the equivalent of the above matrix for three-dimensional vectors?

4. (5 points) Find the least square solution $\overline{x}$ for $Ax = b$ if
   $A = \begin{bmatrix} 2 & 0 \\ -1 & 1 \\ 0 & 2 \end{bmatrix}$, $b = \begin{bmatrix} 1 \\ 0 \\ -1 \end{bmatrix}$
   Verify that the error vector b -$A\overline{x}$ is orthogonal to the columns of A.

5. Matrix K is a discrete, separable 2D filter kernel of size $k \times k$. Assume k is an odd number. After applying filter K on an image I, we get a resulting image I_K.

COMP 4102 Assignment 1

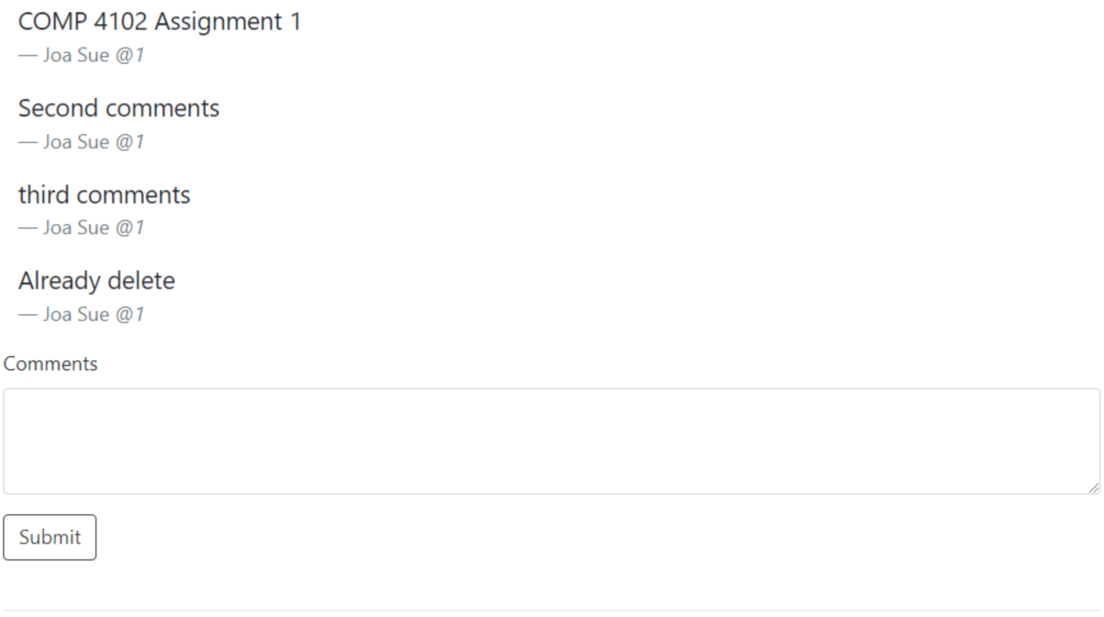
The realization of this function is largely due to the middleware of multer



```
const multer = require('multer');

let reqPath = path.join(__dirname, '../');

const storage = multer.diskStorage({
    destination: function(req, file, cb) {
        cb(null, reqPath+ 'public/upload/');
    },
    filename: function (req, file, cb) {
        cb(null,  Date.now() + '-' + file.originalname);
    }
});
const upload = multer({ storage: storage});
```

Comment System -

Both students and professors can post a comment in each event, and the student's perspective will see the speaker @speaker's unique id. In the professor's perspective, the format will be speaker #comment unique id. Professors can delete inappropriate comments by commenting unique id



COMP 4102 Assignment 1

— Joa Sue @*1*

Second comments

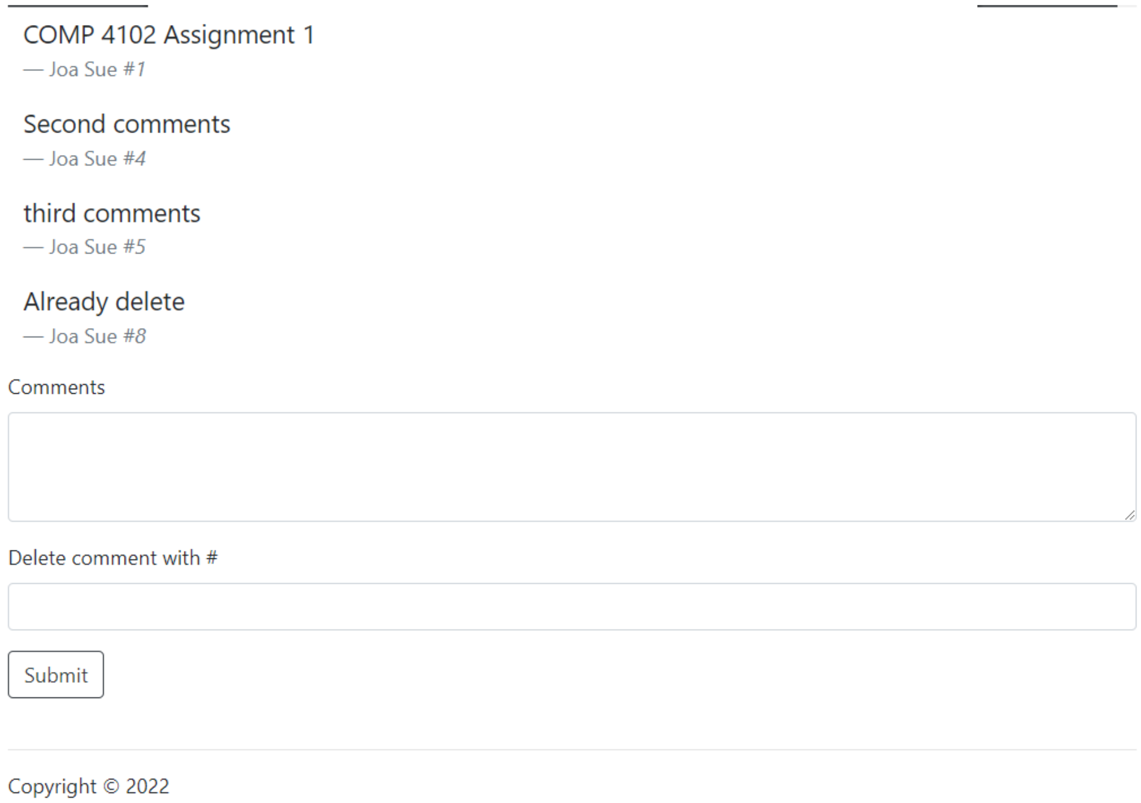
— Joa Sue @*1*

third comments

— Joa Sue @*1*

Already delete

— Joa Sue @*1*



COMP 4102 Assignment 1
— Joa Sue *#1*

Second comments
— Joa Sue *#4*

third comments
— Joa Sue *#5*

Already delete
— Joa Sue *#8*

Copyright © 2022

Deliverable functions

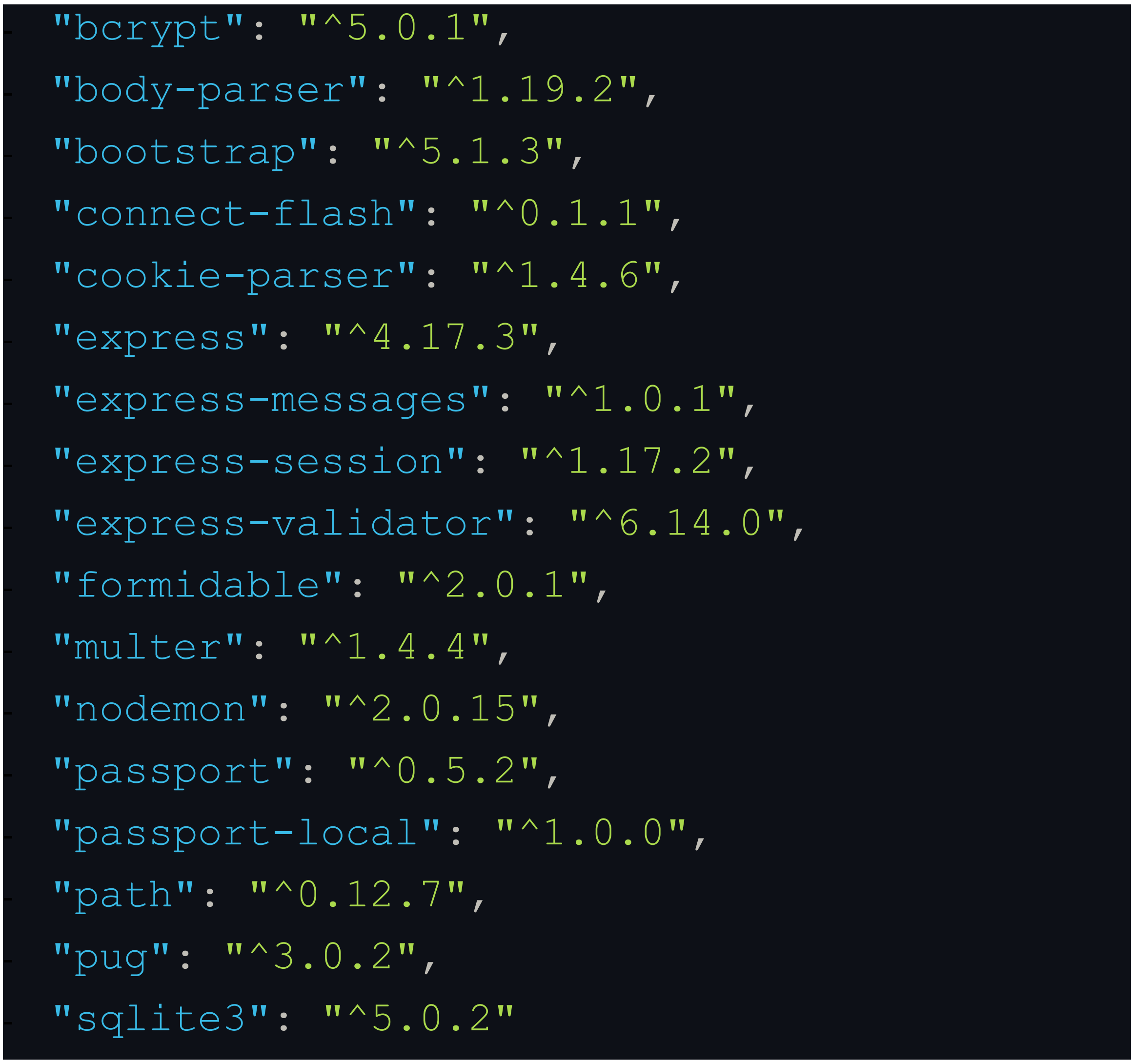
1. Website user login authentication function (Finished)
2. Professor: Create course function (Finished)
3. Student: Join specify course function (Finished)
4. Course function (Finished)
5. Event function include Assignment or Lecutre or quiz. (Finished)
6. Database ER modeling (Finished)
7. Database db and sql document (Finished)
8. Comment function (Finished)
9. Student personalise course list function. (Finished)

References

None

Appendix A: Setup and Execution Instructions

Dependencies



```
"bcrypt": "^5.0.1",
"body-parser": "^1.19.2",
"bootstrap": "^5.1.3",
"connect-flash": "^0.1.1",
"cookie-parser": "^1.4.6",
"express": "^4.17.3",
"express-messages": "^1.0.1",
"express-session": "^1.17.2",
"express-validator": "^6.14.0",
"formidable": "^2.0.1",
"multer": "^1.4.4",
"nodemon": "^2.0.15",
"passport": "^0.5.2",
"passport-local": "^1.0.0",
"path": "^0.12.7",
"pug": "^3.0.2",
"sqlite3": "^5.0.2"
```

Setup and running

1. Open the command terminal
2. Install NPM dependencies:
   - npm install
3. Run
   - nodemon app

4. Use web browser (such as chrome) go to localhost:3000