

Microproject#2 (Beta Var 1.0)

Name: Prabhjot Singh (200566745)

Akash Kaler (200571587)

Vishal (200566563)

Course Name: JavaScript Frameworks

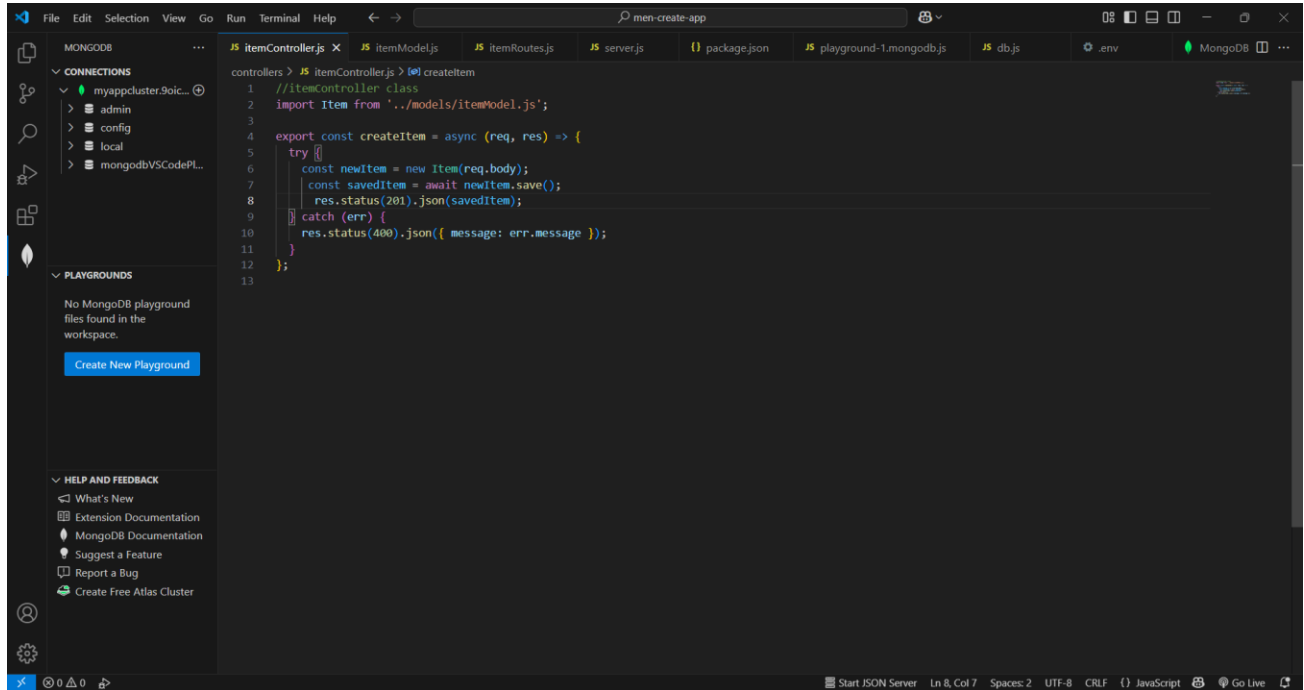
Instructor Name: Anmar Jarjees

Date: April 12th, 2025

MongoDB, Express, and Node.js application

Screenshots of codes

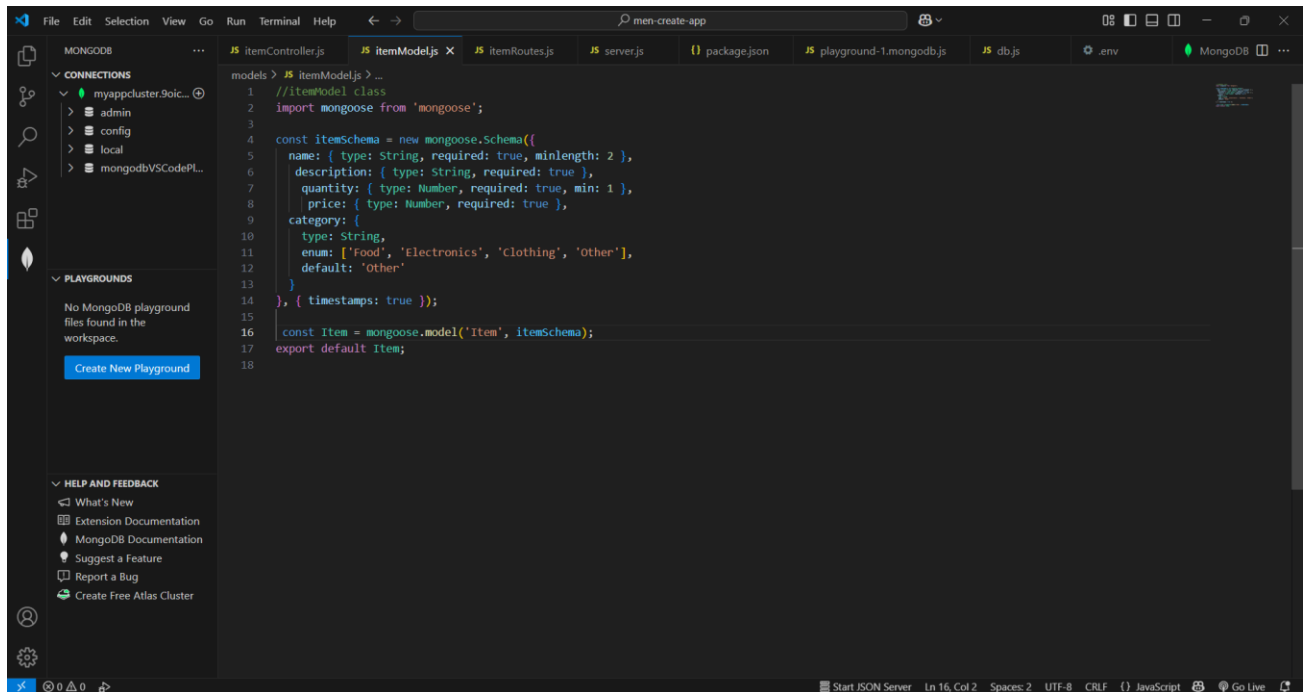
1. itemController class



The screenshot shows the VS Code editor with the file explorer on the left. The file explorer shows a project named 'men-create-app' with a folder 'controllers' containing 'itemController.js'. The 'itemController.js' file is open in the editor, showing the following code:

```
1 //itemController class
2 import Item from '../models/itemModel.js';
3
4 export const createItem = async (req, res) => {
5   try {
6     const newItem = new Item(req.body);
7     const savedItem = await newItem.save();
8     res.status(201).json(savedItem);
9   } catch (err) {
10    res.status(400).json({ message: err.message });
11  }
12 }
13
```

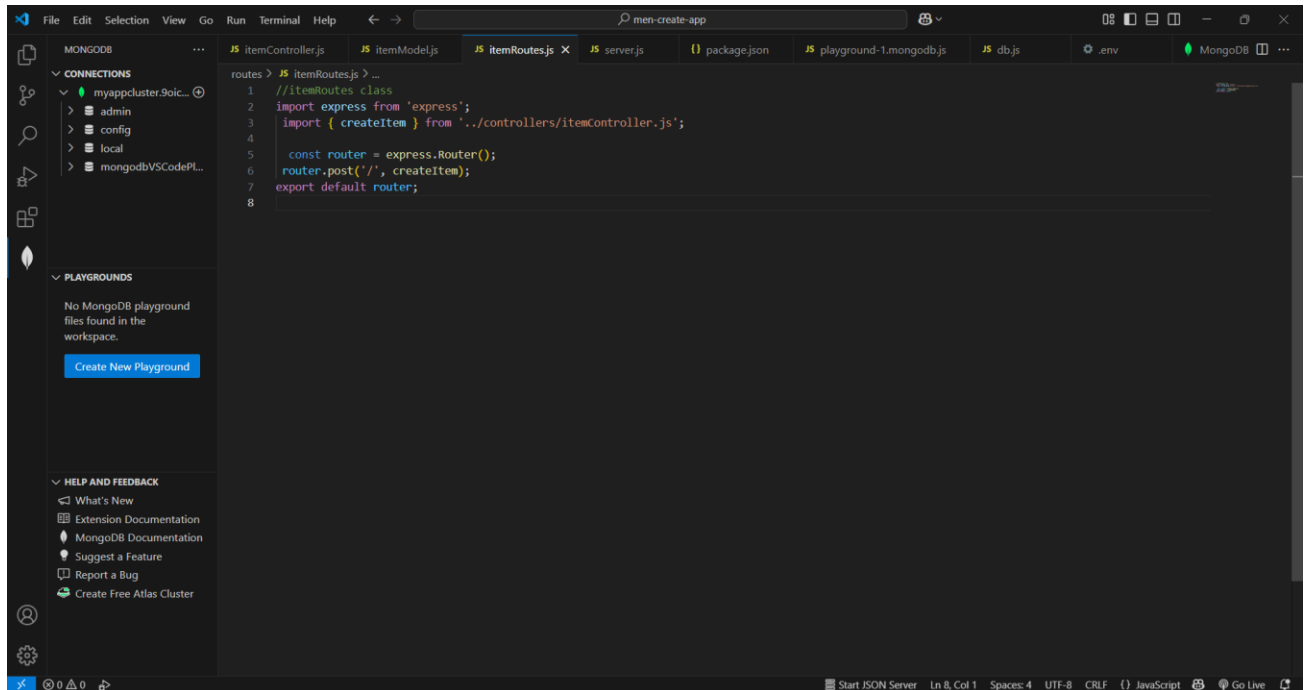
2. itemModel class



The screenshot shows the VS Code editor with the file explorer on the left. The file explorer shows a project named 'men-create-app' with a folder 'models' containing 'itemModel.js'. The 'itemModel.js' file is open in the editor, showing the following code:

```
1 //itemModel class
2 import mongoose from 'mongoose';
3
4 const itemSchema = new mongoose.Schema({
5   name: { type: String, required: true, minlength: 2 },
6   description: { type: String, required: true },
7   quantity: { type: Number, required: true, min: 1 },
8   price: { type: Number, required: true },
9   category: {
10    type: String,
11    enum: ['Food', 'Electronics', 'Clothing', 'Other'],
12    default: 'Other'
13  }
14 }, { timestamps: true });
15
16 const Item = mongoose.model('Item', itemSchema);
17 export default Item;
18
```

3. itemRoutes class

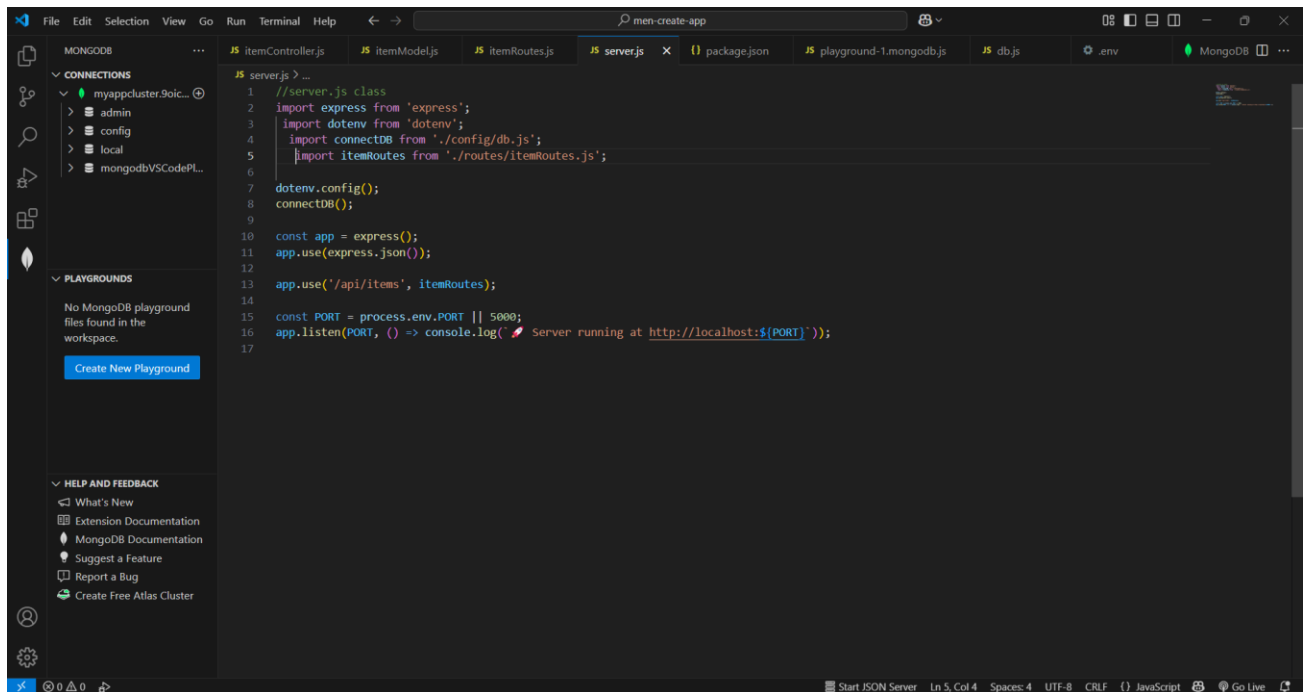


The screenshot shows the VS Code editor with the 'itemRoutes.js' file open. The file contains the following code:

```
1 //itemRoutes class
2 import express from 'express';
3 import { createItem } from '../controllers/itemController.js';
4
5 const router = express.Router();
6 router.post('/', createItem);
7 export default router;
```

The editor interface includes a sidebar on the left with 'CONNECTIONS' and 'PLAYGROUNDS' sections, and a top menu bar with 'File', 'Edit', 'Selection', 'View', 'Go', 'Run', 'Terminal', and 'Help'. The status bar at the bottom indicates 'Start JSON Server', 'Ln 8, Col 1', 'Spaces: 4', 'UTF-8', 'CRLF', 'JavaScript', and 'Go Live'.

4. server.js class

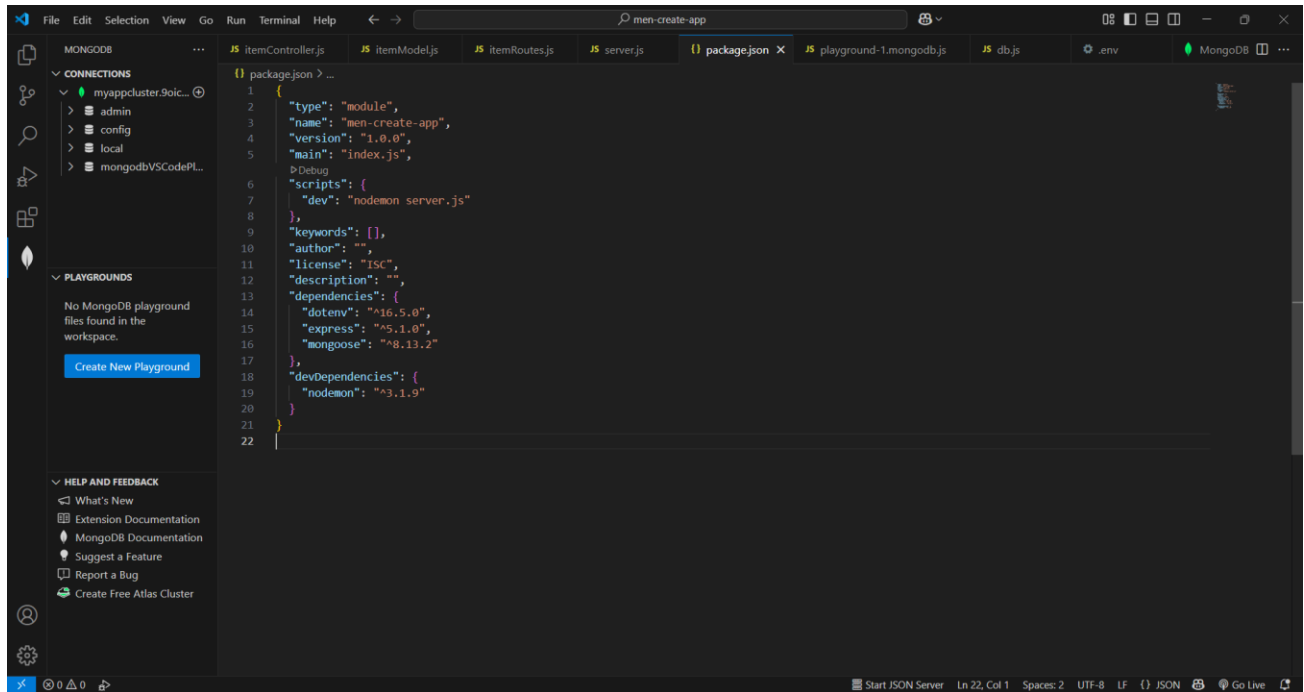


The screenshot shows the VS Code editor with the 'server.js' file open. The file contains the following code:

```
1 //server.js class
2 import express from 'express';
3 import dotenv from 'dotenv';
4 import connectDB from './config/db.js';
5 import itemRoutes from './routes/itemRoutes.js';
6
7 dotenv.config();
8 connectDB();
9
10 const app = express();
11 app.use(express.json());
12
13 app.use('/api/items', itemRoutes);
14
15 const PORT = process.env.PORT || 5000;
16 app.listen(PORT, () => console.log('Server running at http://localhost:${PORT}'));
```

The editor interface is similar to the previous screenshot, showing the same sidebar and menu bar. The status bar at the bottom indicates 'Start JSON Server', 'Ln 5, Col 4', 'Spaces: 4', 'UTF-8', 'CRLF', 'JavaScript', and 'Go Live'.

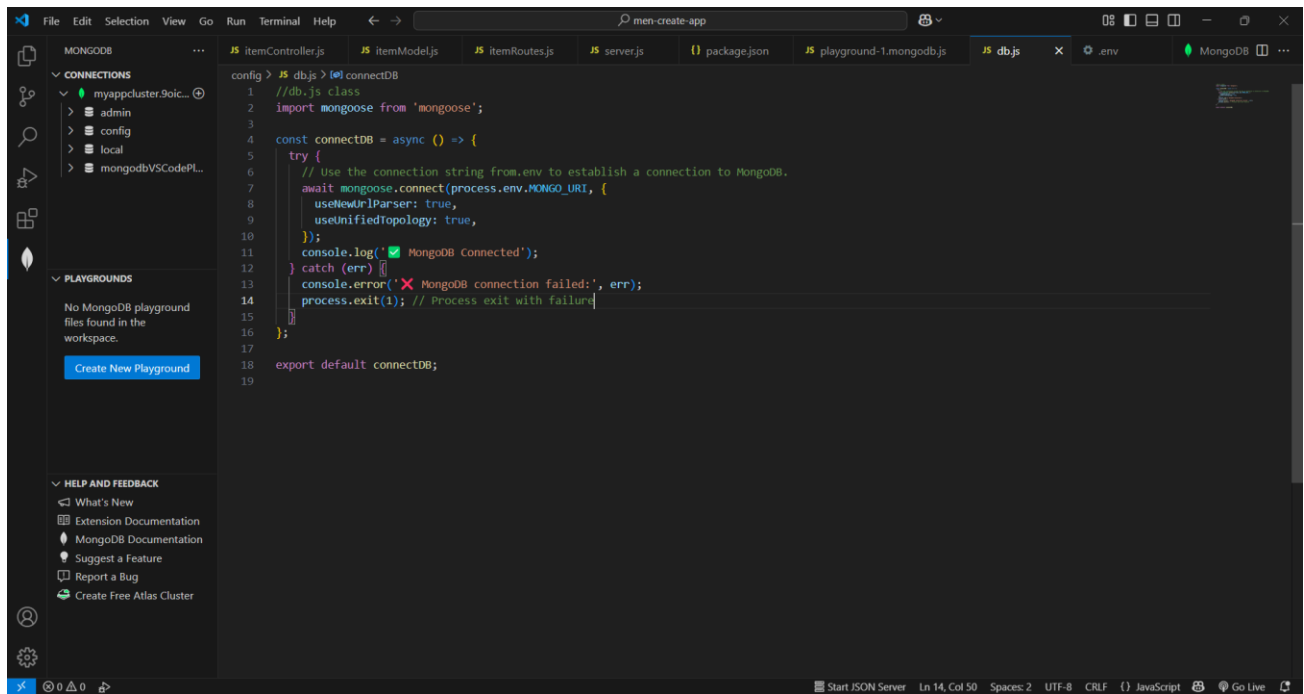
5. package.json class



The screenshot shows the VS Code editor with the 'package.json' file open. The file contains the following JSON structure:

```
1 {
2   "type": "module",
3   "name": "men-create-app",
4   "version": "1.0.0",
5   "main": "index.js",
6   "scripts": {
7     "dev": "nodemon server.js"
8   },
9   "keywords": [],
10  "author": "",
11  "license": "ISC",
12  "description": "",
13  "dependencies": {
14    "dotenv": "^16.5.0",
15    "express": "^5.1.0",
16    "mongoose": "^8.13.2"
17  },
18  "devDependencies": {
19    "nodemon": "^3.1.9"
20  }
21 }
```

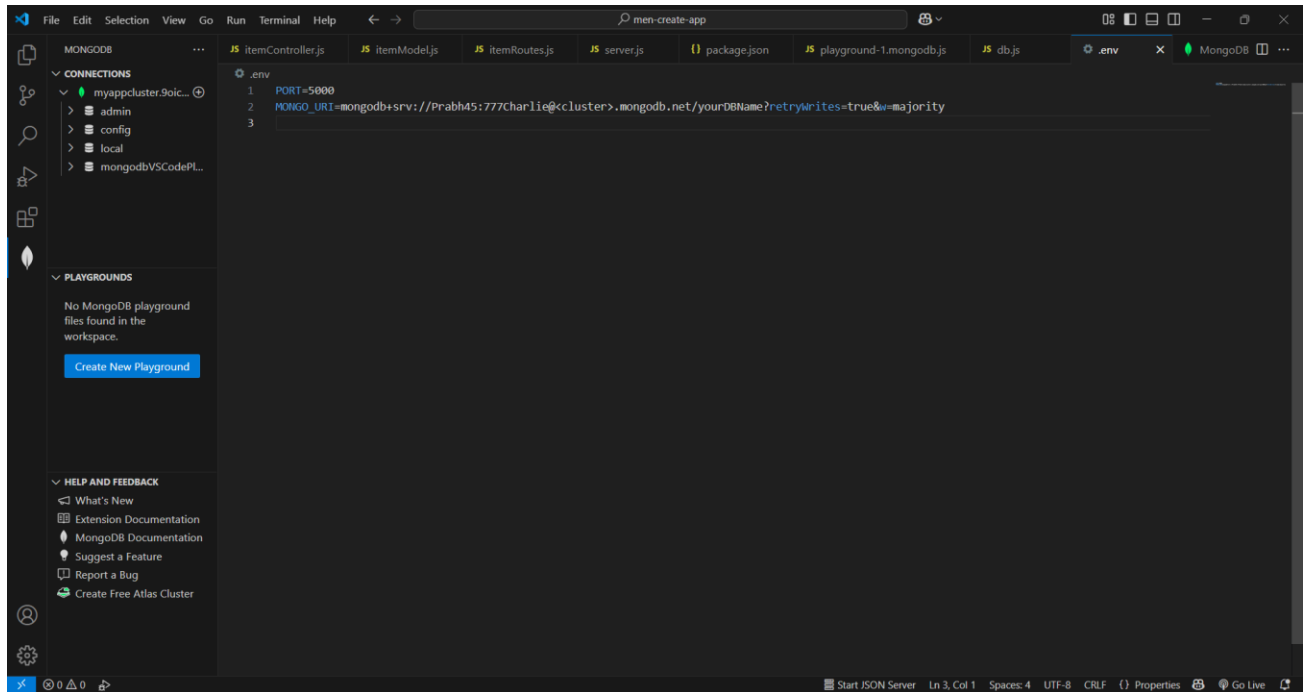
6. db.js class



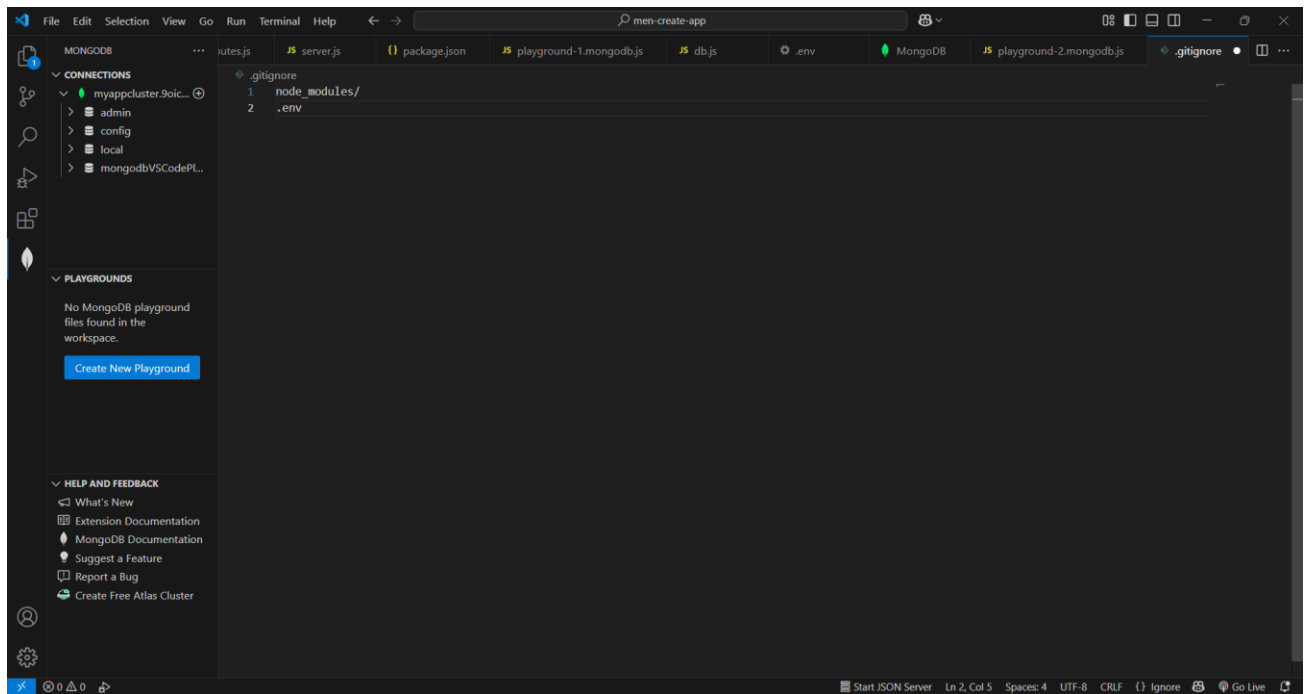
The screenshot shows the VS Code editor with the 'db.js' file open. The file contains the following JavaScript code:

```
1 //db.js class
2 import mongoose from 'mongoose';
3
4 const connectDB = async () => {
5   try {
6     // Use the connection string from .env to establish a connection to MongoDB.
7     await mongoose.connect(process.env.MONGO_URI, {
8       useNewUrlParser: true,
9       useUnifiedTopology: true,
10     });
11     console.log('✅ MongoDB Connected');
12   } catch (err) {
13     console.error('❌ MongoDB connection failed:', err);
14     process.exit(1); // Process exit with failure
15   }
16 };
17
18 export default connectDB;
```

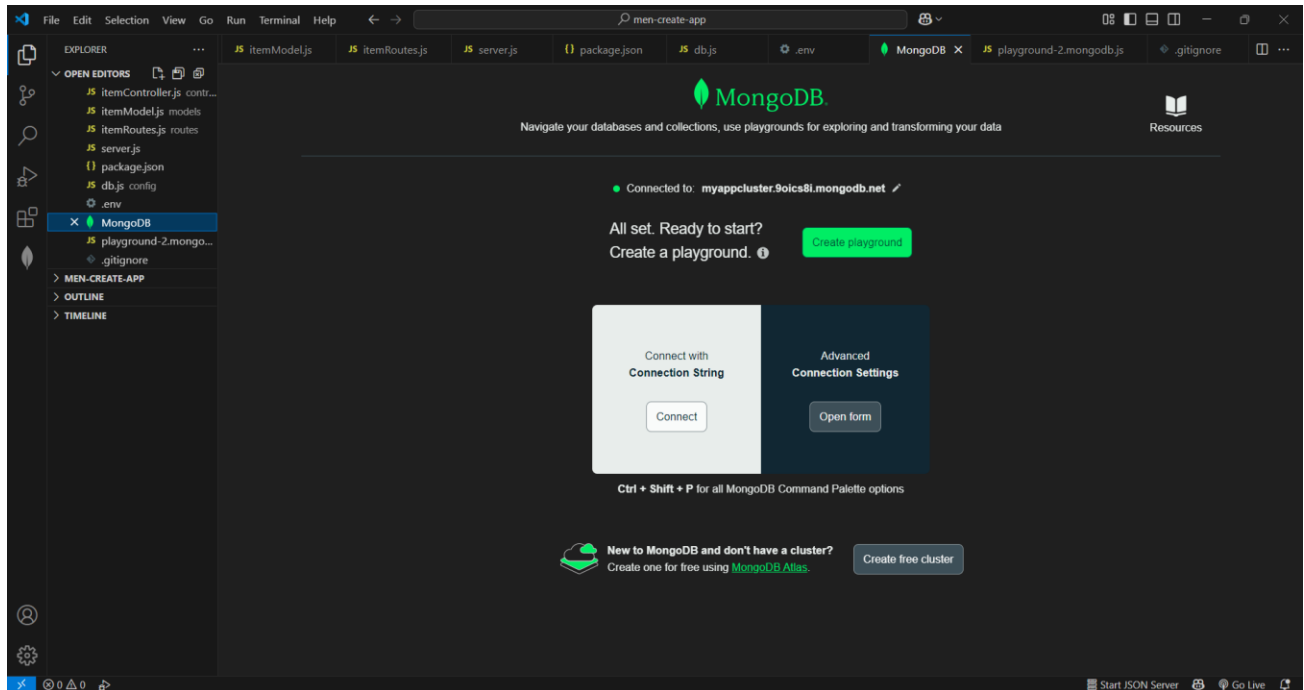
7. .env class



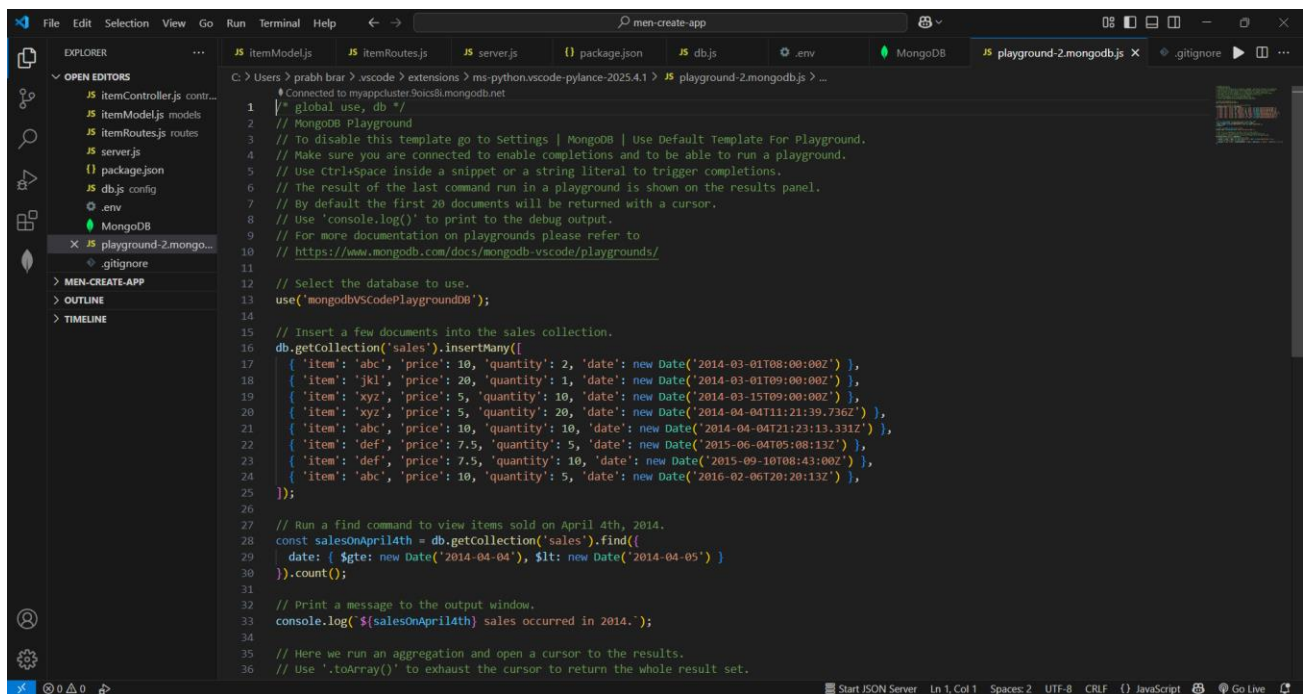
8. Gitignore file



9. MongoDB building connection and create playground

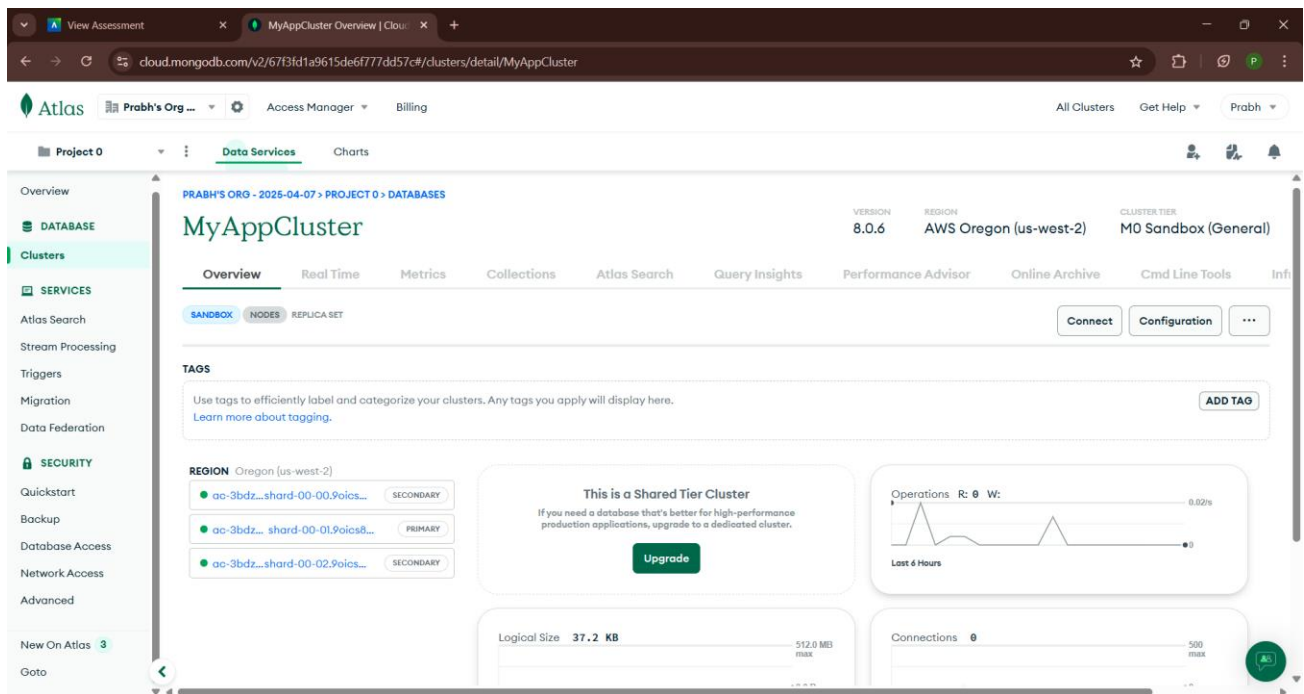


10. Playground-2.mongodb.js class

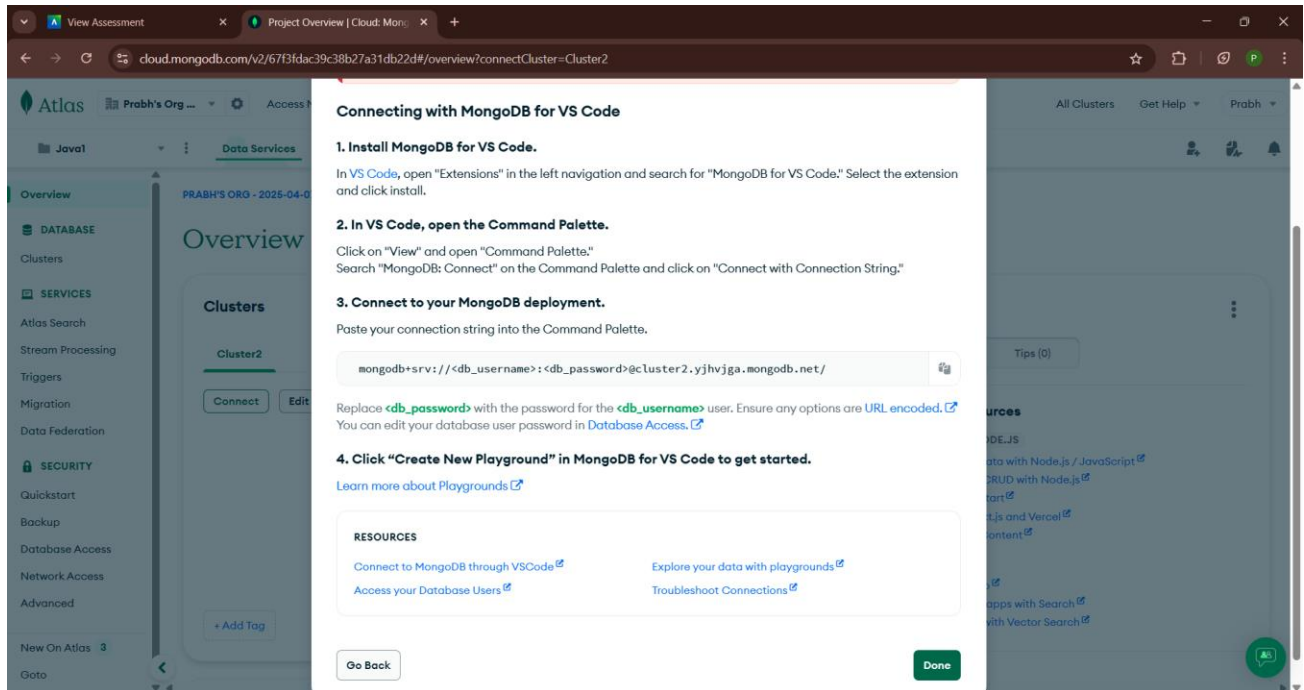


```
15 // Insert a few documents into the sales collection.
16 db.getCollection('sales').insertMany([
17   { 'item': 'abc', 'price': 10, 'quantity': 2, 'date': new Date('2014-03-01T08:00:00Z') },
18   { 'item': 'jkl', 'price': 20, 'quantity': 1, 'date': new Date('2014-03-01T09:00:00Z') },
19   { 'item': 'xyz', 'price': 5, 'quantity': 10, 'date': new Date('2014-03-15T09:00:00Z') },
20   { 'item': 'xyz', 'price': 5, 'quantity': 20, 'date': new Date('2014-04-04T11:21:39.726Z') },
21   { 'item': 'abc', 'price': 10, 'quantity': 10, 'date': new Date('2014-04-04T21:23:13.331Z') },
22   { 'item': 'def', 'price': 7.5, 'quantity': 5, 'date': new Date('2015-06-04T05:08:13Z') },
23   { 'item': 'def', 'price': 7.5, 'quantity': 10, 'date': new Date('2015-09-10T08:43:00Z') },
24   { 'item': 'abc', 'price': 10, 'quantity': 5, 'date': new Date('2016-02-06T20:13Z') },
25 ]);
26
27 // Run a find command to view items sold on April 4th, 2014.
28 const salesOnApril4th = db.getCollection('sales').find({
29   date: { $gte: new Date('2014-04-04'), $lt: new Date('2014-04-05') }
30 }).count();
31
32 // Print a message to the output window.
33 console.log(`${salesOnApril4th} sales occurred in 2014.`);
34
35 // Here we run an aggregation and open a cursor to the results.
36 // Use '.toArray()' to exhaust the cursor to return the whole result set.
37 // You can use '.hasNext().next()' to iterate through the cursor page by page.
38 db.getCollection('sales').aggregate([
39   // Find all of the sales that occurred in 2014.
40   { $match: { date: { $gte: new Date('2014-01-01'), $lt: new Date('2015-01-01') } } },
41   // Group the total sales for each product.
42   { $group: { _id: 'item', totalSaleAmount: { $sum: { $multiply: [ '$price', '$quantity' ] } } } }
43 ]);
44
```

11. MongoDB Atlas cluster making



12. Connection with VS Code



Output:

