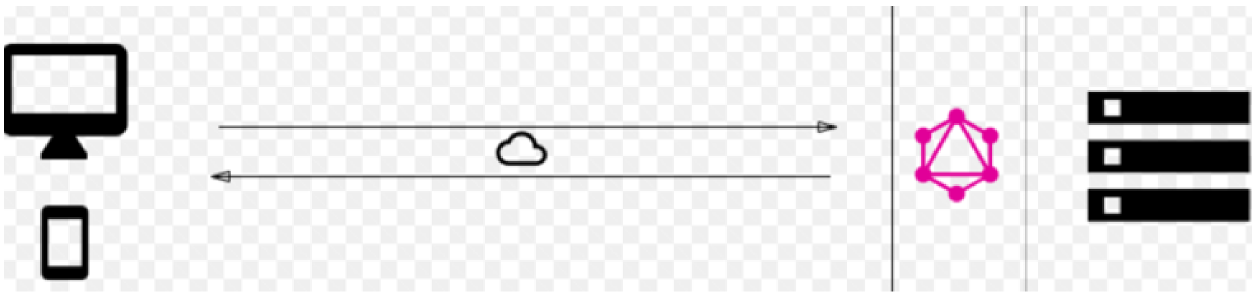
Lab3-Report - Koushik Kumar Kamala – 013766571

Repo: https://github.com/koushik-kumar/Canvas_GraphQL

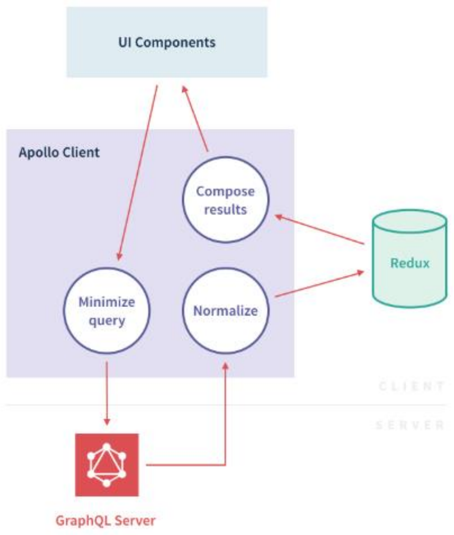
Canvas:

To implement a canvas application by implementing GraphQL as communication between client and server. In addition to that, implementing Frontend using ReactJS and Backend using NodeJS.

Architecture Design Diagram:



System Design Diagram:

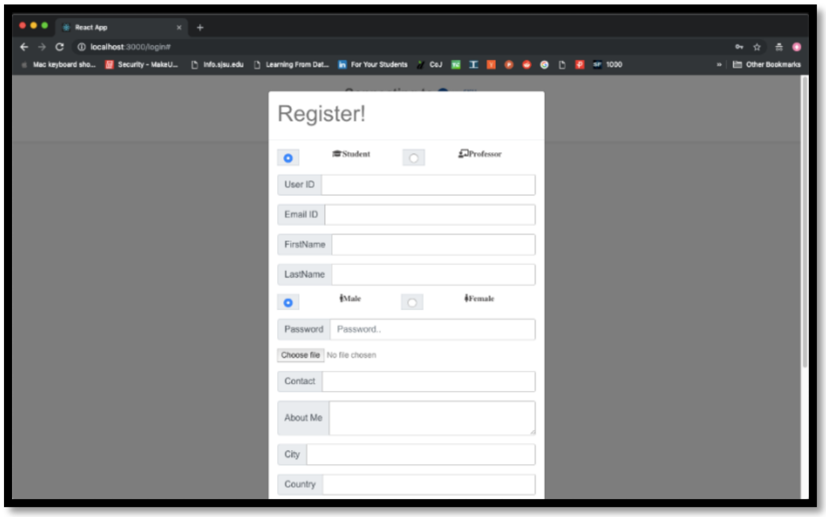


Faculty/Student can access frontend canvas application, which is built using ReactJS. Once user sends a request from frontend, request receives at Server, which is designed using NodeJS. Communication protocol using GraphQL.

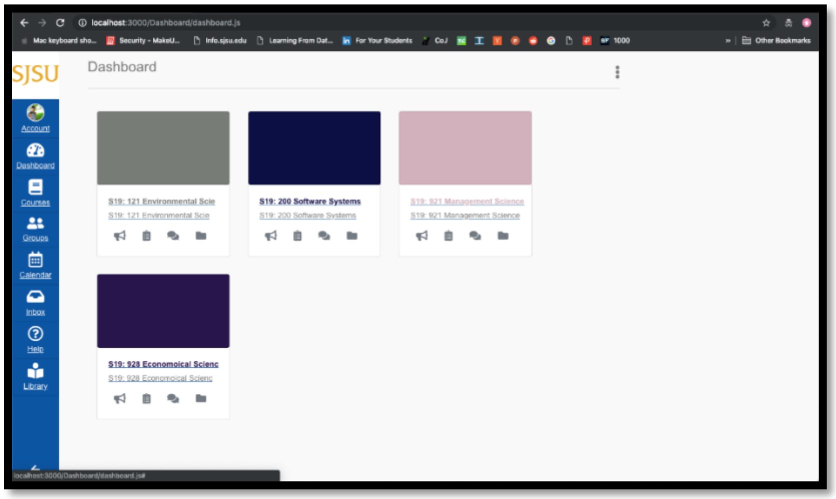
Goal:

- In this Lab, goal is to design a canvas application , which do not uses HTTP Protocol but instead uses GraphQL mutations/queries.
- Used React-Appollo for GraphQL mutations/queries.
- Low payload requests using GraphQL queries.

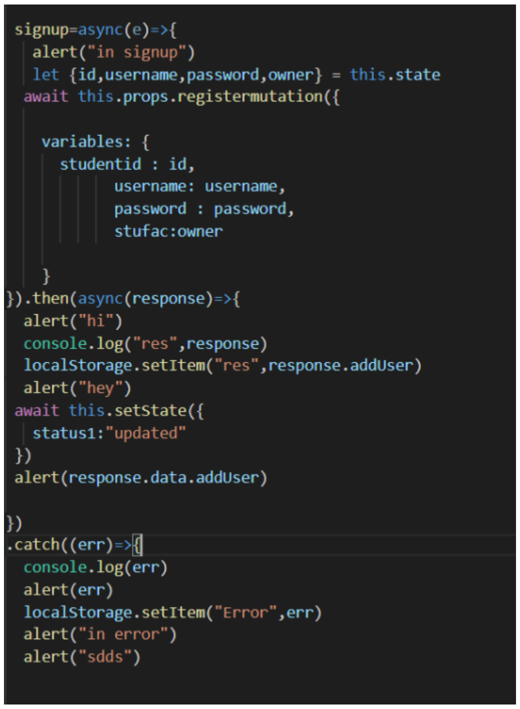
Student Signup:



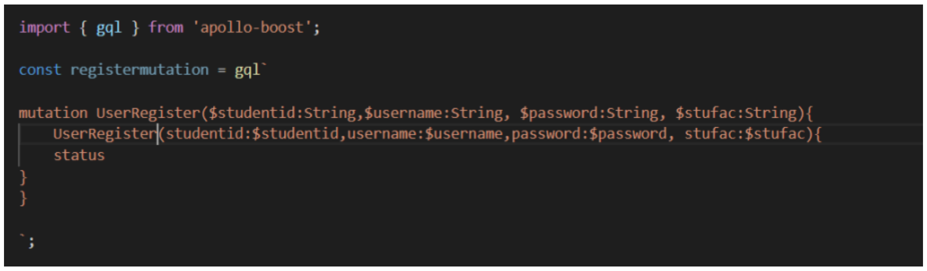
Register Page


SJSU
Dashboard
Account
Dashboard
Courses
Groups
Calendar
Inbox
Help
Library
S19: 121 Environmental Scie
S19: 200 Software Systems
S19: 921 Management Science
S19: 928 Economoical Scienc

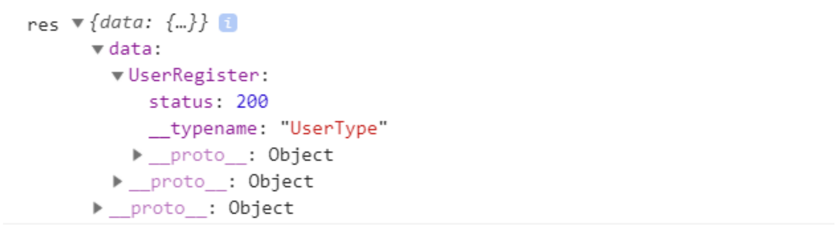
Frontend:



```
signup=async(e)=>{
  alert("in signup")
  let {id,username,password,owner} = this.state
 await this.props.registermutation({

   variables: {
     studentid : id,
           username: username,
           password : password,
           stufac:owner

   }
}).then(async(response)=>{
  alert("hi")
  console.log("res",response)
  localStorage.setItem("res",response.addUser)
  alert("hey")
 await this.setState({
   status1:"updated"
 })
 alert(response.data.addUser)

})
.catch((err)=>{
  console.log(err)
  alert(err)
  localStorage.setItem("Error",err)
  alert("in error")
  alert("sdds")
```

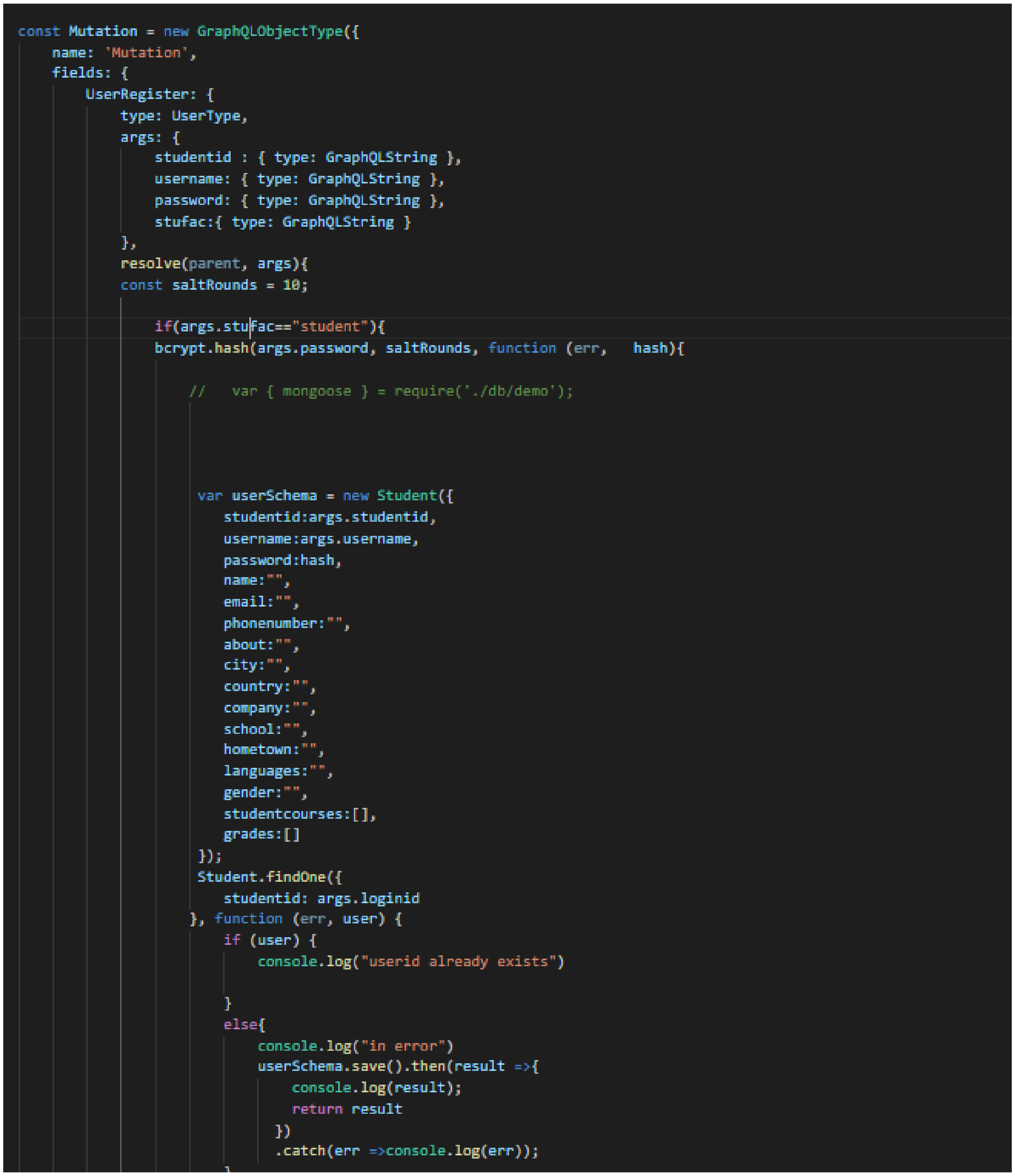


```
import { gql } from 'apollo-boost';

const registermutation = gql`

mutation UserRegister($studentid:String,$username:String, $password:String, $stufac:String){
    UserRegister(studentid:$studentid,username:$username,password:$password, stufac:$stufac){
    status
}
}

`;
```



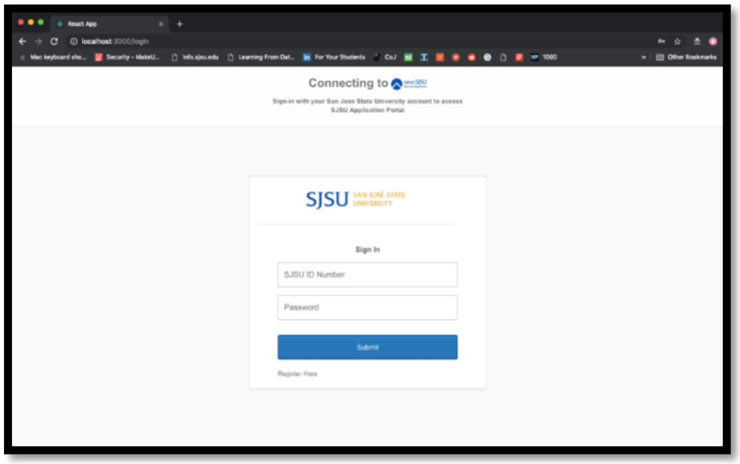
```
res ▼{data: {…}}
      ▼data:
        ▼UserRegister:
            status: 200
            __typename: "UserType"
          ▶__proto__: Object
        ▶__proto__: Object
      ▶__proto__: Object
```

Backend:

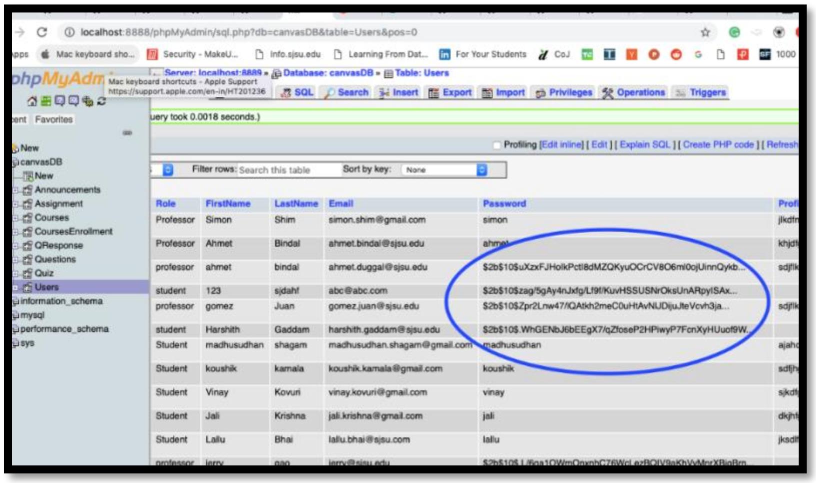


```
const Mutation = new GraphQLObjectType({
    name: 'Mutation',
    fields: {
        UserRegister: {
            type: UserType,
            args: {
                studentid : { type: GraphQLString },
                username: { type: GraphQLString },
                password: { type: GraphQLString },
                stufac:{ type: GraphQLString }
            },
            resolve(parent, args){
            const saltRounds = 10;

                if(args.stufac=="student"){
                bcrypt.hash(args.password, saltRounds, function (err,   hash){

                    //   var { mongoose } = require('./db/demo');



                     var userSchema = new Student({
                        studentid:args.studentid,
                        username:args.username,
                        password:hash,
                        name:"",
                        email:"",
                        phonenumber:"",
                        about:"",
                        city:"",
                        country:"",
                        company:"",
                        school:"",
                        hometown:"",
                        languages:"",
                        gender:"",
                        studentcourses:[],
                        grades:[]
                     });
                     Student.findOne({
                        studentid: args.loginid
                    }, function (err, user) {
                        if (user) {
                            console.log("userid already exists")

                        }
                        else{
                            console.log("in error")
                            userSchema.save().then(result =>{
                                console.log(result);
                                return result
                              })
                              .catch(err =>console.log(err));
```

Login:


React App
localhost:3000/login
Connecting to SJSU
Sign-in with your San Jose State University account to access SJSU Application Portal
SJSU SAN JOSÉ STATE UNIVERSITY
Sign In
SJSU ID Number
Password
Submit
Register Here

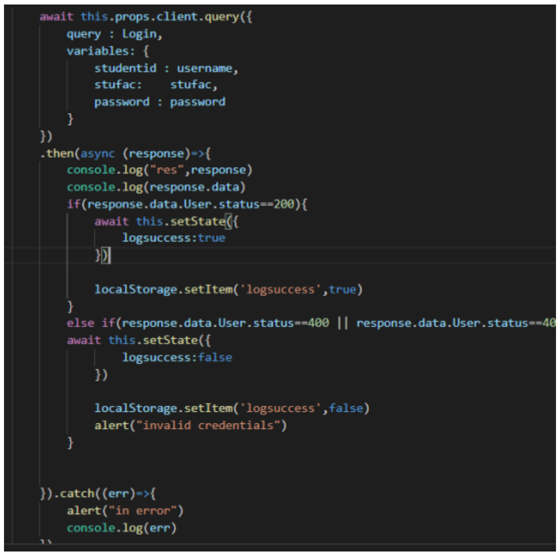
Password encryption:

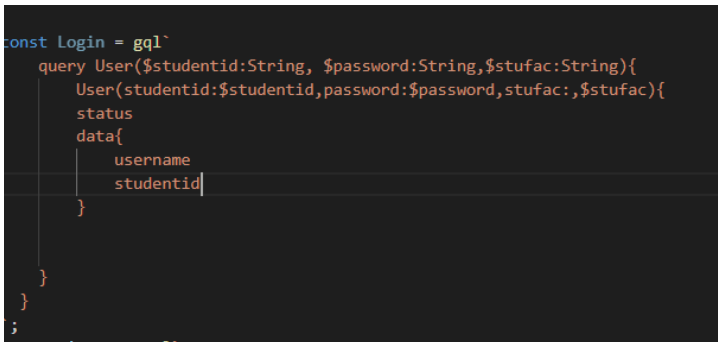

localhost:8888/phpMyAdmin/sql.php?db=canvasDB&table=Users&pos=0
Server: localhost:8889 » Database: canvasDB » Table: Users
SQL
Search
Insert
Export
Import
Privileges
Operations
Triggers
New
canvasDB
Announcements
Assignment
Courses
CoursesEnrollment
QResponse
Questions
Quiz
Users
information_schema
mysql
performance_schema
sys
Profiling [Edit inline] [Edit] [Explain SQL] [Create PHP code] [Refresh
Filter rows: Search this table
Sort by key: None

uery took 0.0018 seconds.)

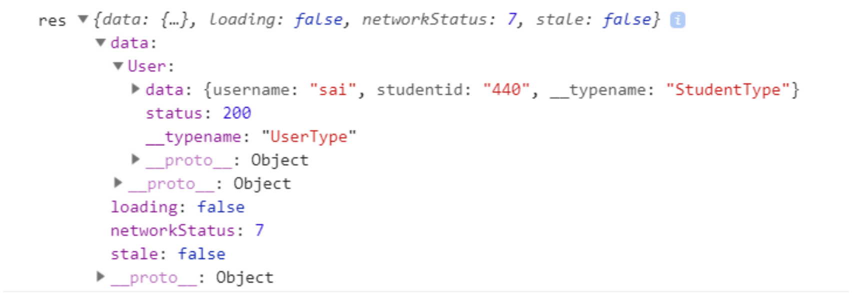
Role	FirstName	LastName	Email	Password	Profi
Professor	Simon	Shim	simon.shim@gmail.com	simon	jlkdfn
Professor	Ahmet	Bindal	ahmet.bindal@sjsu.edu	ahmet	khjdf
professor	ahmet	bindal	ahmet.duggal@sjsu.edu	$2b$10$uXzxFJHolkPctl8dMZQKyuOCrCV8O6mi0ojUinnQykb...	sdjflk
student	123	sjdahf	abc@abc.com	$2b$10$zag/5gAy4nJxfg/Lf9f/KuvHSSUSNrOksUnARpyISAx...	
professor	gomez	Juan	gomez.juan@sjsu.edu	$2b$10$Zpr2Lnw47/IQAtkh2meC0uHtAvNUDijuJteVcvh3ja...	sdjflk
student	Harshith	Gaddam	harshith.gaddam@sjsu.edu	$2b$10$.WhGENbJ6bEEgX7/qZfoseP2HPiwyP7FcnXyHUuof9W...	
Student	madhusudhan	shagam	madhusudhan.shagam@gmail.com	madhusudhan	ajahc
Student	koushik	kamala	koushik.kamala@gmail.com	koushik	sdfjh
Student	Vinay	Kovuri	vinay.kovuri@gmail.com	vinay	sjkdf
Student	Jali	Krishna	jali.krishna@gmail.com	jali	dkjhf
Student	Lallu	Bhai	lallu.bhai@sjsu.com	lallu	jksdf
professor	jerry	gao	jerry@sjsu.edu	$2b$10$.L/6qa1OWmQnxphC76WcLezBQIV9aKhVvMprXBiqBrn...	

Frontend:


```
await this.props.client.query({
    query : Login,
    variables: {
        studentid : username,
        stufac:    stufac,
        password : password
    }
})
.then(async (response)=>{
    console.log("res",response)
    console.log(response.data)
    if(response.data.User.status==200){
        await this.setState({
            logsuccess:true
        })

        localStorage.setItem('logsuccess',true)
    }
    else if(response.data.User.status==400 || response.data.User.status==40
    await this.setState({
            logsuccess:false
        })

        localStorage.setItem('logsuccess',false)
        alert("invalid credentials")
    }


}).catch((err)=>{
    alert("in error")
    console.log(err)
})
```

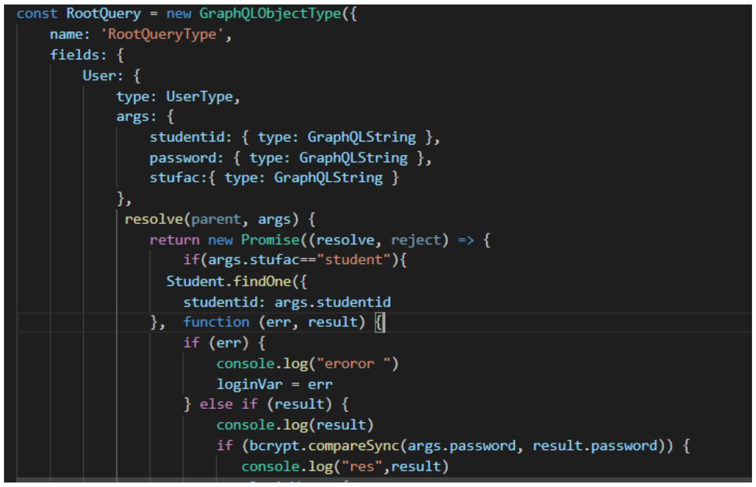


```
onst Login = gql`
    query User($studentid:String, $password:String,$stufac:String){
        User(studentid:$studentid,password:$password,stufac:,$stufac){
        status
        data{
            username
            studentid
        }


    }
  }
;
```



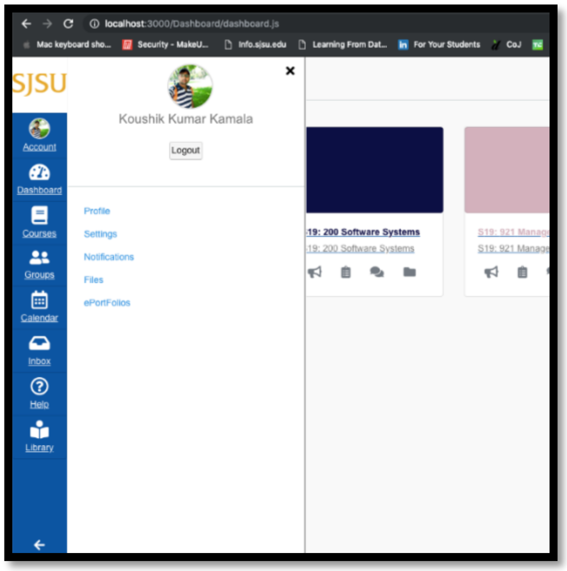
```
res ▼{data: {…}, loading: false, networkStatus: 7, stale: false}
      ▼data:
        ▼User:
          ▶data: {username: "sai", studentid: "440", __typename: "StudentType"}
           status: 200
           __typename: "UserType"
          ▶__proto__: Object
        ▶__proto__: Object
        loading: false
        networkStatus: 7
        stale: false
      ▶__proto__: Object
```

Backend:

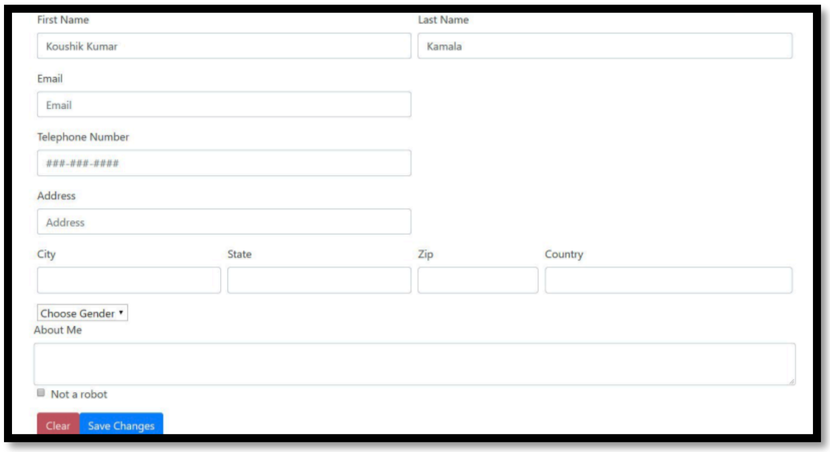


```
const RootQuery = new GraphQLObjectType({
    name: 'RootQueryType',
    fields: {
        User: {
            type: UserType,
            args: {
                studentid: { type: GraphQLString },
                password: { type: GraphQLString },
                stufac:{ type: GraphQLString }
            },
             resolve(parent, args) {
                return new Promise((resolve, reject) => {
                    if(args.stufac=="student"){
                  Student.findOne({
                    studentid: args.studentid
                }, function (err, result) {
                    if (err) {
                        console.log("eroror ")
                        loginVar = err
                    } else if (result) {
                        console.log(result)
                        if (bcrypt.compareSync(args.password, result.password)) {
                           console.log("res",result)
```

Logout:

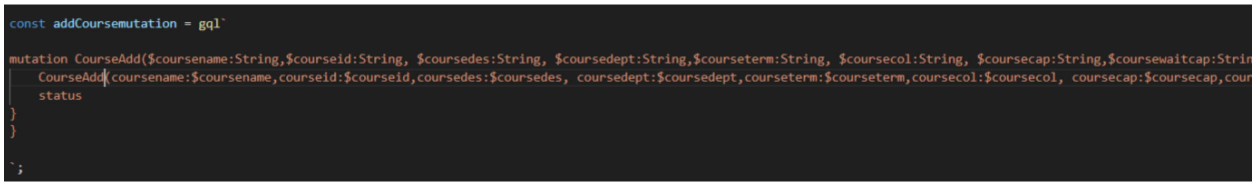


Add Course:



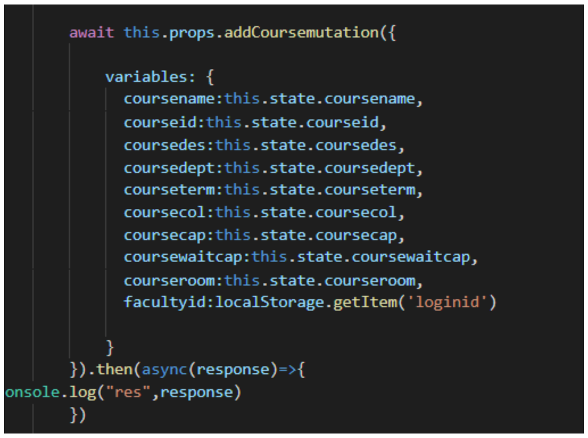
First Name: Koushik Kumar
Last Name: Kamala
Email: Email
Telephone Number: ###-###-####
Address: Address
City:
State:
Zip:
Country:
Choose Gender
About Me:
Not a robot
Clear
Save Changes

Frontend:



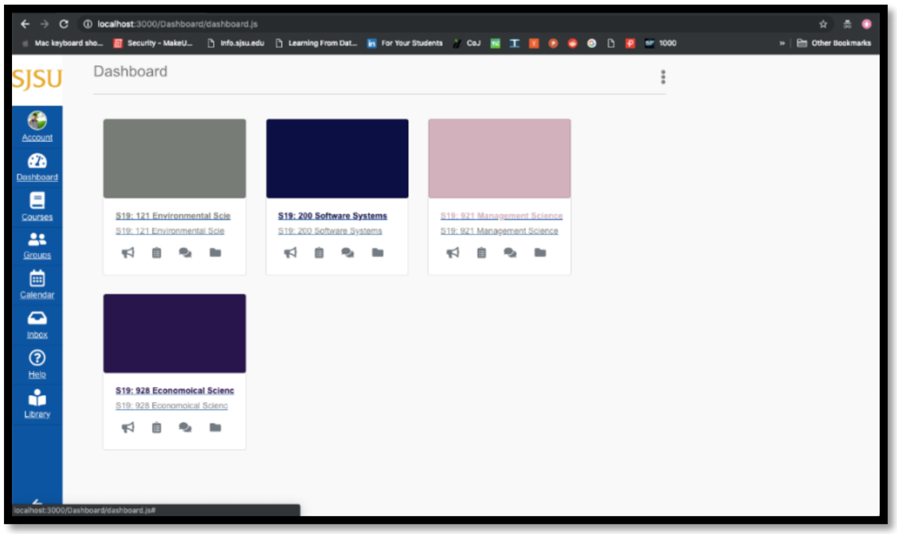
```
const addCoursemutation = gql`

mutation CourseAdd($coursename:String,$courseid:String, $coursedes:String, $coursedept:String,$courseterm:String, $coursecol:String, $coursecap:String,$coursewaitcap:Strin
    CourseAdd(coursename:$coursename,courseid:$courseid,coursedes:$coursedes, coursedept:$coursedept,courseterm:$courseterm,coursecol:$coursecol, coursecap:$coursecap,cour
    status
}
}

`;
```

```
res ▼{data: {…}} ℹ
      ▼data:
        ▶CourseAdd: {status: 200, __typename: "UserType"}
        ▶__proto__: Object
      ▶__proto__: Object
```



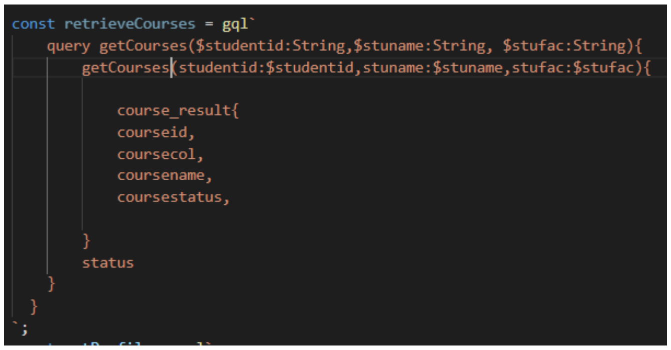
```
        await this.props.addCoursemutation({

            variables: {
              coursename:this.state.coursename,
              courseid:this.state.courseid,
              coursedes:this.state.coursedes,
              coursedept:this.state.coursedept,
              courseterm:this.state.courseterm,
              coursecol:this.state.coursecol,
              coursecap:this.state.coursecap,
              coursewaitcap:this.state.coursewaitcap,
              courseroom:this.state.courseroom,
              facultyid:localStorage.getItem('loginid')

            }
        }).then(async(response)=>{
onsole.log("res",response)
        })
```

Get courses:

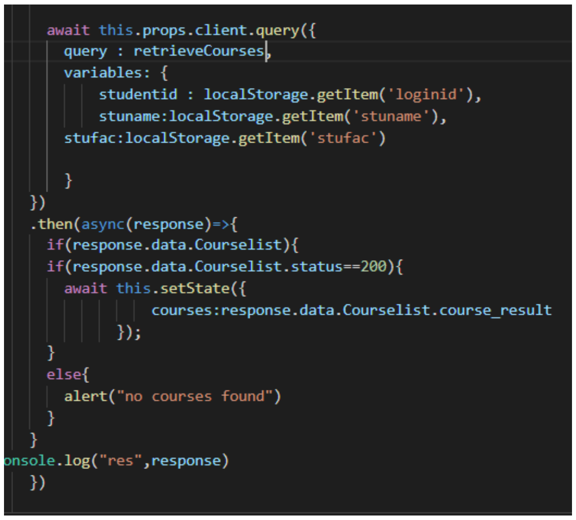


localhost:3000/Dashboard/dashboard.js
Mac keyboard sho...
Security - MakeU...
info.sjsu.edu
Learning From Dat...
For Your Students
CoJ
1000
Other Bookmarks
SJSU
Dashboard
Account
Dashboard
Courses
Groups
Calendar
Inbox
Help
Library
S19: 121 Environmental Scie
S19: 121 Environmental Scie
S19: 200 Software Systems
S19: 200 Software Systems
S19: 921 Management Science
S19: 921 Management Science
S19: 928 Economoical Scienc
S19: 928 Economoical Scienc
localhost:3000/Dashboard/dashboard.js#

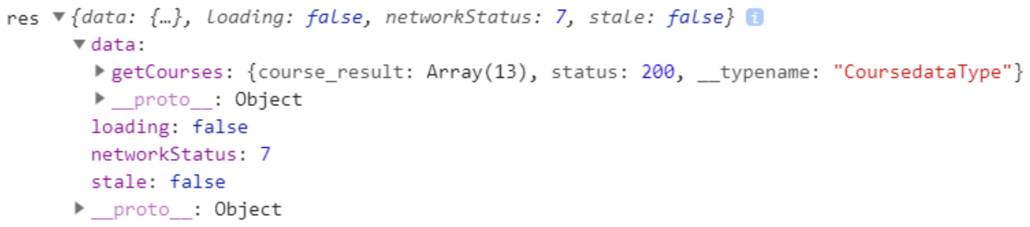
Frontend



```
const retrieveCourses = gql`
    query getCourses($studentid:String,$stuname:String, $stufac:String){
        getCourses(studentid:$studentid,stuname:$stuname,stufac:$stufac){

            course_result{
            courseid,
            coursecol,
            coursename,
            coursestatus,

        }
        status
    }
  }
`;
```

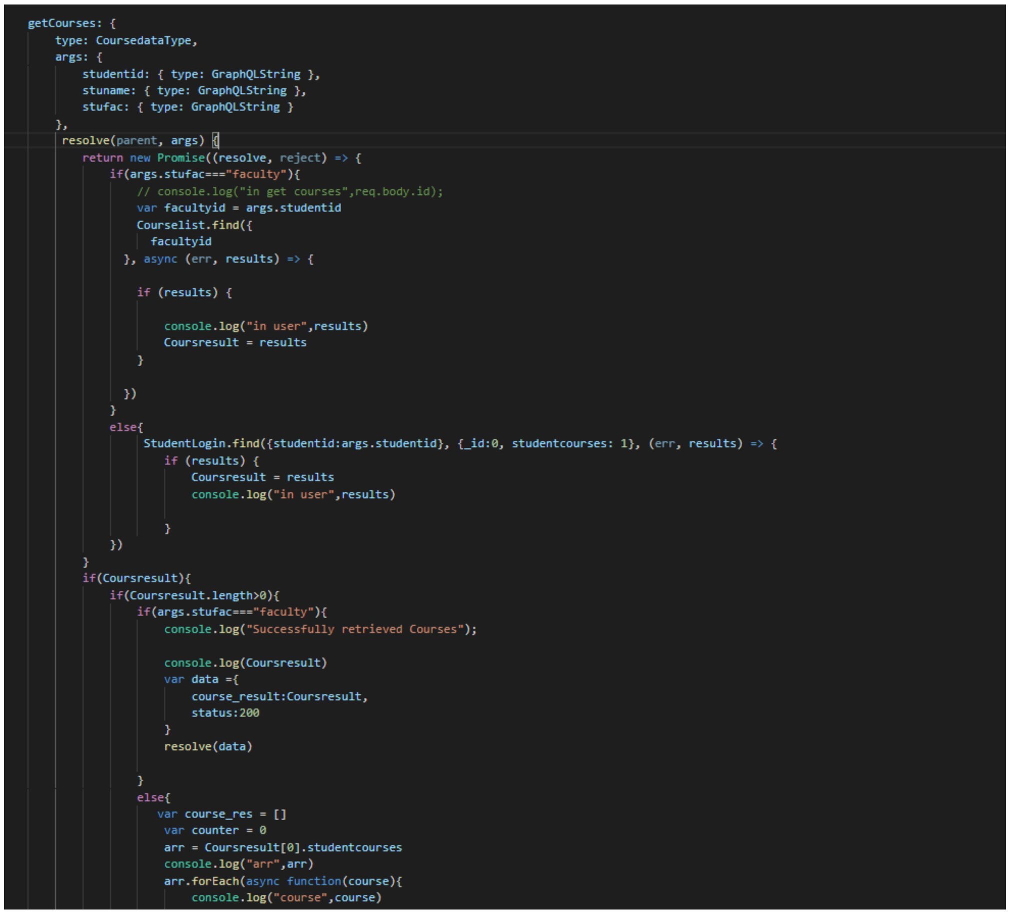


```
    await this.props.client.query({
      query : retrieveCourses,
      variables: {
          studentid : localStorage.getItem('loginid'),
          stuname:localStorage.getItem('stuname'),
      stufac:localStorage.getItem('stufac')

      }
  })
  .then(async(response)=>{
    if(response.data.Courselist){
    if(response.data.Courselist.status==200){
      await this.setState({
                courses:response.data.Courselist.course_result
            });
    }
    else{
      alert("no courses found")
    }
  }
onsole.log("res",response)
  })
```



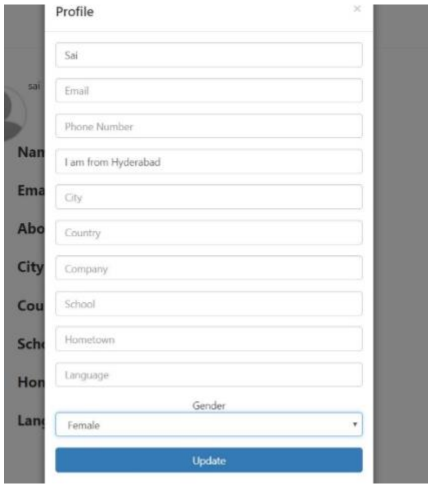
```
res ▼{data: {…}, loading: false, networkStatus: 7, stale: false}
    ▼data:
      ▶getCourses: {course_result: Array(13), status: 200, __typename: "CoursedataType"}
      ▶__proto__: Object
     loading: false
     networkStatus: 7
     stale: false
    ▶__proto__: Object
```

Backend:



```
getCourses: {
    type: CoursedataType,
    args: {
        studentid: { type: GraphQLString },
        stuname: { type: GraphQLString },
        stufac: { type: GraphQLString }
    },
     resolve(parent, args) {
        return new Promise((resolve, reject) => {
            if(args.stufac==="faculty"){
                // console.log("in get courses",req.body.id);
                var facultyid = args.studentid
                Courselist.find({
                  facultyid
              }, async (err, results) => {

                if (results) {

                    console.log("in user",results)
                    Coursresult = results
                }

              })
            }
            else{
                 StudentLogin.find({studentid:args.studentid}, {_id:0, studentcourses: 1}, (err, results) => {
                    if (results) {
                        Coursresult = results
                        console.log("in user",results)

                    }
            })
        }
        if(Coursresult){
            if(Coursresult.length>0){
                if(args.stufac==="faculty"){
                    console.log("Successfully retrieved Courses");

                    console.log(Coursresult)
                    var data ={
                        course_result:Coursresult,
                        status:200
                    }
                    resolve(data)

                }
                else{
                   var course_res = []
                    var counter = 0
                    arr = Coursresult[0].studentcourses
                    console.log("arr",arr)
                    arr.forEach(async function(course){
                        console.log("course",course)
```

Edit Profile:



Profile

Sai

Email

Phone Number

I am from Hyderabad

City

Country

Company

School

Hometown

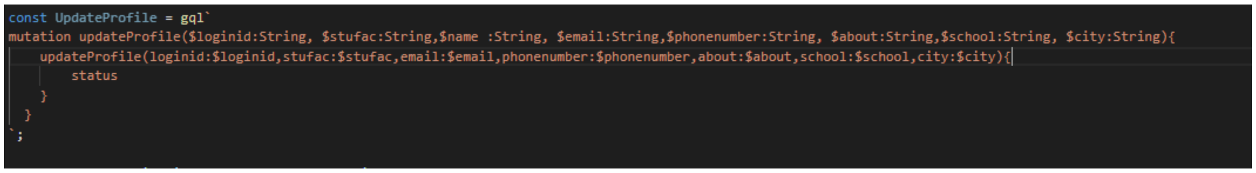
Language

Gender

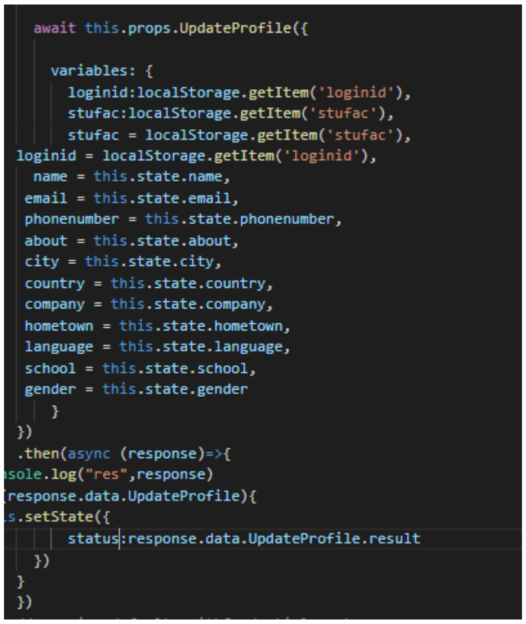
Female

Update

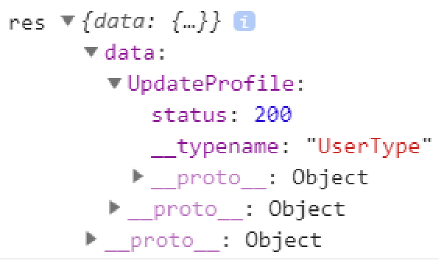
Frontend:



```
const UpdateProfile = gql`
mutation updateProfile($loginid:String, $stufac:String,$name :String, $email:String,$phonenumber:String, $about:String,$school:String, $city:String){
    updateProfile(loginid:$loginid,stufac:$stufac,email:$email,phonenumber:$phonenumber,about:$about,school:$school,city:$city){
        status
    }
  }
`;
```

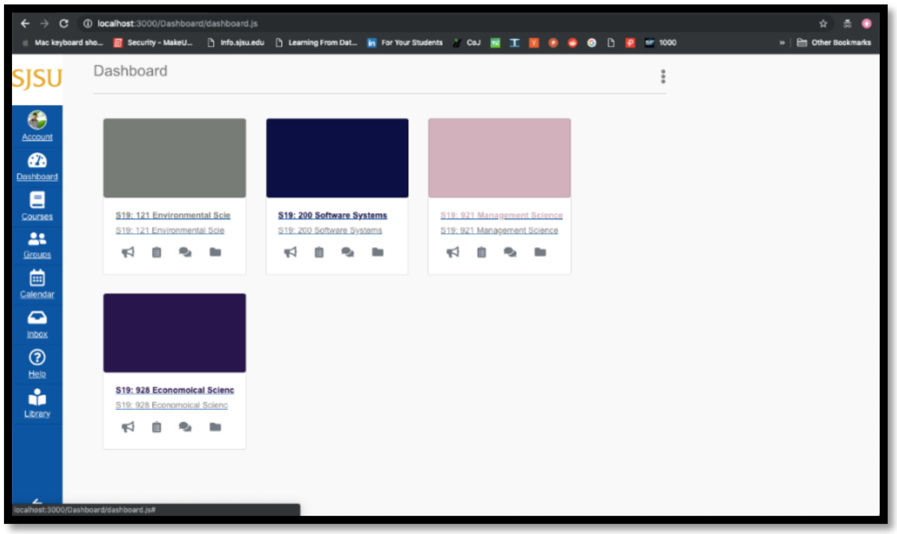


```
  await this.props.UpdateProfile({

    variables: {
      loginid:localStorage.getItem('loginid'),
      stufac:localStorage.getItem('stufac'),
      stufac = localStorage.getItem('stufac'),
loginid = localStorage.getItem('loginid'),
  name = this.state.name,
 email = this.state.email,
 phonenumber = this.state.phonenumber,
 about = this.state.about,
 city = this.state.city,
 country = this.state.country,
 company = this.state.company,
 hometown = this.state.hometown,
 language = this.state.language,
 school = this.state.school,
 gender = this.state.gender
    }
})
.then(async (response)=>{
sole.log("res",response)
response.data.UpdateProfile){
s.setState({
      status:response.data.UpdateProfile.result
  })
}
})
```

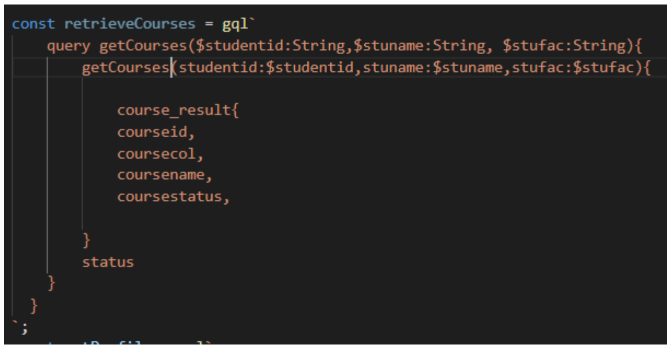


```
res ▼{data: {…}}
      ▼data:
        ▼UpdateProfile:
            status: 200
            __typename: "UserType"
          ▶ __proto__: Object
        ▶ __proto__: Object
      ▶ __proto__: Object
```

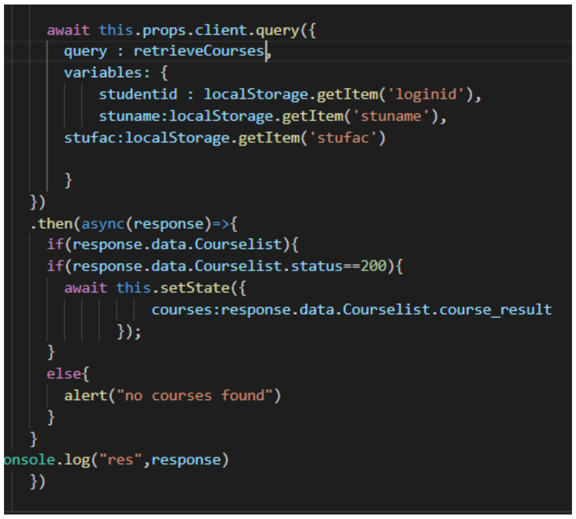
View Courses:


localhost:3000/Dashboard/dashboard.js
Mac keyboard sho...
Security - MakeU...
info.sjsu.edu
Learning From Dat...
For Your Students
CoJ
1000
Other Bookmarks
SJSU
Dashboard
Account
Dashboard
Courses
Groups
Calendar
Inbox
Help
Library
S19: 121 Environmental Scie
S19: 121 Environmental Scie
S19: 200 Software Systems
S19: 200 Software Systems
S19: 921 Management Science
S19: 921 Management Science
S19: 928 Economoical Scienc
S19: 928 Economoical Scienc
localhost:3000/Dashboard/dashboard.js#

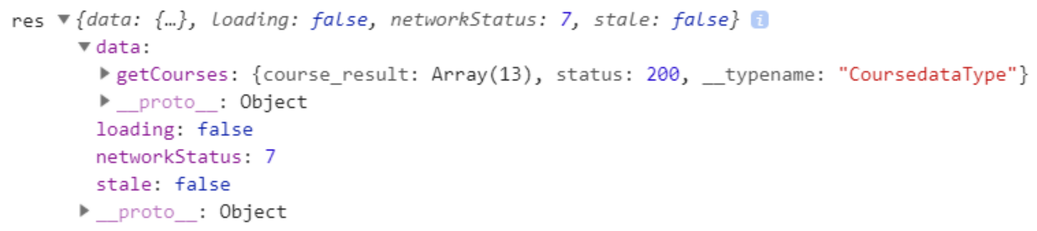
Frontend



```
const retrieveCourses = gql`
    query getCourses($studentid:String,$stuname:String, $stufac:String){
        getCourses(studentid:$studentid,stuname:$stuname,stufac:$stufac){

            course_result{
            courseid,
            coursecol,
            coursename,
            coursestatus,

        }
        status
    }
  }
`;
```

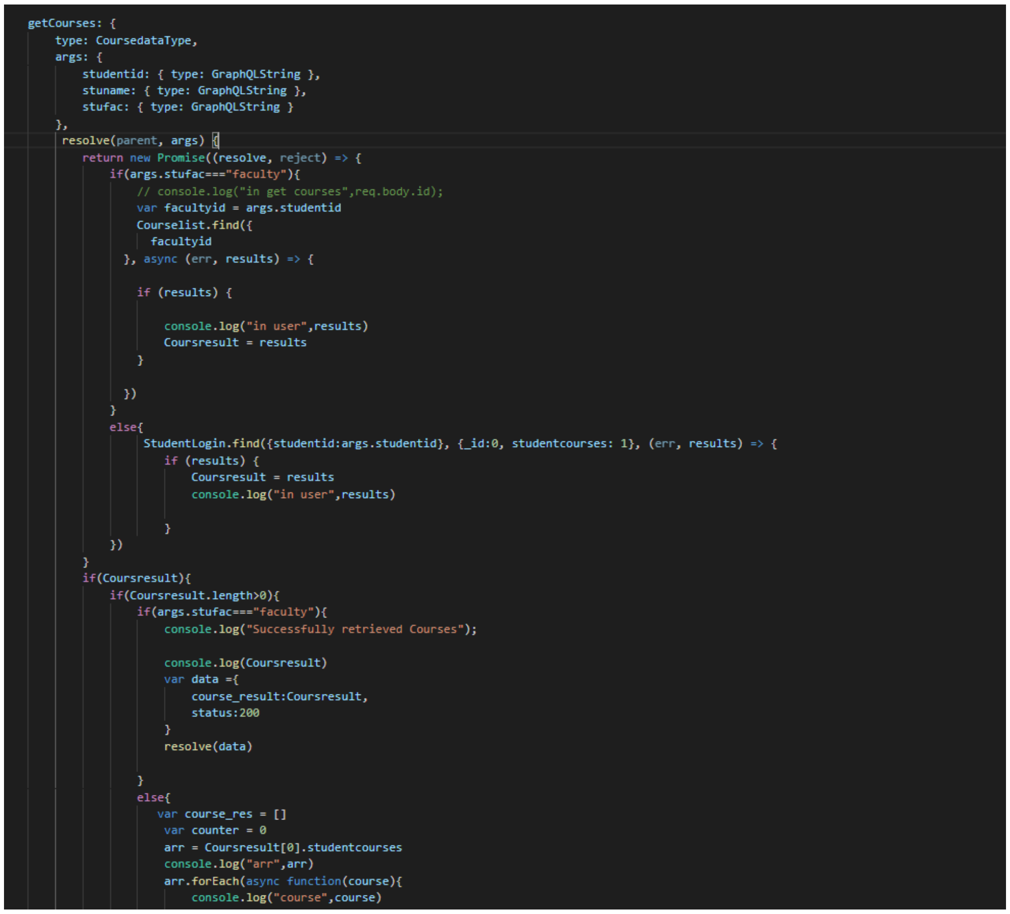


```
    await this.props.client.query({
      query : retrieveCourses,
      variables: {
          studentid : localStorage.getItem('loginid'),
          stuname:localStorage.getItem('stuname'),
      stufac:localStorage.getItem('stufac')

      }
  })
  .then(async(response)=>{
    if(response.data.Courselist){
    if(response.data.Courselist.status==200){
      await this.setState({
                courses:response.data.Courselist.course_result
              });
    }
    else{
      alert("no courses found")
    }
  }
onsole.log("res",response)
  })
```



```
res ▼{data: {…}, loading: false, networkStatus: 7, stale: false}
     ▼data:
       ▶getCourses: {course_result: Array(13), status: 200, __typename: "CoursedataType"}
       ▶__proto__: Object
      loading: false
      networkStatus: 7
      stale: false
     ▶__proto__: Object
```

Backend:



```
getCourses: {
    type: CoursedataType,
    args: {
        studentid: { type: GraphQLString },
        stuname: { type: GraphQLString },
        stufac: { type: GraphQLString }
    },
     resolve(parent, args) {
        return new Promise((resolve, reject) => {
            if(args.stufac==="faculty"){
                // console.log("in get courses",req.body.id);
                var facultyid = args.studentid
                Courselist.find({
                  facultyid
              }, async (err, results) => {

                if (results) {

                    console.log("in user",results)
                    Coursresult = results
                }

              })
            }
            else{
                 StudentLogin.find({studentid:args.studentid}, {_id:0, studentcourses: 1}, (err, results) => {
                    if (results) {
                        Coursresult = results
                        console.log("in user",results)

                    }
            })
        }
        if(Coursresult){
            if(Coursresult.length>0){
                if(args.stufac==="faculty"){
                    console.log("Successfully retrieved Courses");

                    console.log(Coursresult)
                    var data ={
                        course_result:Coursresult,
                        status:200
                    }
                    resolve(data)

                }
                else{
                   var course_res = []
                    var counter = 0
                    arr = Coursresult[0].studentcourses
                    console.log("arr",arr)
                    arr.forEach(async function(course){
                        console.log("course",course)
```

Ques & Ans:

1. Uploading files to the backend server has never been easy using basic base64 approach, as it may cause high load as each file has to independently should finish with uploading and this will occur in sequence, could cause very less performance.

   To overcome this problem, you can use Mutil-form data, which is one of the best performance model w.r.t multiple file upload. Here is follows a basic architecture.
   - Frontend filters data and maps to different keys. Basically it stores data with some hashing methods.
   - Backend server makes sure access and parse this data using these keys. Sample Mutation could be like below.

     ```
     this.props.mutate({variables: {file: yourFile}})
     ```

2. Graphql-upload would be the open source middleware, which I would prefer. I strongly support this middleware, as it can add additional feature to normal process of multi-form and can distribute the request to multiple node servers, which could essentially increase the performance.
   Sample code:

```js
import { GraphQLSchema, GraphQLObjectType, GraphQLBoolean } from 'graphql'
import { GraphQLUpload } from 'graphql-upload'

export const schema = new GraphQLSchema({
  mutation: new GraphQLObjectType({
    name: 'Mutation',
    fields: {
      uploadImage: {
        description: 'Uploads an image.',
        type: GraphQLBoolean,
        args: {
          image: {
            description: 'Image file.',
            type: GraphQLUpload
          }
        },
        async resolve(parent, { image }) {
          const { filename, mimetype, createReadStream } = await image
          const stream = createReadStream()
          // Promisify the stream and store the file, then…
          return true
        }
      }
    }
  })
})
```