*BED Assignment Documentation & Walk-through*

*P2112646*

*25/12/2021*

*DAAA/FT/1B/04*

*Justin Wong Juin Hng*

**Endpoint 1 POST /users/: Used to add a new user to the database.**

**Postman:**

A screenshot of a computer screen

Description automatically generated

**Graphical user interface, application

Description automatically generated with medium confidence**

**Code:**

//endpoint 1  
***app***.post('/users',function(req, res){  
 let username = req.body.username  
 let contact = req.body.contact  
 let password = req.body.password  
 let type = req.body.type  
 let profile\_pic\_url = req.body.profile\_pic\_url  
 ***dbFunc***.addUser(username,contact,password,type, profile\_pic\_url, function(err, result){ //call function addUser  
 if(err){  
 res.sendStatus(err)  
 }else{  
 res.status(201).send(result) //user added successfully. success code 201 passed.  
 }  
 })  
})

//Add user to userinfo table by POST method  
addUser : function(username,contact,password,type,profile\_pic\_url,callback){  
 let conn = ***db***.getConnection();  
 conn.connect(function(err){  
 if (err){  
 log('\naddUser : Error. Could not connect to SQL Server.\n'+err) //check for sql connection  
 return callback(500,null)  
 }else{  
 if(isNaN(contact)){  
 log('\naddUser : Bad Request. Input received does not match required datatype.')  
 return callback(400,null)  
 }  
 if(type=='customer'|| type=='admin'){  
 let sqlquery = 'INSERT INTO userinfo(username,contact,password,type, profile\_pic\_url) VALUES(?,?,?,?,?)'; //create new user  
 conn.query(sqlquery, [username, contact, password, type, profile\_pic\_url], function (err){  
 conn.end()  
 if (err){  
 if (err.***errno***===1062){  
 log('\naddUser : Duplicate Entry. The new username OR new email provided already exists.\n'+err); //duplicate users. 422 error code passed  
 return callback(422,null)  
 }  
 else{  
 log('\naddUser : Internal Server Error\n'+err) //internal sql error. error code 500 passed  
 return callback(500,null)  
 }  
 }else{  
 let conn = ***db***.getConnection()  
 conn.connect(function(err){  
 if (err) {  
 log('\naddUser : Error. Could not connect to SQL Server.\n' + err) //check for sql connection  
 return callback(500, null)  
 }else{  
 let sqlquery = `SELECT userid FROM userinfo where username='${username}'`  
 conn.query(sqlquery,function(err,result){  
 if(err){  
 log(err)  
 return callback(err,null)  
 }  
 else {  
 let string\_id = ***JSON***.stringify(result[0].userid)  
 log(`addUser : Connected to SQL Server: Added User ID: ${string\_id}`) //successful connection. user added  
 let sqlquery = `UPDATE userinfo SET password = MD5(password) WHERE userid =?`  
 conn.query(sqlquery,string\_id,function(err,result){  
 conn.end()  
 if (err){  
 log(err)  
 return callback(err,null)  
 }else{  
 return callback(null,`User ID: ${string\_id}`)  
 }  
 })  
 }  
 })  
 }  
 })  
 }  
 })  
 }else{  
 log('\naddUser : Bad Request. Input received mismatches required datatype.')  
 return callback(400,null)  
 }  
  
 }  
 })  
},

**Endpoint 2 GET /users/**

**Postman:A screenshot of a computer

Description automatically generated with medium confidence**

**A screenshot of a computer

Description automatically generated with medium confidence**

**Code:**

//endpoint 2  
***app***.get('/users', function(req, res){  
 ***dbFunc***.getAllUsers(function(err,result){ //call function getAllUsers  
 if(err){  
 res.sendStatus(err)  
 }else{  
 res.status(200).send(result) //task successful. success code 200 passed  
 }  
 })  
})

//Retrieve All Data within 'userinfo' Table by GET method  
getAllUsers : function (callback){  
 let conn = ***db***.getConnection();  
 conn.connect(function (err){  
 if (err){  
 log('\ngetAllUsers : Error. Could not connect to SQL Server.\n'+err); //check for sql connection  
 return callback(500,null)  
 }else{  
 let sqlquery = 'SELECT \* FROM userinfo' //retrieve all users  
 conn.query(sqlquery, function(err,result){  
 conn.end();  
 if (err){  
 log('\ngetAllUsers : Internal Server Error.\n'+err) //internal sql error. error code 500 passed  
 return callback(500,null)  
 }else{  
 log(`getAllUsers : Connected to SQL Server: Printing All Users.`) //successful connection. all users printed  
 return callback(null,result)  
 }  
 })  
 }  
 })  
  
},

**Endpoint 3 GET /users/:id/**

**Postman:**

A screenshot of a computer

Description automatically generated with medium confidence

**Graphical user interface, application, Teams

Description automatically generated**

**Graphical user interface, application, Teams

Description automatically generated**

**Code:**

// Retrieve Specific User Data by GET method  
getUser: function (id,callback){  
 let conn = ***db***.getConnection();  
 conn.connect(function(err){  
 if (err){  
 log('\ngetUser: Error. Could not connect to SQL Server.\n'+err); //internal sql error. error code 500 passed  
 return callback(500,null)  
 }  
 else{  
 if (isNaN(id)){  
 log(`\ngetUser: Not connected to SQL Server: Invalid ID (${id})\n`+err); //check if id entered in a number  
 return callback(400, null)  
 }  
 let sqlquery = 'SELECT \* FROM userinfo WHERE userid = ?' //retrieve specific user info  
 conn.query(sqlquery, [id], function (err,result){  
 conn.end();  
 if (err){  
 log('\ngetUser: Internal Server Error.\n'+err) //internal sql error. error code 500 passed  
 return callback(500,null);  
 }else{  
 if (result[0] === undefined){  
 log(`getUser: Connected to SQL Server: User ${id} not found.`); //user not found, 404 error code passed  
 return callback(404,null)  
 }  
 else{  
 log(`getUser: Connected to SQL Server: Printing User ${id}`); //successful connection. user info retrieved  
 return callback(null,result[0])  
 }  
 }  
 })  
  
  
 }  
 })  
},

**Endpoint 4 PUT /users/:id/**

**Postman:**

**A screenshot of a computer

Description automatically generated with medium confidence**

**A screenshot of a computer

Description automatically generated with medium confidence**

**A screenshot of a computer

Description automatically generated with medium confidenceA screenshot of a computer

Description automatically generated with medium confidence**

**Code:**

//endpoint 4  
***app***.put('/users/:id', function(req, res){  
 let userid = req.params.id  
 let username = req.body.username  
 let contact = req.body.contact  
 let password = req.body.password  
 let type = req.body.type  
 let profile\_pic\_url = req.body.profile\_pic\_url  
 ***dbFunc***.editUser(userid,username,contact,password,type, profile\_pic\_url,function(err,result){ //call function editUser  
 if (err){  
 res.sendStatus(err); //user edit failed. error code passed.  
 }else {  
 res.sendStatus(204) //user edit successful. success code 204 passed.  
 }  
 })  
})

//Inserts new user info into existing entries within userinfo database using PUT method  
editUser : function(id, username,contact,password,type,profile\_pic\_url,callback){  
 let conn = ***db***.getConnection();  
 conn.connect(function(err){  
 if (err){  
 log('\neditUser : Error. Could not connect to SQL Server.\n'+err) //check for sql connection  
 return callback(500,null)  
 }  
 else{  
 if(isNaN(contact)){  
 log('\neditUser : Bad Request. Input received mismatches required datatype.')  
 return callback(400,null)  
 }  
 if(type=='customer'|| type=='admin'){  
 let sqlquery = 'SELECT \* FROM userinfo WHERE userid = ?'  
 conn.query(sqlquery, [id], function (err,result){  
 if (err){  
 log('\neditUser : Internal Server Error.\n'+err) //internal sql error. error code 500 passed  
 return callback(500,null);  
 } else{  
 if (result[0] === undefined){  
 log(`editUser : Connected to SQL Server: User ${id} not found.`); //404 error code passed. user not found  
 return callback(404,null)  
 }  
 else{  
 if(((result[0].username)==username||(result[0].profile\_pic\_url)==profile\_pic\_url)){ //check for duplicate information  
 log('\neditUser : Duplicate Entry. The new username OR new email provided already exists.'); //422 error code. duplicate entries found  
 return callback(422,null)  
 }  
 let sqlquery = `UPDATE userinfo SET username='${username}',contact='${contact}',password='${password}',type='${type}',profile\_pic\_url='${profile\_pic\_url}' WHERE userid=${id}` //update user info  
 conn.query(sqlquery, [username,contact,password,type,profile\_pic\_url],function(err,result){  
 conn.end();  
 if (err) {  
 if (err.***errno*** === 1062) { //check if duplicate error  
 log('\neditUser : Duplicate Entry. The new username OR new email provided already exists.\n'+err); //duplicate error found. 422 error code passed  
 return callback(422, null)  
 } else {  
 log('\neditUser : Internal Server Error\n'+err) //internal sql error. error code 500 passed  
 return callback(500, null)  
 }  
 } else{  
 return callback(null,result) //user edit successful.  
 }  
 })  
 }  
 }  
 })  
 }else{  
 log('\neditUser : Bad Request. Input received mismatches required datatype.')  
 return callback(400,null)  
 }  
  
 }  
 })  
},

**Endpoint 5 POST /category**

**Postman:**

**A screenshot of a computer

Description automatically generated with medium confidence**

**Code:**

//endpoint 5  
***app***.post('/category', function(req, res){  
 let category = req.body.Category  
 let description = req.body.Description  
 ***dbFunc***.addCategory(category,description,function(err,result){  
 if (err){  
 res.sendStatus(err)  
 }else{  
 res.status(204).send(result)  
 }  
 })  
})

addCategory : function(category, description, callback){  
 let conn = ***db***.getConnection();  
 conn.connect(function(err){  
 if (err){  
 log('\naddCategory: Internal Server Error\n'+err)  
 return callback(500,null)  
 }else {  
 if (isNaN(description)) {  
 let sqlquery = 'INSERT INTO product\_categories (category, description) VALUES (?,?)'  
 conn.query(sqlquery, [category, description], function (err, result) {  
 if (err) {  
 if (err.***errno*** === 1062) { //check if duplicate error  
 log('\naddCategory : Duplicate Entry. The new username OR new email provided already exists.\n'+err); //duplicate error found. 422 error code passed  
 return callback(422, null)  
 } else {  
 log('\naddCategory : Internal Server Error\n'+err) //internal sql error. error code 500 passed  
 return callback(500, null)  
 }  
 } else {  
 log('addCategory : Product Category Added.')  
 return callback(null, result)  
 }  
 })  
 }else{  
 log('\naddCategory : Bad Request. Input received mismatches required datatype.')  
 return callback(400, null)  
 }  
 }  
 })  
},

**Endpoint 6 GET /category:**

**Postman:**

**Text

Description automatically generated**

**Code:**

//endpoint 6  
***app***.get('/category',function(req,res){  
 ***dbFunc***.getAllCategories(function(err,result){  
 if(err){  
 res.sendStatus(err)  
 }else{  
 res.status(200).send(result)  
 }  
 })  
  
})

getAllCategories : function(callback){  
 let conn = ***db***.getConnection()  
 conn.connect(function(err){  
 if(err){  
 log('\getAllCategories : Internal Server Error\n'+err)  
 return callback(500,null)  
 }else{  
 let sqlquery = 'SELECT \* FROM product\_categories'  
 conn.query(sqlquery,function(err,result){  
 if(err){  
 log('\ngetAllCategories : Internal Server Error.\n'+err) //internal sql error. error code 500 passed  
 return callback(500,null)  
 }else{  
 log('\ngetAllCategories : Connected to SQL Server. Printing All Product Categories')  
 return callback(null,result)  
 }  
 })  
 }  
 })  
},

**Endpoint 7 POST /product/**

**Postman:**

**A screenshot of a computer screen

Description automatically generated**

**A screenshot of a computer screen

Description automatically generated with medium confidence**

**Code:**

//endpoint 7  
***app***.post('/product',upload.single('file'),function(req,res){  
 let name = req.body.name  
 let description = req.body.description  
 let category\_id = req.body.category\_id  
 let brand = req.body.brand  
 let price = req.body.price  
 let img\_name = req.file.filename  
 fs.stat(`uploads/${img\_name}`, (err, stats) => {  
 if (err) {  
 ***console***.log(`File doesn't exist.`)  
 res.sendStatus(500)  
 } else {  
 if (stats.size<=1000000){  
 ***console***.log(stats.size)  
 ***dbFunc***.addProduct(name,description,category\_id,brand,price,img\_name,function(err,result){  
 if(err){  
 res.sendStatus(err)  
 }else{  
 res.status(201).send(result)  
 }  
 });  
 }  
 else{  
 ***console***.log('addProduct : File size too big.')  
 res.sendStatus(500)  
 }  
 }  
 });  
})

addProduct : function(name,description,category\_id,brand,price,img\_src,callback){  
 let conn = ***db***.getConnection();  
 conn.connect(function(err){  
 if (err){  
 log('\naddProduct : Error. Could not connect to SQL Server.\n'+err) //check for sql connection  
 return callback(500,null)  
 }else{  
 if(isNaN(category\_id)){  
 log('\naddProduct : Bad Request. Input received does not match required datatype.')  
 return callback(400,null)  
 }  
 if(!isNaN(price)){  
 let sqlquery = 'INSERT INTO products(name, description, category\_id, brand, price,product\_image\_src) VALUES(?,?,?,?,?,?)'; //create new user  
 conn.query(sqlquery, [name, description, category\_id, brand, price,img\_src], function (err){  
  
 if (err){  
 if (err.***errno***===1062){  
 log('\naddProduct : Duplicate Entry. The new name provided already exists.\n'+err); //duplicate users. 422 error code passed  
 return callback(422,null)  
 }  
 else{  
 log('\naddProduct : Internal Server Error\n'+err) //internal sql error. error code 500 passed  
 return callback(500,null)  
 }  
 }else{  
 let conn = ***db***.getConnection()  
 conn.connect(function(err){  
 if (err) {  
 log('\naddProduct : Error. Could not connect to SQL Server.\n' + err) //check for sql connection  
 return callback(500, null)  
 }else{  
 let sqlquery = `SELECT product\_id FROM products WHERE name='${name}'`  
 conn.query(sqlquery,function(err,result){  
 conn.end()  
 if(err){  
 log(err)  
 return callback(err,null)  
 }  
 else {  
 let string\_id = ***JSON***.stringify(result[0].product\_id)  
 log(`\naddProduct : Connected to SQL Server: Added Product ID: ${string\_id}`) //successful connection. user added  
 return callback(null, result)  
 }  
 })  
 }  
 })  
  
 }  
 })  
 }else{  
 log('\naddUser : Bad Request. Input received mismatches required datatype.')  
 return callback(400,null)  
 }  
  
 }  
 })  
},

**Endpoint 8: GET /product/:id**

**Postman:**

**A screenshot of a computer

Description automatically generated with medium confidence**

**A hand holding a remote control

Description automatically generated with low confidence**

**A screenshot of a computer

Description automatically generated with medium confidence**

**Code:**

//endpoint 8 GET /product/:id  
***app***.get('/product/:id', function(req,res){  
 let id = req.params.id  
 ***dbFunc***.getProduct(id,function(err,result){  
 if(err){  
 res.sendStatus(err)  
 }else{  
 res.status(200).render('index', { title: 'Hey', src: `http://localhost:8001/${result[0].product\_image\_src}`,result})  
  
  
 }  
 })  
})

getProduct : function(id,callback){  
 let conn = ***db***.getConnection()  
 conn.connect(function(err){  
 if(err){  
 log('\ngetProduct : Error. Could not connect to SQL Server.\n'+err)  
 return callback(500,null)  
 }else{  
 if(!isNaN(id)){  
 let sqlquery1 = 'CREATE TEMPORARY TABLE temp\_table AS SELECT products.name, products.description, products.category\_id,products.price,products.product\_id,products.brand,products.product\_image\_src,product\_categories.category FROM products INNER JOIN product\_categories ON products.category\_id=product\_categories.category\_id'  
 conn.query(sqlquery1,[id],function(err){  
 if (err){  
 log('\ngetProduct : Internal Server Error.\n'+err) //internal sql error. error code 500 passed  
 return callback(500,null)  
 }else{  
 let sqlquery2 = 'SELECT name,description,category\_id,category,brand,price,product\_image\_src FROM temp\_table WHERE product\_id=?'  
 conn.query(sqlquery2,[id],function(err,result){  
 conn.end()  
 if (err){  
 log('\ngetProduct : Internal Server Error.\n'+err) //internal sql error. error code 500 passed  
 return callback(500,null)  
 }else{  
 if(result[0]===undefined){  
 log(`\ngetProduct : Bad Request. Product ${id} not found.`)  
 return callback(404,null)  
 }else{  
 result[0].price = result[0].price.toFixed(2)  
 log(`\ngetProduct : Connected to SQL Server: Product ${id}'s Information.`) //successful connection. all users printed  
 return callback(null,result)  
 }  
 }  
 })  
 }  
 })  
 }else{  
 log('\ngetProduct : Bad Request. Input received mismatches required datatype.')  
 return callback(400,null)  
 }  
 }  
 })  
},

**Endpoint 9** **DELETE /product/:id/**

**Postman:**

**Graphical user interface, application, Teams

Description automatically generated**

**Code:**

//endpoint 9 DELETE /product/:id  
***app***.delete('/product/:id', function(req,res){  
 let id = req.params.id  
  
 ***dbFunc***.deleteProduct(id,function(err,result){  
 if(err){  
 res.send(err)  
 }else{  
 res.status(204).send(result)  
 }  
 })  
})

deleteProduct: function (id,callback){  
 let conn = ***db***.getConnection();  
 conn.connect(function(err){  
 if (err){  
 log('\ndeleteProduct: Error. Could not connect to SQL Server.\n'+err); //internal sql error. error code 500 passed  
 return callback(500,null)  
 }  
 else{  
 if (isNaN(id)){  
 log(`\ndeleteProduct: Not connected to SQL Server: Invalid ID (${id})\n`+err); //check if id entered in a number  
 return callback(400, null)  
 }  
 let sqlquery = 'DELETE FROM products where product\_id=?' //retrieve specific user info  
 conn.query(sqlquery, [id], function (err,result){  
 conn.end();  
 if (err){  
 log('\ndeleteProduct: Internal Server Error.\n'+err) //internal sql error. error code 500 passed  
 return callback(500,null);  
 }else{  
 log(`deleteProduct: Connected to SQL Server: Deleting Product ${id}`); //successful connection. user info retrieved  
 return callback(null,result[0])  
  
 }  
 })  
  
  
 }  
 })  
},

**Endpoint 10 POST /product/:id/review/**

**Postman:**

**Graphical user interface, application

Description automatically generated**

**Graphical user interface, application

Description automatically generated**

**Code:**

//endpoint 10 POST /product/:id/review/  
***app***.post('/product/:id/review', function(req,res){  
 let userid = req.body.user\_id  
 let product\_id = req.params.id  
 let rating = req.body.rating  
 let review = req.body.review  
 ***dbFunc***.addReview(userid,product\_id,rating,review,function(err,result){  
 if(err){  
 res.send(err)  
 }else{  
 res.status(200).send(result)  
 }  
 })  
})

addReview : function(userid,product\_id,rating,review,callback){  
 let conn = ***db***.getConnection();  
 conn.connect(function(err){  
 if (err){  
 log('\naddReview : Error. Could not connect to SQL Server.\n'+err) //check for sql connection  
 return callback(500,null)  
 }else{  
 if(isNaN(userid)||isNaN(product\_id)||isNaN(rating)){  
 ***console***.log(userid, product\_id, rating)  
 log('\naddReview : Bad Request. Input received does not match required datatype.')  
 return callback(400,null)  
 }  
 if(isNaN(review)){  
 let sqlquery = 'INSERT INTO product\_reviews(userid, product\_id, rating, review) VALUES(?,?,?,?)'; //create new user  
 conn.query(sqlquery, [userid,product\_id,rating,review], function (err){  
 conn.end()  
 if (err){  
 if (err.***errno***===1062){  
 log('\naddReview : Duplicate Entry. The provided input already exists.\n'+err); //duplicate users. 422 error code passed  
 return callback(422,null)  
 }  
 else{  
 log('\naddReview : Internal Server Error\n'+err) //internal sql error. error code 500 passed  
 return callback(500,null)  
 }  
 }else{  
 let conn = ***db***.getConnection()  
 conn.connect(function(err){  
 if (err) {  
 log('\naddReview : Error. Could not connect to SQL Server.\n' + err) //check for sql connection  
 return callback(500, null)  
 }else{  
 let sqlquery = `SELECT reviewid FROM product\_reviews WHERE userid='${userid}' AND product\_id='${product\_id}' AND rating='${rating}' AND review='${review}'`  
 conn.query(sqlquery,function(err,result){  
 conn.end()  
 if(err){  
 log(err)  
 return callback(err,null)  
 }  
 else {  
 let string\_id = ***JSON***.stringify(result[0].reviewid)  
 log(`\naddReview : Connected to SQL Server: Added Product ID: ${string\_id}`) //successful connection. user added  
 return callback(null, result)  
 }  
 })  
 }  
 })  
  
 }  
 })  
 }else{  
 log('\naddReview : Bad Request. Input received mismatches required datatype.')  
 return callback(400,null)  
 }  
  
 }  
 })  
},

**Endpoint 11 GET /product/:id/reviews**

**Postman:**

**A screenshot of a computer screen

Description automatically generated with medium confidence**

**Graphical user interface, application

Description automatically generated**

**Code:**

//endpoint 11 GET /product/:id/reviews  
***app***.get('/product/:id/reviews', function(req,res){  
 let productid = req.params.id  
 ***dbFunc***.getSpecificReview(productid,function(err,result){  
 if(err){  
 res.send(err)  
 }else{  
 res.status(200).send(result)  
 }  
 })  
})

getSpecificReview : function (productid,callback){  
 let conn = ***db***.getConnection();  
 conn.connect(function(err){  
 if (err){  
 log('\ngetSpecificReview: Error. Could not connect to SQL Server.\n'+err); //internal sql error. error code 500 passed  
 return callback(500,null)  
 }  
 else{  
 if (isNaN(productid)){  
 log(`\ngetSpecificReview: Not connected to SQL Server: Invalid ID (${productid})\n`+err); //check if reviewid entered in a number  
 return callback(400, null)  
 }  
 let sqlquery = 'SELECT product\_id, product\_reviews.userid, username,rating,review,product\_reviews.created\_at FROM product\_reviews JOIN userinfo on product\_reviews.userid = userinfo.userid WHERE product\_id = ?' //retrieve specific user info  
 conn.query(sqlquery, [productid], function (err,result){  
 conn.end();  
 if (err){  
 log('\ngetSpecificReview: Internal Server Error.\n'+err) //internal sql error. error code 500 passed  
 return callback(500,null);  
 }else{  
 if (result[0] === undefined){  
 log(`\ngetSpecificReview: Connected to SQL Server: Product ${productid} not found.`); //user not found, 404 error code passed  
 return callback(404,null)  
 }  
 else{  
 log(`\ngetSpecificReview: Connected to SQL Server: Printing Review for Product ${productid}`); //successful connection. user info retrieved  
 return callback(null,result)  
 }  
 }  
 })  
 }  
 })  
},

**Endpoint 12: POST /interest/:userid**

**Postman:**

**Graphical user interface, application

Description automatically generated** **Graphical user interface, application

Description automatically generated** A screenshot of a computer

Description automatically generated with medium confidence

**Code:**

//endpoint 12 POST /interest/:userid  
***app***.post('/interest/:userid', function(req,res){  
 let id = req.params.userid  
 let categoryids = req.body.categoryids.split(',')  
 ***dbFunc***.createInterest(id,categoryids,function(err){  
 if(err){  
 res.sendStatus(err)  
 }else{  
 res.sendStatus(201)  
 }  
 })  
})

createInterest : function (id,categoryids,callback){  
 let errorCheck = false  
 let conn = ***db***.getConnection();  
 conn.connect(function(err){  
 if (err){  
 log('\createInterest: Error. Could not connect to SQL Server.\n'+err); //internal sql error. error code 500 passed  
 return callback(500,null)  
 }  
 else{  
 if (isNaN(id)){  
 log(`\createInterest: Not connected to SQL Server: Invalid ID (${id})\n`+err); //check if id entered in a number  
 return callback(400, null)  
 }  
 for (let i=0; i<categoryids.length;i++){  
 let sqlquery = 'INSERT INTO `bed\_assignment1`.`userinfo\_product\_categories\_junction` (`user\_id`, `category\_id`) VALUES (?,?);' //retrieve specific user info  
 conn.query(sqlquery, [id,categoryids[i]], function (err){  
 if (err){  
 log('\createInterest: Internal Server Error.\n'+err)//internal sql error. error code 500 passed  
 errorCheck = true  
 return callback(500,null);  
 }else{  
 log(`\createInterest: Connected to SQL Server: Added Interests For User ${id} @ Category ${categoryids[i]}`); //successful connection. user info retrieved  
 }  
  
 })  
 }  
 if (errorCheck == false){  
 return callback(null,null)  
 }  
 }  
 })  
},

**Endpoint 13: TRUNCATE TABLE /users/delete**

**Postman:**

**Graphical user interface, text, application

Description automatically generated**

**Code:**

//endpoint 13  
***app***.delete('/users/delete', function(req,res){  
 let table = req.body.table  
 ***dbFunc***.deleteTable(table,function(err,result){  
 if (err)  
 {  
 res.sendStatus(err)  
 }  
 else{  
 switch (table) {  
 case 'userinfo':  
 ***dbFunc***.getAllUsers(function (err, result) {  
 if (err) {  
 res.status(500).send('Unable delete table. Server Error.')  
 } else {  
 res.status(200).send(result) //user deleted successfully. success code 200 passed  
 }  
 })  
 break;  
 case 'products':  
 ***dbFunc***.getAllProducts(function(err,result){  
 if(err){  
 res.sendStatus(err).send('Unable delete table. Server Error.')  
 }else{  
 res.status(200).send(result)  
 }  
 })  
 break;  
 case 'product\_categories':  
 ***dbFunc***.getAllCategories(function (err,result){  
 if(err){  
 res.status(500).send('Unable delete table. Server Error.')  
 }else{  
 res.status(200).send(result)  
 }  
 })  
 break;  
 case 'product\_reviews':  
 ***dbFunc***.getAllReviews(function(err,result){  
 if (err){  
 res.send(err)  
 }  
 else{  
 res.status(200).send(result)  
 }  
 })  
 break;  
 default:  
 res.sendStatus(500)  
 }  
  
 }  
 })  
})

**Endpoint 14: DELETE /users/delete/:id**

**Postman:**

**A screenshot of a computer

Description automatically generated with medium confidence**

**Code:**

//endpoint 14  
***app***.delete('/users/delete/:id', function(req, res){  
 let userid = req.params.id  
 ***dbFunc***.deleteUser(userid, function(err,result){  
 if (err){  
 res.sendStatus(err)  
 }else{  
 ***dbFunc***.getAllUsers(function(err,result){  
 if(err){  
 res.status(500).send('Unable to retrieve all users. Server Error.')  
 }else{  
 res.status(200).send(result) //specific user deleted. success code 200 passed  
 }  
 })  
 }  
 }  
 )  
})

//delete specific user  
deleteUser : function(id,callback){  
 let conn = ***db***.getConnection();  
 conn.connect(function(err){  
 if (err){  
 log('\ndeleteUser : Error. Could not connect to SQL Server.\n'+err) //check for sql connection  
 return callback(500,null)  
 }else{  
 if (isNaN(id)){  
 log(`\ndeleteUser : Not connected to SQL Server: Invalid ID (${id})\n`+err); //check if id entered in a number  
 return callback(400, null)  
 }  
 let sqlquery = 'SELECT \* FROM userinfo WHERE userid = ?' //sql query  
 conn.query(sqlquery, [id], function (err,result){  
 if (err){  
 log('\ndeleteUser : Internal Server Error.\n'+err) //internal sql error. error code 500 passed.  
 return callback(500,null);  
 } else{  
 if (result[0] === undefined){  
 log(`deleteUser : Connected to SQL Server: User ${id} not found.`); //user not found. 404 error code passed  
 return callback(404,null)  
 }  
 else{  
 let sqlquery = 'DELETE FROM userinfo WHERE userid = ?' //sql query to delete user  
 conn.query(sqlquery, [id], function(err, result){  
 conn.end();  
 if (err){  
 log('\ndeleteUser : Internal Server Error.\n'+err) //internal sql error. error code 500 passed  
 return callback(500,null)  
 }else{  
 log(`deleteUser : Connected to SQL Server: User ${id} deleted.`)  
 return callback(null,result) // user deleted successfully.  
 }  
 })  
 }  
 }  
 })  
 }  
 })  
},

**Endpoint 15 GET /product**

**Postman:**

**A screenshot of a computer

Description automatically generated with medium confidence**

**Code:**

//endpoint 15  
***app***.get('/product', function(req,res){  
 ***dbFunc***.getAllProducts(function (err,result){  
 if(err){  
 res.sendStatus(err)  
 }else{  
 res.status(200).send(result)  
 }  
 })  
})

getAllProducts : function(callback){  
 let conn = ***db***.getConnection();  
 conn.connect(function (err){  
 if (err){  
 log('\ngetAllProducts : Error. Could not connect to SQL Server.\n'+err); //check for sql connection  
 return callback(500,null)  
 }else{  
 let sqlquery = 'SELECT \* FROM products' //retrieve all users  
 conn.query(sqlquery, function(err,result){  
 conn.end();  
 if (err){  
 log('\ngetAllProducts : Internal Server Error.\n'+err) //internal sql error. error code 500 passed  
 return callback(500,null)  
 }else{  
 log(`getAllProducts : Connected to SQL Server: Printing All Products.`) //successful connection. all users printed  
 return callback(null,result)  
 }  
 })  
 }  
 })  
},

**Endpoint 16 GET /category/:id**

**Postman:**

**A screenshot of a computer

Description automatically generated with medium confidence**

**Code:**

//endpoint 16  
***app***.get('/category/:id', function(req,res){  
 let id = req.params.id  
 ***dbFunc***.getAlProductsWithinCategory(id,function(err,result){  
 if(err){  
 res.sendStatus(err)  
 }else{  
 res.status(200).send(result)  
 }  
 })  
})

getAlProductsWithinCategory : function(id,callback){  
 let conn = ***db***.getConnection()  
 conn.connect(function(err){  
 if(err){  
 log('\ngetAlProductsWithinCategory : Error. Could not connect to SQL Server.\n'+err)  
 return callback(500,null)  
 }else{  
 if(!isNaN(id)){  
 let sqlquery1 = 'CREATE TEMPORARY TABLE temp\_products AS SELECT \* FROM products;'  
 conn.query(sqlquery1,[id],function(err){  
 if (err){  
 log('\ngetAlProductsWithinCategory : Internal Server Error.\n'+err) //internal sql error. error code 500 passed  
 }else{  
 let sqlquery2 = 'SELECT category\_id,product\_id,name,description,price,brand FROM temp\_products WHERE category\_id=?'  
 conn.query(sqlquery2,[id],function(err,result){  
 conn.end()  
 if(err){  
 log('\ngetAllProductsWithinCategory : Internal Server Error.\n'+err) //internal sql error. error code 500 passed  
 return callback(500,null)  
 }else {  
 if (result === undefined) {  
 log(`\ngetAllProductsWithinCategory : Bad Request. Product Category ${id} not found.`)  
 return callback(400, null)  
 } else {  
 log(`ngetAllProductsWithinCategory : Connected to SQL Server: Category ${id}'s Information.`) //successful connection. all users printed  
 return callback(null, result)  
 }  
 }  
 })  
  
 }  
 })  
 }else{  
 log('\ngetAllProductsWithinCategory : Bad Request. Input received mismatches required datatype.')  
 return callback(400,null)  
 }  
 }  
 })  
},

**Endpoint 17 GET /products/reviews**

**Postman:**

**Graphical user interface, text

Description automatically generated with medium confidence**

**Code:**

getAllReviews : function (callback){  
 let conn = ***db***.getConnection();  
 conn.connect(function(err){  
 if (err){  
 log('\ngetAllReviews: Error. Could not connect to SQL Server.\n'+err); //internal sql error. error code 500 passed  
 return callback(500,null)  
 }  
 else{  
 let sqlquery = 'SELECT \* FROM product\_reviews' //retrieve specific user info  
 conn.query(sqlquery, function (err,result){  
 conn.end();  
 if (err){  
 log('\ngetAllReviews: Internal Server Error.\n'+err) //internal sql error. error code 500 passed  
 return callback(500,null);  
 }else{  
 log(`\ngetAllReviews: Connected to SQL Server: Printing Table`); //successful connection. user info retrieved  
 return callback(null,result)  
 }  
 })  
 }  
 })  
},