GRAPHQL using Apollo Server and Connected with MongoDB

Index.js

```
const { ApolloServer, gql } = require('apollo-server');
const mongoose = require('mongoose');
const User = require('./user');

// Connect to MongoDB (local or Atlas)
mongoose.connect('mongodb://127.0.0.1:27017/graphql_demo', {
  useNewUrlParser: true,
  useUnifiedTopology: true
}).then(() => {
  console.log("☑ MongoDB Connected");
}).catch(err => console.error(err));

// Schema definition
const typeDefs = gql`
  type User {
    id: ID!
    name: String!
    email: String!
  }

  type Query {
    user(id: ID!): User
    users: [User]
  }

  type Mutation {
  addUser(name: String!, email: String!): User
  updateUser(id: ID!, name: String, email: String): User
  deleteUser(id: ID!): User
}

`;

// Resolvers
const resolvers = {
  Query: {
    user: async (_, { id }) => await User.findById(id),
    users: async () => await User.find()
  },
  Mutation: {
    addUser: async (_, { name, email }) => {
      const user = new User({ name, email });
      return await user.save();
    },
    updateUser: async (_, { id, name, email }) => {
```

```
      const user = await User.findById(id);
      if (!user) return null;
      if (name) user.name = name;
      if (email) user.email = email;
      return await user.save();
    },
    deleteUser: async (_, { id }) => {
      return await User.findByIdAndDelete(id);
    }
  }
};


// Create and start the server
const server = new ApolloServer({ typeDefs, resolvers });

server.listen().then(({ url }) => {
  console.log(` Server ready at ${url}`);
});
```

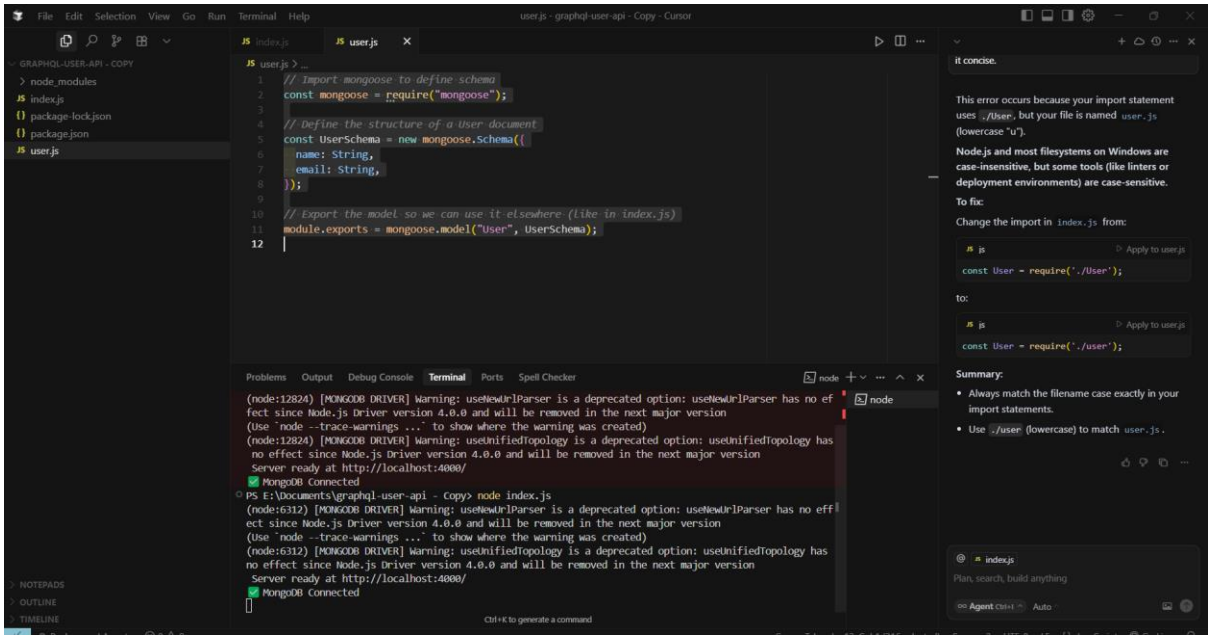
User.js

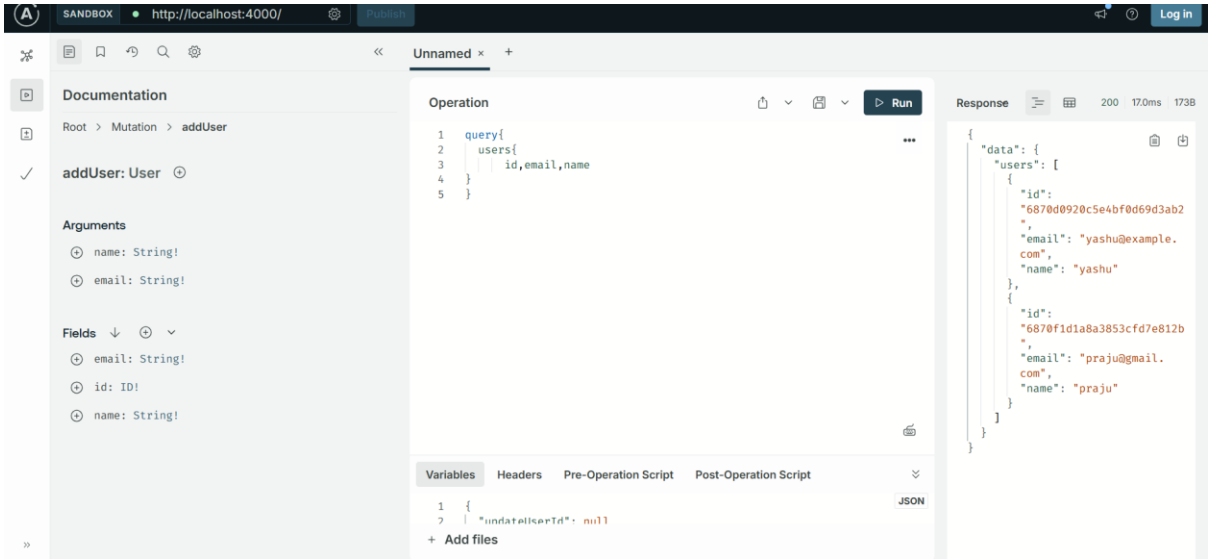
```
// Import mongoose to define schema
const mongoose = require("mongoose");

// Define the structure of a User document
const UserSchema = new mongoose.Schema({
  name: String,
  email: String,
});

// Export the model so we can use it elsewhere (like in index.js)
module.exports = mongoose.model("User", UserSchema);
```

Output:


File Edit Selection View Go Run Terminal Help
user.js - graphql-user-api - Copy - Cursor
GRAPHQL-USER-API - COPY
node_modules
index.js
package-lock.json
package.json
user.js
// Import mongoose to define schema
const mongoose = require("mongoose");
// Define the structure of a User document
const UserSchema = new mongoose.Schema({
name: String,
email: String,
});
// Export the model so we can use it elsewhere (like in index.js)
module.exports = mongoose.model("User", UserSchema);
Problems Output Debug Console Terminal Ports Spell Checker
(node:12824) [MONGODB DRIVER] Warning: useNewUrlParser is a deprecated option: useNewUrlParser has no effect since Node.js Driver version 4.0.0 and will be removed in the next major version
(Use `node --trace-warnings ...` to show where the warning was created)
(node:12824) [MONGODB DRIVER] Warning: useUnifiedTopology is a deprecated option: useUnifiedTopology has no effect since Node.js Driver version 4.0.0 and will be removed in the next major version
Server ready at http://localhost:4000/
MongoDB Connected
PS E:\Documents\graphql-user-api - Copy> node index.js
(node:6312) [MONGODB DRIVER] Warning: useNewUrlParser is a deprecated option: useNewUrlParser has no effect since Node.js Driver version 4.0.0 and will be removed in the next major version
(Use `node --trace-warnings ...` to show where the warning was created)
(node:6312) [MONGODB DRIVER] Warning: useUnifiedTopology is a deprecated option: useUnifiedTopology has no effect since Node.js Driver version 4.0.0 and will be removed in the next major version
Server ready at http://localhost:4000/
MongoDB Connected
Ctrl+K to generate a command
it concise.
This error occurs because your import statement uses ./User, but your file is named user.js (lowercase "u").
Node.js and most filesystems on Windows are case-insensitive, but some tools (like linters or deployment environments) are case-sensitive.
To fix:
Change the import in index.js from:
const User = require('./User');
to:
const User = require('./user');
Summary:
Always match the filename case exactly in your import statements.
Use ./user (lowercase) to match user.js.
Plan, search, build anything
Agent Auto


SANDBOX http://localhost:4000/
Publish
Log in
Unnamed
Documentation
Root > Mutation > addUser
addUser: User
Arguments
name: String!
email: String!
Fields
email: String!
id: ID!
name: String!
Operation
Run
query{
users{
id,email,name
}
}
Variables Headers Pre-Operation Script Post-Operation Script
JSON
Add files
Response
200 17.0ms 173B
{
"data": {
"users": [
{
"id": "6870d0920c5e4bf0d69d3ab2",
"email": "yashu@example.com",
"name": "yashu"
},
{
"id": "6870f1d1a8a3853cfd7e812b",
"email": "praju@gmail.com",
"name": "praju"
}
]
}
}