Coventry University

FACULTY OF ENGINEERING AND COMPUTING

Coursework Cover Sheet

Please ensure that you complete all relevant sections legibly
First Copy: *Attach top copy to the front of your assignment.*
Second Copy: *Keep safety as your receipt*

Module Code VTC303COM	**Student Card ID Number** 197101307	
Module Title Individual Project		
Deadline date	**Actual word Count**	**Tutor**
Coursework Title/Number **Developing a Full Stack Web Application**		

This assessment is all my own work and has not been copied in part or in whole from any other source, except for any clearly marked up quotations. It complies with the university regulations on plagiarism, which I have read and understood.

Please print in BLOCK CAPITALS

Surname……………………………………………

Other names…………………………………………

Signature……………………………………………

FEEDBACK ON MARKED WORK: *Lecturers will complete this section when work is marked.*
NB. All marks notified during the year are provisional until confirmed by the end of year Assessment Board

STRENGTHS

WEAKNESSES

ADVICE ON HOW WORK COULD BE IMPROVED AND FURTHER COMMENTS

If you require more feedback, please contact your tutor or see module web.
See assignment sheet for assessment criteria for this assignment.

MARKED AWARDED	
Less any late penalty	-
Adjusted mark if penalty	

Marker's Signature ……………………………………………………………… **Date** ………………………………………………………

Students are reminded that reference must be given for any previously published work used to gather information to help write assignments, including internet sources, but these sources should not be copied directly.

Second Marker Additional comments Signature ………………………………………… **Date** …………………………

Blackground

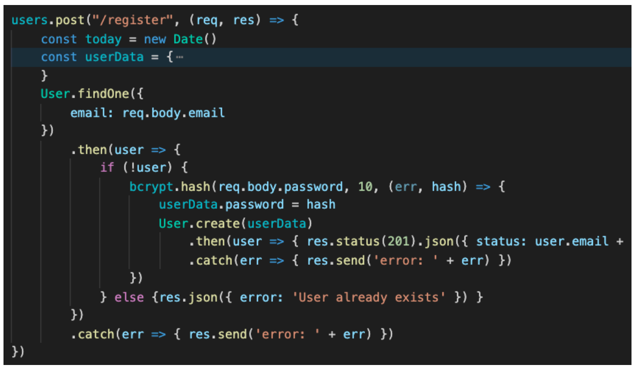
This is a school attendance system. The student attendance monitoring system provides monitoring, tracking, and reporting of student attendance in the classroom. The system will be a web-based system with a user-friendly GUI. In addition to the above functions, web-based systems also provide account management, analysis, and reporting functions for monitoring student attendance.
Basically, the system includes teachers, students and IT administrators.
• Teacher: Can attendance, modify attendance records, and generate reports.
• it administrator: full access to the system, including account management.

Backend Git remote repositories:
https://github.com/Pollyhsuhsu/Pollyhsuhsu-304CEM_Assignment_backend.git
Frontend Git remote repositories:
https://github.com/Pollyhsuhsu/Pollyhsuhsu-304CEM_Assignment_frontend.git

Task 1: Backend System (Public Web API)

I build a RESTful API using NodeJS, Express

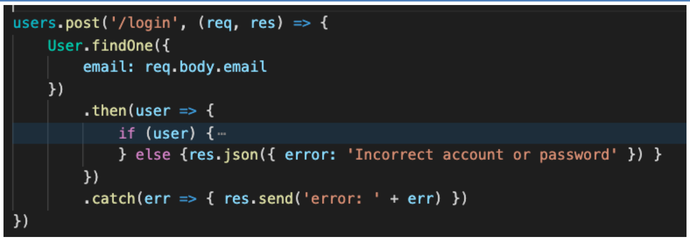
User Registrations



```
users.post("/register", (req, res) => {
    const today = new Date()
    const userData = {⋯
    }
    User.findOne({
        email: req.body.email
    })
        .then(user => {
            if (!user) {
                bcrypt.hash(req.body.password, 10, (err, hash) => {
                    userData.password = hash
                    User.create(userData)
                        .then(user => { res.status(201).json({ status: user.email +
                        .catch(err => { res.send('error: ' + err) })
                })
            } else {res.json({ error: 'User already exists' }) }
        })
        .catch(err => { res.send('error: ' + err) })
})
```

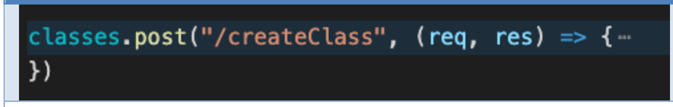
This API for user Registration. Also, the registered email cannot be repeated.

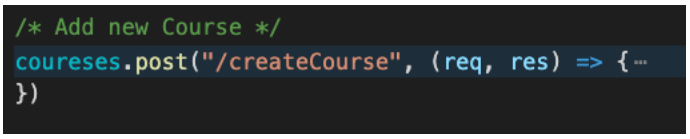
Login System and Authentication

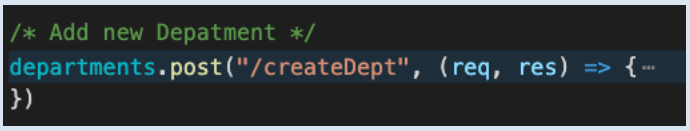


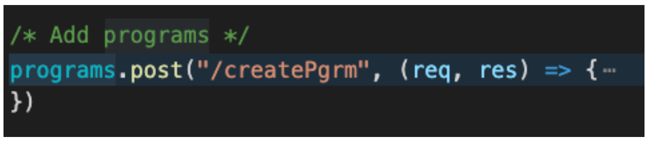
```
users.post('/login', (req, res) => {
    User.findOne({
        email: req.body.email
    })
        .then(user => {
            if (user) {⋯
            } else {res.json({ error: 'Incorrect account or password' }) }
        })
        .catch(err => { res.send('error: ' + err) })
})
```

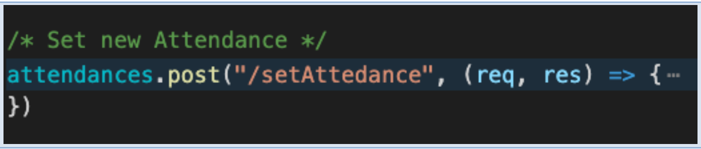
This API for user login. The system will find the corresponding email and then check the password.

At least two CREATE methods (Post)	
 `classes.post("/createClass", (req, res) => {⋯` `})`	This API for create a Class.

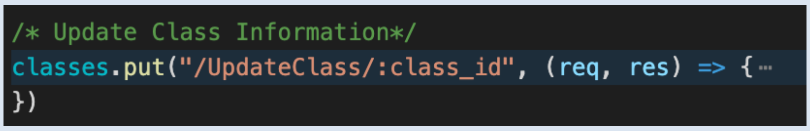
 `/* Add new Course */`
`coureses.post("/createCourse", (req, res) => {⋯`
`})`
	This API for create a Course.

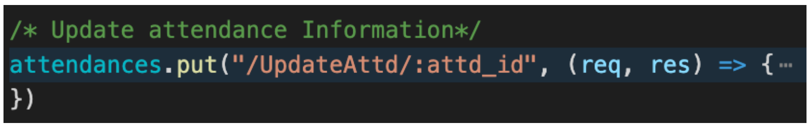
 `/* Add new Depatment */`
`departments.post("/createDept", (req, res) => {⋯`
`})`
	This API for create a Department.

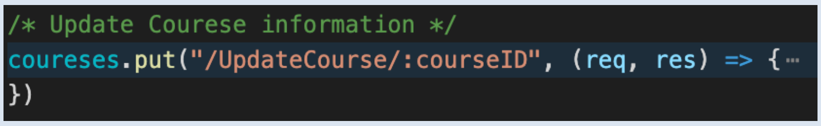
 `/* Add programs */`
`programs.post("/createPgrm", (req, res) => {⋯`
`})`
	This API for create a Program.

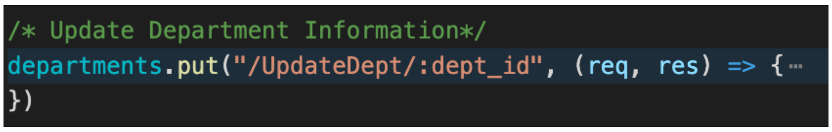
 `/* Set new Attendance */`
`attendances.post("/setAttedance", (req, res) => {⋯`
`})`
	This API for create a Attendance.

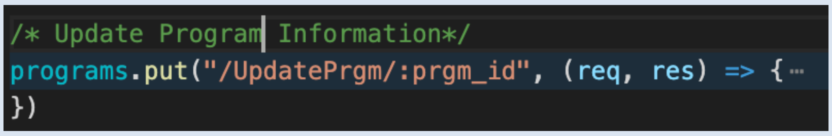
At least two UPDATE(Put)	

 `/* Update Class Information*/`
`classes.put("/UpdateClass/:class_id", (req, res) => {⋯`
`})`
	This API for update a Class.

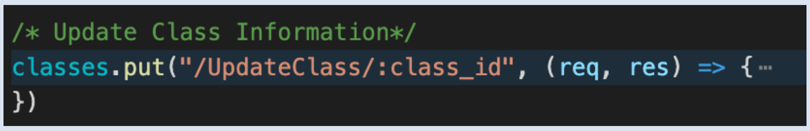
 `/* Update attendance Information*/`
`attendances.put("/UpdateAttd/:attd_id", (req, res) => {⋯`
`})`
	This API for update a attendance.

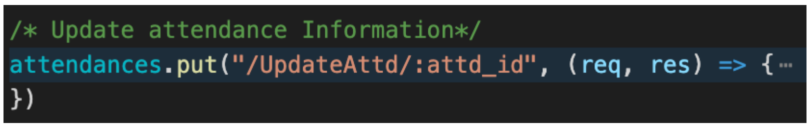
 `/* Update Courese information */`
`coureses.put("/UpdateCourse/:courseID", (req, res) => {⋯`
`})`
	This API for update a course.

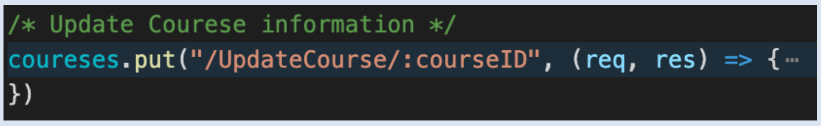
 `/* Update Department Information*/`
`departments.put("/UpdateDept/:dept_id", (req, res) => {⋯`
`})`
	This API for update a Department.

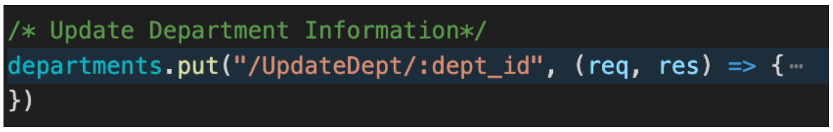
 `/* Update Program Information*/`
`programs.put("/UpdatePrgm/:prgm_id", (req, res) => {⋯`
`})`
	This API for update a program.

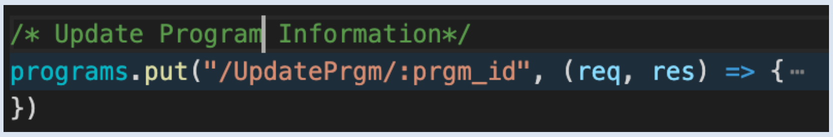
At least two UPDATE(Put)	

 `/* Update Class Information*/`
`classes.put("/UpdateClass/:class_id", (req, res) => {⋯`
`})`
	This API for update a Class.

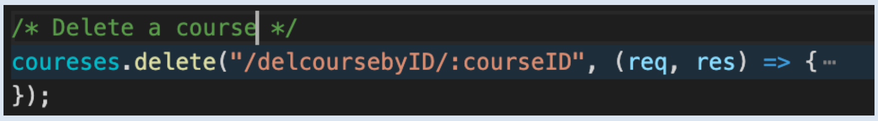
 `/* Update attendance Information*/`
`attendances.put("/UpdateAttd/:attd_id", (req, res) => {⋯`
`})`
	This API for update a attendance.

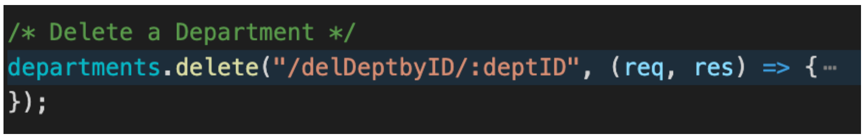
 `/* Update Courese information */`
`coureses.put("/UpdateCourse/:courseID", (req, res) => {⋯`
`})`
	This API for update a course.

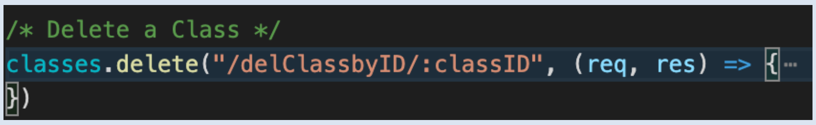
 `/* Update Department Information*/`
`departments.put("/UpdateDept/:dept_id", (req, res) => {⋯`
`})`
	This API for update a Department.

 `/* Update Program Information*/`
`programs.put("/UpdatePrgm/:prgm_id", (req, res) => {⋯`
`})`
	This API for update a program.

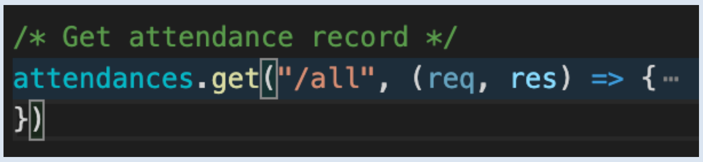
At least two DELETE methods(Delete)	

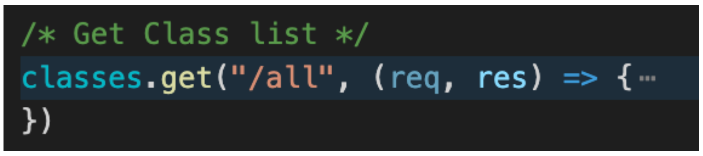
 `/* Delete a course */`
`coureses.delete("/delcoursebyID/:courseID", (req, res) => {⋯`
`});`
	This API for delete a course.

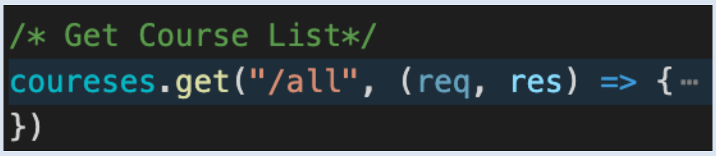
 `/* Delete a Department */`
`departments.delete("/delDeptbyID/:deptID", (req, res) => {⋯`
`});`
	This API for delete a department.

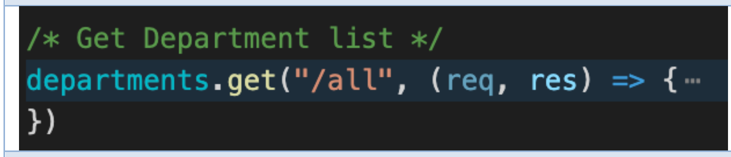
 `/* Delete a Class */`
`classes.delete("/delClassbyID/:classID", (req, res) => {⋯`
`})`
	This API for delete a class.

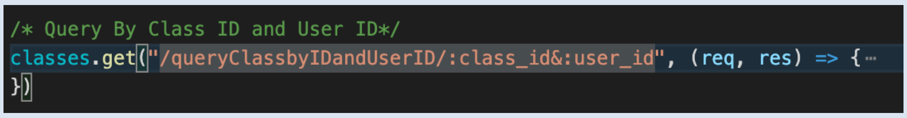
At least two DELETE methods(Delete)	

 `/* Get attendance record */`
`attendances.get("/all", (req, res) => {⋯`
`})`
	This API for get all attendances.

 `/* Get Class list */`
`classes.get("/all", (req, res) => {⋯`
`})`
	This API for get all classes.

 `/* Get Course List*/`
`coureses.get("/all", (req, res) => {⋯`
`})`
	This API for get all courses.

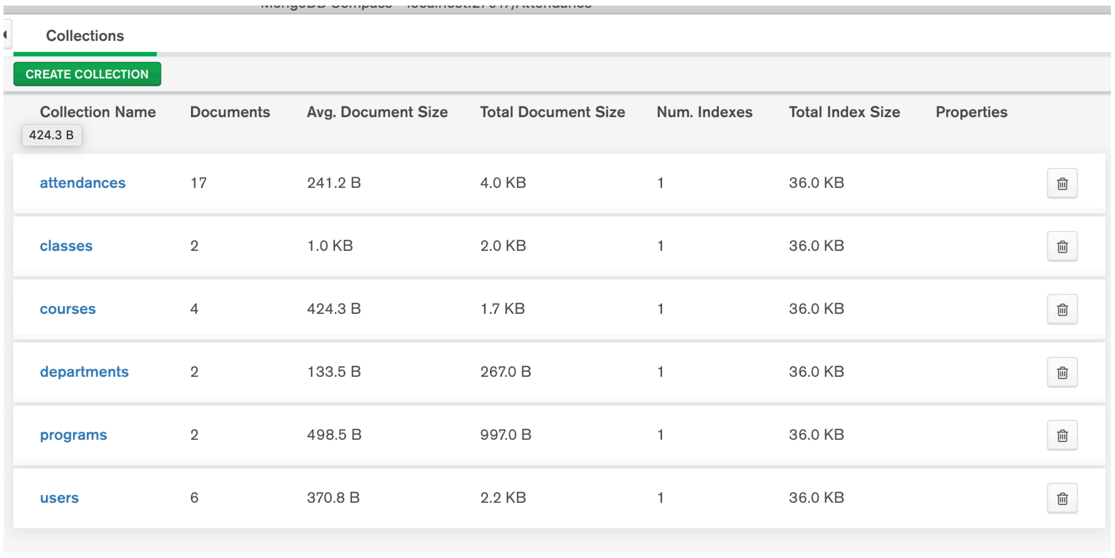
 `/* Get Department list */`
`departments.get("/all", (req, res) => {⋯`
`})`
	This API for get all Department.

 `/* Query By Class ID and User ID*/`
`classes.get("/queryClassbyIDandUserID/:class_id&:user_id", (req, res) => {⋯`
`})`
	This API for get specific user list. (Here is to extract the designated teacher, he has taught

	courses.)

Task 2: Persistent Storage

For the persistent storage I used a noSQL database - Mongo DB



Collections

CREATE COLLECTION

Collection Name	Documents	Avg. Document Size	Total Document Size	Num. Indexes	Total Index Size	Properties
attendances	17	241.2 B	4.0 KB	1	36.0 KB	
classes	2	1.0 KB	2.0 KB	1	36.0 KB	
courses	4	424.3 B	1.7 KB	1	36.0 KB	
departments	2	133.5 B	267.0 B	1	36.0 KB	
programs	2	498.5 B	997.0 B	1	36.0 KB	
users	6	370.8 B	2.2 KB	1	36.0 KB	

For better understanding, I set up some simulation data here

Table	Simulation data
User	“Polly”
Department	“Information Technology”
Course	“ITP4909-Object Oriented Technology”
Program	“IT114105 Higher Diploma in Software Engineering”
Class	“IT114105/1A”
Attendance	…

The table of **User** is used to store user information, such as email, password, date of birth, phone number, etc.

The table of **Department** is used to store the department details, such as ‘IT’, ‘Applied Science’ etc.

The table of **Course** is used to store the course details, such as course name, course code, etc.

The table of **Program** is used to store the program details, such as program name, program code,course etc.

The table of **Class** is used to store the class details, such as department belong to, class A, Class B etc.

The table of **Attendance** is used to store the student attendances.