PROJECT NAME: COURSE SELECTION SYSTEM

COURSE NAME: SOFTWARE ENGINEERING FALL 2020

Student Name: MD MAHFUZUR RAHMAN 毒液

Student ID: 20183290375

Instructor Name: 谢成

TABLE OF CONTENTS

Project Overview

This course selection system which allows students to select courses and giving student access to course offerings via online as well as the ability to complete various administrative functions allowing for better management of curriculum decisions in the context of academic objectives. The objective of these systems is to make this process more convenient and easier to achieve which has been met with varying levels of success. This Course Selection Portal will be operated by three users, the administrator, students and teachers.

Key Features

Admin

Admin have almost all access like create, delete and modify.

Teacher

Teacher can create, delete and modify the course, also can give the scores to the students and see his schedule.

Student

Students can select, withdraw course, can his scores and schedule

Common

In this system have some common feature for all user-like Registration, Login, Logout, view personal info, change pass, retrieve pass, 404 page and etc.

Technology I used for implementation

Frontend

- Vue.js
- HTML
- CSS

Backend

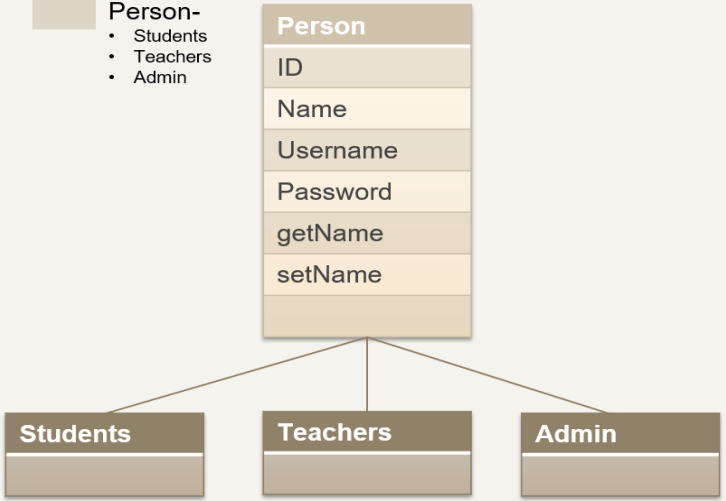
- Node.js (express js)
- For accessing data use MySQL

IDE

- WebStorm
- MySQL workbench
- **use mysql server 5x

Class Design

Person-
- Students
- Teachers
- Admin



Course

Course
C_ID
Name
Credit
C_Hours
……
ChangeCredit()
ChangeHours()
…….

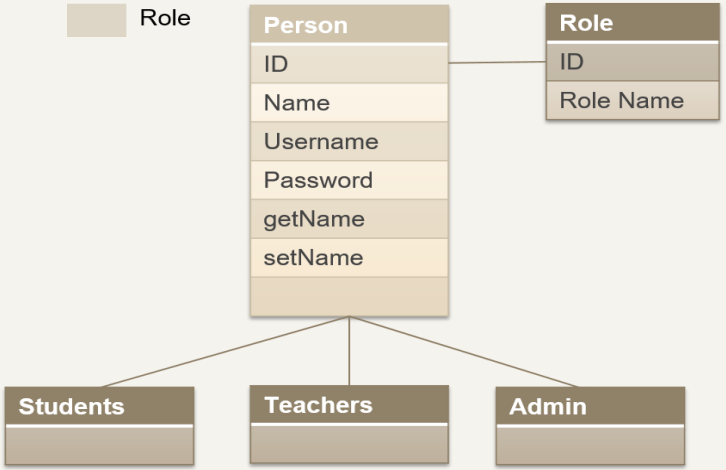
Arrangement

Course
C_ID
Semester
Credit
Time
……
Arrangeing()
ChangeCourse()
…….

Class(course)

Course
List<students>
Course Information
……
Select Course()
Withdraw Course()
…….

Role



Database Table

Score	
S_ID	int(11)
C_ID	int(11)
Score	int(3)

Course	
C_ID	int(11)
P_ID	int(11)
C_Name	varchar(20)
C_Hours	int(3)
C_credit	int(2)
SelectCount	int(5)
MaxCournt	int(5)

Class	
Classroom	varchar(20)
CA_ID	int(11)
P_ID	int(11)

Students	
S.ID	int(11)
S.Name	varchar(255)
S.Phone	int(11)
S.Username	varchar(20)
Password	varchar(50)

Teachers	
T.ID	int(11)
T.Name	varchar(255)
T.Phone	int(11)
T.Username	varchar(20)
T.Password	varchar(50)

Role		
R_ID	(FK)	int(11)
R_Name		varchar(20)

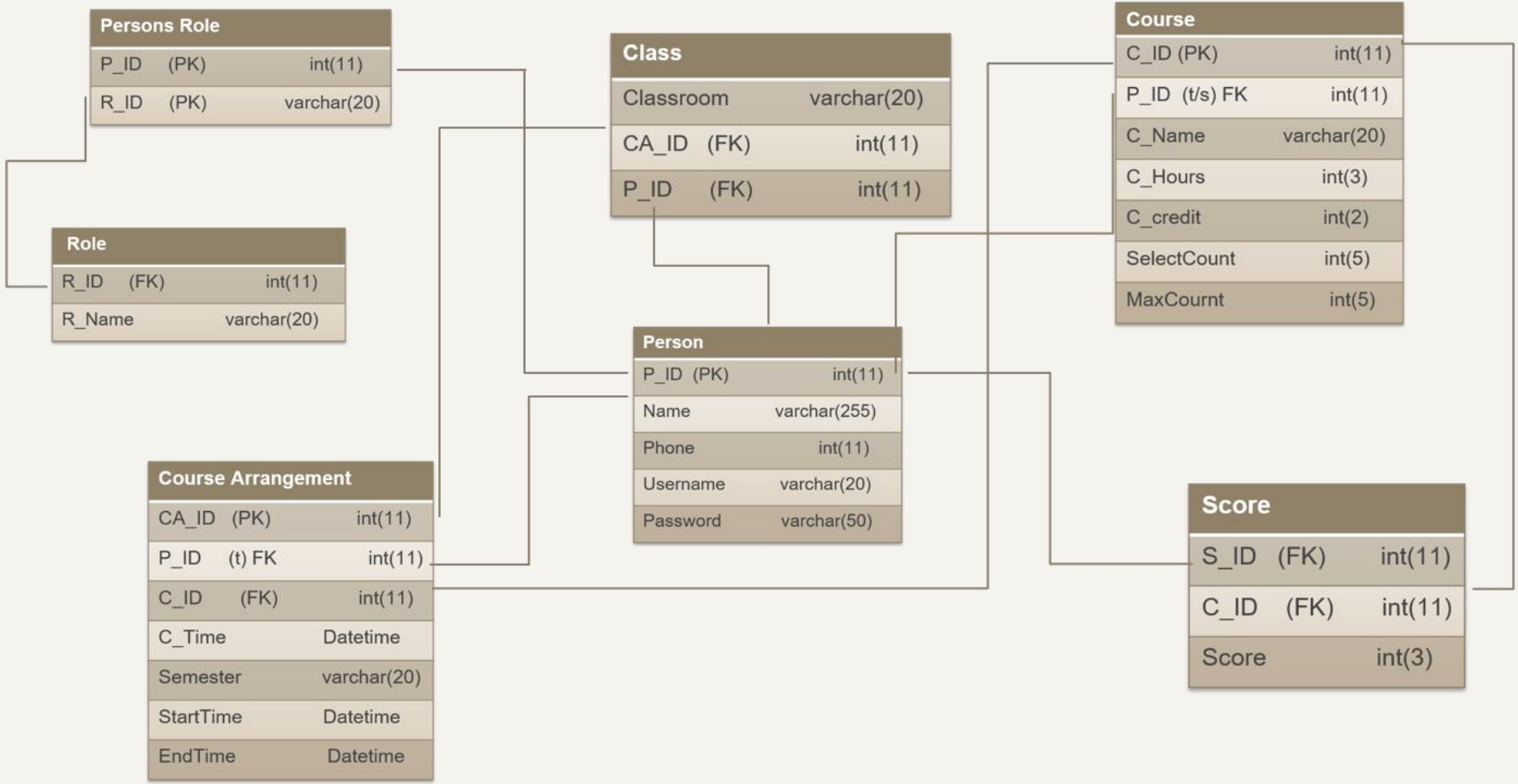
Course Arrangement	
CA_ID	int(11)
P_ID	int(11)
C_ID	int(11)
C_Time	Datetime
Semester	varchar(20)
StartTime	Datetime
EndTime	Datetime

Admin	
A.ID	int(11)
A.Name	varchar(255)
A.Phone	int(11)
A.Username	varchar(20)
A.Password	varchar(50)

Person	
ID	int(11)
Name	varchar(255)
Phone	int(11)
Username	varchar(20)
Password	varchar(50)

Persons Role		
P_ID	(PK)	int(11)
R_ID	(PK)	varchar(20)

Database Diagram



Core code and screenshot

Connect with database

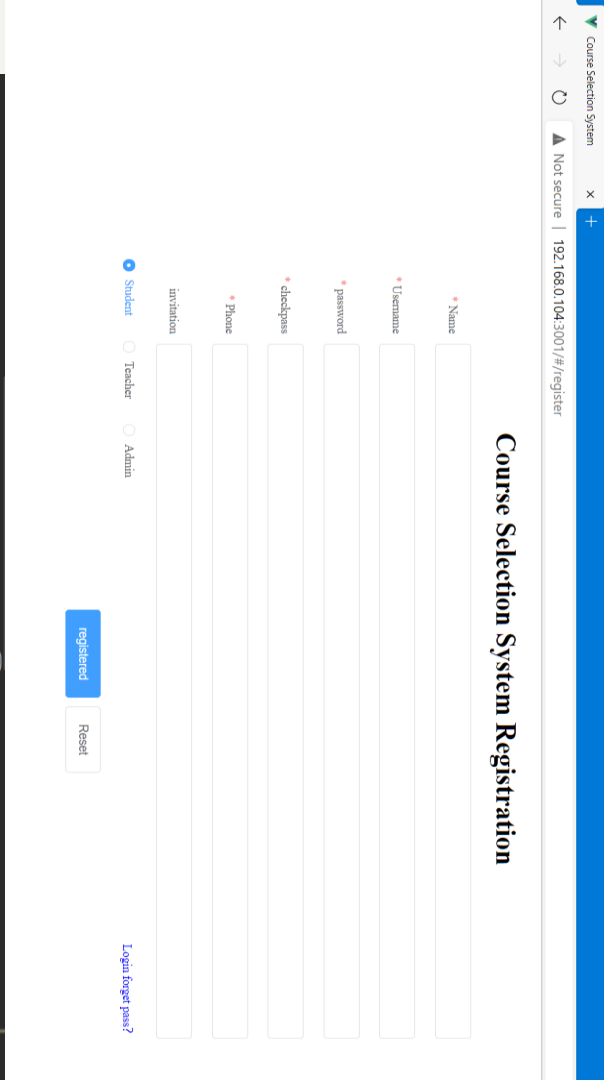
```
let mysql = require('mysql')
let pool = mysql.createPool( config: {
    host: 'localhost',
    user: 'root',
    password: '123456',
    database: 'courseselection',
    multipleStatements: true,
});

exports.getLimitStr = function(query) {
    if (!query.page || !query.count) throw Error('parameter error')
    return `LIMIT ${(query.page - 1) * query.count},${query.count}`
}

exports.querySql = function(callback) {
    pool.getConnection(function(err, connection) {
        if (err) throw err
        callback(connection)
        connection.release()
    })
}
```

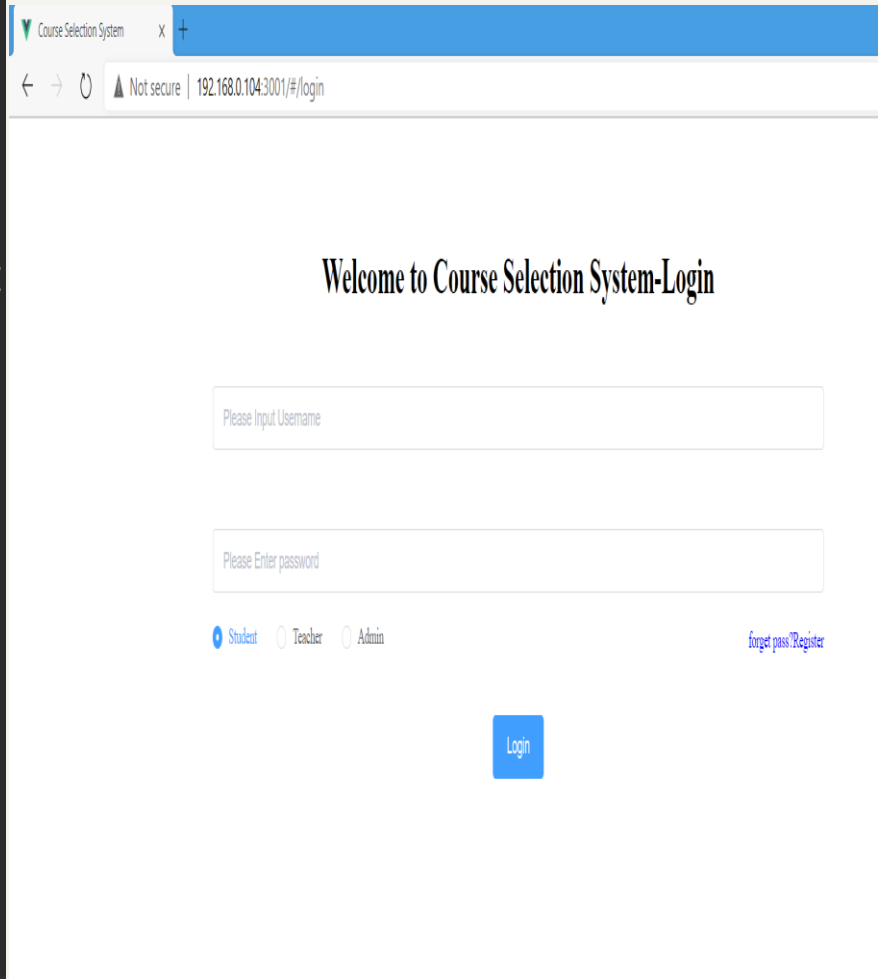
Registration

```
*Register-->Student invitation code 111 Teacher 222 Administrator 333*/
r.post( path: '/api/register', handlers: function(req : Request<P, ResBody, ReqBody, ReqQuery> , res : Response<ResBody> ) {
  let body = req.body
  let invitation = parseInt(body.invitation)
  let level = body.level
  console.log('post-->register:', level, 'name', body.name, 'username', body.username, 'password', body.password, 'phone', body.phone, 'invitation', body.invitation)
    /*verify invitation code*/
  if ((level === 'students' && invitation !== 111) || (level === 'teachers' && invitation !== 222) ||
    (level === 'admins') && invitation !== 333) return res.json( body: { message: 'wrong invitation code', status: 1 })
  querySql( callback: function(connection) {
    connection.query(`SELECT count(1) as count FROM ${body.level} WHERE username = '${body.username}'`, function(error, results) {
      if (error) throw error
      if (results.flat(Infinity).length === 0) return res.json( body: { message: 'no data available', status: 1 })
      if (results[0].count > 0) return res.json( body: { message: 'username already exists', status: 1 })
      connection.query(`INSERT ${level}(name, phone, username,password) VALUES(?,?,?,?);`, [body.name, body.phone, body.username, body.password], function(error2, results2)
        if (error2) throw error2
        if (results2.affectedRows > 0) {
          res.json( body: { message: 'registration success', token: getJwt(body, level, expiresIn: 3600 * 12), status: 0, id: results2.insertId })
        } else {
          res.json( body: { message: 'registration failed', status: 1 })
        }
      })
    })
  })
)
```



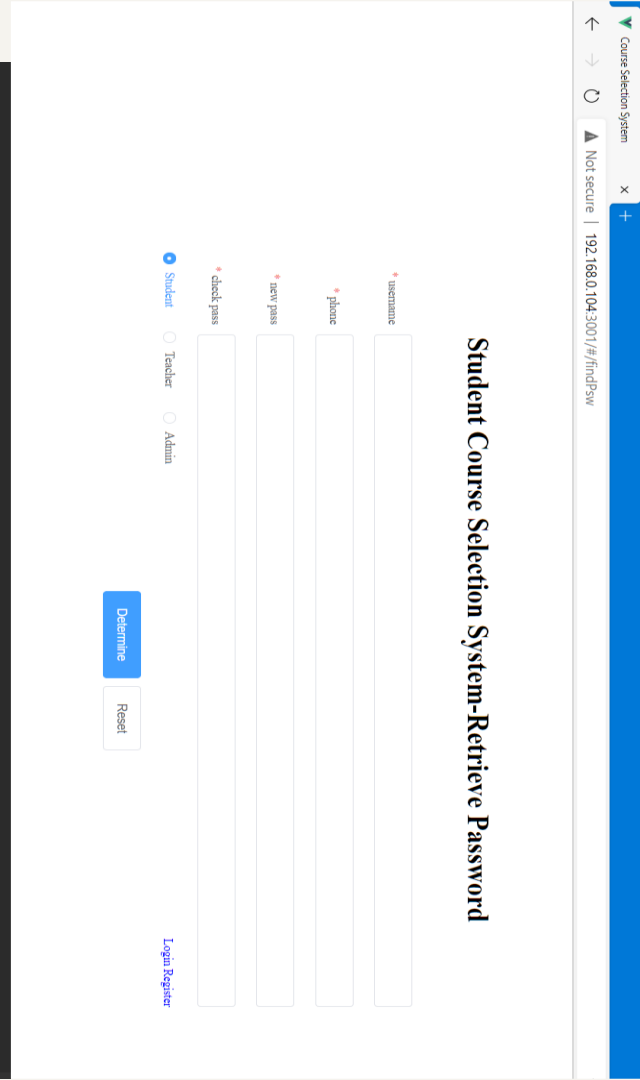
```
/*login*/
router.post( path: '/api/login', handlers: function(req : Request<P, ResBody, ReqBody, ReqQuery> , res : Response<ResBody> ) {
    let body = req.body
    let level = body.level
    console.log('post-->login:username', body.username, 'password', body.password, 'level', level)
    if (level !== 'students' && level !== 'teachers' && level !== 'admins') {
        return res.json( body: { message: 'Unknown identity', status: 1, notLogin: true })
    }
    querySql( callback: function(connection) {
        connection.query(`SELECT password,id,phone,name FROM ${level} WHERE username = '${body.username}' LIMIT 1;`, function(error, results) {
            if (error) throw error
            if (results.length === 0) return res.json( body: { message: 'username does not exist', status: 1 })
            let sqlData = results[0]
            let password = sqlData.password
            if (body.password !== password) return res.json( body: { message: 'wrong password', status: 1 })
            res.json( body: {
                    message: 'login successful!',
                    token: getJwt(body, level, expiresIn: 3600 * 12),
                    // refreshToken: getJwt(body, level, 3600 * 24 * 7),
                    status: 0,
                    id: sqlData.id,
                    phone: sqlData.phone,
                    name: sqlData.name
                })
        })
    })
})
```

Login page for Students, teachers and Admin



Retrieve password

```
/*Retrieve password*/
router.post( path: '/api/findPsw', handlers: function(req : Request<P, ResBody, ReqBody, ReqQuery> , res : Response<ResBody> ) {
    let body = req.body
    let level = body.level
    console.log('post-->findPsw:', level, 'username', body.username, 'phone', body.phone, 'password', body.password)
    querySql( callback: function(connection) {
        connection.query(`SELECT count(1) as count,phone,id,name FROM ${level} WHERE username = ?`, [body.username], function(error, results) {
            if (error) throw error
            let sqlData = results[0]
            if (sqlData.count === 0) return res.json( body: { message: 'username does not exist', status: 1 })
            if (sqlData.phone !== body.phone) return res.json( body: { message: 'secret phone error', status: 1 })
            connection.query(`UPDATE ${level} SET password=? WHERE username=?`, [body.password, body.username], function(error2, results2) {
                if (error2) throw error2
                if (results2.affectedRows > 0) {
                    res.json( body: {
                        message: 'successfully modified',
                        token: getJwt(body, level, expiresIn: 3600 * 12),
                        status: 0,
                        id: sqlData.id,
                        phone: sqlData.phone,
                        name: sqlData.name
                    })
                } else {
                    res.json( body: { message: 'fail to edit', status: 1 })
                }
            })
        })
    })
})
```



```
/*change password*/
router.post( path: '/api/updatePsw', handlers: function(req : Request<P, ResBody, ReqBody, ReqQuery> , res : Response<ResBody> ) {
    let body = req.body
    console.log('post-->updatePsw:id', body.id, 'level', body.level, 'oldPsw', body.oldPsw, 'newPsw', body.newPsw)
    if (!checkNumParam(body.id)) return res.json( body: { message: 'parameter error', status: 1 })
    querySql( callback: function(connection) {
        connection.query(`SELECT password FROM ${body.level} WHERE id = ${body.id}`, function(error, results) {
            if (error) throw error
            if (results.length === 0) return res.json( body: { message: 'identity error', status: 1 })
            if (results[0].password !== body.oldPsw) return res.json( body: { message: 'old pass error', status: 1 })
            connection.query(`UPDATE ${body.level} SET password = ? WHERE id = ?`, [body.newPsw, parseInt(body.id)], function(error2, results2) {
                if (error2) throw error2
                if (results2.affectedRows > 0) {
                    res.json( body: { message: 'successfully modified', token: getJwt(body, level, expiresIn: 3600 * 12), status: 0 })
                } else {
                    res.json( body: { message: 'fail to edit', status: 1 })
                }
            })
        })
    })
})
```

Change Pass

ChangePasss

* OldPass

* NewPass

* CheckPass

Determine Reset

Course Selection System

Not secure | 192.168.0.104:3001/#/logout

404

Personal Information and 404 page

Setup

Personal Information

Name: MD MAHFUZUR RAHMAN

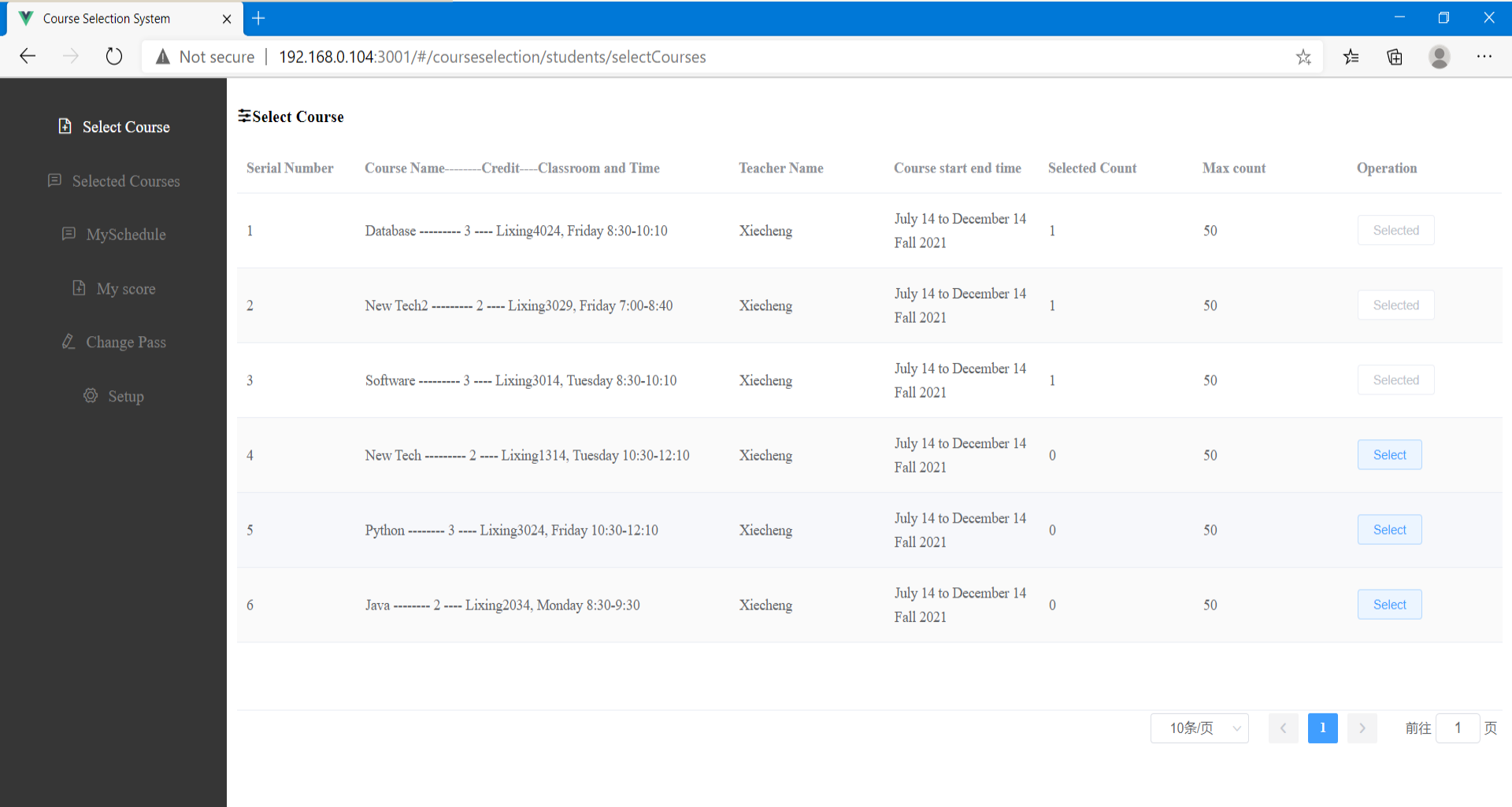
username: rony

Phone: 15587009804

Select course, students can view the all course information and choose the course

```
studentRouter.post( path: '/api/students/selectCourse', handlers: function(req : Request<P, ResBody, ReqBody, ReqQuery> , res : Response<ResBody> ) {
    let body = req.body
    console.log('post-->students/selectCourse:courseId', body.courseId, 'stuId', body.id)
    if (!checkNumParam( params: [body.id, body.courseId])) return res.json( body: { message: 'Parameter type error', status: 1 })
    if (parseInt(body.courseId) !== parseInt(body.courseId)) return res.json( body: { message: 'Parameter type error', status: 1 })
    querySql( callback: function(connection) {
        connection.query(`SELECT selectedCourseId FROM students WHERE id = ${body.id};
            SELECT count(1) AS selectedCount FROM scores WHERE stuId=${body.id} AND courseId = ${body.courseId};
            SELECT selectedCount,maxCount FROM courses WHERE courseId=${body.courseId}`, function(error, results) {
            if (error) throw error
            if (results.flat(Infinity).length === 0) return res.json( body: { message: 'no data available', status: 1 })
            results = results.flat(1)
            if (results[2].selectedCount >= results[2].maxCount) return res.json( body: { message: 'the number of people reached the limit', status: 1 })
            if (results[1].selectedCount > 0) return res.json( body: { message: 'course repeat', status: 1 })
            let selectedCourseId = results[0].selectedCourseId
            if (!selectedCourseId) {
                selectedCourseId = body.courseId + ''
            } else {
                if (selectedCourseId.split(',').indexOf(body.courseId + '') > -1) return res.json( body: {
                    message: 'course repeat',
                    status: 1
                })
                selectedCourseId = selectedCourseId + ',' + body.courseId
            }
            connection.query(`UPDATE students SET selectedCourseId = '${selectedCourseId}' WHERE id = ${body.id};
                UPDATE courses SET selectedCount = selectedCount+1 WHERE courseId = ${body.courseId};
                INSERT scores (stuId,courseId,score) VALUES(?,?,?)`, [body.id, body.courseId, -1], function(error2, results2) {
                if (error2) throw error2
                if (results2[0].affectedRows < 1 || results2[1].affectedRows < 1) {
                    res.json( body: { message: 'operation failed', status: 1 })
                } else {
                    res.json( body: { message: 'operation success', status: 0 })
```

Select Courses



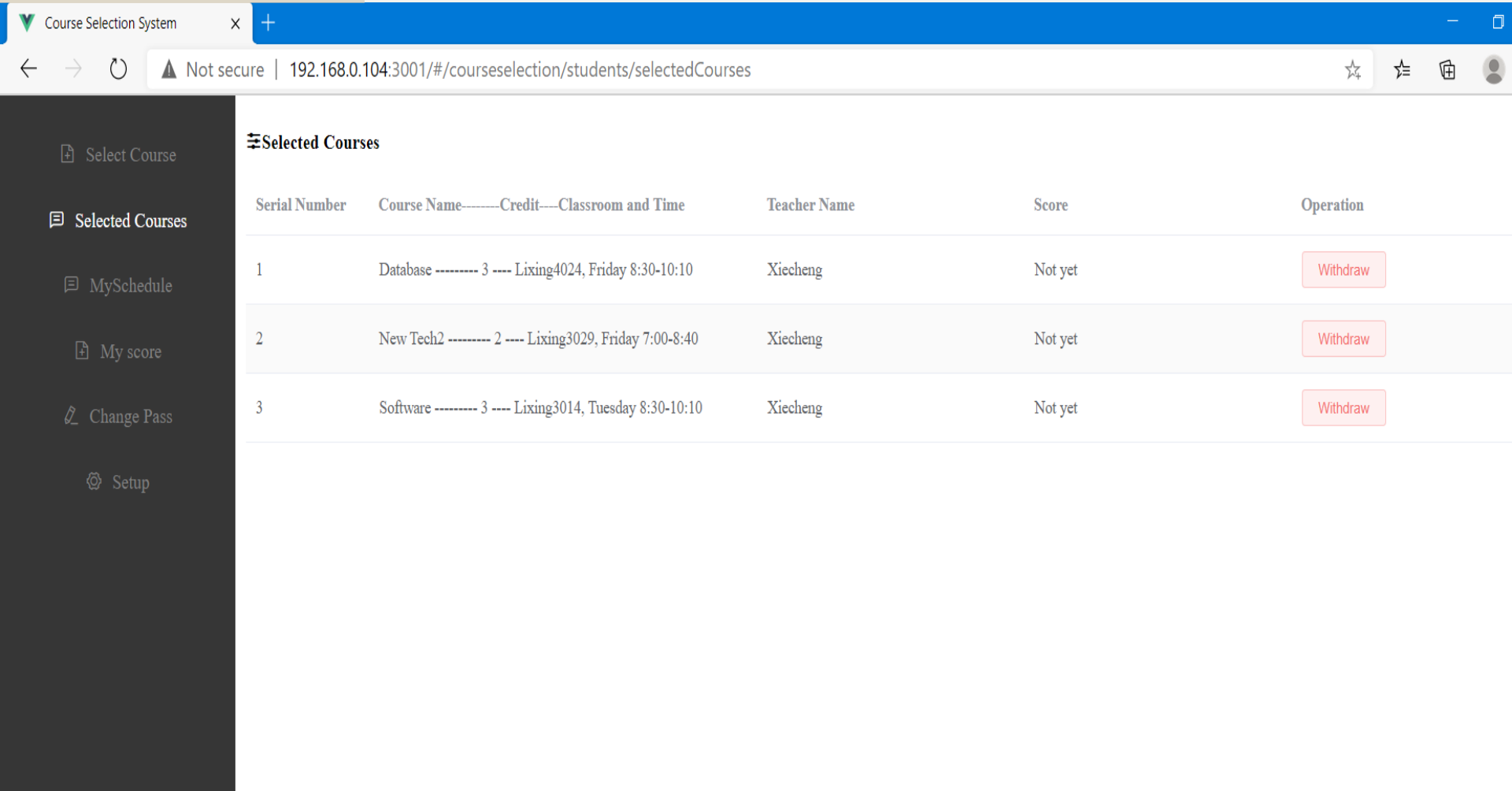
Selected course

```
    /*students get selected course*/
studentRouter.get( path: '/api/students/selectedCourses', handlers: function(req : Request<P, ResBody, ReqBody, ReqQuery> , res : Response<ResBody> ) {
    let query = filterQuery(req.query)
    console.log('get-->students/selectedCourses:page', query.page, 'count', query.count, 'id', query.id)
    if (!checkNumParam(query.id)) return res.json( body: { message: 'parameter error', status: 1 })
    querySql( callback: function(connection) {
        connection.query(`SELECT selectedCourseId FROM students WHERE id = ${query.id}`, function(error, results) {
            if (error) throw error
            if (results.flat(Infinity).length === 0) return res.json( body: { message: 'no data available', status: 1 })
            let courseIds = results[0].selectedCourseId
            if (!courseIds) return res.json( body: {
                message: 'no data available',
                status: 0,
                data: [
                    [], { totalCount: 0 }
                ]
            })
            connection.query(`SELECT courseName,teachers.name AS teacherName,score,courses.courseId
FROM courses, teachers, scores
WHERE courses.courseId in (${courseIds}) AND courses.teacherId=teachers.id AND scores.stuId=${query.id} AND scores.courseId = courses.courseId
ORDER BY courses.created_at DESC ${getLimitStr(query)};
SELECT count(1) AS totalCount FROM courses WHERE courseId in (${courseIds});`, function(error2, results2) {
                if (error2) throw error2
                results2[1] = { totalCount: results2[1][0].totalCount }
                res.json( body: { message: 'ok', status: 0, data: results2 })
            })
        })
    })
```

Withdraw course

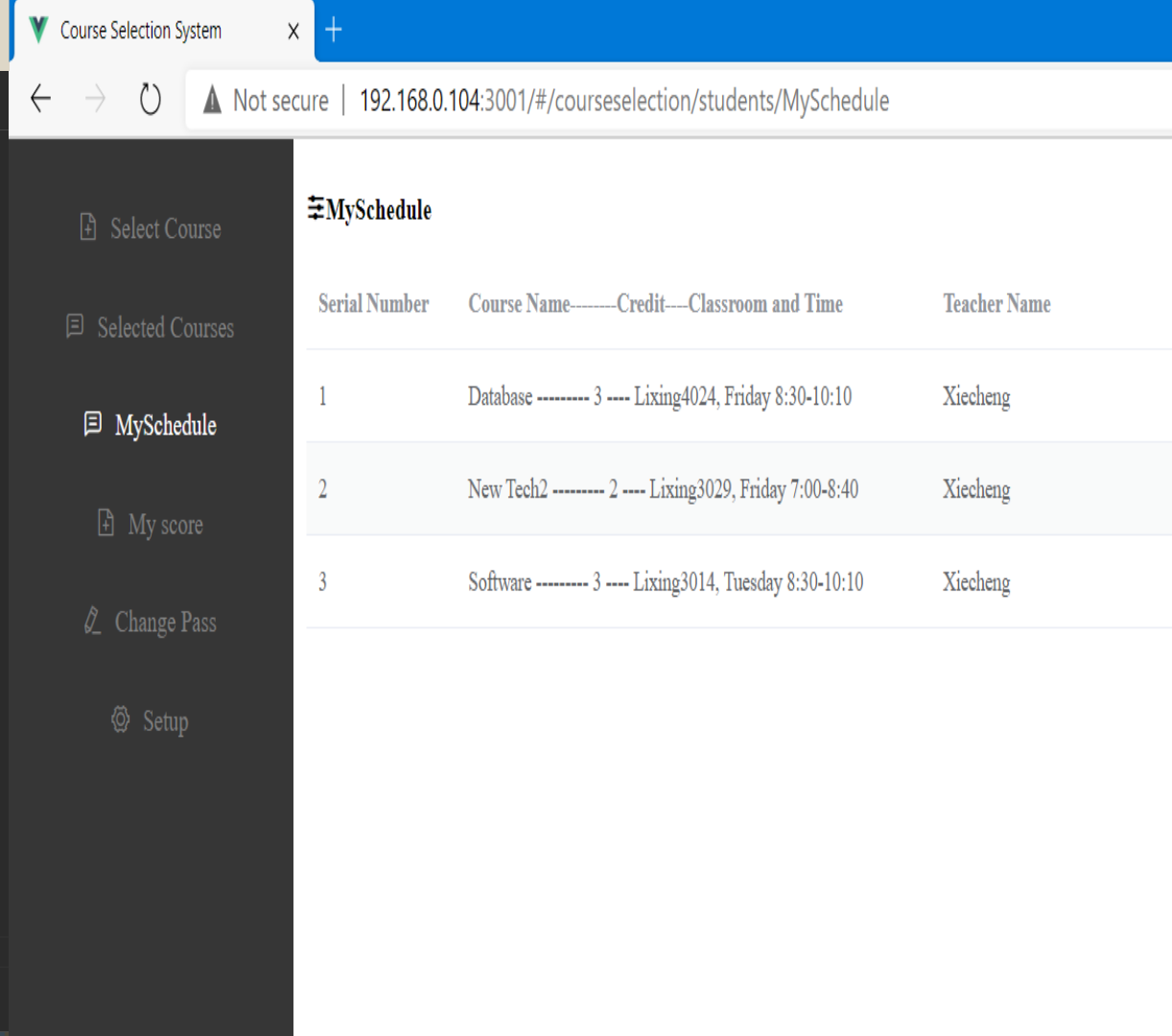
```
    /*students withdraw courses*/
studentRouter.post( path: '/api/students/deleteCourse', handlers: function(req : Request<P, ResBody, ReqBody, ReqQuery> , res : Response<ResBody> ) {
        let body = req.body
        console.log('post-->students/deleteCourse:id', body.id, 'courseId', body.courseId)
        if (!checkNumParam( params: [body.id, body.courseId])) return res.json( body: { message: 'parameter error', status: 1 })
        querySql( callback: function(connection) {
            connection.query(`SELECT selectedCourseId FROM students WHERE id = ${body.id}; SELECT score FROM scores
                WHERE stuId=${body.id} AND courseId=${body.courseId}`, function(error, results) {
                if (error) throw error
                if (results.flat(Infinity).length === 0) return res.json( body: { message: 'no data', status: 1 })
                results = results.flat(1)
                let selectedIds = results[0].selectedCourseId
                if (!selectedIds) return res.json( body: { message: 'this course is not selected', status: 1 })
                selectedIds = selectedIds.split(',')
                let courseIndex = selectedIds.indexOf(body.courseId + '')
                if (courseIndex < 0) return res.json( body: { message: 'this course is not selected', status: 1 })
                if (results[1].score > -1) return res.json( body: { message: 'this course has beed scored, and cannot be withdraw', status: 1 })
                selectedIds.splice(courseIndex, deleteCount: 1)
                let newCourseIds = selectedIds.join(',')
                connection.query(`UPDATE students SET selectedCourseId=? WHERE id=?; DELETE FROM scores WHERE stuId=? AND courseId=?`,
                    [newCourseIds, body.id, body.id, body.courseId], function(error2, results2) {
                    if (error2) throw error2
                    if (results2[0].affectedRows > 0 && results2[1].affectedRows > 0) {
                        res.json( body: { message: 'operation successful', status: 0 })
                    } else {
                        res.json( body: { message: 'operation failed', status: 1 })
                    }
                })
            })
        })
    })
```

Withdraw course

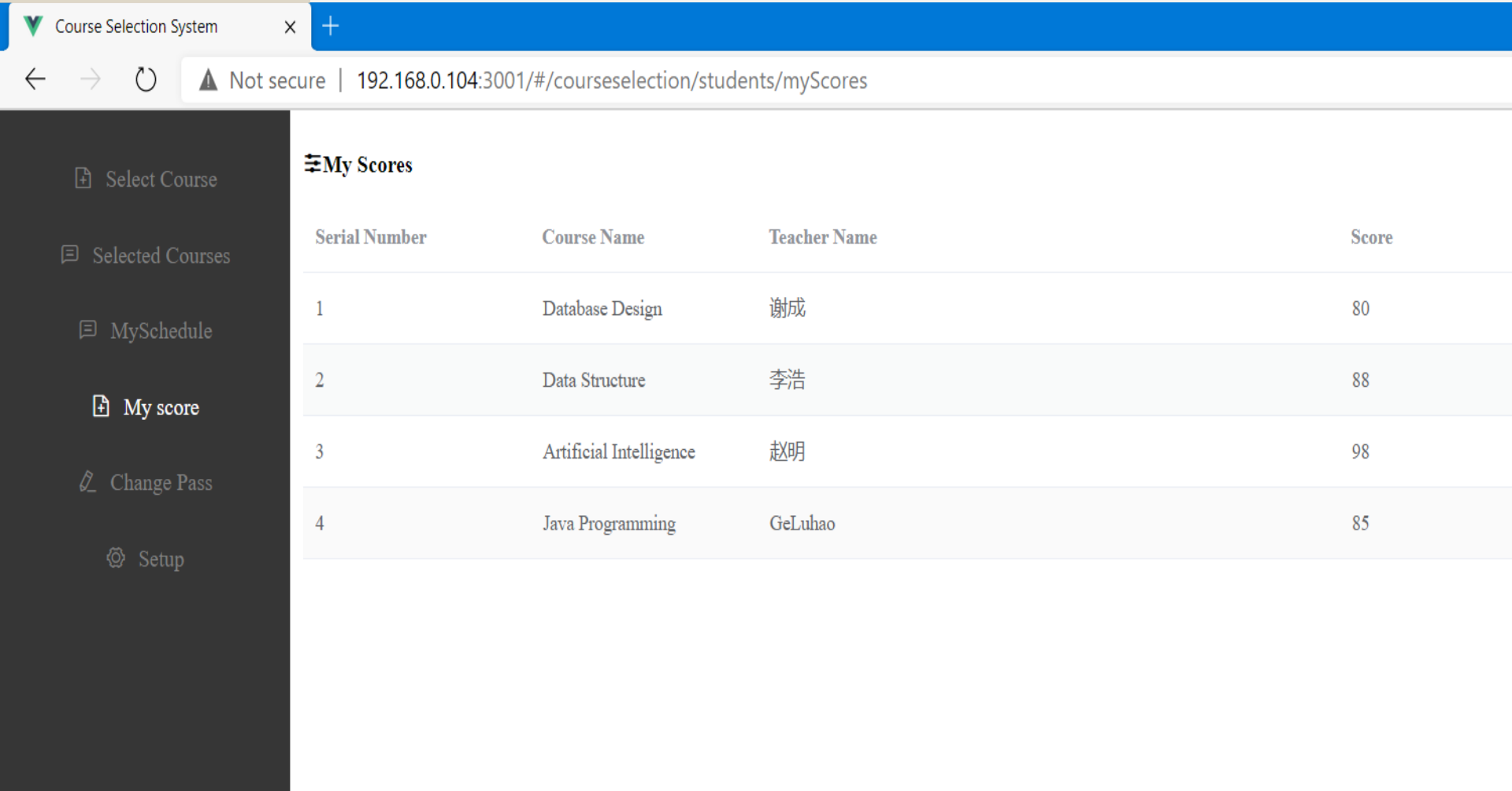


Student timetable

```
studentRouter.get( path: '/api/students/MySchedule', handlers: function(req : Request<P, ResBody, ReqBody, ReqQuery>, res : Response<ResBody> ) {
    let query = filterQuery(req.query)
    console.log('get-->students/MySchedule:page', query.page, 'count', query.count, 'id', query.id)
    if (!checkNumParam(query.id)) return res.json( body: { message: 'parameter error', status: 1 })
    querySql( callback: function(connection) {
        connection.query(`SELECT selectedCourseId FROM students WHERE id = ${query.id}`, function(error, results) {
            if (error) throw error
            if (results.flat(Infinity).length === 0) return res.json( body: { message: 'no data available', status: 1 })
            let courseIds = results[0].selectedCourseId
            if (!courseIds) return res.json( body: {
                message: 'no data available',
                status: 0,
                data: [
                    [], { totalCount: 0 }
                ]
            })
```



Student Scores



Add new course

```
/*teacher add new course*/
teacherRouter.post( path: '/api/teachers/addCourse', handlers: function(req : Request<P, ResBody, ReqBody, ReqQuery> , res : Response<ResBody> ) {
    let body = req.body
    console.log('post-->teachers/addCourse:courseName', body.courseName, 'teacherId', body.teacherId, 'maxCount', body.maxCount, 'classroom', body.classroom)
    if (!checkNumParam( params: [body.teacherId, body.maxCount])) return res.json( body: { message: 'parameter type errot', status: 1 })
    querySql( callback: function(connection) {
        connection.query(`INSERT courses(courseName,teacherId,classroom,selectedCount,maxCount) values(?,?,?,?,?);`,
            [body.courseName, body.teacherId, body.classroom, 0, body.maxCount], function(error, results) {
            if (error) throw error
            if (results.affectedRows > 0) {
                res.json( body: { message: 'added successfully', status: 0 })
            } else {
                res.json( body: { message: 'add failed', status: 1 })
            }
        })
    })
})
```

See added course

```
/*teachers added  courses*/
teacherRouter.get( path: '/api/teachers/addedCourses', handlers: function(req : Request<P, ResBody, ReqBody, ReqQuery> , res : Response<ResBody> ) {
    let query = filterQuery(req.query)
    console.log('get-->teachers/addedCourses:page', query.page, 'count', query.count)
    querySql( callback: function(connection) {
        connection.query(`SELECT courseName,classroom,selectedCount,courseId FROM courses
            WHERE teacherId = ${query.teacherId} ORDER BY courses.created_at DESC ${getLimitStr(query)};
            SELECT count(1) as totalCount FROM courses`, function(error, results) {
            if (error) throw error
            if (results.flat(Infinity).length === 0) return res.json( body: { message: 'no data', status: 1 })
            results[1] = { totalCount: results[1][0].totalCount }
            res.json( body: { message: 'ok', status: 0, data: results })
        })
    })
})
```

All courses

```
/*teacher all courses*/
teacherRouter.get( path: '/api/teachers/allCourses', handlers: function(req : Request<P, ResBody, ReqBody, ReqQuery> , res : Response<ResBody> ) {
    let query = filterQuery(req.query)
    console.log('get-->teachers/allCourses:page', query.page, 'count', query.count)
    querySql( callback: function(connection) {
        connection.query(`SELECT courseName,teachers.name as teacherName,classroom,maxCount,selectedCount
            FROM courses INNER JOIN teachers WHERE courses.teacherId = teachers.id ORDER BY courses.created_at DESC ${getLimitStr(query)};
            SELECT count(1) as totalCount FROM courses`, function(error, results) {
            if (error) throw error
            if (results.flat(Infinity).length === 0) return res.json( body: { message: 'No data', status: 1 })
            results[1] = { totalCount: results[1][0].totalCount }
            res.json( body: { message: 'ok', status: 0, data: results })
        });
    })
})
```

Course Selection System

Not secure | 192.168.0.104:3001/#/courseselection/teachers/allCourses

- All Courses
- Student Result
- Edit Courses
- mySchedule
- Change Password
- Setup

All Courses

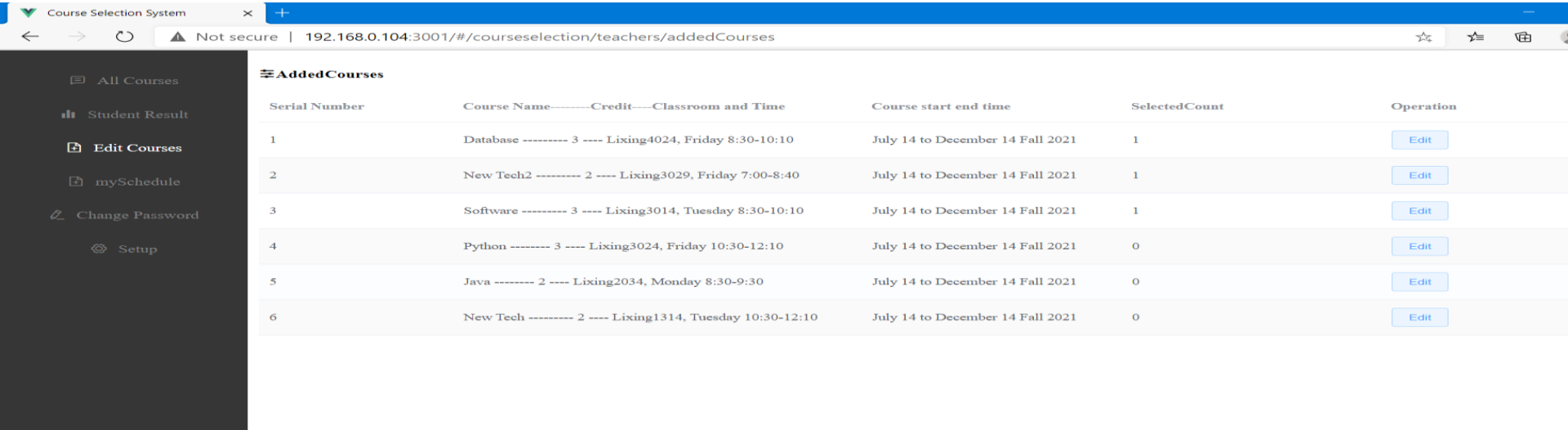
Add Course

Course ID	Course Name---------Credit----Classroom and Time	Teacher Name	Course start end time	SelectedCount	MaxCount
1	Database --------- 3 ---- Lixing4024, Friday 8:30-10:10	Xiecheng	July 14 to December 14 Fall 2021	1	50
2	New Tech2 --------- 2 ---- Lixing3029, Friday 7:00-8:40	Xiecheng	July 14 to December 14 Fall 2021	1	50
3	Software --------- 3 ---- Lixing3014, Tuesday 8:30-10:10	Xiecheng	July 14 to December 14 Fall 2021	1	50
4	New Tech --------- 2 ---- Lixing1314, Tuesday 10:30-12:10	Xiecheng	July 14 to December 14 Fall 2021	0	50
5	Python -------- 3 ---- Lixing3024, Friday 10:30-12:10	Xiecheng	July 14 to December 14 Fall 2021	0	50
6	Java -------- 2 ---- Lixing2034, Monday 8:30-9:30	Xiecheng	July 14 to December 14 Fall 2021	0	50

10条/页 1 前往 1 页

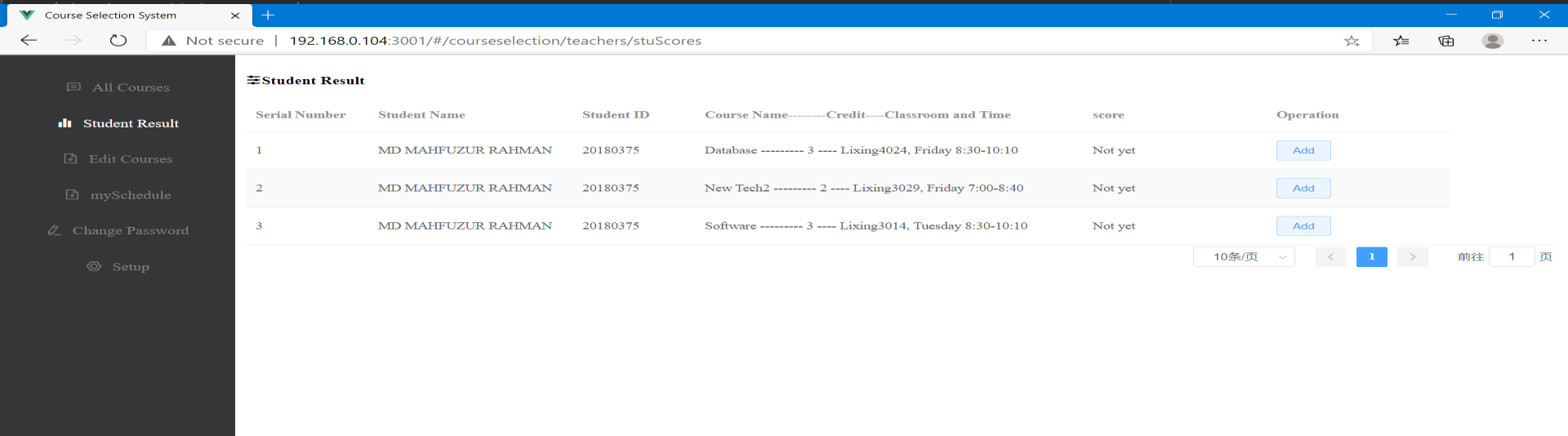
Modify Course

```
    /*teacher modify the course information he added*/
teacherRouter.post( path: '/api/teachers/updateCourse', handlers: function(req : Request<P, ResBody, ReqBody, ReqQuery> , res : Response<ResBody> ) {
    let body = req.body
    console.log('post-->teachers/updateCourse:courseId', body.courseId, 'teacherId', body.teacherId, 'courseName', body.courseName, 'classroom', body.classroom)
    if (!checkNumParam( params: [body.courseId, body.teacherId])) return res.json( body: { message: 'Parameter type error', status: 1 })
    querySql( callback: function(connection) {
        connection.query(`UPDATE courses SET courseName = ?,classroom  = ? WHERE courseId = ? AND teacherId=?`, [body.courseName, body.classroom, body.courseId, body.teacherId]
            if (error) throw error
            if (results.affectedRows > 0) {
                res.json( body: { message: 'successfully modified', status: 0 })
            } else {
                res.json( body: { message: 'fail to edit', status: 1 })
            }
        })
    })
})
```



Add score to students

```
    /*teacher add student score*/
teacherRouter.post( path: '/api/teachers/addScore', handlers: function(req : Request<P, ResBody, ReqBody, ReqQuery> , res : Response<ResBody> ) {
        let body = req.body
        console.log('post-->teachers/addScore:courseId', body.courseId, 'stuId', body.stuId, 'score', body.score)
        if (!checkNumParam( params: [body.courseId, body.stuId, body.score])) return res.json( body: { message: 'parameter type error', status: 1 })
        querySql( callback: function(connection) {
            connection.query(`UPDATE scores SET score = ${body.score} WHERE stuId = ${body.stuId} AND courseId = ${body.courseId}`, function(error, results) {
                if (error) throw error
                if (results.affectedRows > 0) {
                    res.json( body: { message: 'add sccessfully', status: 0 })
                } else {
                    res.json( body: { message: 'add failed', status: 1 })
                }
            })
        })
    })
```



TimeTable

```
    /*teachers | time table*/
teacherRouter.get( path: '/api/teachers/mySchedule',  handlers: function(req : Request<P, ResBody, ReqBody, ReqQuery> , res : Response<ResBody> ) {
        let query = filterQuery(req.query)
        console.log('get-->teachers/mySchedule:page', query.page, 'count', query.count)
        querySql( callback: function(connection) {
            connection.query(`SELECT courseName,classroom,selectedCount,courseId FROM courses WHERE teacherId = ${query.teacherId} ORDER BY courses.created
                if (error) throw error
                if (results.flat(Infinity).length === 0) return res.json( body: { message: 'no data', status: 1 })
                results[1] = { totalCount: results[1][0].totalCount }
                res.json( body: { message: 'ok', status: 0, data: results })
            })
        })
    })
```

Course Selection System

Not secure | 192.168.0.104:3001/#/courseselection/teachers/mySchedule

- All Courses
- Student Result
- Edit Courses
- mySchedule
- Change Password
- Setup

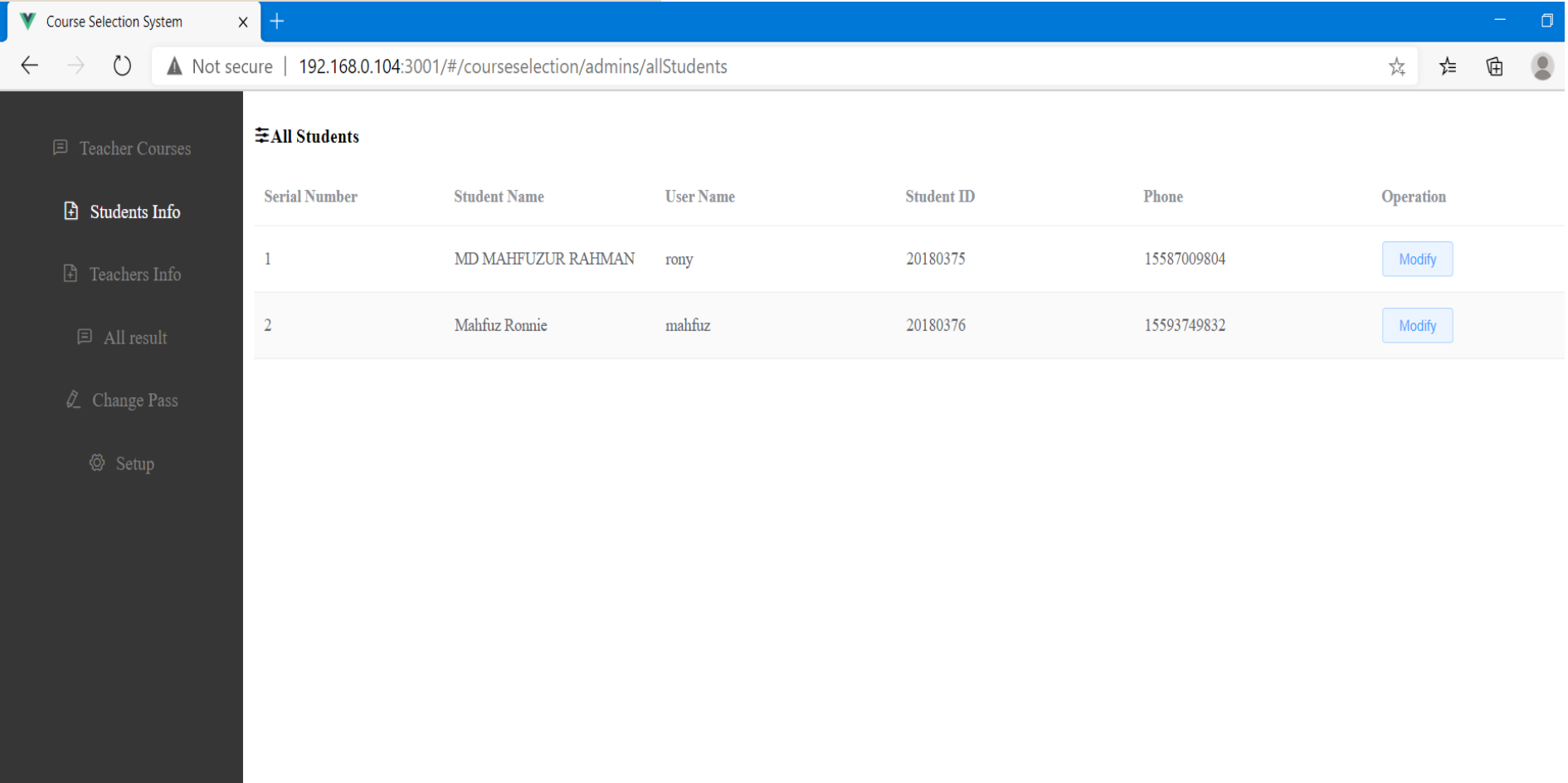
mySchedule

Serial Number	Course Name---------Credit----Classroom and Time	Course start end time
1	Database --------- 3 ---- Lixing4024, Friday 8:30-10:10	July 14 to December 14 Fall 2021
2	New Tech2 --------- 2 ---- Lixing3029, Friday 7:00-8:40	July 14 to December 14 Fall 2021
3	Software --------- 3 ---- Lixing3014, Tuesday 8:30-10:10	July 14 to December 14 Fall 2021
4	Python -------- 3 ---- Lixing3024, Friday 10:30-12:10	July 14 to December 14 Fall 2021
5	Java -------- 2 ---- Lixing2034, Monday 8:30-9:30	July 14 to December 14 Fall 2021
6	New Tech --------- 2 ---- Lixing1314, Tuesday 10:30-12:10	July 14 to December 14 Fall 2021

Admin can see students info and can modify

```
    /*The administrator gets all student information*/
adminRouter.get( path: '/api/admins/allStudents', handlers: function(req : Request<P, ResBody, ReqBody, ReqQuery> , res : Response<ResBody> ) {
        let query = filterQuery(req.query)
        console.log('get-->admins/allStudents:page', query.page, 'count', query.count)
        querySql( callback: function(connection) {
            connection.query(`SELECT id AS stuId,name AS studentName,phone,username
FROM students ORDER BY students.created_at DESC ${getLimitStr(query)};
SELECT count(1) as totalCount FROM students`, function(error, results) {
                if (error) throw error
                if (results.flat(Infinity).length === 0) return res.json( body: { message: 'no data available', status: 1 })
                results[1] = { totalCount: results[1][0].totalCount }
                res.json( body: { message: 'ok', status: 0, data: results })
            })
        })
    /*Admin modify student information*/
adminRouter.post( path: '/api/admins/updateStudent', handlers: function(req : Request<P, ResBody, ReqBody, ReqQuery> , res : Response<ResBody> ) {
    let body = req.body
    console.log('post-->admins/updateStudent:stuId', body.stuId, 'studentName', body.studentName, 'phone', body.phone)
    if (!checkNumParam( params: [body.stuId, body.phone])) return res.json( body: { message: 'parameter type error', status: 1 })
    querySql( callback: function(connection) {
        connection.query(`UPDATE students SET name=?,phone=? WHERE id=?`, [body.studentName, body.phone, body.stuId], function(error, results) {
            if (error) throw error
            if (results.affectedRows > 0) {
                res.json( body: { message: 'successfully modified', status: 0 })
            } else {
                res.json( body: { message: 'fail to edit', status: 1 })
            }
```

Students Info in Admin page

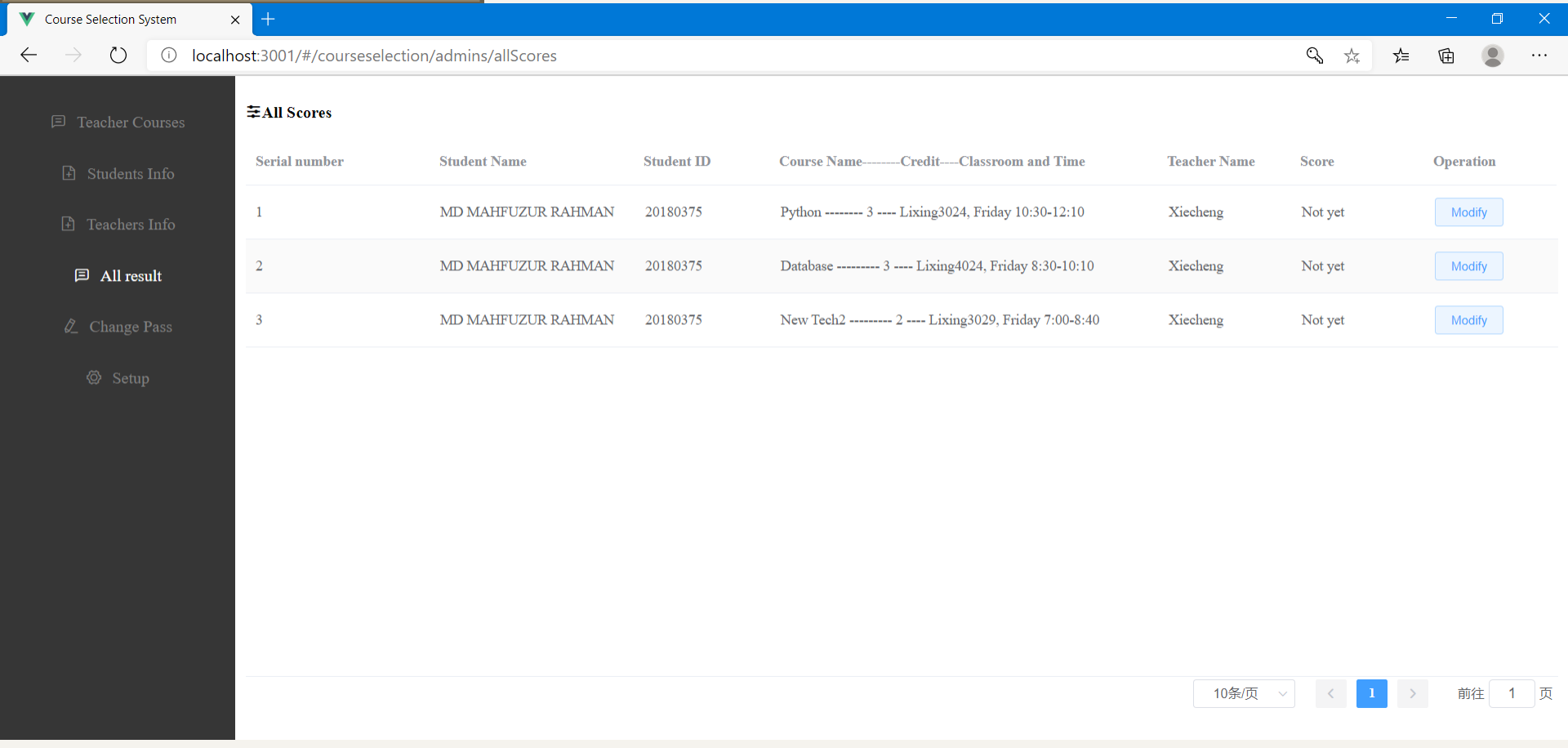


View and modify students scores

```
    /*Admin obtains student grades*/
adminRouter.get( path: '/api/admins/allScores', handlers: function(req : Request<P, ResBody, ReqBody, ReqQuery> , res : Response<ResBody> ) {
        let query = filterQuery(req.query)
        console.log('get-->admins/allScores:page', query.page, 'count', query.count)
        querySql( callback: function(connection) {
            connection.query(`SELECT courses.courseId,students.id AS stuId,score,courseName,students.name AS studentName,teachers.name AS teacherName
FROM ((scores INNER JOIN courses ON scores.courseId=courses.courseId) INNER JOIN students ON students.id=scores.stuId)
INNER JOIN teachers ON teachers.id=courses.teacherId ORDER BY scores.created_at DESC ${getLimitStr(query)}; SELECT count(1) as totalCount FROM scores`, function(error, results)
                if (error) throw error
                if (results.flat(Infinity).length === 0) return res.json( body: { message: 'No data available', status: 1 })
                results[1] = { totalCount: results[1][0].totalCount }
                res.json( body: { message: 'ok', status: 0, data: results })
            })
        })
    })
```

```
    /*Administrator modify student grades*/
adminRouter.post( path: '/api/admins/updateScore', handlers: function(req : Request<P, ResBody, ReqBody, ReqQuery> , res : Response<ResBody> ) {
        let body = req.body
        console.log('post-->admins/updateScore:courseId', body.courseId, 'stuId', body.stuId, 'score', body.score)
        if (!checkNumParam( params: [body.courseId, body.stuId, body.score])) return res.json( body: { message: 'Parameter type error', status: 1 })
        querySql( callback: function(connection) {
            connection.query(`UPDATE scores SET score = ? WHERE courseId=? AND stuId=?`, [body.score, body.courseId, body.stuId], function(error, results) {
                if (error) throw error
                if (results.affectedRows > 0) {
                    res.json( body: { message: 'Successfully modified', status: 0 })
                } else {
                    res.json( body: { message: 'fail to edit', status: 1 })
                }
            })
        })
    })
```

View and modify students Scores

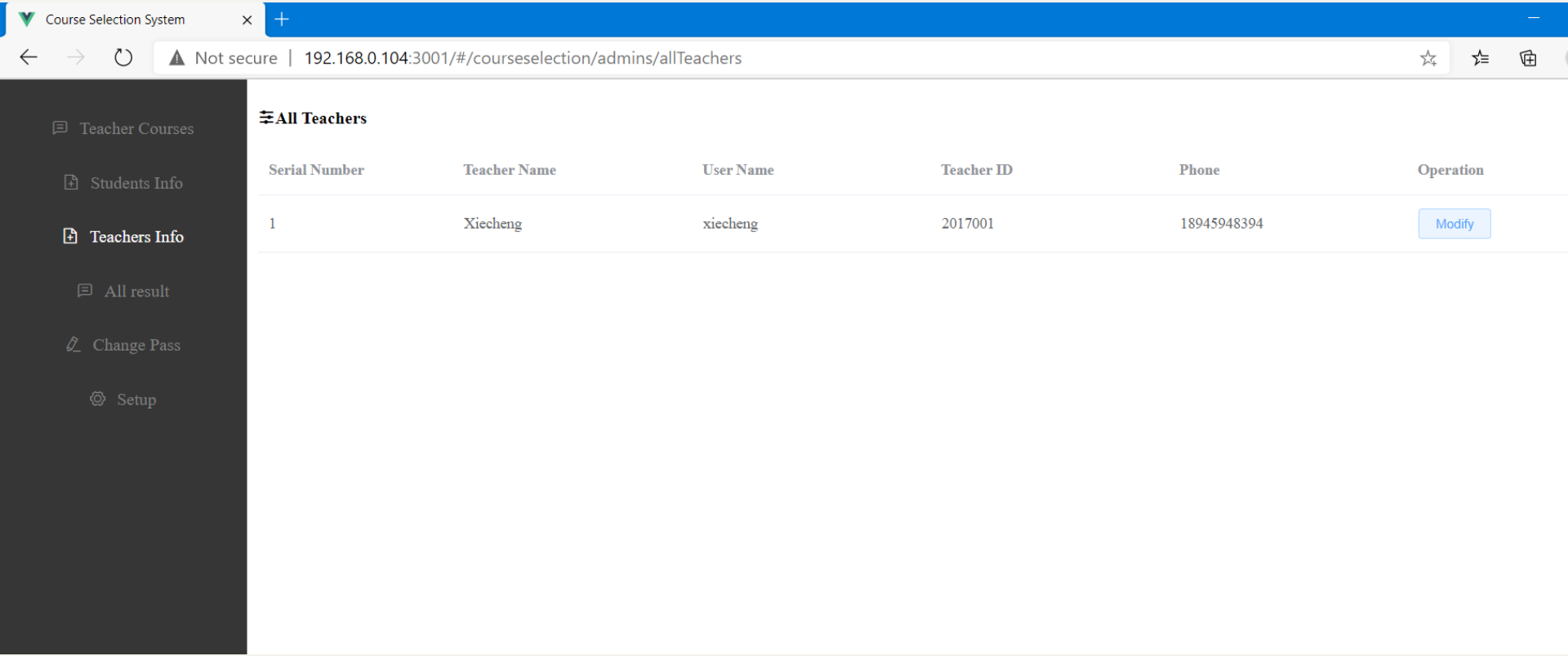


View and modify teacher info

```
    /*The administrator gets all teacher information*/
adminRouter.get( path: '/api/admins/allTeachers', handlers: function(req : Request<P, ResBody, ReqBody, ReqQuery> , res : Response<ResBody> ) {
        let query = filterQuery(req.query)
        console.log('get-->admins/allTeachers:page', query.page, 'count', query.count)
        querySql( callback: function(connection) {
            connection.query(`SELECT id AS teacherId,name AS teacherName,phone,username FROM teachers ORDER BY teachers.created_at DESC ${getLimitStr(query)};
SELECT count(1) as totalCount FROM teachers`, function(error, results) {
                if (error) throw error
                if (results.flat(Infinity).length === 0) return res.json( body: { message: 'no data available', status: 1 })
                results[1] = { totalCount: results[1][0].totalCount }
                res.json( body: { message: 'ok', status: 0, data: results })
            })
        })
    })
```

```
    /*The administrator modifies the teacher information*/
adminRouter.post( path: '/api/admins/updateTeacher', handlers: function(req : Request<P, ResBody, ReqBody, ReqQuery> , res : Response<ResBody> ) {
        let body = req.body
        console.log('post-->admins/updateTeacher:teacherId', body.teacherId, 'teacherName', body.teacherName, 'phone', body.phone)
        if (!checkNumParam( params: [body.teacherId, body.phone])) return res.json( body: { message: 'Parameter type error', status: 1 })
        querySql( callback: function(connection) {
            connection.query(`UPDATE teachers SET name=?,phone=? WHERE id=?`, [body.teacherName, body.phone, body.teacherId], function(error, results) {
                if (error) throw error
                if (results.affectedRows > 0) {
                    res.json( body: { message: 'successfully modified', status: 0 })
                } else {
                    res.json( body: { message: 'fail to edit', status: 1 })
                }
            })
        })
    })
```

View and Modify Teachers info

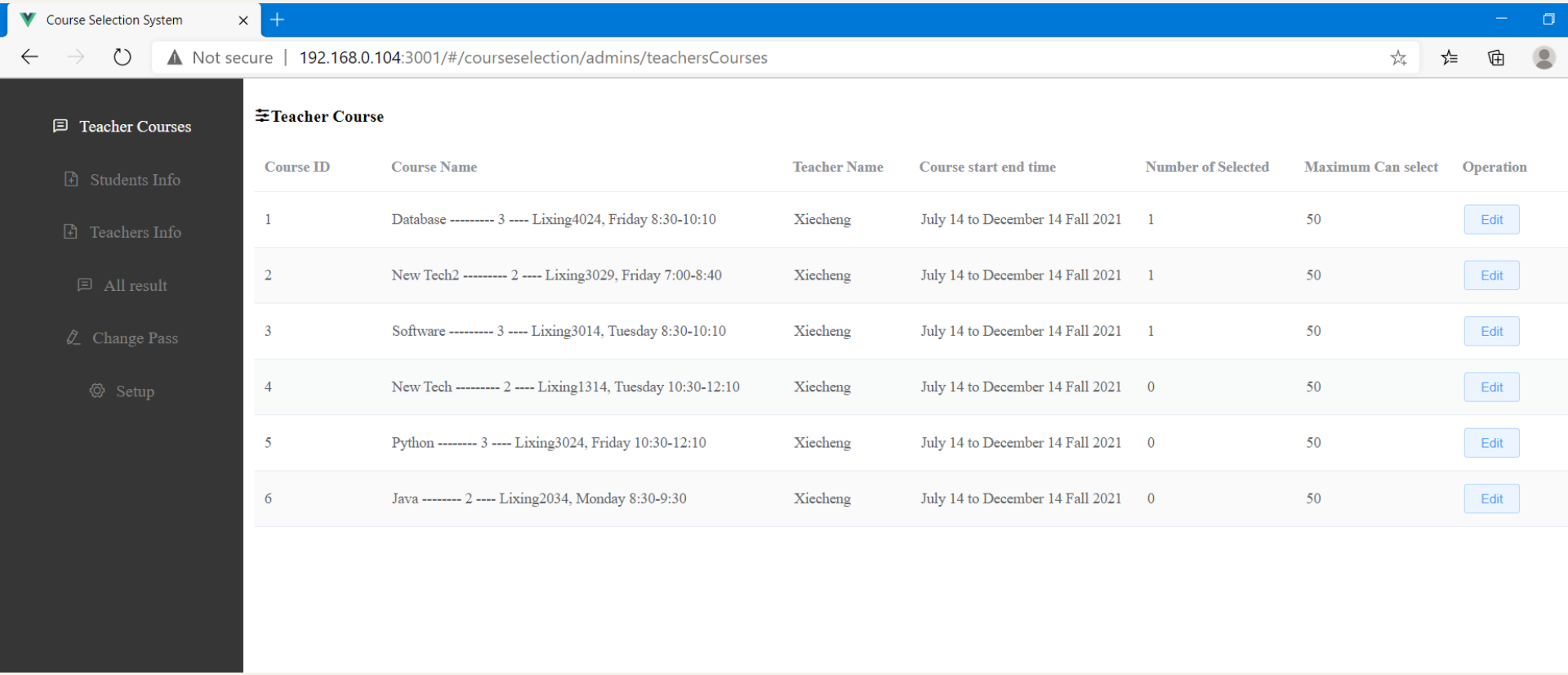


Teachers course info and modify

```
    /*The administrator obtains teacher course information*/
adminRouter.get( path: '/api/admins/teacherCourses', handlers: function(req : Request<P, ResBody, ReqBody, ReqQuery> , res : Response<ResBody> ) {
        let query = filterQuery(req.query)
        console.log('get-->admins/teacherCourses:page', query.page, 'count', query.count)
        querySql( callback: function(connection) {
            connection.query(`SELECT courseId,courseName,teachers.name AS teacherName,classroom,selectedCount,maxCount,
teachers.id AS teacherId FROM courses INNER JOIN teachers ON courses.teacherId=teachers.id ORDER BY courses.created_at DESC ${getLimitStr(query)};
SELECT count(1) as totalCount FROM courses`, function(error, results) {
                if (error) throw error
                if (results.flat(Infinity).length === 0) return res.json( body: { message: 'No Data Available', status: 1 })
                results[1] = { totalCount: results[1][0].totalCount }
                res.json( body: { message: 'ok', status: 0, data: results })
            })
        })
    })
```

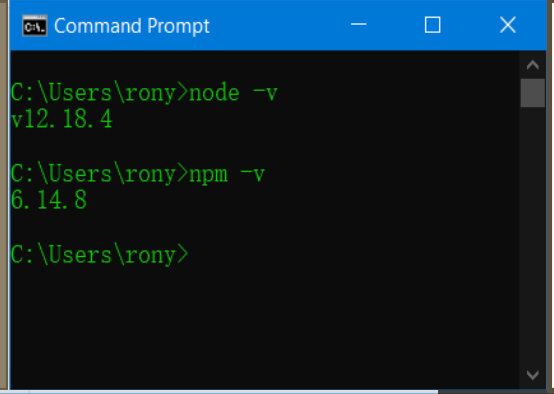
```
    /*The administrator obtains teacher course information*/
adminRouter.post( path: '/api/admins/updateCourse', handlers: function(req : Request<P, ResBody, ReqBody, ReqQuery> , res : Response<ResBody> ) {
        let body = req.body
        console.log('post-->admins/updateCourse:courseId', body.courseId, 'teacherId', body.teacherId, 'courseName', body.courseName, 'classroom', body.classroom, 'maxCount', bod
        if (!checkNumParam( params: [body.courseId, body.teacherId, body.maxCount])) return res.json( body: { message: 'Parameter type error', status: 1 })
        querySql( callback: function(connection) {
            connection.query(`UPDATE courses SET courseName = ?,classroom  = ?,maxCount = ? WHERE courseId = ? AND teacherId=?`,
                [body.courseName, body.classroom, body.maxCount, body.courseId, body.teacherId], function(error, results) {
                if (error) throw error
                if (results.affectedRows > 0) {
                    res.json( body: { message: 'Successfully modify', status: 0 })
                } else {
                    res.json( body: { message: 'Fail to edit', status: 1 })
                }
            })
        })
    })
```

Teachers course info and modify



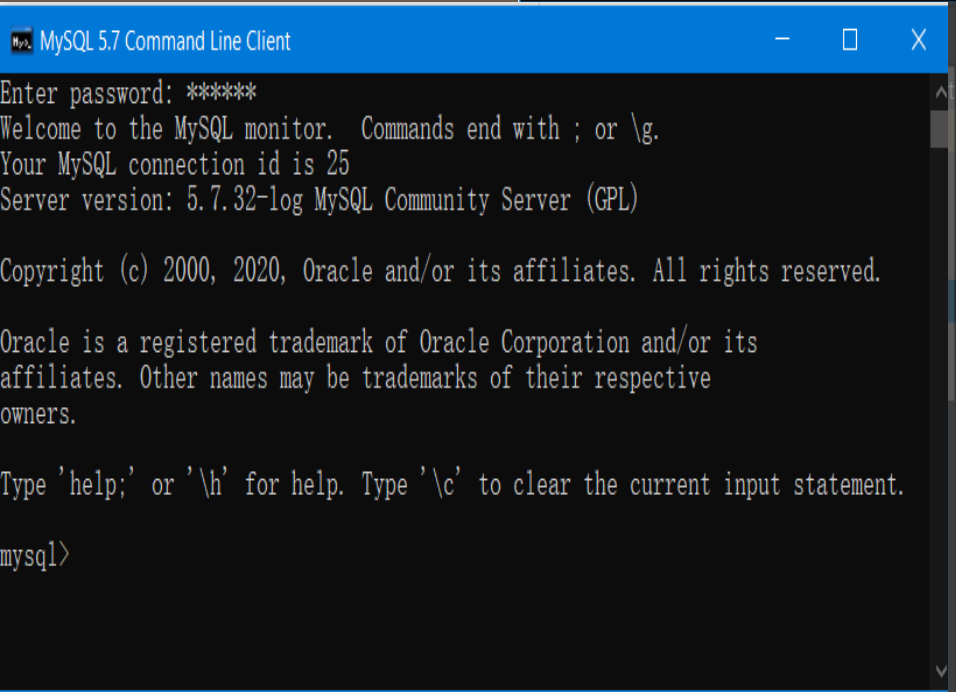
Show the runnable Project

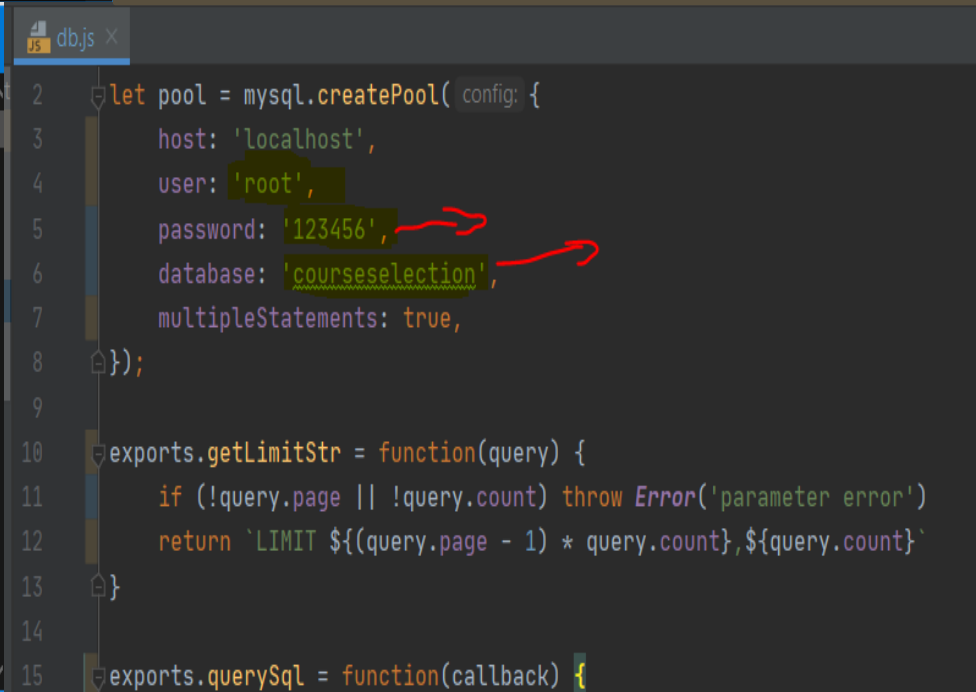
How to run this system in your computer



**Must use MySQL 5x
After open this project in you IDE, change username and password in db.js file, use your own username and password.

- node app.js
- npm run serve






Thank You

What I send before

- Requirement version 1,2,3
- Design Document V1, V2
- Implementation V 0.5, V1, V2, V3

What I am going to deliver

- Organized source code with README.md file
- Final Report of this project