

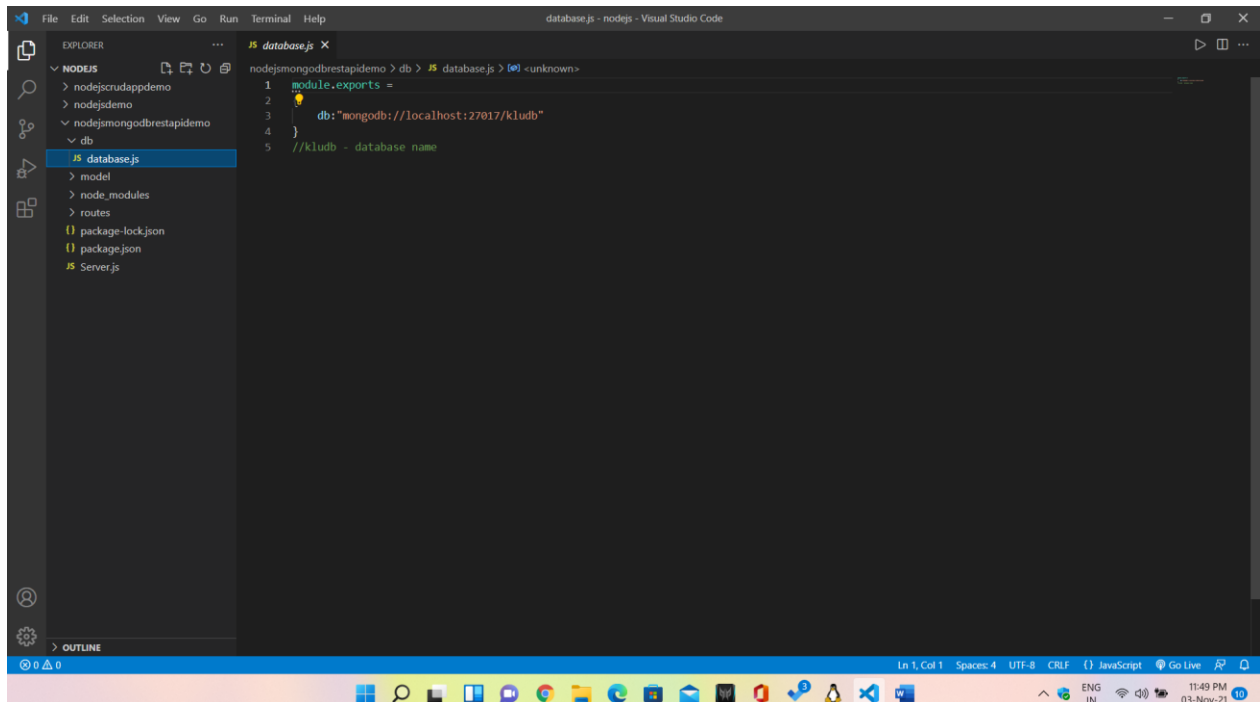
# Skill 10

ID: 2000030384

Name: Jatla Nikhil Sai Lalith

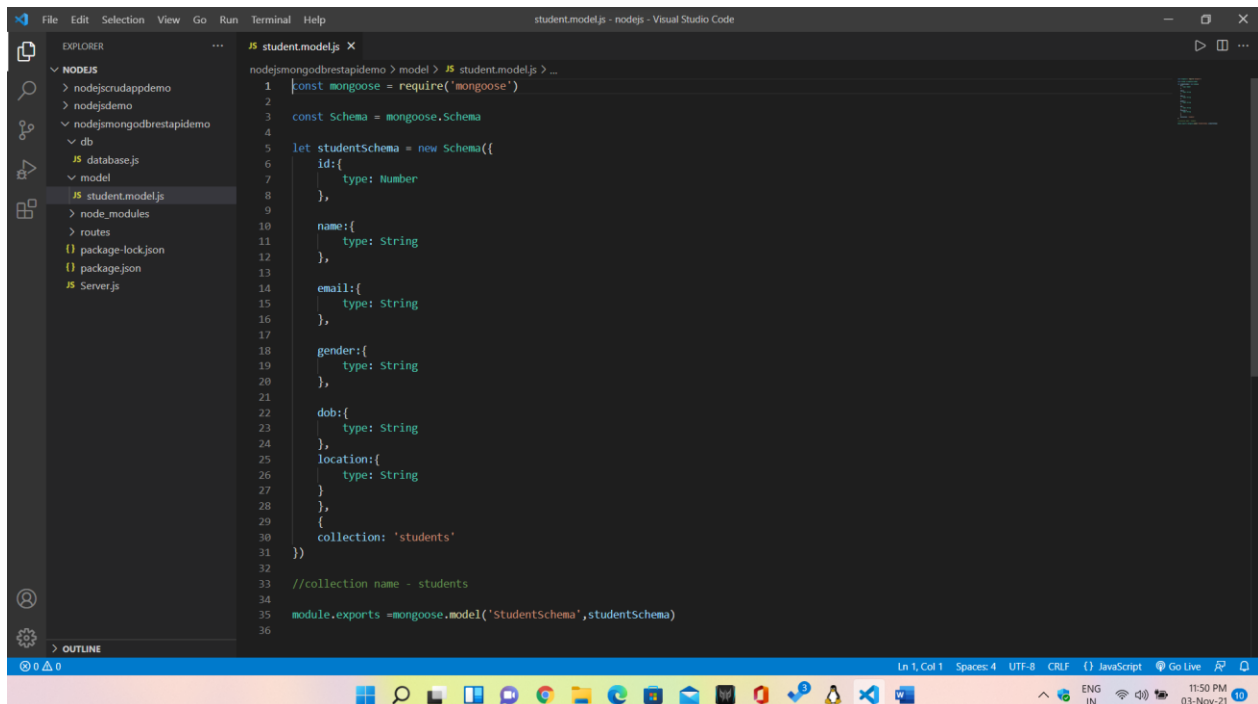
Section No: S09

## Database.js:



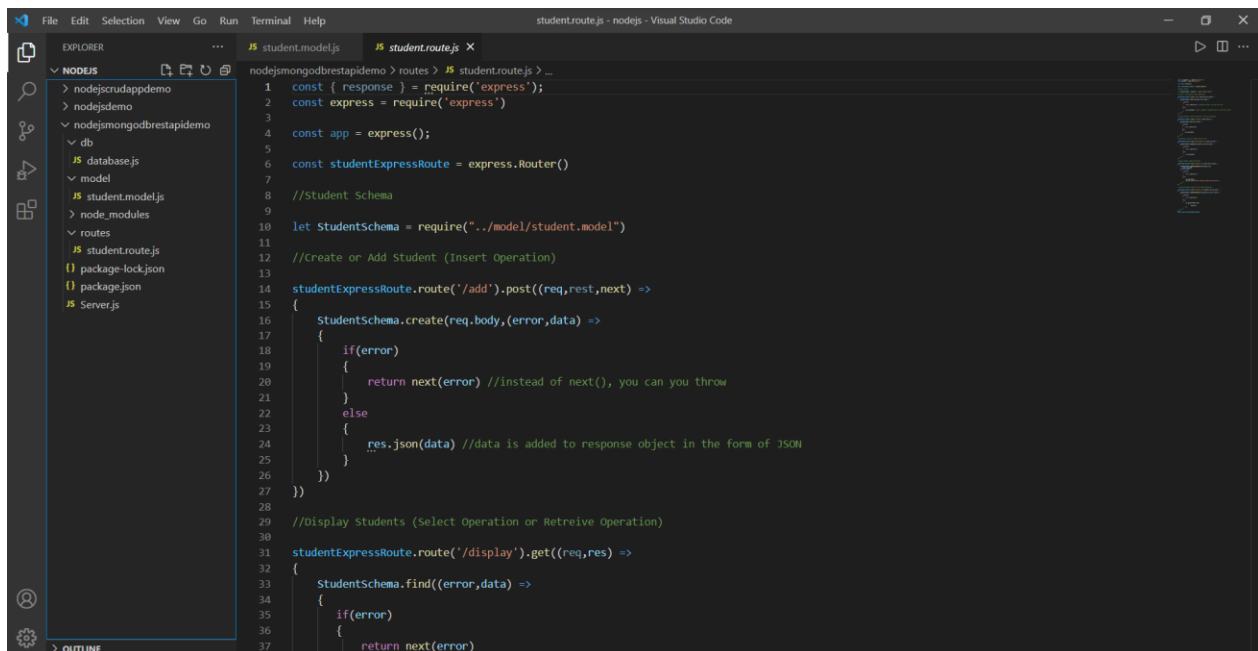
```
1 module.exports =  
2 {  
3   db: "mongodb://localhost:27017/kludb"  
4 }  
5 //kludb - database name
```

## Student.model.js:

A screenshot of the Visual Studio Code editor. The Explorer sidebar on the left shows a project structure with folders like 'node\_modules' and 'routes', and files like 'package-lock.json' and 'package.json'. The 'student.model.js' file is selected and open in the main editor. The code defines a Mongoose schema for a student with fields: id (Number), name (String), email (String), gender (String), dob (String), and location (String). It also sets the collection name to 'students' and exports the model.

```
1 const mongoose = require('mongoose')
2
3 const Schema = mongoose.Schema
4
5 let studentSchema = new Schema({
6   id: {
7     type: Number
8   },
9
10  name: {
11    type: String
12  },
13
14  email: {
15    type: String
16  },
17
18  gender: {
19    type: String
20  },
21
22  dob: {
23    type: String
24  },
25  location: {
26    type: String
27  }
28 },
29 {
30   collection: 'students'
31 })
32
33 //collection name - students
34
35 module.exports = mongoose.model('StudentSchema', studentSchema)
36
```

## Student.route.js:

A screenshot of the Visual Studio Code editor. The Explorer sidebar on the left shows the same project structure as the previous screenshot. The 'student.route.js' file is selected and open in the main editor. The code sets up an Express application, creates a router for the student schema, and defines two routes: a POST route for adding a student and a GET route for displaying all students.

```
1 const { response } = require('express');
2 const express = require('express')
3
4 const app = express();
5
6 const studentExpressRoute = express.Router()
7
8 //Student Schema
9
10 let StudentSchema = require("../model/student.model")
11
12 //Create or Add Student (Insert Operation)
13
14 studentExpressRoute.route('/add').post((req, res, next) =>
15 {
16   StudentSchema.create(req.body, (error, data) =>
17   {
18     if(error)
19     {
20       return next(error) //instead of next(), you can you throw
21     }
22     else
23     {
24       res.json(data) //data is added to response object in the form of JSON
25     }
26   })
27 })
28
29 //Display Students (Select Operation or Retrieve Operation)
30
31 studentExpressRoute.route('/display').get((req, res) =>
32 {
33   StudentSchema.find((error, data) =>
34   {
35     if(error)
36     {
37       return next(error)
38     }
39   })
40 })
41
```

```
nodejscrudappdemo
nodejsdemo
nodejsmongodbrestapidemo
  db
    database.js
  model
    student.model.js
  node_modules
  routes
    student.routes.js
package-lock.json
package.json
Server.js

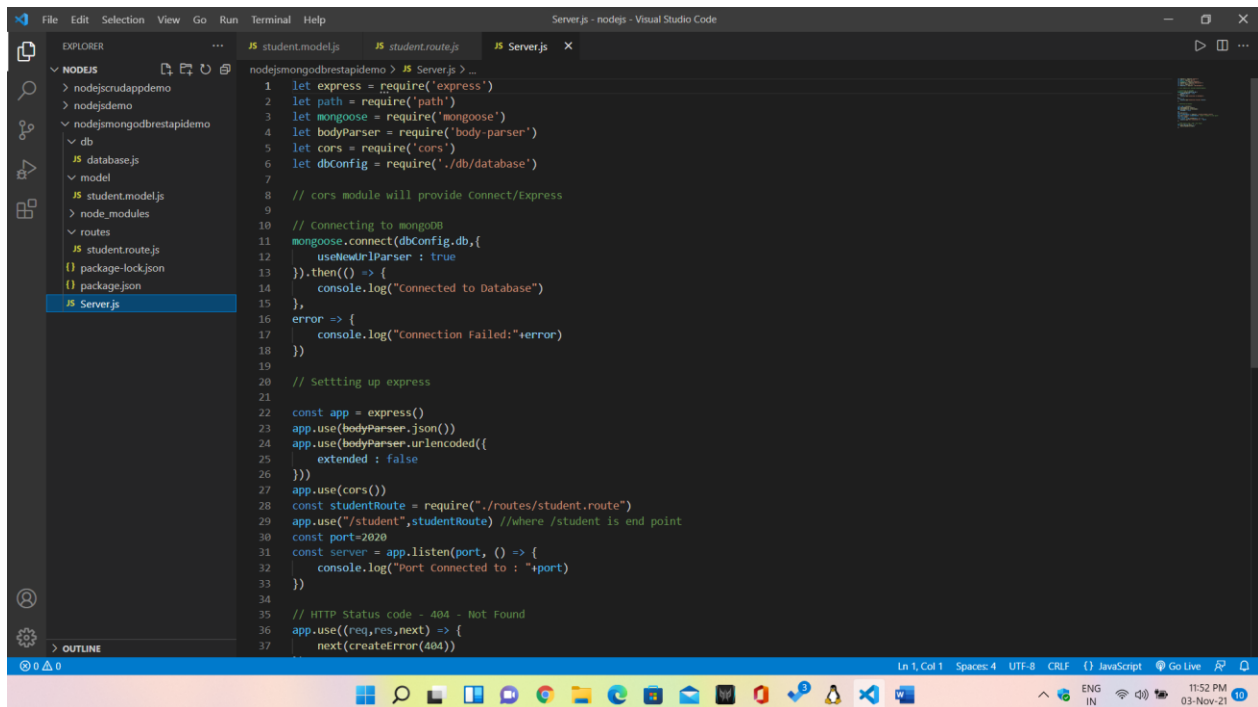
38      }
39      else
40      {
41          res.json(data)
42      }
43  })
44  })
45  //Displaying a Specific Student Record by ID
46  studentExpressRoute.route("/getstudent/:id").get((req,res) =>
47  {
48      StudentSchema.findById(req.params.id,(error,data) =>
49      {
50          if(error)
51          {
52              return next(error)
53          }
54          else
55          {
56              res.json(data)
57          }
58      })
59  })
60  })
61  //Update Student (Update Operation)
62  studentExpressRoute.route("/update/:id").put((req,res,next) =>
63  {
64      StudentSchema.findByIdAndUpdate(req.params.id,{
65          $set: req.body
66      },(error,data)=>
67      {
68          if(error)
69          {
70              return next(error)
71          }
72      })
73  })
74  })
```

```
nodejscrudappdemo
nodejsdemo
nodejsmongodbrestapidemo
  db
    database.js
  model
    student.model.js
  node_modules
  routes
    student.routes.js
package-lock.json
package.json
Server.js

75      }
76      else
77      {
78          res.json(data)
79          console.log("Student Document Updated Successfully")
80      }
81  })
82  // Deleting Student Document by ID (Delete Operation)
83  studentExpressRoute.route("/delete/:id").delete((req,res,next) =>
84  {
85      StudentSchema.findByIdAndRemove(req.params.id,(error,data) =>
86      {
87          if(error)
88          {
89              return next(error)
90          }
91          else
92          {
93              res.status(200).json({
94                  msg:data
95              })
96          }
97      })
98  })
99  })
100  })
101  module.exports=studentExpressRoute
102
```

Ln 63, Col 2   Spaces: 4   UTF-8   CRUF   JavaScript   Go Live   11:51 PM 03-Nov-21

# Server.js:



```
1 let express = require('express')
2 let path = require('path')
3 let mongoose = require('mongoose')
4 let bodyParser = require('body-parser')
5 let cors = require('cors')
6 let dbConfig = require('./db/database')
7
8 // cors module will provide Connect/Express
9
10 // Connecting to mongoDB
11 mongoose.connect(dbConfig.db,{
12   useNewUrlParser : true
13 }).then(() => {
14   console.log("Connected to Database")
15 },
16 error => {
17   console.log("Connection Failed:"+error)
18 })
19
20 // Setting up express
21
22 const app = express()
23 app.use(bodyParser.json())
24 app.use(bodyParser.urlencoded({
25   extended : false
26 }))
27 app.use(cors())
28 const studentRoute = require("./routes/student.route")
29 app.use("/student",studentRoute) //where /student is end point
30 const port=2020
31 const server = app.listen(port, () => {
32   console.log("Port connected to : "+port)
33 })
34
35 // HTTP Status code - 404 - Not Found
36 app.use((req,res,next) => {
37   next(createError(404))
38 })
```