Module 3 Lab 1: Implementing MongoDB Operations in Node.js

Mohiddeen Vilak Mohammad

Department of Information Technology, Arizona State University

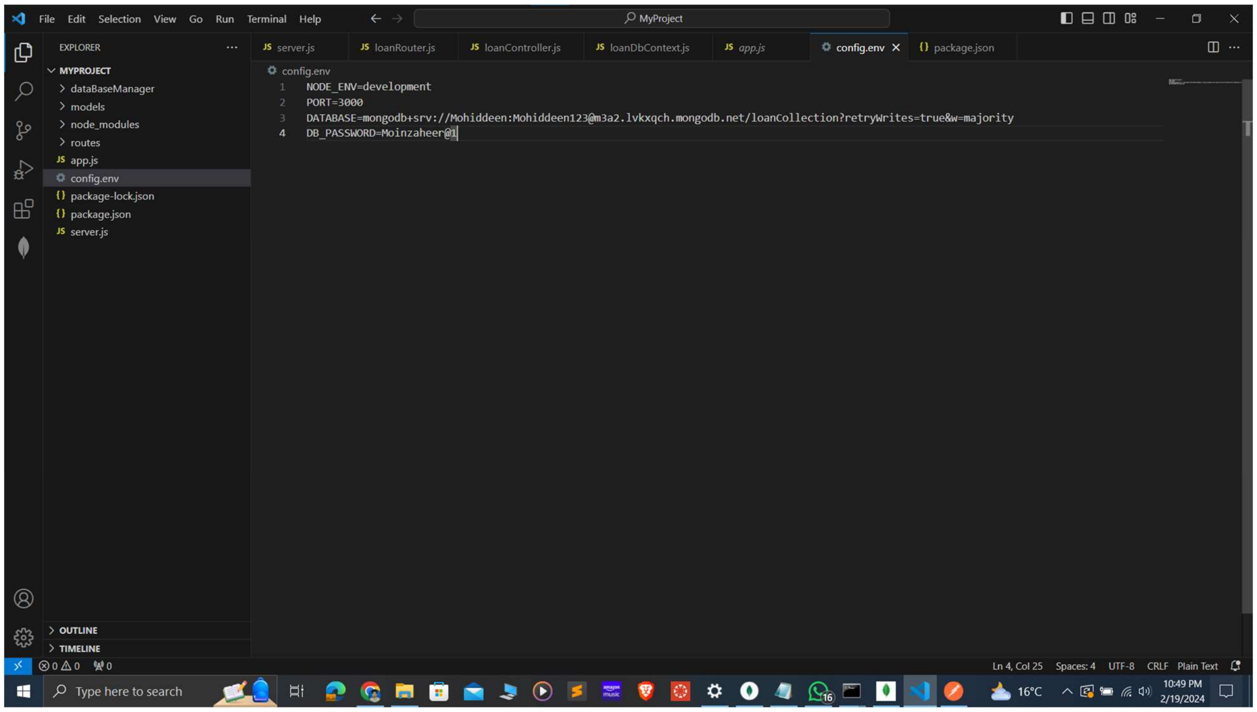
IFT 554: Middleware Programming & Database Security

Dinesh Sthapit

February 18, 2024

Figure 1:

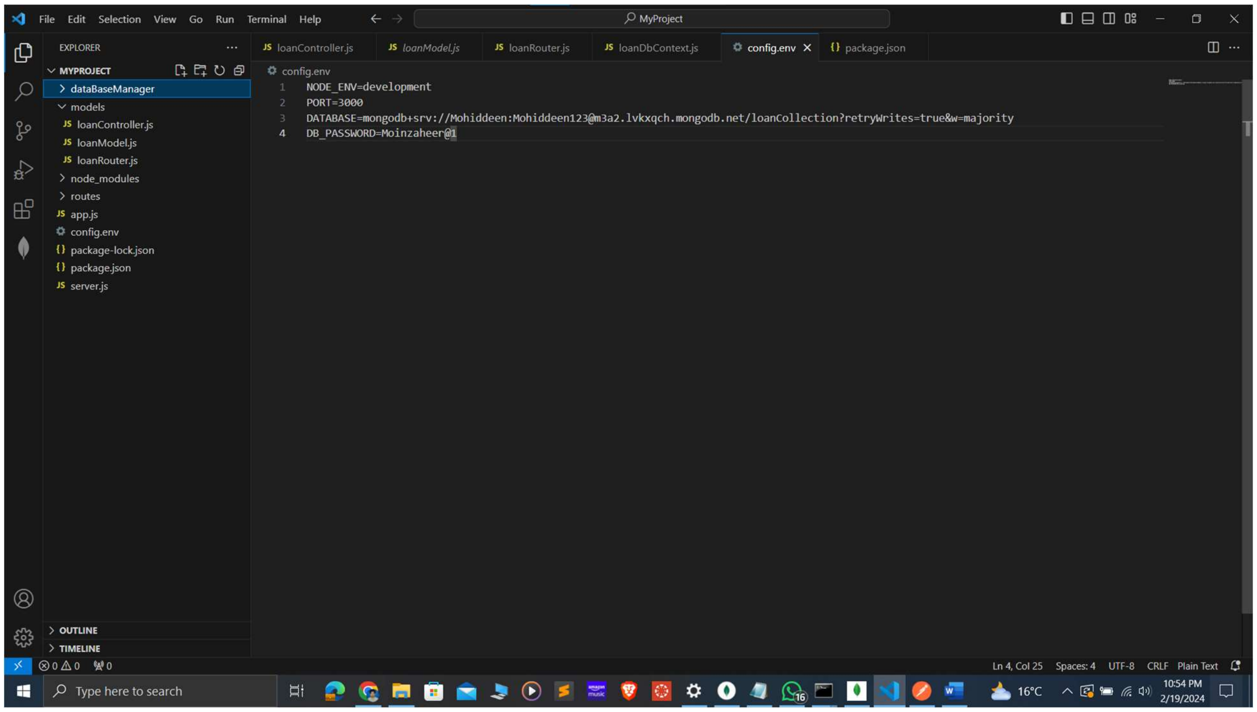
Manually adding config.env file



```
NODE_ENV=development
PORT=3000
DATABASE=mongodb+srv://Mohiddeen:Mohiddeen123@m3a2.lvkxqch.mongodb.net/loanCollection?retryWrites=true&w=majority
DB_PASSWORD=Moinzaheer@1
```

Figure 2:

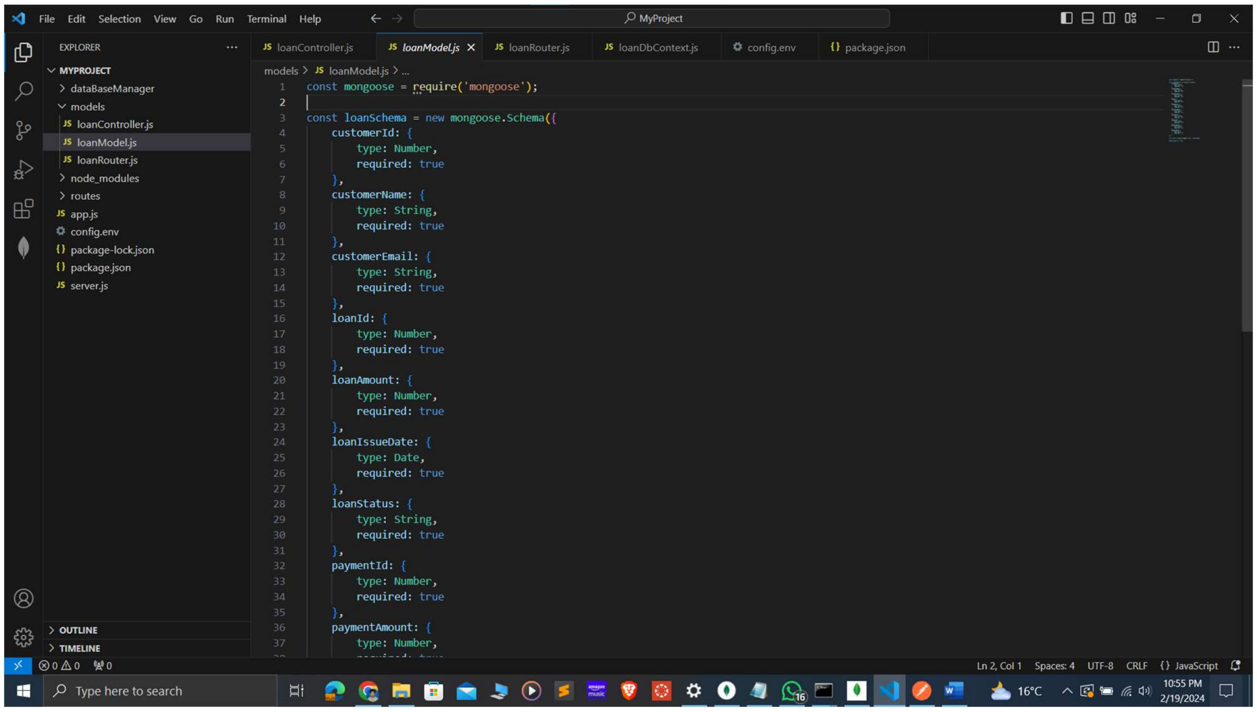
Creating a folder with all the three files



```
NODE_ENV=development
PORT=3000
DATABASE=mongodb+srv://Mohiddeen:Mohiddeen123@m3a2.lvkxqch.mongodb.net/loanCollection?retryWrites=true&w=majority
DB_PASSWORD=Moinzaheer@1
```

Figure 3:

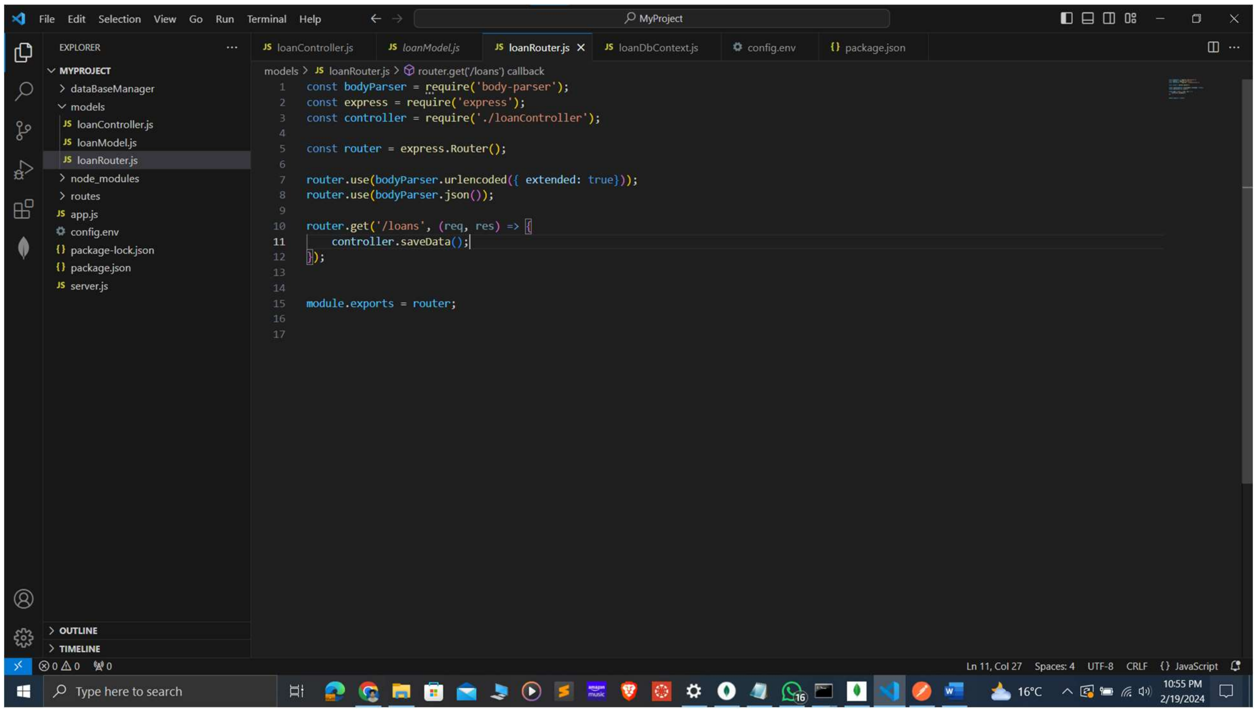
loanModel.js file



```
const mongoose = require('mongoose');

const loanSchema = new mongoose.Schema({
    customerId: {
        type: Number,
        required: true
    },
    customerName: {
        type: String,
        required: true
    },
    customerEmail: {
        type: String,
        required: true
    },
    loanId: {
        type: Number,
        required: true
    },
    loanAmount: {
        type: Number,
        required: true
    },
    loanIssueDate: {
        type: Date,
        required: true
    },
    loanStatus: {
        type: String,
        required: true
    },
    paymentId: {
        type: Number,
        required: true
    },
    paymentAmount: {
        type: Number,
```

Figure 4:

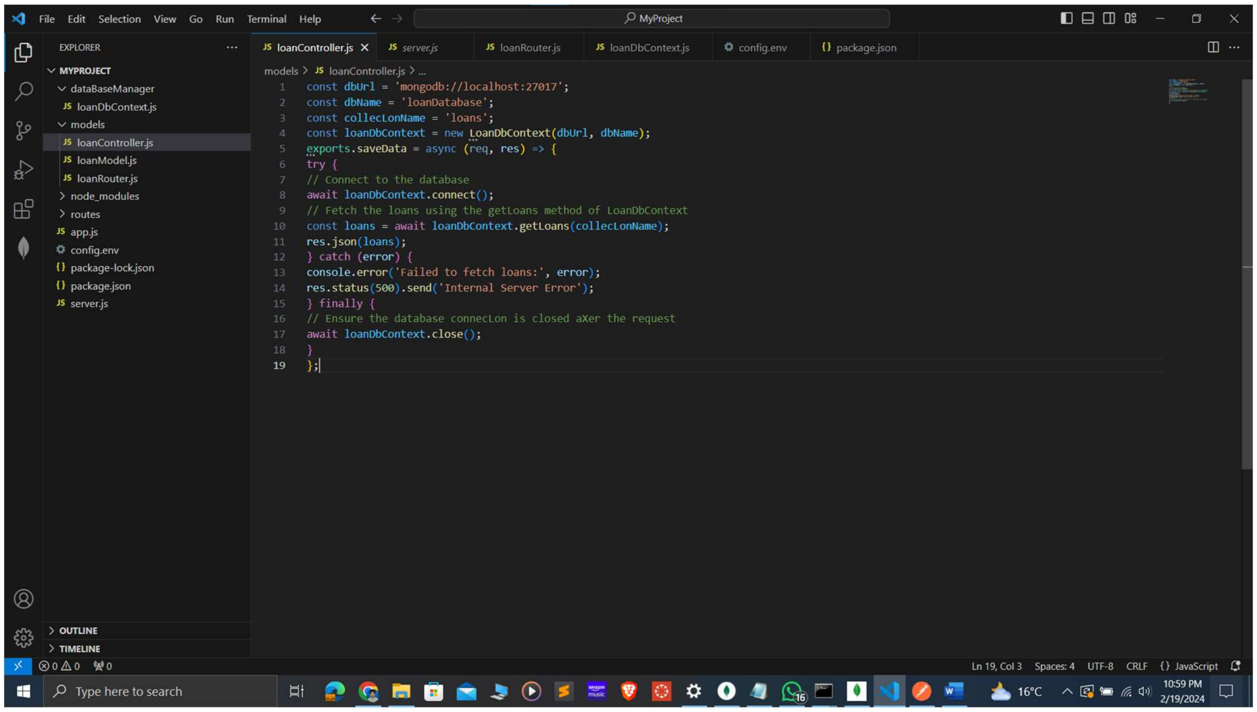
loanRoutes.js



```javascript
const bodyParser = require('body-parser');
const express = require('express');
const controller = require('./loanController');

const router = express.Router();

router.use(bodyParser.urlencoded({ extended: true}));
router.use(bodyParser.json());

router.get('/loans', (req, res) => {
    controller.saveData();
});


module.exports = router;
```

Figure 5:

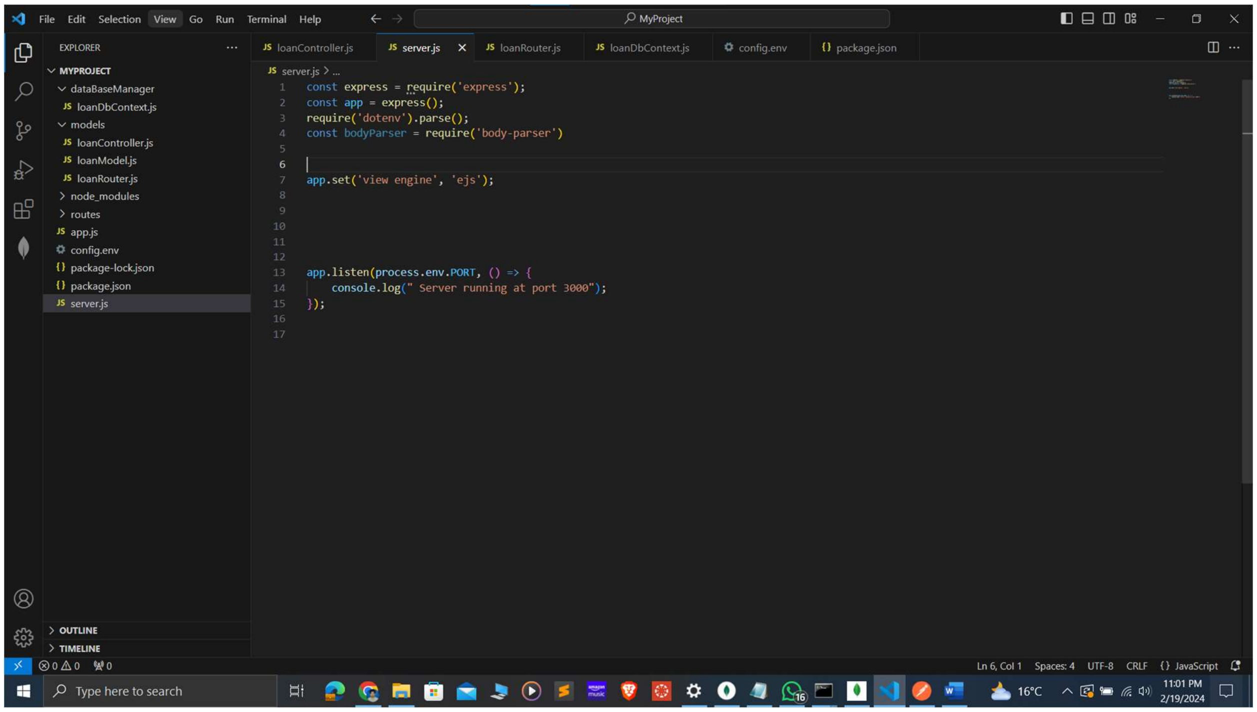
loanController.js



```
const dbUrl = 'mongodb://localhost:27017';
const dbName = 'loanDatabase';
const collecLonName = 'loans';
const loanDbContext = new LoanDbContext(dbUrl, dbName);
exports.saveData = async (req, res) => {
try {
// Connect to the database
await loanDbContext.connect();
// Fetch the loans using the getLoans method of LoanDbContext
const loans = await loanDbContext.getLoans(collecLonName);
res.json(loans);
} catch (error) {
console.error('Failed to fetch loans:', error);
res.status(500).send('Internal Server Error');
} finally {
// Ensure the database connecLon is closed aXer the request
await loanDbContext.close();
}
};
```

Figure 6:

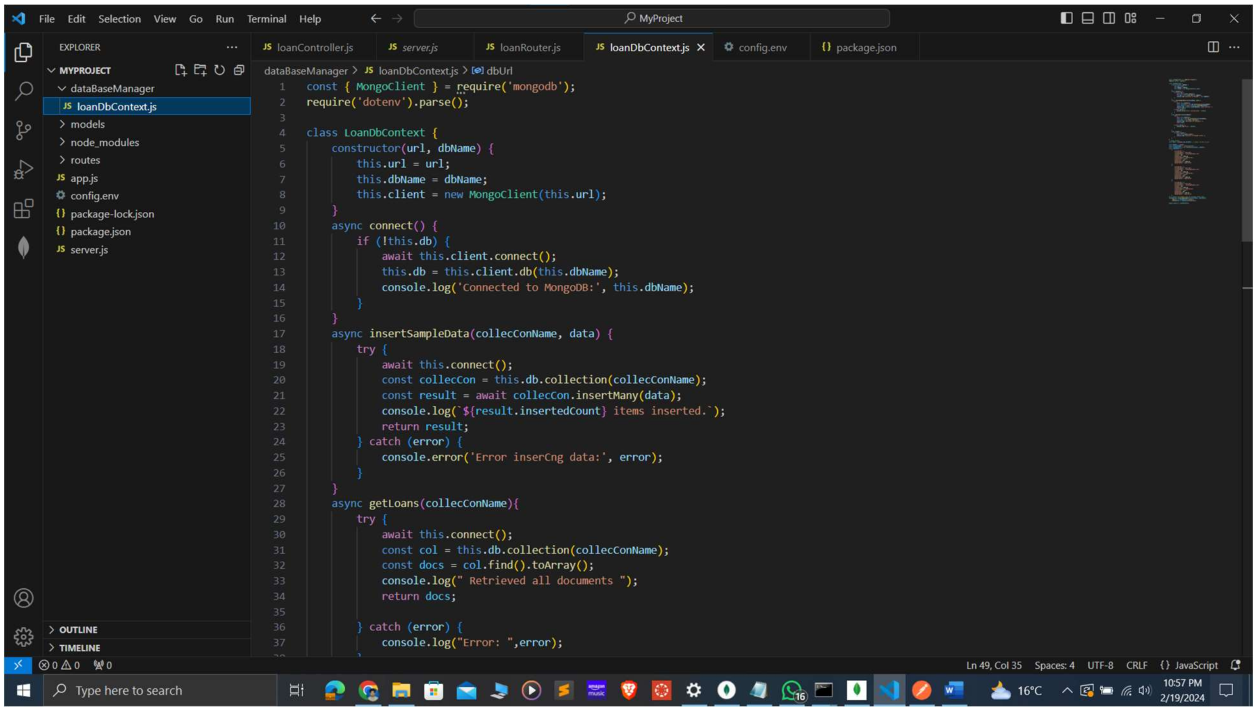
Server.js



```
const express = require('express');
const app = express();
require('dotenv').parse();
const bodyParser = require('body-parser')


app.set('view engine', 'ejs');





app.listen(process.env.PORT, () => {
    console.log(" Server running at port 3000");
});
```

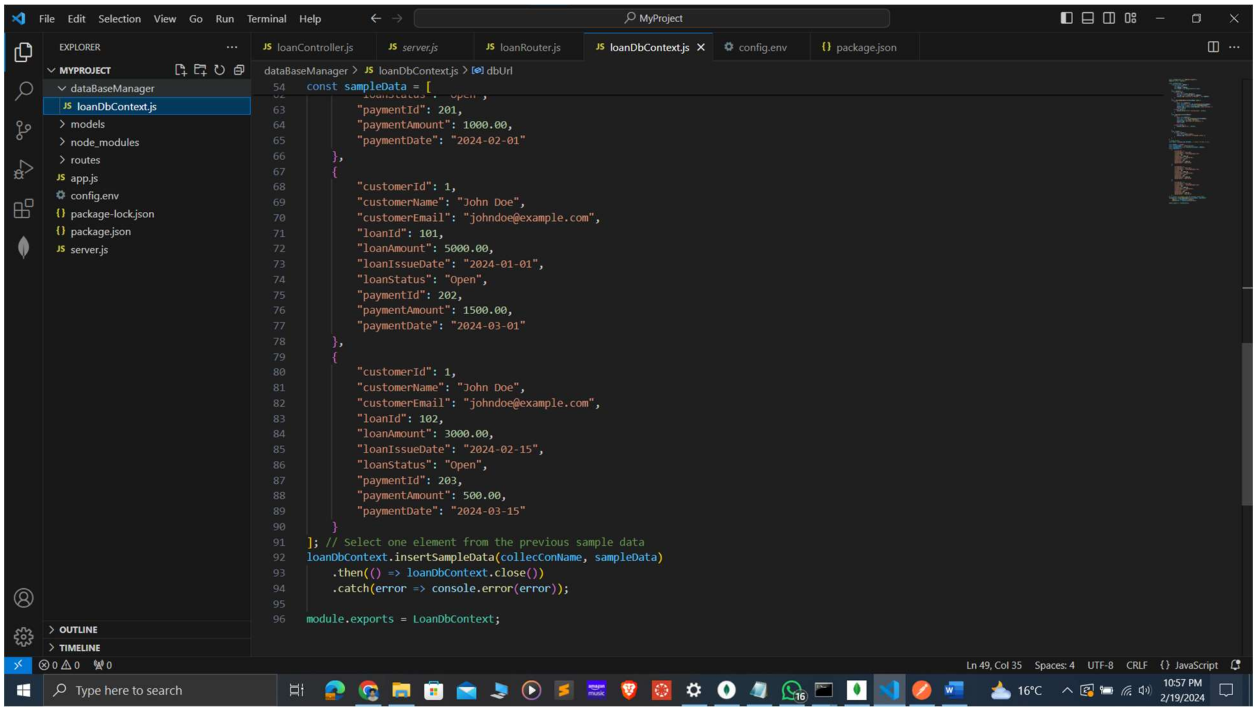
Figure 7:

loanDBcontext.js



```
const { MongoClient } = require('mongodb');
require('dotenv').parse();

class LoanDbContext {
    constructor(url, dbName) {
        this.url = url;
        this.dbName = dbName;
        this.client = new MongoClient(this.url);
    }
    async connect() {
        if (!this.db) {
            await this.client.connect();
            this.db = this.client.db(this.dbName);
            console.log('Connected to MongoDB:', this.dbName);
        }
    }
    async insertSampleData(collecConName, data) {
        try {
            await this.connect();
            const collecCon = this.db.collection(collecConName);
            const result = await collecCon.insertMany(data);
            console.log(`${result.insertedCount} items inserted.`);
            return result;
        } catch (error) {
            console.error('Error inserCng data:', error);
        }
    }
    async getLoans(collecConName){
        try {
            await this.connect();
            const col = this.db.collection(collecConName);
            const docs = col.find().toArray();
            console.log(" Retrieved all documents ");
            return docs;

        } catch (error) {
            console.log("Error: ",error);
```

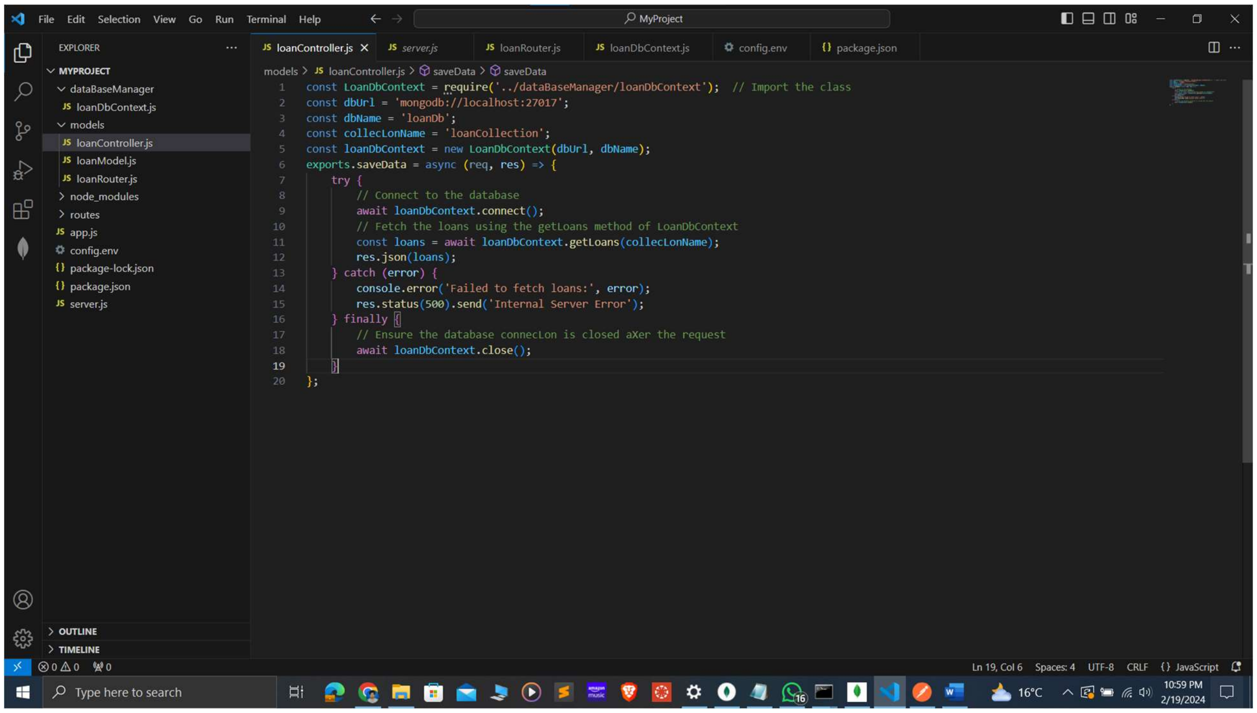
Figure 8:



```
const sampleData = [
            "paymentId": 201,
            "paymentAmount": 1000.00,
            "paymentDate": "2024-02-01"
        },
        {
            "customerId": 1,
            "customerName": "John Doe",
            "customerEmail": "johndoe@example.com",
            "loanId": 101,
            "loanAmount": 5000.00,
            "loanIssueDate": "2024-01-01",
            "loanStatus": "Open",
            "paymentId": 202,
            "paymentAmount": 1500.00,
            "paymentDate": "2024-03-01"
        },
        {
            "customerId": 1,
            "customerName": "John Doe",
            "customerEmail": "johndoe@example.com",
            "loanId": 102,
            "loanAmount": 3000.00,
            "loanIssueDate": "2024-02-15",
            "loanStatus": "Open",
            "paymentId": 203,
            "paymentAmount": 500.00,
            "paymentDate": "2024-03-15"
        }
]; // Select one element from the previous sample data
loanDbContext.insertSampleData(collecConName, sampleData)
    .then(() => loanDbContext.close())
    .catch(error => console.error(error));

module.exports = LoanDbContext;
```

Figure 9:

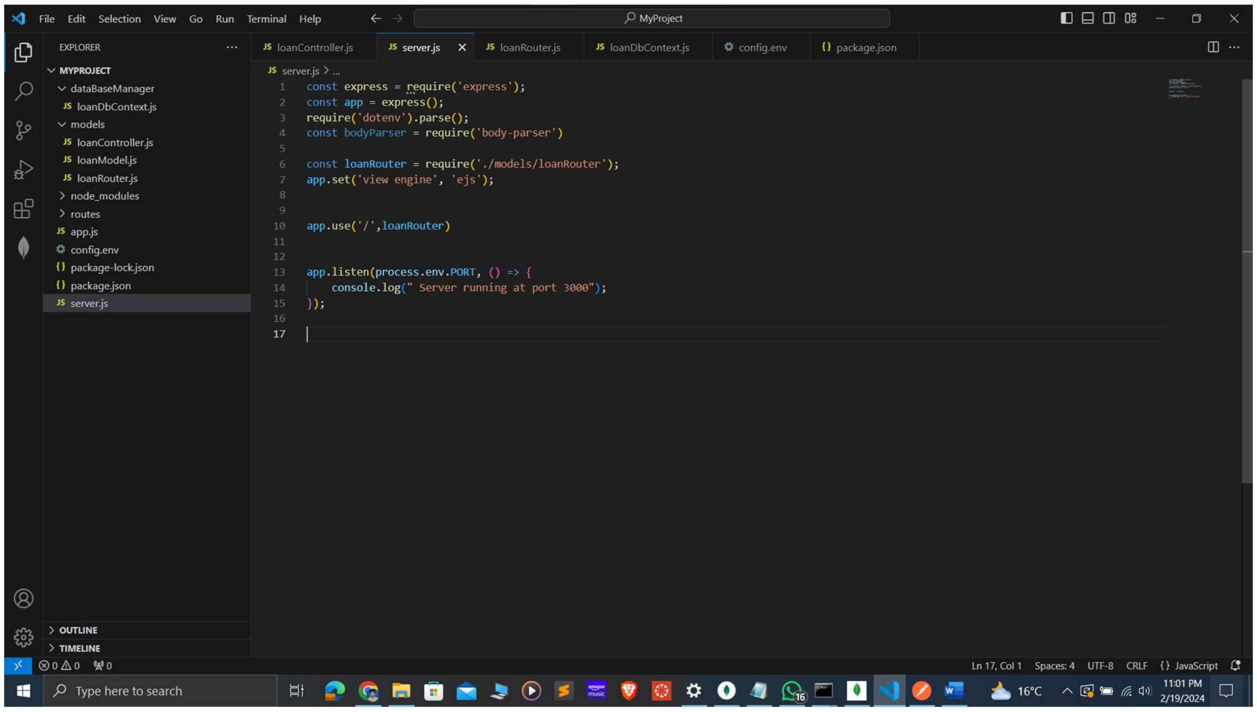
Updated controller.js



```
const LoanDbContext = require('../dataBaseManager/loanDbContext');  // Import the class
const dbUrl = 'mongodb://localhost:27017';
const dbName = 'loanDb';
const collecLonName = 'loanCollection';
const loanDbContext = new LoanDbContext(dbUrl, dbName);
exports.saveData = async (req, res) => {
    try {
        // Connect to the database
        await loanDbContext.connect();
        // Fetch the loans using the getLoans method of LoanDbContext
        const loans = await loanDbContext.getLoans(collecLonName);
        res.json(loans);
    } catch (error) {
        console.error('Failed to fetch loans:', error);
        res.status(500).send('Internal Server Error');
    } finally {
        // Ensure the database connecLon is closed aXer the request
        await loanDbContext.close();
    }
};
```

Figure 10:

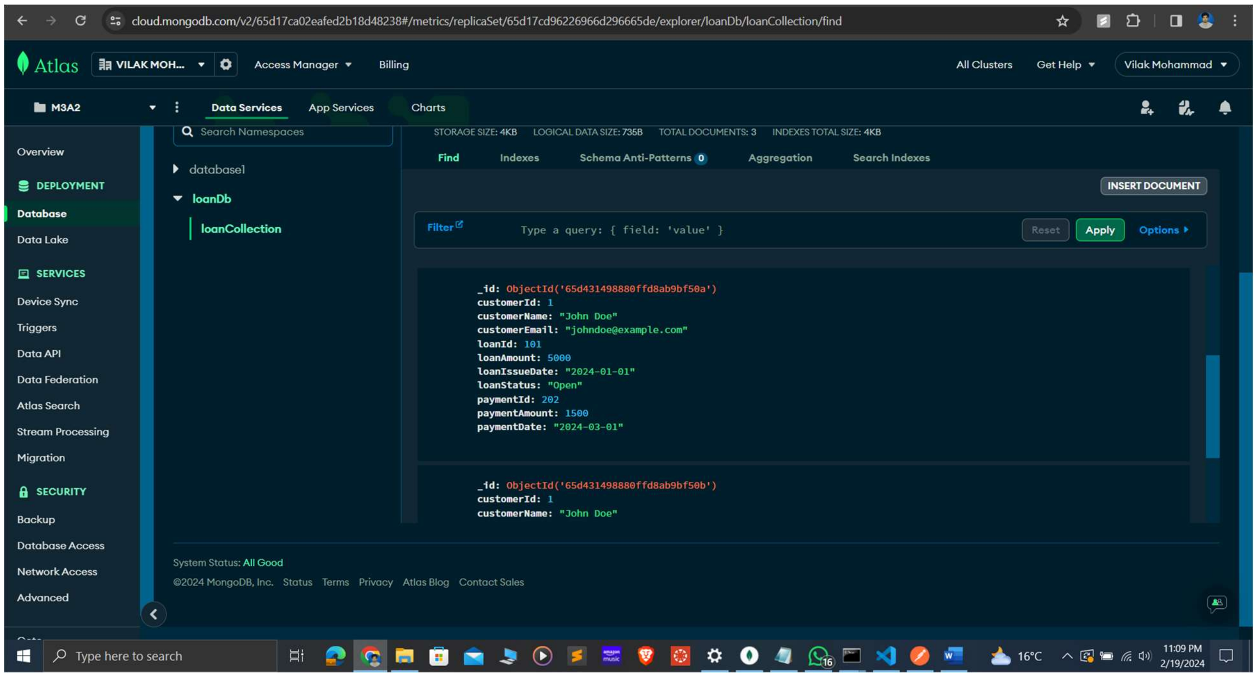
Updated server.js with routes



```
const express = require('express');
const app = express();
require('dotenv').parse();
const bodyParser = require('body-parser')

const loanRouter = require('./models/loanRouter');
app.set('view engine', 'ejs');


app.use('/',loanRouter)


app.listen(process.env.PORT, () => {
    console.log(" Server running at port 3000");
});

```

Figure 11:

Data is sent and retrieved.



```
_id: ObjectId('65d431498880ffd8ab9bf50a')
customerId: 1
customerName: "John Doe"
customerEmail: "johndoe@example.com"
loanId: 101
loanAmount: 5000
loanIssueDate: "2024-01-01"
loanStatus: "Open"
paymentId: 202
paymentAmount: 1500
paymentDate: "2024-03-01"
```

```
_id: ObjectId('65d431498880ffd8ab9bf50b')
customerId: 1
customerName: "John Doe"
```