Mongoose

Foundation Questions

1. **Mongoose** is an Object Data Modeling (ODM) library built for MongoDB and JavaScript. It's to help with data modeling, schema, model validation, and general data manipulation in MongoDB.

2. Database Interaction, Data Modeling and Validation, Querying and Poppulation, Middleware and hooks, Abstraction and Productivity.

3. **Mongoose schemas** are how to tell Mongoose what your documents look like. It is important to define the structure of our document and casting of properties, and also define the document instance of methods, static model methods, compound indexes, and document lifecycle hooks called middleware.

4. Models are modified constructors compiled from schema definitions. An instance of a model is called a document. Models are responsible for creating and reading documents from the underlying MongoDB database.

5. **save()** can be used to both create a new record and update an existing record. **create()** is used to create a new record by providing all required field at one time.

6. **CRUD operations(Create, Read, Update, Delete)**
   Create: In MongoDB, you can create a new document using the insertOne() or insertMany() methods.

   Read: You can read or retrieve documents from a MongoDB database using the find() method.

   Update: Updating documents in MongoDB can be done using the updateOne(), updateMany(), or replaceOne() methods.

   Delete: You can delete documents using the deleteOne() or deleteMany() methods.

7. **Middleware**, also known as pre and post-hooks, are functions that intercept the execution of asynchronous operations in Mongoose. It acts as a bridge between diverse technologies, tools, and databases

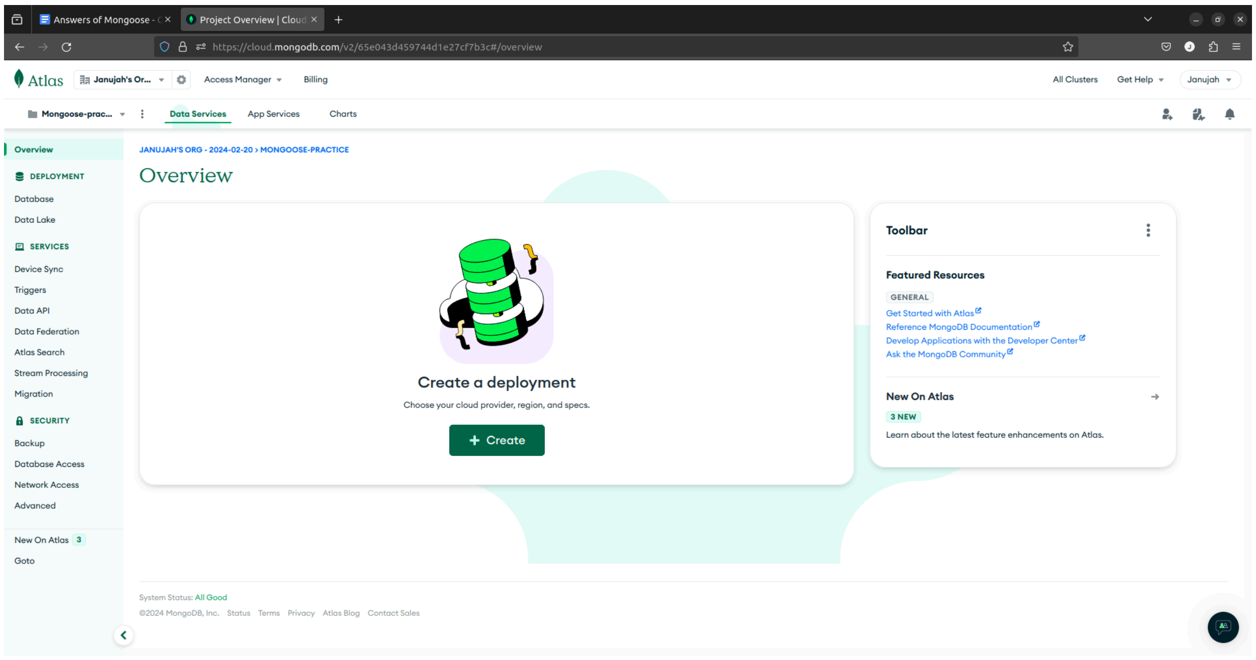
8. mongoose.connect(insert your MongoDB Atlas URI)

9.

Deep Dive into the Concepts

1. Async is multi-thread, which means operations or programs can run in parallel. Sync is a single thread, so only one operation or program will run at a time

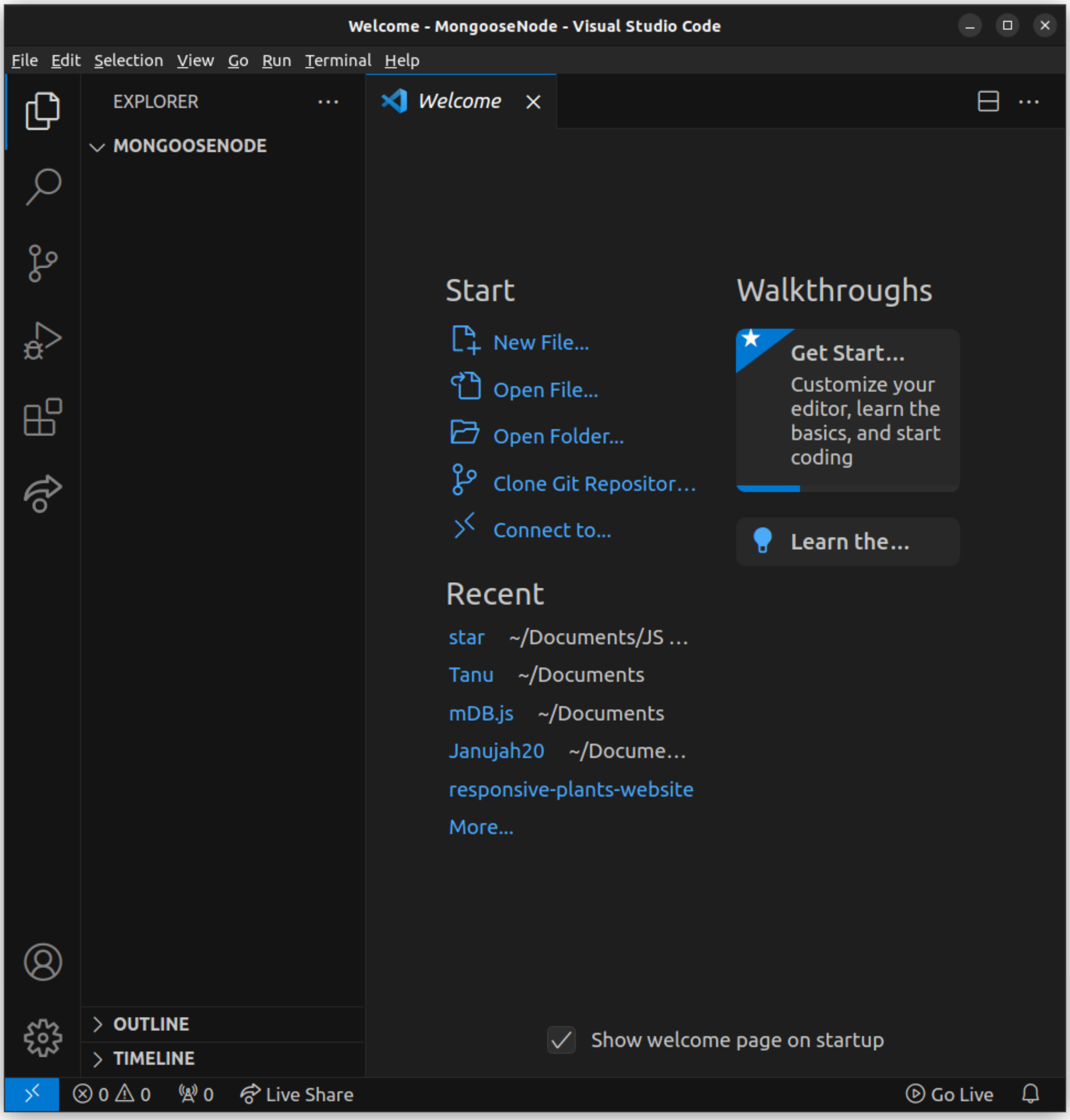
2.



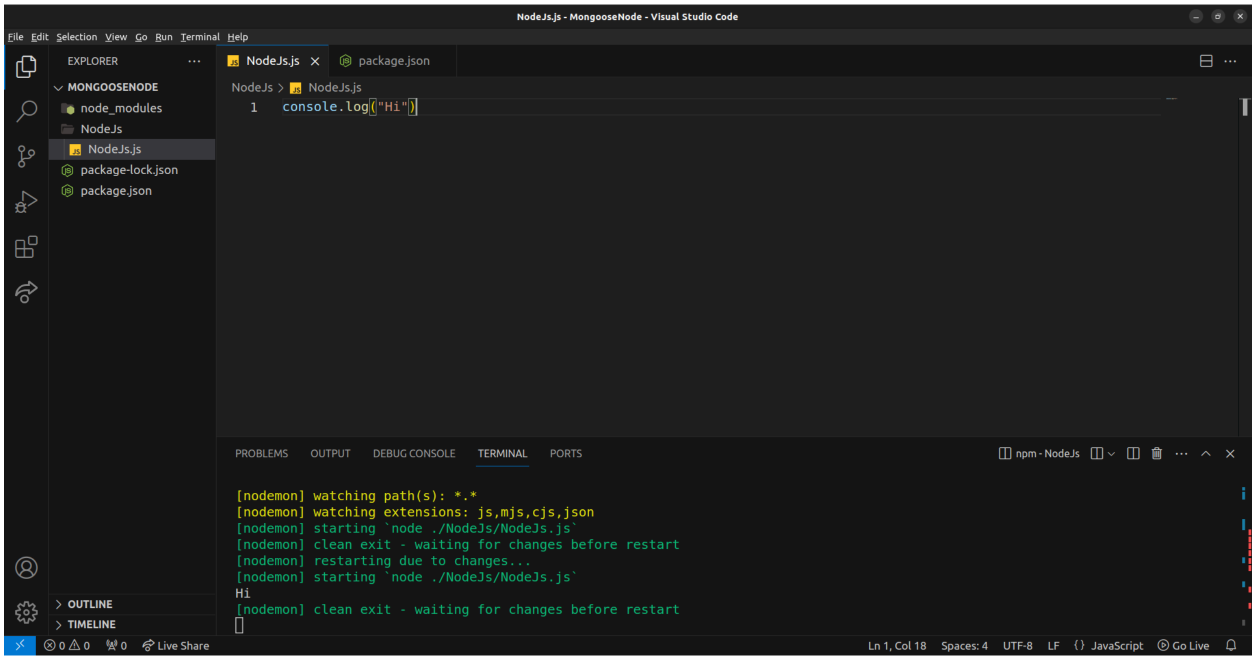
Choose your cloud provider, region, and specs.

Learn about the latest feature enhancements on Atlas.

3.


Welcome - MongooseNode - Visual Studio Code
File Edit Selection View Go Run Terminal Help
EXPLORER
MONGOOSENODE
Welcome
Start
New File...
Open File...
Open Folder...
Clone Git Repositor…
Connect to...
Walkthroughs
Get Start…
Customize your editor, learn the basics, and start coding
Learn the…
Recent
star ~/Documents/JS …
Tanu ~/Documents
mDB.js ~/Documents
Janujah20 ~/Docume…
responsive-plants-website
More...
Show welcome page on startup
OUTLINE
TIMELINE
0 0 0
Live Share
Go Live

4.



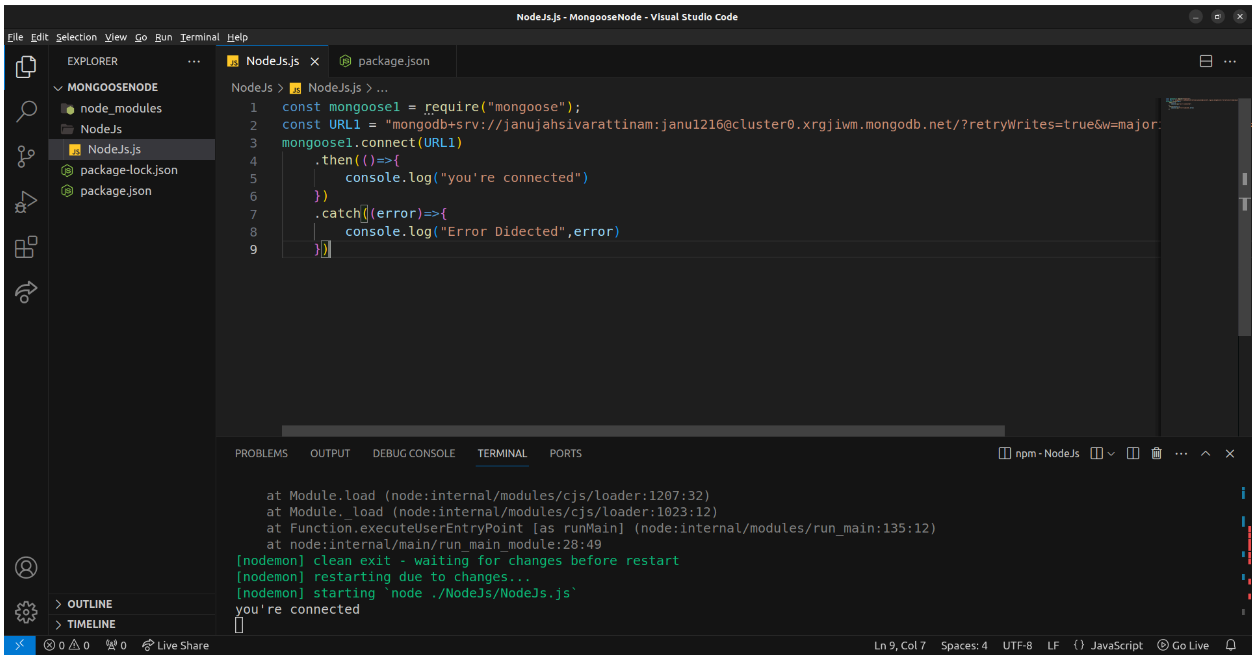
NodeJs.js - MongooseNode - Visual Studio Code
File Edit Selection View Go Run Terminal Help
EXPLORER
MONGOOSENODE
node_modules
NodeJs
NodeJs.js
package-lock.json
package.json
OUTLINE
TIMELINE
NodeJs.js
package.json
NodeJs > NodeJs.js
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS
npm - NodeJs
Ln 1, Col 18 Spaces: 4 UTF-8 LF JavaScript Go Live
Live Share

```
console.log("Hi")
```

```
[nodemon] watching path(s): *.*
[nodemon] watching extensions: js,mjs,cjs,json
[nodemon] starting `node ./NodeJs/NodeJs.js`
[nodemon] clean exit - waiting for changes before restart
[nodemon] restarting due to changes...
[nodemon] starting `node ./NodeJs/NodeJs.js`
Hi
[nodemon] clean exit - waiting for changes before restart
```

5. **Nodemon** is a command line tool that helps develop Node.js based applications by automatically restarting the code application when file changes in the directory are detected.

6.



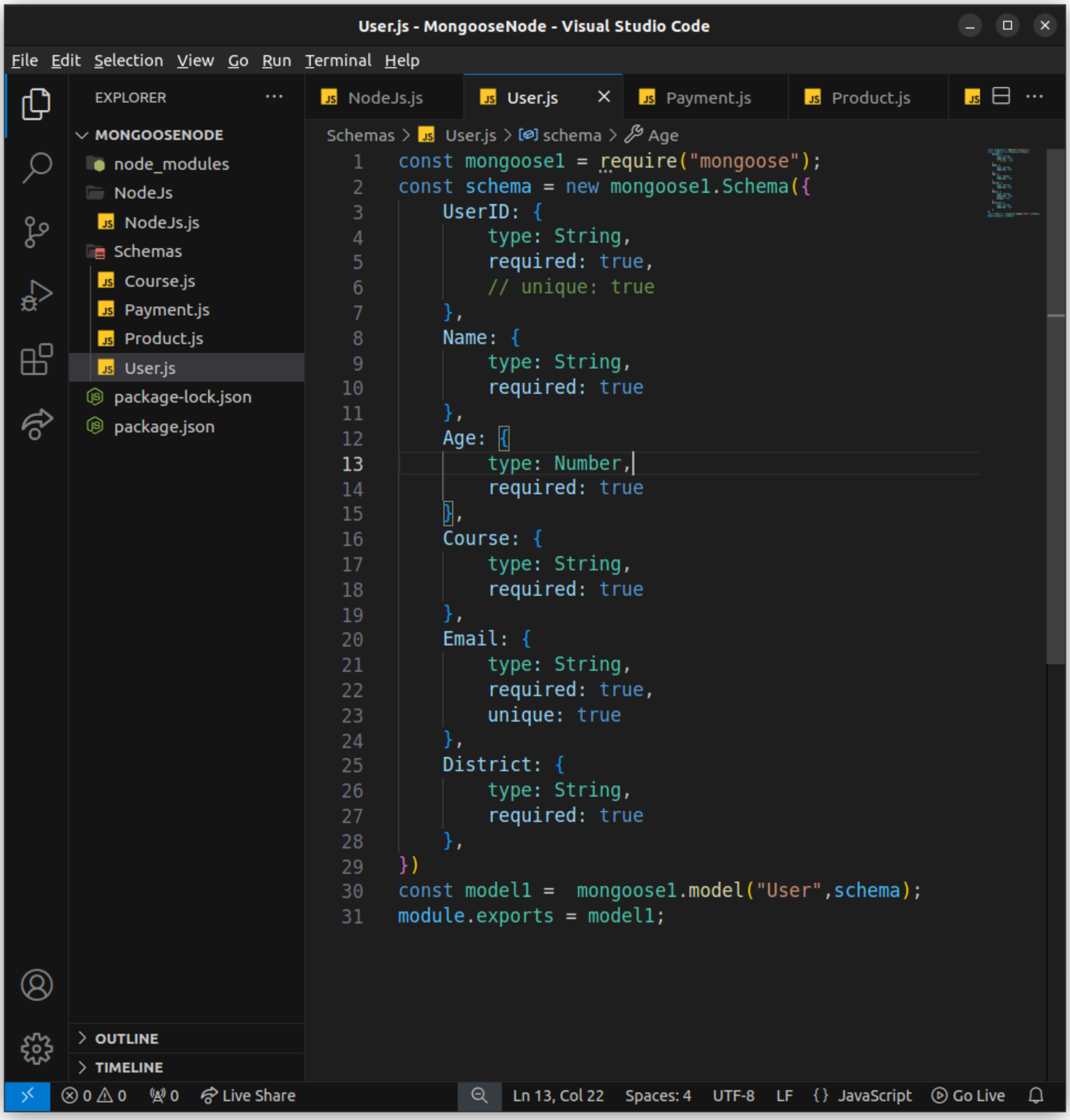
NodeJs.js - MongooseNode - Visual Studio Code
File Edit Selection View Go Run Terminal Help
EXPLORER
MONGOOSENODE
node_modules
NodeJs
NodeJs.js
package-lock.json
package.json
OUTLINE
TIMELINE
NodeJs.js
package.json
NodeJs > NodeJs.js > ...
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS
npm - NodeJs
Ln 9, Col 7 Spaces: 4 UTF-8 LF JavaScript Go Live
Live Share

```
const mongoose1 = require("mongoose");
const URL1 = "mongodb+srv://janujahsivarattinam:janu1216@cluster0.xrgjiwm.mongodb.net/?retryWrites=true&w=major
mongoose1.connect(URL1)
    .then(()=>{
        console.log("you're connected")
    })
    .catch((error)=>{
        console.log("Error Didected",error)
    })
```

```
    at Module.load (node:internal/modules/cjs/loader:1207:32)
    at Module._load (node:internal/modules/cjs/loader:1023:12)
    at Function.executeUserEntryPoint [as runMain] (node:internal/modules/run_main:135:12)
    at node:internal/main/run_main_module:28:49
[nodemon] clean exit - waiting for changes before restart
[nodemon] restarting due to changes...
[nodemon] starting `node ./NodeJs/NodeJs.js`
you're connected
```

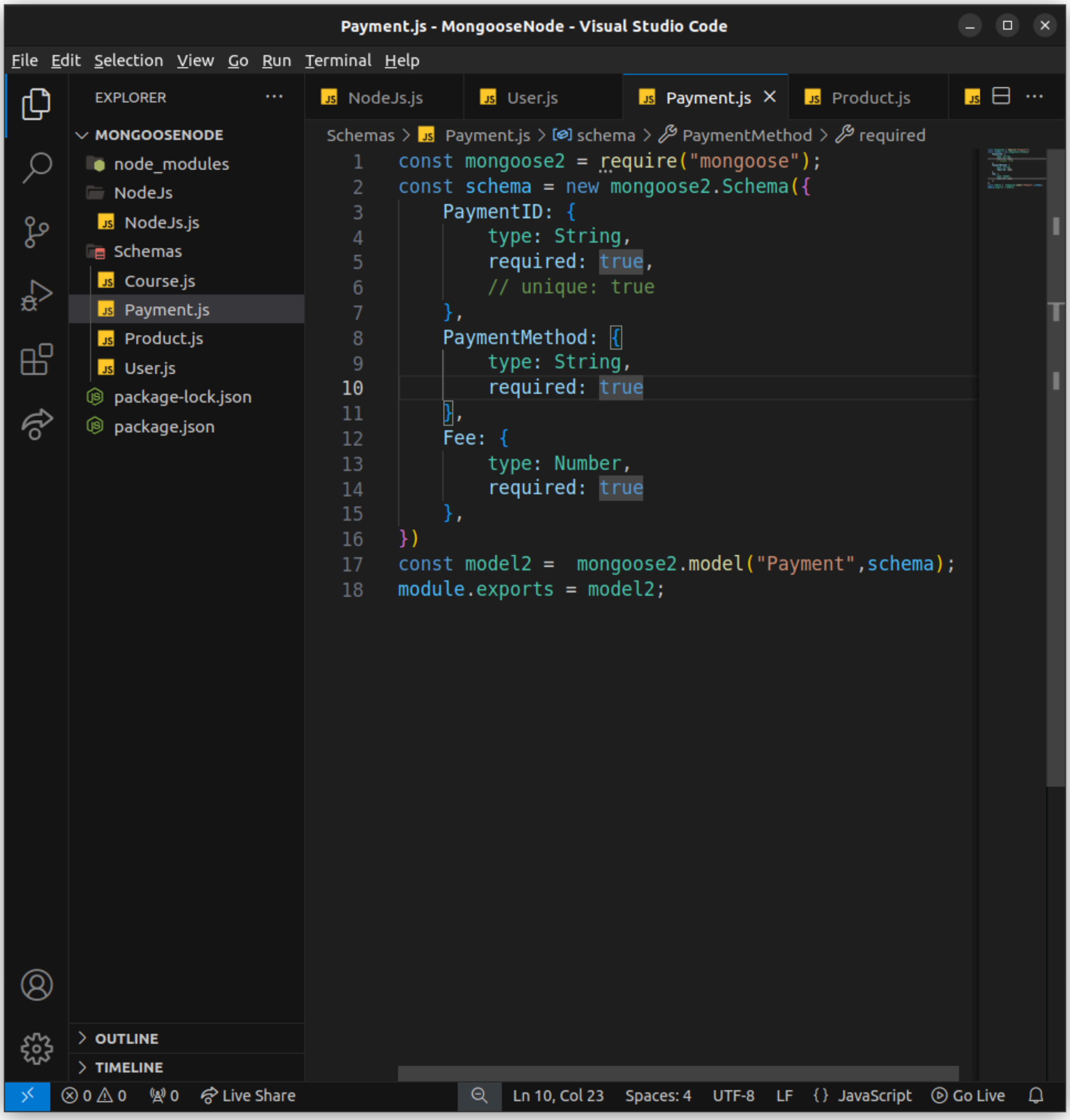
7.

a. User's Schema



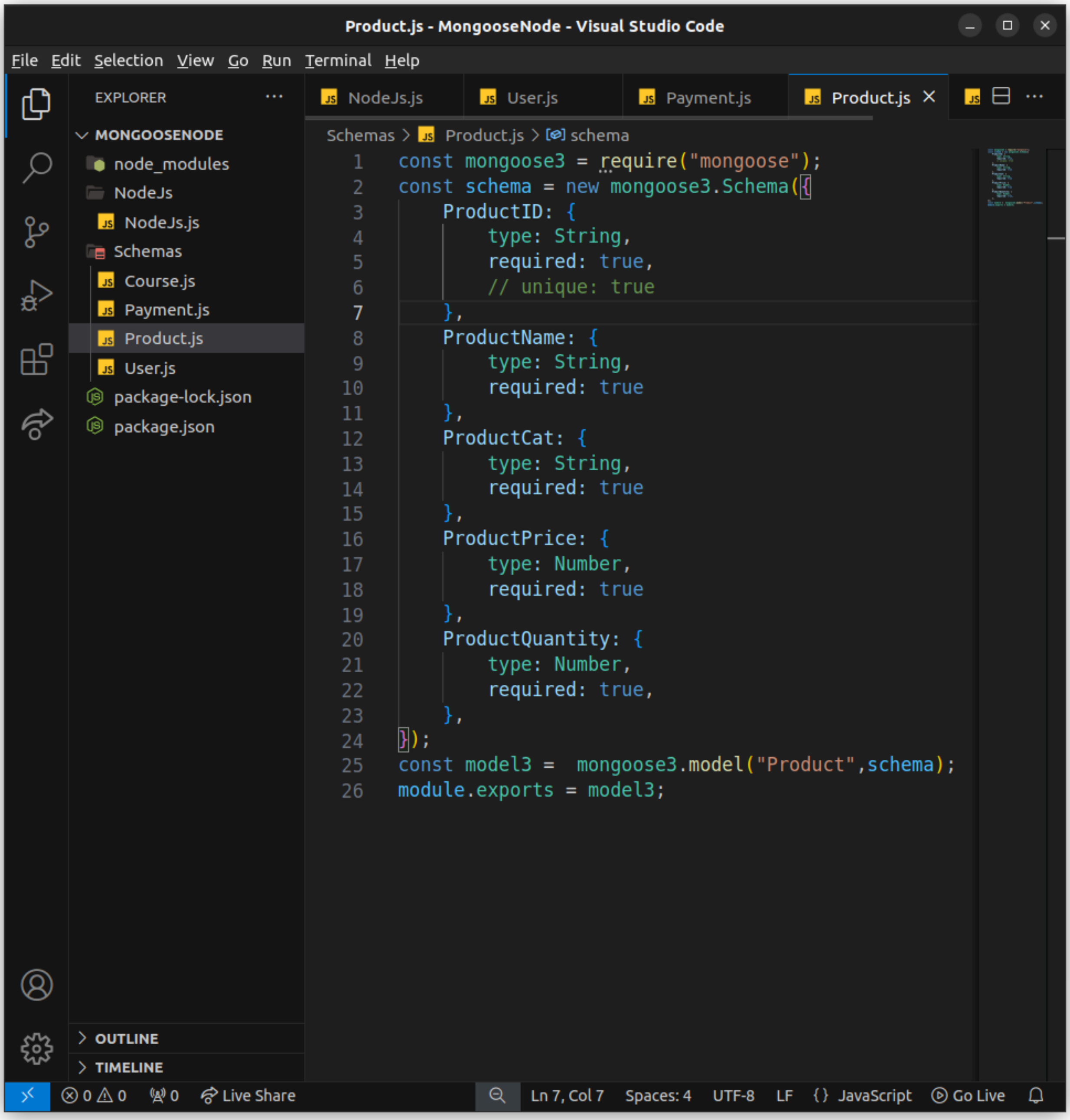
```
const mongoose1 = require("mongoose");
const schema = new mongoose1.Schema({
    UserID: {
        type: String,
        required: true,
        // unique: true
    },
    Name: {
        type: String,
        required: true
    },
    Age: {
        type: Number,
        required: true
    },
    Course: {
        type: String,
        required: true
    },
    Email: {
        type: String,
        required: true,
        unique: true
    },
    District: {
        type: String,
        required: true
    },
})
const model1 =  mongoose1.model("User",schema);
module.exports = model1;
```

b. Payment’s Schema


Payment.js - MongooseNode - Visual Studio Code
File Edit Selection View Go Run Terminal Help
EXPLORER
MONGOOSENODE
node_modules
NodeJs
NodeJs.js
Schemas
Course.js
Payment.js
Product.js
User.js
package-lock.json
package.json
OUTLINE
TIMELINE
NodeJs.js
User.js
Payment.js
Product.js
Schemas > Payment.js > schema > PaymentMethod > required
Ln 10, Col 23 Spaces: 4 UTF-8 LF {} JavaScript Go Live
Live Share

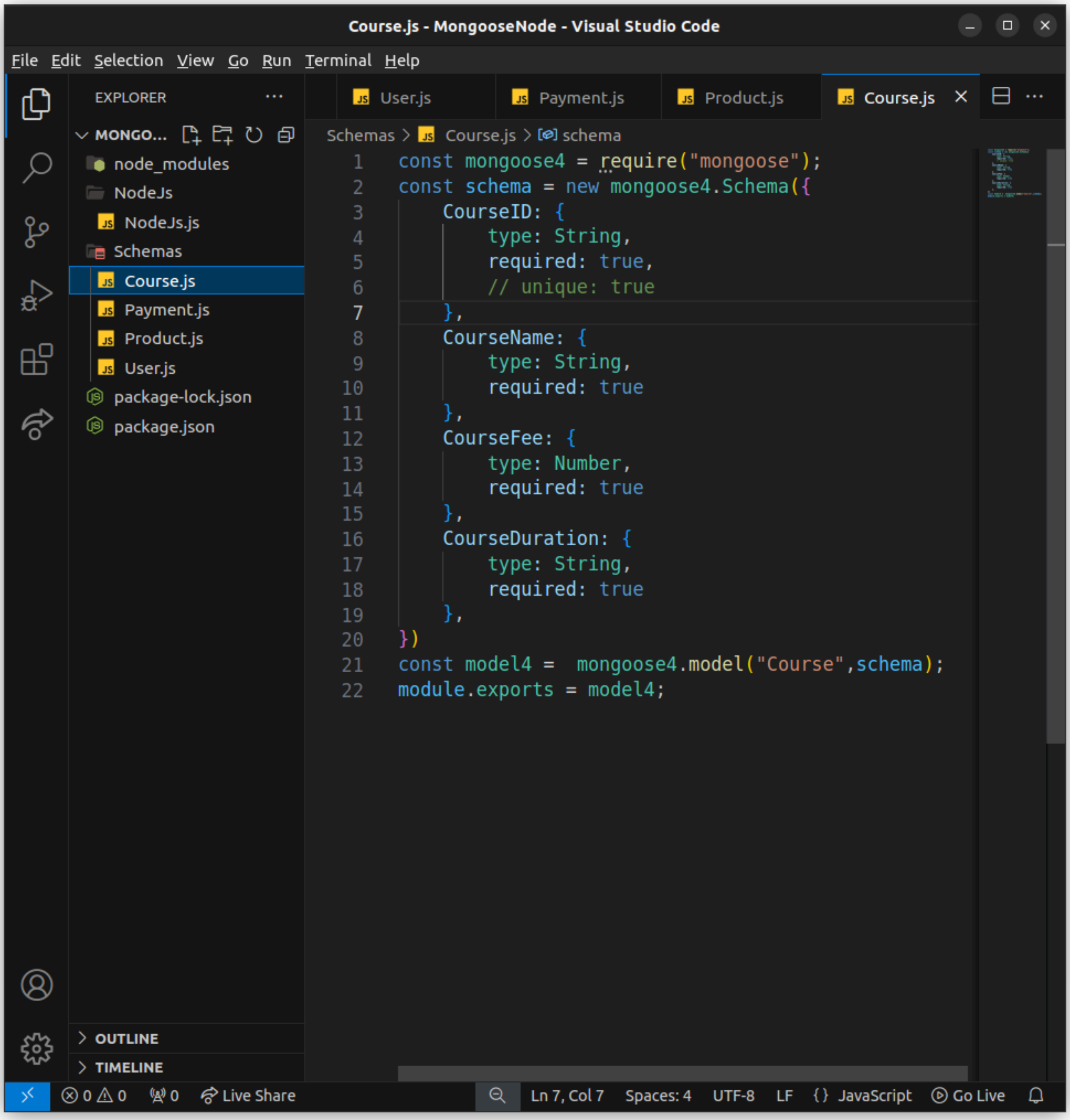
```
const mongoose2 = require("mongoose");
const schema = new mongoose2.Schema({
    PaymentID: {
        type: String,
        required: true,
        // unique: true
    },
    PaymentMethod: {
        type: String,
        required: true
    },
    Fee: {
        type: Number,
        required: true
    },
})
const model2 =  mongoose2.model("Payment",schema);
module.exports = model2;
```

c. Product’s Schema



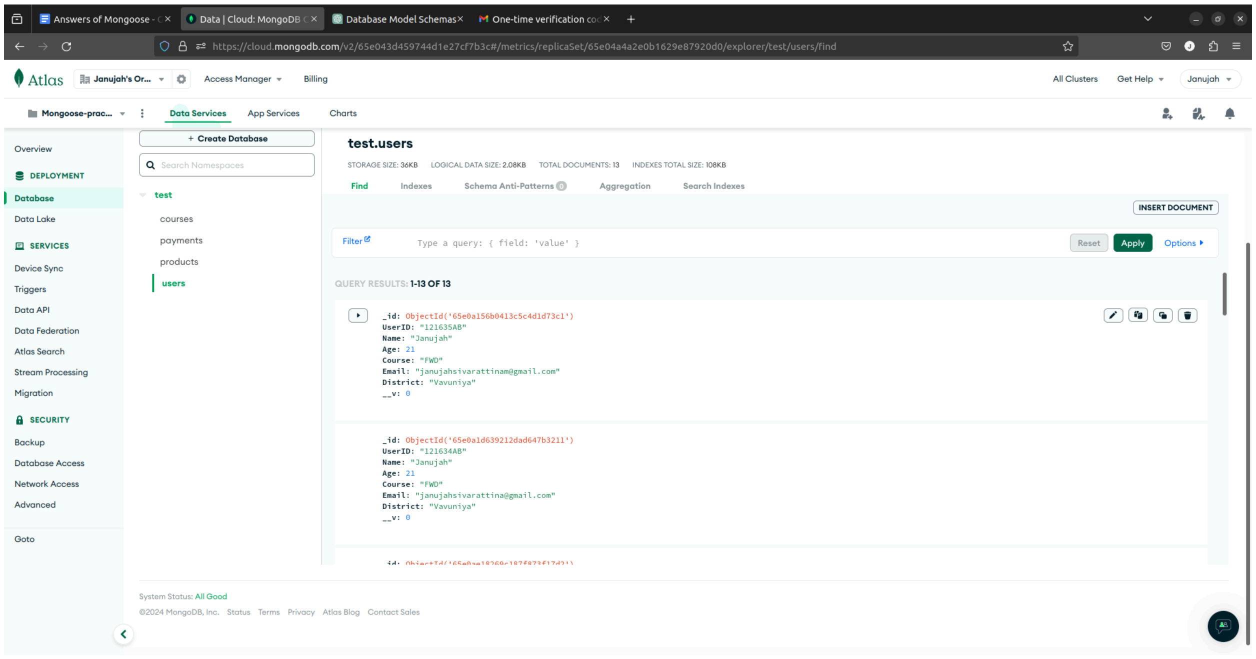
```
const mongoose3 = require("mongoose");
const schema = new mongoose3.Schema({
    ProductID: {
        type: String,
        required: true,
        // unique: true
    },
    ProductName: {
        type: String,
        required: true
    },
    ProductCat: {
        type: String,
        required: true
    },
    ProductPrice: {
        type: Number,
        required: true
    },
    ProductQuantity: {
        type: Number,
        required: true,
    },
});
const model3 =  mongoose3.model("Product",schema);
module.exports = model3;
```

d. Course’s Schema



```
const mongoose4 = require("mongoose");
const schema = new mongoose4.Schema({
    CourseID: {
        type: String,
        required: true,
        // unique: true
    },
    CourseName: {
        type: String,
        required: true
    },
    CourseFee: {
        type: Number,
        required: true
    },
    CourseDuration: {
        type: String,
        required: true
    },
})
const model4 =  mongoose4.model("Course",schema);
module.exports = model4;
```

8. Users



test.users

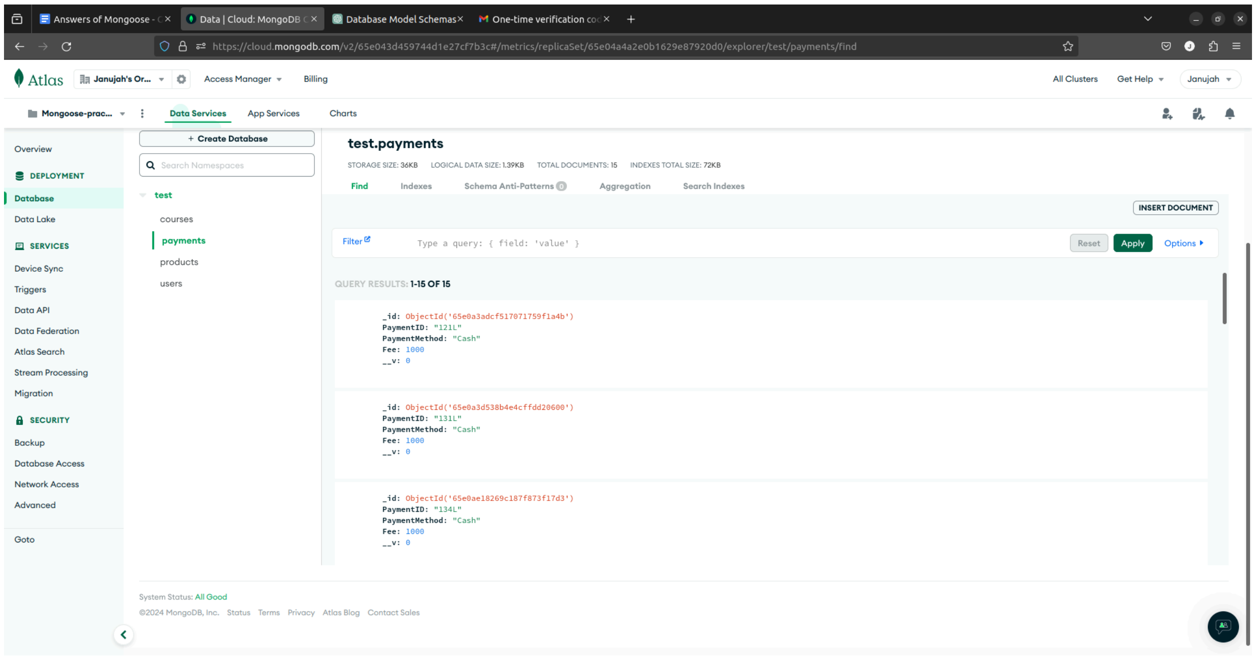
STORAGE SIZE: 36KB LOGICAL DATA SIZE: 2.08KB TOTAL DOCUMENTS: 13 INDEXES TOTAL SIZE: 108KB

QUERY RESULTS: 1-13 OF 13

```
_id: ObjectId('65e0a156b0413c5c4d1d73c1')
UserID: "121635AB"
Name: "Janujah"
Age: 21
Course: "FWD"
Email: "janujahsivarattinam@gmail.com"
District: "Vavuniya"
__v: 0

_id: ObjectId('65e0a1d639212dad647b3211')
UserID: "121634AB"
Name: "Janujah"
Age: 21
Course: "FWD"
Email: "janujahsivarattina@gmail.com"
District: "Vavuniya"
__v: 0
```

Payments



test.payments

STORAGE SIZE: 36KB LOGICAL DATA SIZE: 1.39KB TOTAL DOCUMENTS: 15 INDEXES TOTAL SIZE: 72KB

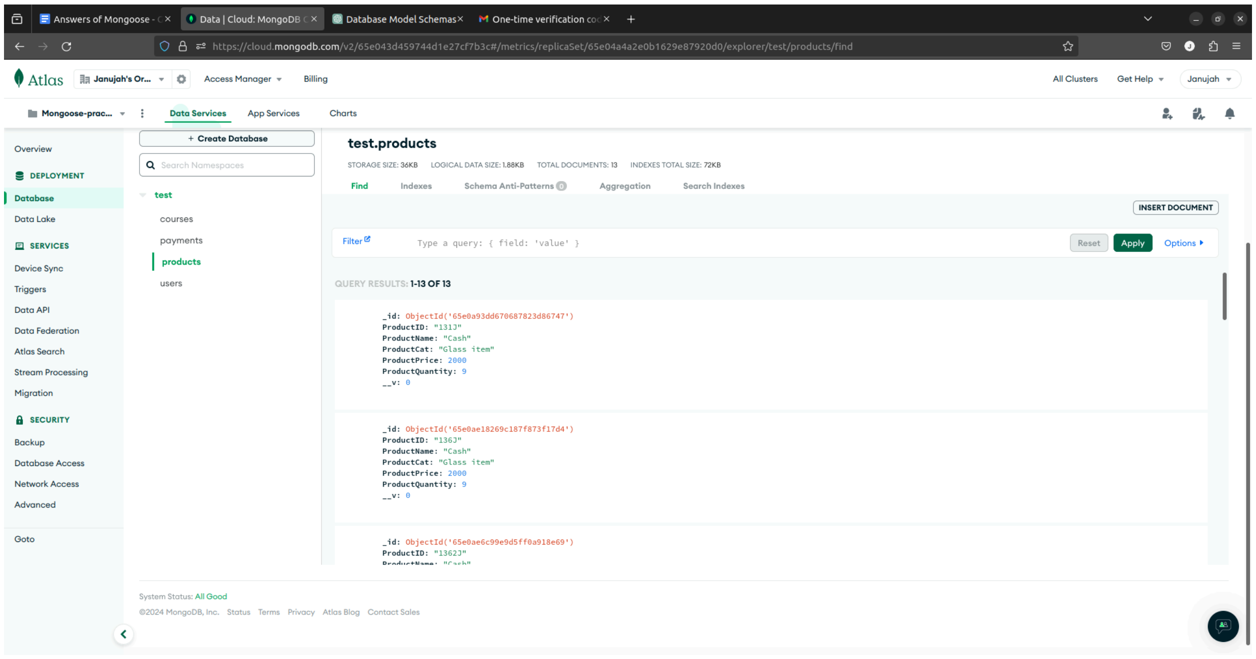
QUERY RESULTS: 1-15 OF 15

```
_id: ObjectId('65e0a3adcf517071759f1a4b')
PaymentID: "121L"
PaymentMethod: "Cash"
Fee: 1000
__v: 0

_id: ObjectId('65e0a3d538b4e4cffdd20600')
PaymentID: "131L"
PaymentMethod: "Cash"
Fee: 1000
__v: 0

_id: ObjectId('65e0ae18269c187f873f17d3')
PaymentID: "134L"
PaymentMethod: "Cash"
Fee: 1000
__v: 0
```

Products



https://cloud.mongodb.com/v2/65e043d459744d1e27cf7b3c#/metrics/replicaSet/65e04a4a2e0b1629e87920d0/explorer/test/products/find

test.products

STORAGE SIZE: 36KB LOGICAL DATA SIZE: 1.88KB TOTAL DOCUMENTS: 13 INDEXES TOTAL SIZE: 72KB

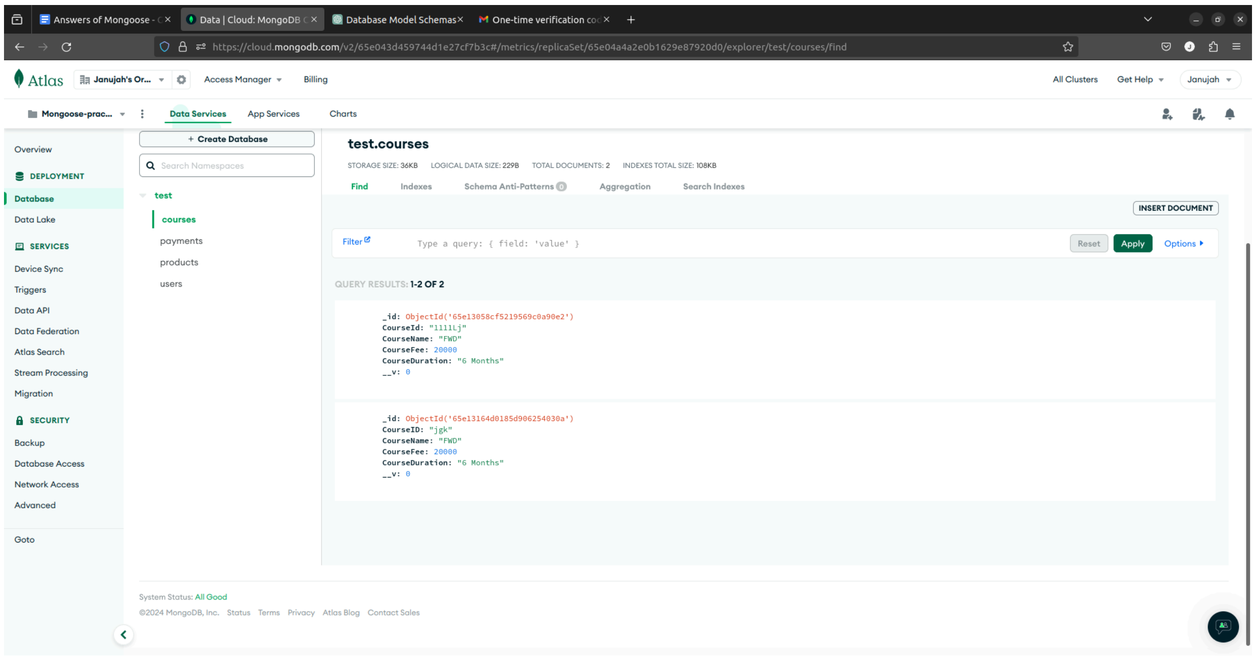
QUERY RESULTS: 1-13 OF 13

```
_id: ObjectId('65e0a93dd670687823d86747')
ProductID: "131J"
ProductName: "Cash"
ProductCat: "Glass item"
ProductPrice: 2000
ProductQuantity: 9
__v: 0

_id: ObjectId('65e0ae18269c187f873f17d4')
ProductID: "136J"
ProductName: "Cash"
ProductCat: "Glass item"
ProductPrice: 2000
ProductQuantity: 9
__v: 0

_id: ObjectId('65e0ae6c99e9d5ff0a918e69')
ProductID: "1362J"
```

Courses



https://cloud.mongodb.com/v2/65e043d459744d1e27cf7b3c#/metrics/replicaSet/65e04a4a2e0b1629e87920d0/explorer/test/courses/find

test.courses

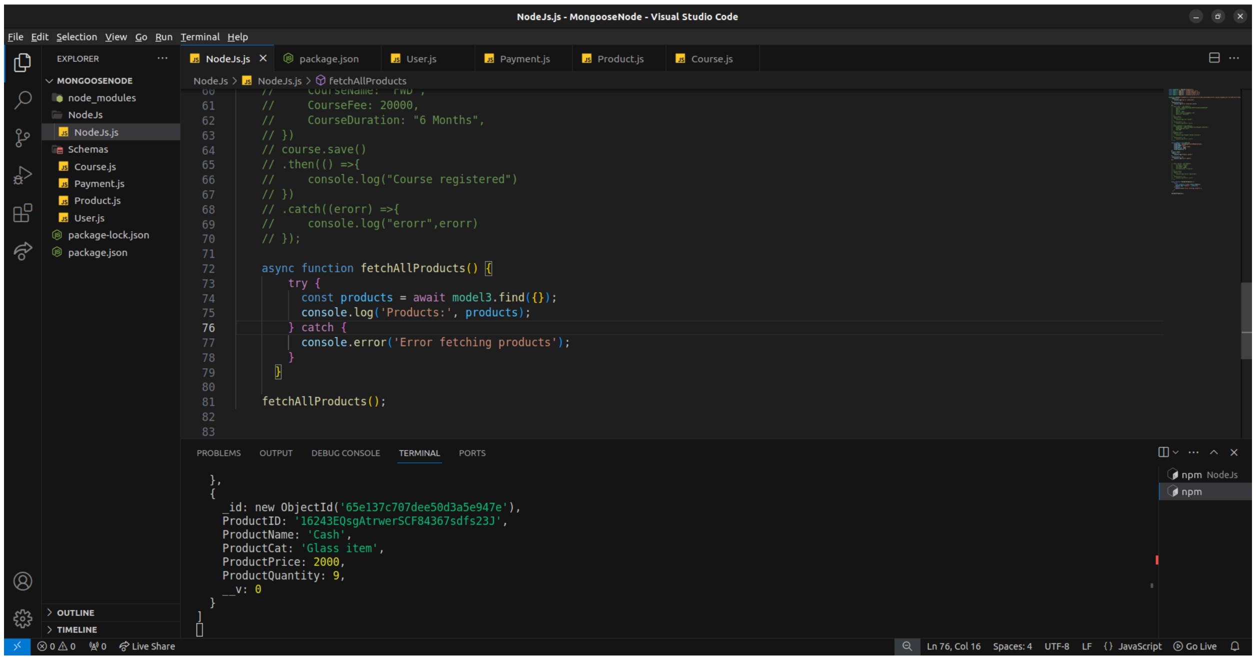
STORAGE SIZE: 36KB LOGICAL DATA SIZE: 229B TOTAL DOCUMENTS: 2 INDEXES TOTAL SIZE: 108KB

QUERY RESULTS: 1-2 OF 2

```
_id: ObjectId('65e13058cf5219569c0a90e2')
CourseId: "1111Lj"
CourseName: "FWD"
CourseFee: 20000
CourseDuration: "6 Months"
__v: 0

_id: ObjectId('65e13164d0185d906254030a')
CourseID: "jgk"
CourseName: "FWD"
CourseFee: 20000
CourseDuration: "6 Months"
__v: 0
```

9.



NodeJs.js - MongooseNode - Visual Studio Code

```javascript
// 	CourseFee: 20000,
// 	CourseDuration: "6 Months",
// })
// course.save()
// .then(() =>{
// 	console.log("Course registered")
// })
// .catch((erorr) =>{
// 	console.log("erorr",erorr)
// });

async function fetchAllProducts() {
    try {
      const products = await model3.find({});
      console.log('Products:', products);
    } catch {
      console.error('Error fetching products');
    }
  }

fetchAllProducts();
```

```
  },
  {
    _id: new ObjectId('65e137c707dee50d3a5e947e'),
    ProductID: '16243EQsgAtrwerSCF84367sdfs23J',
    ProductName: 'Cash',
    ProductCat: 'Glass item',
    ProductPrice: 2000,
    ProductQuantity: 9,
    __v: 0
  }
]
```