

Graphiql is dummy front end application to test queries

**Steps**

Install node first

Create folder graphql

Create folder server inside graphql

Npm init

Enter 6 times

Inside server there is packge .jspn fie where all info is saved

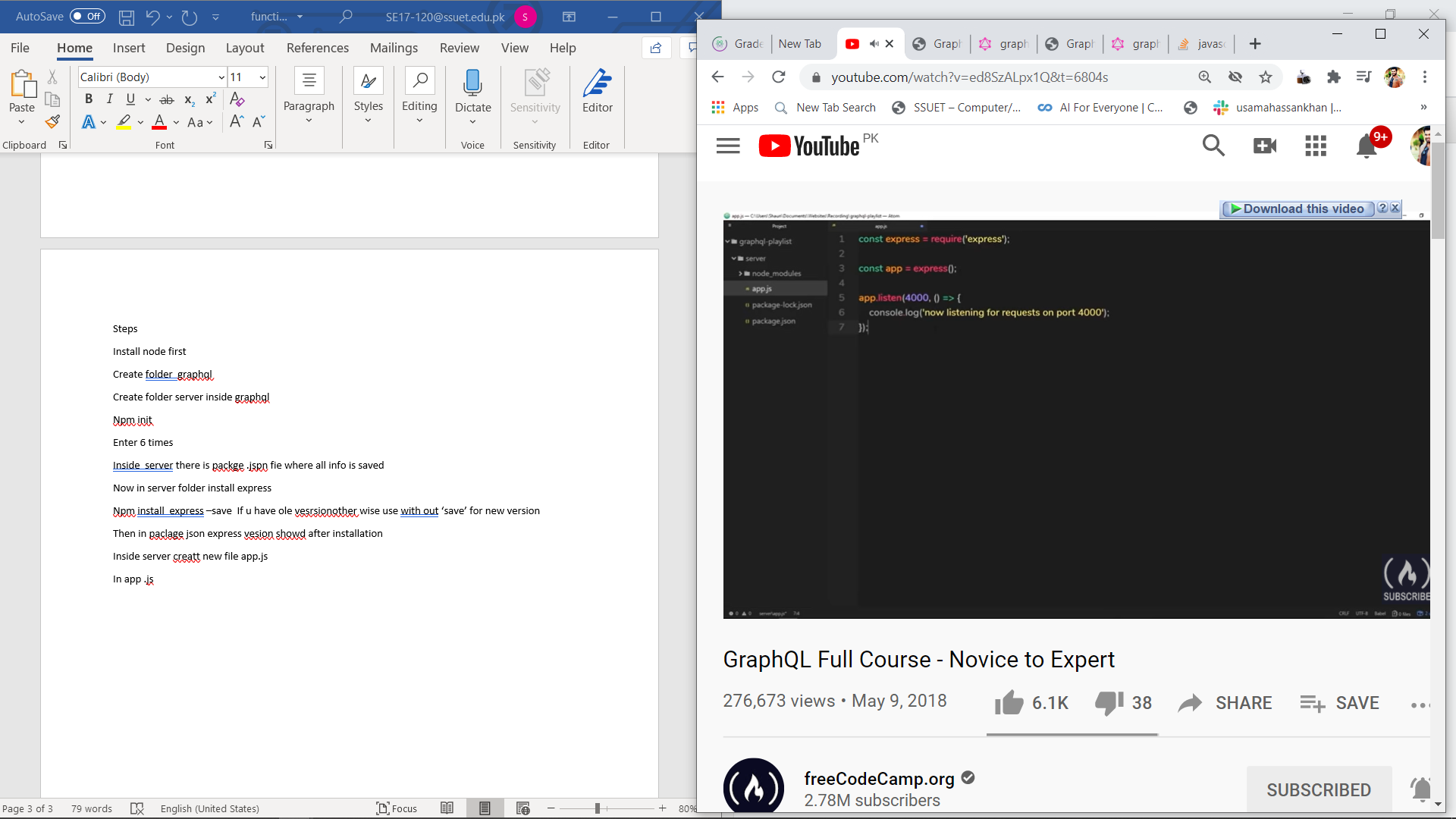
Now in server folder install express

Npm install express –save If u have ole vesrsionother wise use with out ‘save’ for new version

Then in paclage json express vesion showd after installation

Inside server creatt new file app.js

In app .js



Then type node app and run localhost 4000 got a message now listinig for request on port 4000

Install nodemon

Npm install nodemon -g

Type **nodemon app.js or app to run server**

**Npm install graphql expresss-graphql**

**Express-graphql for undersd express about graphql**

**In app.js**

const express = require('express');

const { graphqlHTTP } = require('express-graphql');

const schema=require('./schema/schema');

const app =express();

app.use(

  '/graphql',

  graphqlHTTP({

    schema:schema,

    graphiql:true

  }),

);

app.listen(4000,()=>{

  console.log('now listning request on port 4000')

});

**Link express graph ql**

**https://github.com/graphql/express-graphql?fbclid=IwAR1FxUeL34DmqIUf6dYAyndWj9s1\_5x9F1byw0JKs9ZfeOpEMsULe6FVKCg**

**Schema**

**Install lodash which helpd to find or get data from array**

**Npm install lodash**

**npm i underscore –save**

const graphql =require("graphql");

const\_=require("lodash");

const { GraphQLObjectType,GraphQLString,GraphQLSchema }=graphql;

var books=[

  {name:'name of the wind',genre:'fantsy',id :'1'},

  {name:'name of the empire',genre:'sci fic',id :'2'},

  {name:'name of the shore',genre:'novel',id :'3'},

]

var \_ = require('underscore');

const BookType = new GraphQLObjectType({

  name: "Book",

  fields:()=>({

    id: { type:GraphQLString},

    name: { type:GraphQLString},

    genre: { type:GraphQLString},

  }),

});

const RootQuery = new GraphQLObjectType({

  name: "RootQueryType",

  fields: {

    book: {

      type: BookType,

      args: { id: { type: GraphQLString } },

      resolve(parent, args) {

*//code to get data from db*

*return* \_.find(books,{id:args.id});

      },

    },

   author:{

      type:AuthorType,

      args:{id:{type:GraphQLID}},

      resolve(parent,args){

*return* \_.find(authors,{id:args.id});

      }

    }

  },

});

module.exports = new GraphQLSchema({

  query: RootQuery,

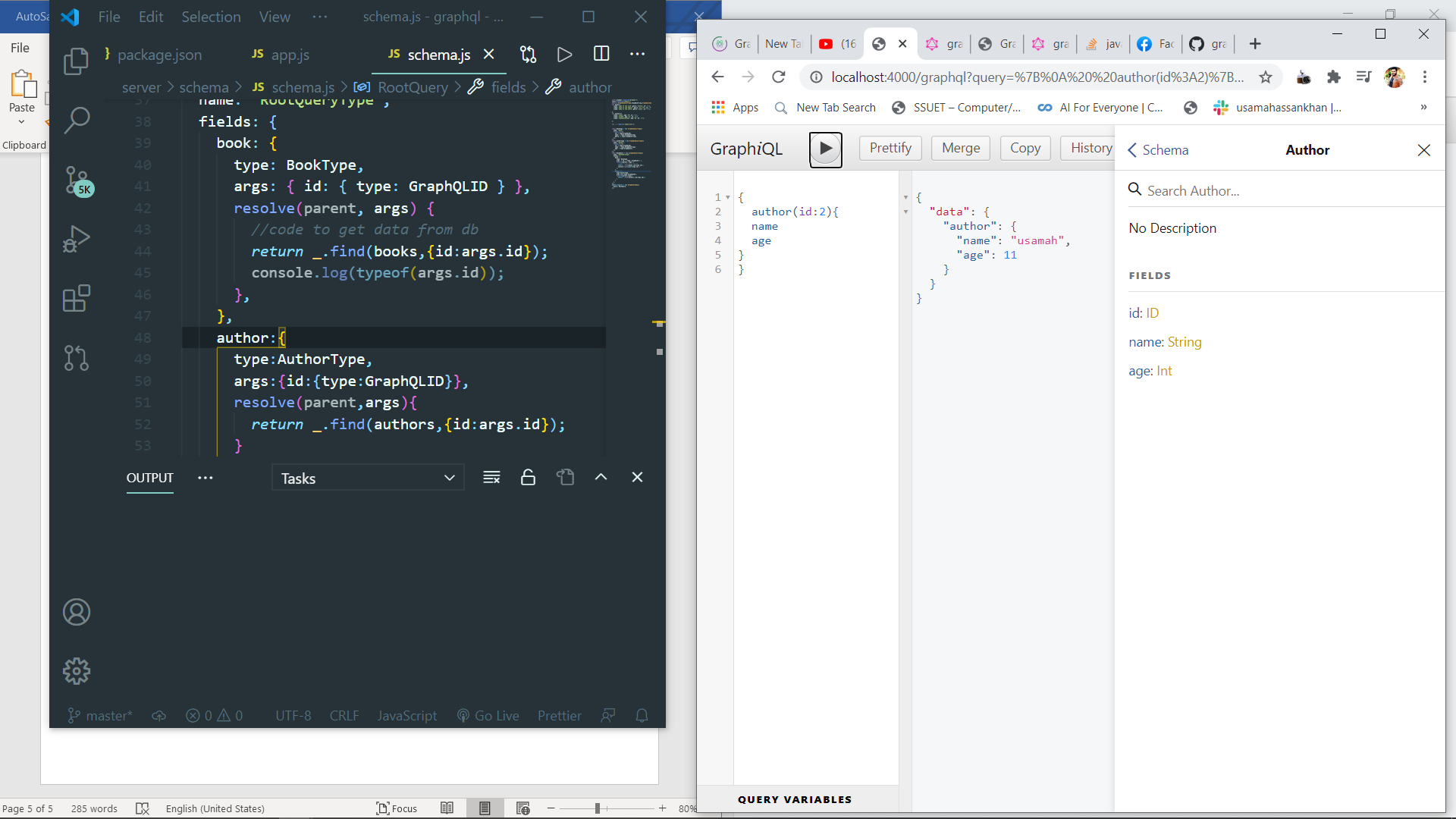
});

**For id**

GraphQLID

Its string in arrya but in graphqli we can wrir like int

Output



For relation with author

const graphql =require("graphql");

const\_=require("lodash");

const { GraphQLObjectType,GraphQLString,GraphQLSchema ,GraphQLID,GraphQLInt}=graphql;

var books=[

  {name:'name of the wind',genre:'fantsy',id :'1',authorid:'1'},

  {name:'name of the empire',genre:'sci fic',id :'2',authorid:'2'},

  {name:'name of the shore',genre:'novel',id :'3',authorid:'3'},

];

var authors=[

  {name:'usama',age:'19',id :'1'},

  {name:'usamah',age:'11',id :'2'},

  {name:'usamahasankhan',age:'66',id :'3'},

];

var \_ = require('underscore');

const BookType = new GraphQLObjectType({

  name: "Book",

  fields:()=>({

    id: { type:GraphQLID},

    name: { type:GraphQLString},

    genre: { type:GraphQLString},

    author:{

      type:AuthorType,

      resolve(parent,args){

*return* \_.find(authors,{id:parent.authorid});

      }

    }

  }),

});

const AuthorType = new GraphQLObjectType({

  name: "Author",

  fields:()=>({

    id: { type:GraphQLID},

    name: { type:GraphQLString},

    age: { type:GraphQLInt},

  }),

});

const RootQuery = new GraphQLObjectType({

  name: "RootQueryType",

  fields: {

    book: {

      type: BookType,

      args: { id: { type: GraphQLID } },

      resolve(parent, args) {

*//code to get data from db*

*return* \_.find(books,{id:args.id});

        console.log(typeof(args.id));

      },

    },

    author:{

      type:AuthorType,

      args:{id:{type:GraphQLID}},

      resolve(parent,args){

*return* \_.find(authors,{id:args.id});

      }

    }

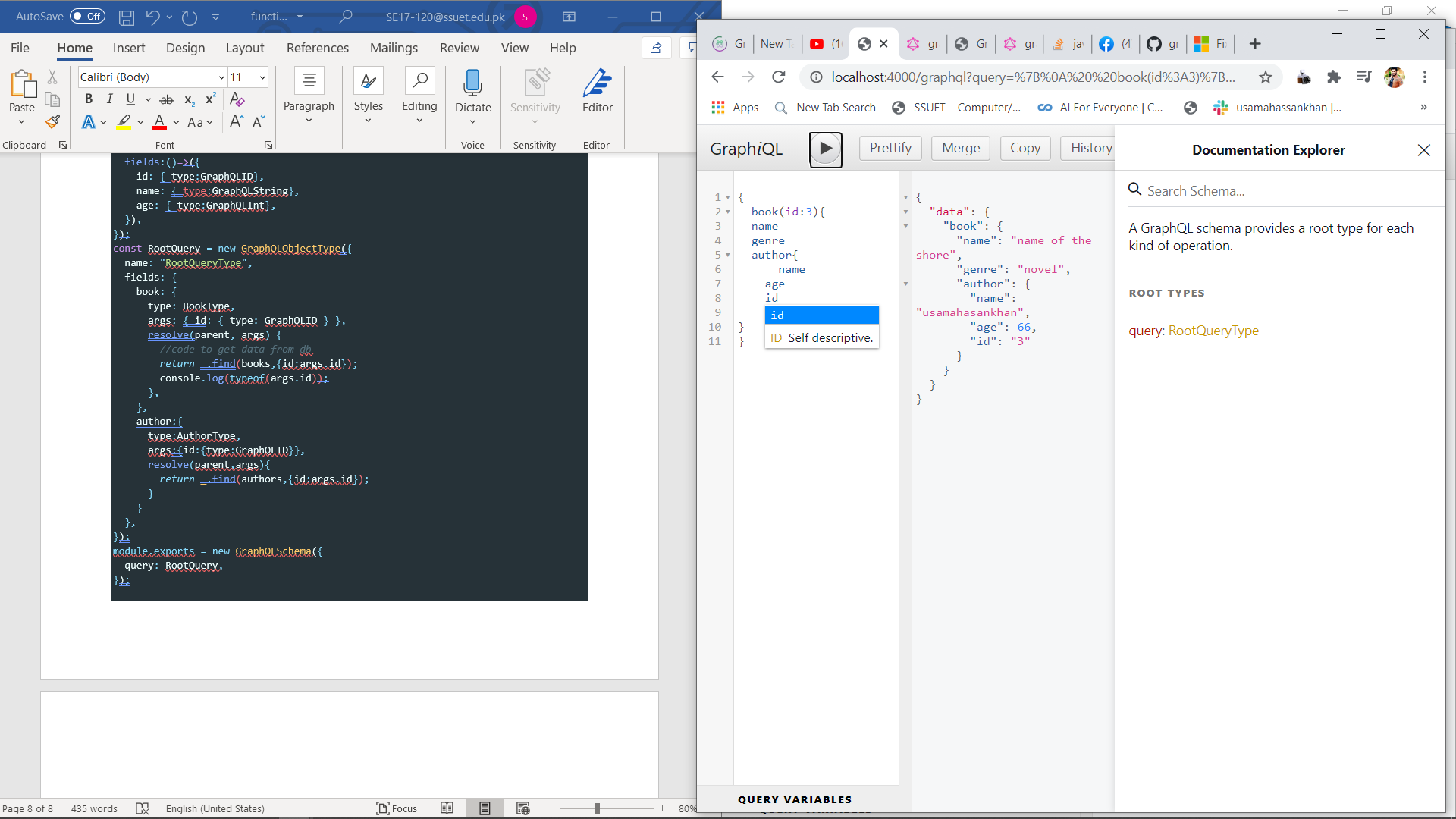
  },

});

module.exports = new GraphQLSchema({

  query: RootQuery,

});



Not from author to books

const graphql =require("graphql");

const\_=require("lodash");

const { GraphQLObjectType,GraphQLString,GraphQLSchema ,GraphQLID,GraphQLInt,GraphQLList}=graphql;

var books=[

  {name:'name of the wind',genre:'fantsy',id :'1',authorid:'1'},

  {name:'name of the empire',genre:'sci fic',id :'2',authorid:'2'},

  {name:'name of the shore',genre:'novel',id :'3',authorid:'3'},

  {name:'name of the fore',genre:'fantasy',id :'3',authorid:'3'},

];

var authors=[

  {name:'usama',age:'19',id :'1'},

  {name:'usamah',age:'11',id :'2'},

  {name:'usamahasankhan',age:'66',id :'3'},

];

var \_ = require('underscore');

const BookType = new GraphQLObjectType({

  name: "Book",

  fields:()=>({

    id: { type:GraphQLID},

    name: { type:GraphQLString},

    genre: { type:GraphQLString},

    author:{

      type:AuthorType,

      resolve(parent,args){

*return* \_.find(authors,{id:parent.authorid});

      }

    }

  }),

});

const AuthorType = new GraphQLObjectType({

  name: "Author",

  fields:()=>({

    id: { type:GraphQLID},

    name: { type:GraphQLString},

    age: { type:GraphQLInt},

    books:{

      type:new GraphQLList(BookType),

      resolve(parent,args){

*return* \_.filter(books,{authorid:parent.id});

      }

    }

  }),

});

const RootQuery = new GraphQLObjectType({

  name: "RootQueryType",

  fields: {

    book: {

      type: BookType,

      args: { id: { type: GraphQLID } },

      resolve(parent, args) {

*//code to get data from db*

*return* \_.find(books,{id:args.id});

        console.log(typeof(args.id));

      },

    },

    author:{

      type:AuthorType,

      args:{id:{type:GraphQLID}},

      resolve(parent,args){

*return* \_.find(authors,{id:args.id});

      }

    }

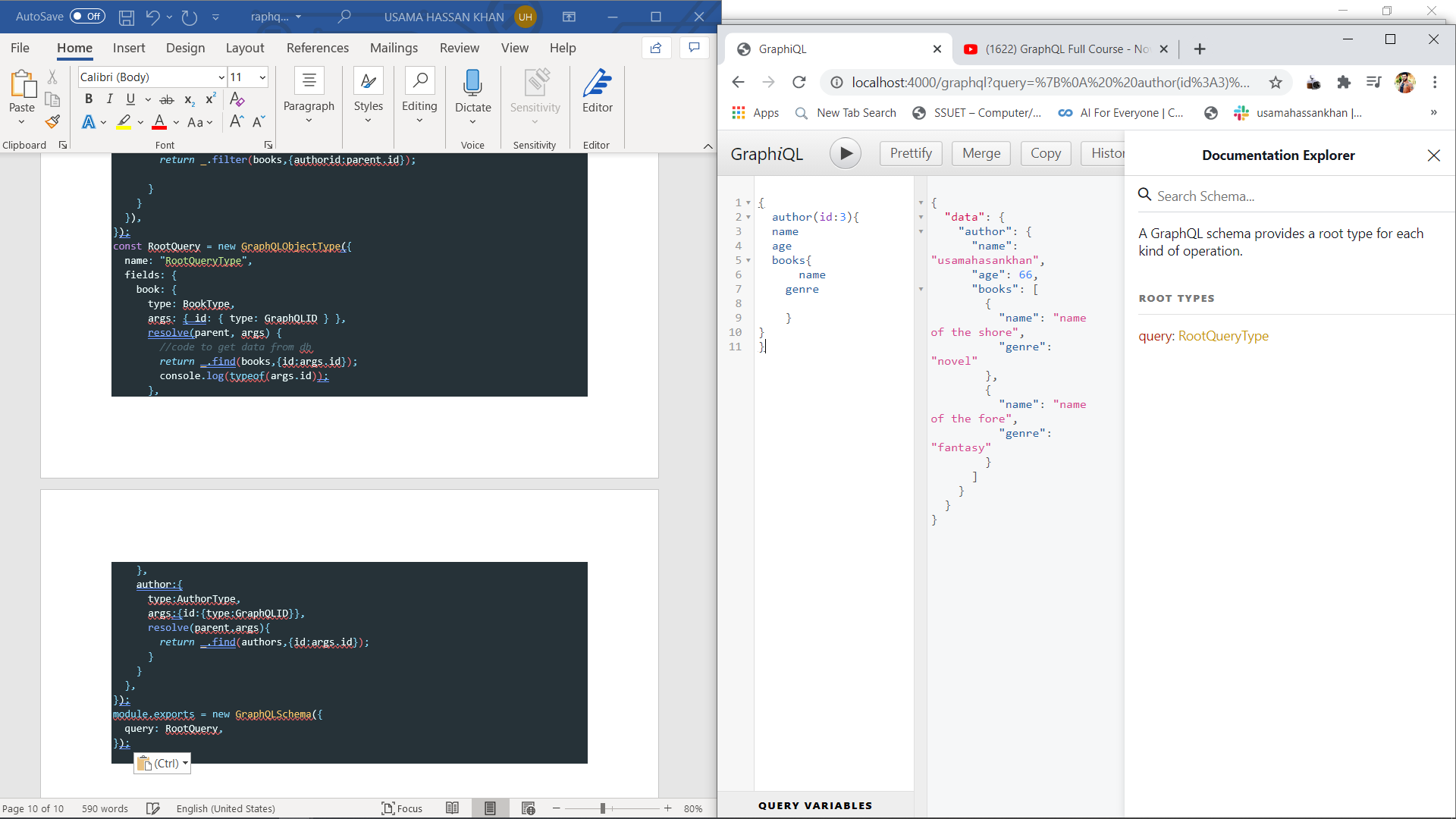
  },

});

module.exports = new GraphQLSchema({

  query: RootQuery,

});



Get complete books

So jst add in root query

const graphql =require("graphql");

const\_=require("lodash");

const { GraphQLObjectType,GraphQLString,GraphQLSchema ,GraphQLID,GraphQLInt,GraphQLList}=graphql;

var books=[

  {name:'name of the wind',genre:'fantsy',id :'1',authorid:'1'},

  {name:'name of the empire',genre:'sci fic',id :'2',authorid:'2'},

  {name:'name of the shore',genre:'novel',id :'3',authorid:'3'},

  {name:'name of the fore',genre:'fantasy',id :'3',authorid:'3'},

];

var authors=[

  {name:'usama',age:'19',id :'1'},

  {name:'usamah',age:'11',id :'2'},

  {name:'usamahasankhan',age:'66',id :'3'},

];

var \_ = require('underscore');

const BookType = new GraphQLObjectType({

  name: "Book",

  fields:()=>({

    id: { type:GraphQLID},

    name: { type:GraphQLString},

    genre: { type:GraphQLString},

    author:{

      type:AuthorType,

      resolve(parent,args){

*return* \_.find(authors,{id:parent.authorid});

      }

    }

  }),

});

const AuthorType = new GraphQLObjectType({

  name: "Author",

  fields:()=>({

    id: { type:GraphQLID},

    name: { type:GraphQLString},

    age: { type:GraphQLInt},

    books:{

      type:new GraphQLList(BookType),

      resolve(parent,args){

*return* \_.filter(books,{authorid:parent.id});

      }

    }

  }),

});

const RootQuery = new GraphQLObjectType({

  name: "RootQueryType",

  fields: {

    book: {

      type: BookType,

      args: { id: { type: GraphQLID } },

      resolve(parent, args) {

*//code to get data from db*

*return* \_.find(books,{id:args.id});

        console.log(typeof(args.id));

      },

    },

    author:{

      type:AuthorType,

      args:{id:{type:GraphQLID}},

      resolve(parent,args){

*return* \_.find(authors,{id:args.id});

      }

    },

    books:{

      type:new GraphQLList(BookType),

      resolve(parent,args){

*return* books

      }

    }

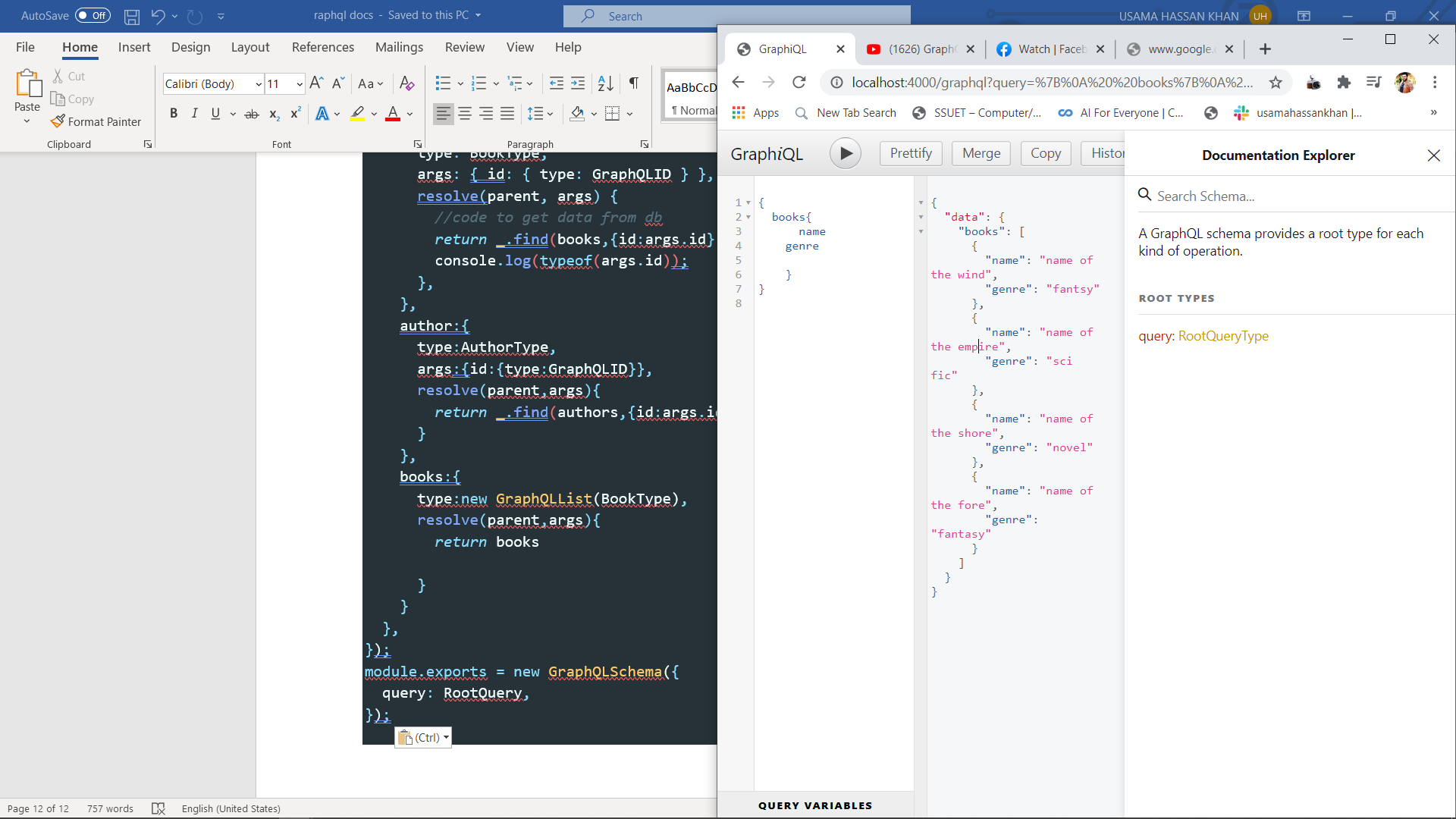
  },

});

module.exports = new GraphQLSchema({

  query: RootQuery,

});



List of authors

const graphql =require("graphql");

const\_=require("lodash");

const { GraphQLObjectType,GraphQLString,GraphQLSchema ,GraphQLID,GraphQLInt,GraphQLList}=graphql;

var books=[

  {name:'name of the wind',genre:'fantsy',id :'1',authorid:'1'},

  {name:'name of the empire',genre:'sci fic',id :'2',authorid:'2'},

  {name:'name of the shore',genre:'novel',id :'3',authorid:'3'},

  {name:'name of the fore',genre:'fantasy',id :'3',authorid:'3'},

];

var authors=[

  {name:'usama',age:'19',id :'1'},

  {name:'usamah',age:'11',id :'2'},

  {name:'usamahasankhan',age:'66',id :'3'},

];

var \_ = require('underscore');

const BookType = new GraphQLObjectType({

  name: "Book",

  fields:()=>({

    id: { type:GraphQLID},

    name: { type:GraphQLString},

    genre: { type:GraphQLString},

    author:{

      type:AuthorType,

      resolve(parent,args){

*return* \_.find(authors,{id:parent.authorid});

      }

    }

  }),

});

const AuthorType = new GraphQLObjectType({

  name: "Author",

  fields:()=>({

    id: { type:GraphQLID},

    name: { type:GraphQLString},

    age: { type:GraphQLInt},

    books:{

      type:new GraphQLList(BookType),

      resolve(parent,args){

*return* \_.filter(books,{authorid:parent.id});

      }

    }

  }),

});

const RootQuery = new GraphQLObjectType({

  name: "RootQueryType",

  fields: {

    book: {

      type: BookType,

      args: { id: { type: GraphQLID } },

      resolve(parent, args) {

*//code to get data from db*

*return* \_.find(books,{id:args.id});

        console.log(typeof(args.id));

      },

    },

    author:{

      type:AuthorType,

      args:{id:{type:GraphQLID}},

      resolve(parent,args){

*return* \_.find(authors,{id:args.id});

      }

    },

    books:{

      type:new GraphQLList(BookType),

      resolve(parent,args){

*return* books

      }

    }

    ,

    authors:{

      type:new GraphQLList(AuthorType),

      resolve(parent,args){

*return* authors

      }

    }

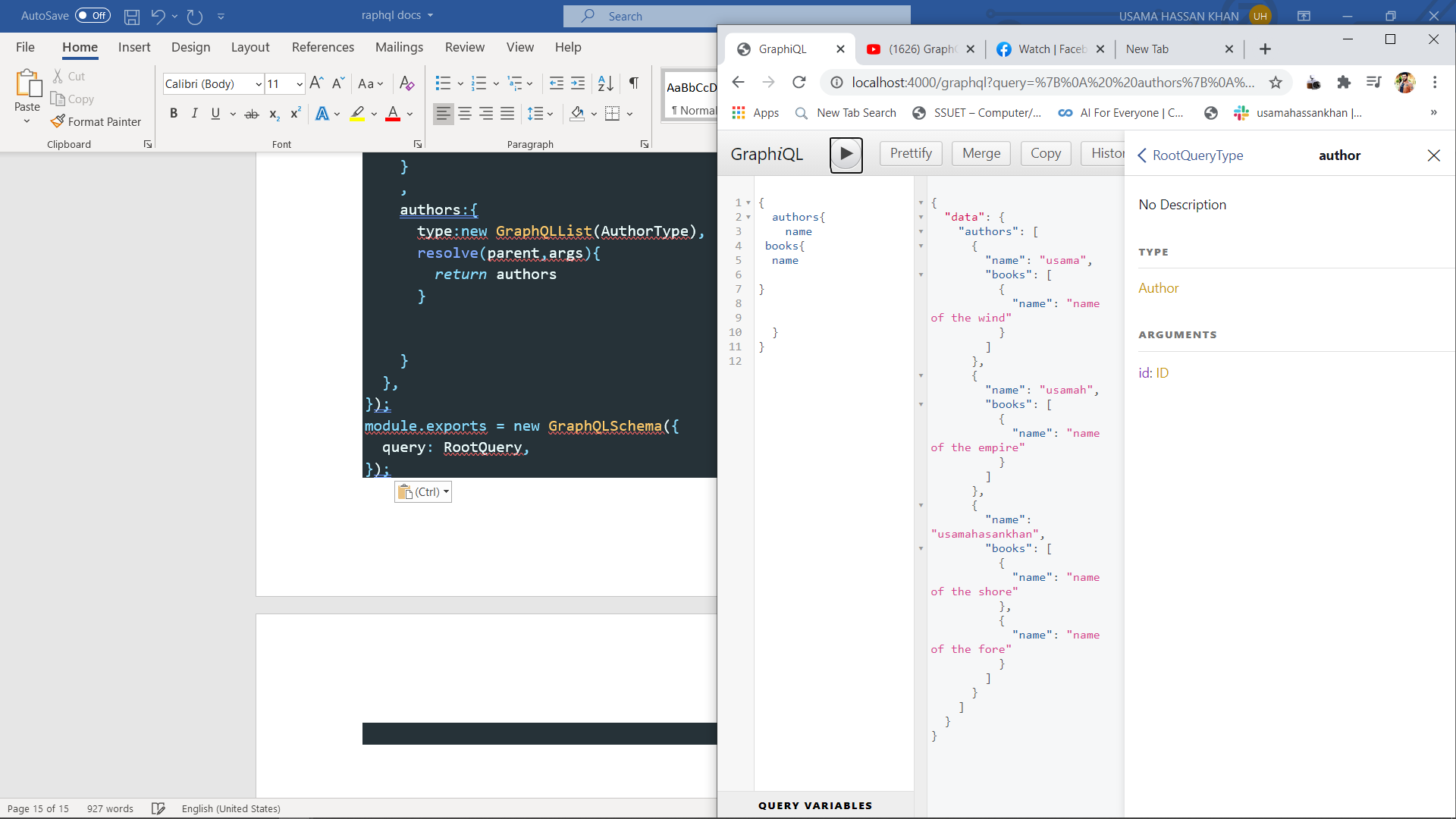
  },

});

module.exports = new GraphQLSchema({

  query: RootQuery,

});

c

connect to database mLab

const express = require('express');

const { graphqlHTTP } = require('express-graphql');

const schema=require('./schema/schema');

const mongoose=require('mongoose');

const app =express();

mongoose.connect("mongodb+srv://usama:se2017120@cluster0.lqkdl.mongodb.net/<dbname>?retryWrites=true&w=majority");

mongoose.connection.once('open',()=>{

  console.log("connected to database");

})

app.use(

  '/graphql',

  graphqlHTTP({

    schema:schema,

    graphiql:true

  }),

);

app.listen(4000,()=>{

  console.log('now listning request on port 4000')

});

Then

Create folder in server folder name models

And add author.js and book.js

And

Author .js

const mongoose =require('mongoose');

const Schema=mongoose.Schema;

const authorSchema=new Schema({

    name:String,

    age:Number

});

module.exports=mongoose.model('Author',authorSchema);

book.js

const mongoose =require('mongoose');

const Schema=mongoose.Schema;

const bookSchema=new Schema({

    name:String,

    genre:String,

    authorId:String

});

module.exports=mongoose.model('Book',bookSchema);

add this in schema.js

const Book=require('../models/book');

const Author=require('../models/author');

mutation

schema.js

const graphql =require("graphql");

const\_=require("lodash");

const Book=require('../models/book');

const Author=require('../models/author');

const { GraphQLObjectType,GraphQLString,GraphQLSchema ,GraphQLID,GraphQLInt,GraphQLList}=graphql;

*// var books=[*

*//   {name:'name of the wind',genre:'fantsy',id :'1',authorid:'1'},*

*//   {name:'name of the empire',genre:'sci fic',id :'2',authorid:'2'},*

*//   {name:'name of the shore',genre:'novel',id :'3',authorid:'3'},*

*//   {name:'name of the fore',genre:'fantasy',id :'3',authorid:'3'},*

*// ];*

*// var authors=[*

*//   {name:'usama',age:'19',id :'1'},*

*//   {name:'usamah',age:'11',id :'2'},*

*//   {name:'usamahasankhan',age:'66',id :'3'},*

*// ];*

var \_ = require('underscore');

const BookType = new GraphQLObjectType({

  name: "Book",

  fields:()=>({

    id: { type:GraphQLID},

    name: { type:GraphQLString},

    genre: { type:GraphQLString},

    author:{

      type:AuthorType,

      resolve(parent,args){

*return* \_.find(authors,{id:parent.authorid});

      }

    }

  }),

});

const AuthorType = new GraphQLObjectType({

  name: "Author",

  fields:()=>({

    id: { type:GraphQLID},

    name: { type:GraphQLString},

    age: { type:GraphQLInt},

    books:{

      type:new GraphQLList(BookType),

      resolve(parent,args){

*// return \_.filter(books,{authorid:parent.id});*

      }

    }

  }),

});

const RootQuery = new GraphQLObjectType({

  name: "RootQueryType",

  fields: {

*// book: {*

*//   type: BookType,*

*//   args: { id: { type: GraphQLID } },*

*//   resolve(parent, args) {*

*//     //code to get data from db*

*//     // return \_.find(books,{id:args.id});*

*//     // console.log(typeof(args.id));*

*//   },*

*// },*

    author:{

      type:AuthorType,

      args:{id:{type:GraphQLID}},

      resolve(parent,args){

*// return \_.find(authors,{id:args.id});*

      }

    },

*// books:{*

*//   type:new GraphQLList(BookType),*

*//   resolve(parent,args){*

*//     // return books*

*//   }*

*// }*

*// ,*

*// authors:{*

*//   type:new GraphQLList(AuthorType),*

*//   resolve(parent,args){*

*//     // return authors*

*//   }*

*// }*

  },

});

const Mutation=new GraphQLObjectType(

  {

    name:'Mutation',

    fields:{

      addAuthor:{

      type:AuthorType,

      args:{

        name:{type:GraphQLString},

        age:{type:GraphQLInt}},

        resolve(parent,args){

          let author=new Author({

            name:args.name,

            age:args.age

          });

*return* author.save();

        }

      }}}

)

module.exports = new GraphQLSchema({

  query: RootQuery,

  mutation:Mutation

});

Book mutation

const graphql =require("graphql");

const\_=require("lodash");

const Book=require('../models/book');

const Author=require('../models/author');

const { GraphQLObjectType,GraphQLString,GraphQLSchema ,GraphQLID,GraphQLInt,GraphQLList}=graphql;

*// var books=[*

*//   {name:'name of the wind',genre:'fantsy',id :'1',authorid:'1'},*

*//   {name:'name of the empire',genre:'sci fic',id :'2',authorid:'2'},*

*//   {name:'name of the shore',genre:'novel',id :'3',authorid:'3'},*

*//   {name:'name of the fore',genre:'fantasy',id :'3',authorid:'3'},*

*// ];*

*// var authors=[*

*//   {name:'usama',age:'19',id :'1'},*

*//   {name:'usamah',age:'11',id :'2'},*

*//   {name:'usamahasankhan',age:'66',id :'3'},*

*// ];*

var \_ = require('underscore');

const BookType = new GraphQLObjectType({

  name: "Book",

  fields:()=>({

    id: { type:GraphQLID},

    name: { type:GraphQLString},

    genre: { type:GraphQLString},

    author:{

      type:AuthorType,

      resolve(parent,args){

*return* \_.find(authors,{id:parent.authorid});

      }

    }

  }),

});

const AuthorType = new GraphQLObjectType({

  name: "Author",

  fields:()=>({

    id: { type:GraphQLID},

    name: { type:GraphQLString},

    age: { type:GraphQLInt},

    books:{

      type:new GraphQLList(BookType),

      resolve(parent,args){

*// return \_.filter(books,{authorid:parent.id});*

      }

    }

  }),

});

const RootQuery = new GraphQLObjectType({

  name: "RootQueryType",

  fields: {

*// book: {*

*//   type: BookType,*

*//   args: { id: { type: GraphQLID } },*

*//   resolve(parent, args) {*

*//     //code to get data from db*

*//     // return \_.find(books,{id:args.id});*

*//     // console.log(typeof(args.id));*

*//   },*

*// },*

    author:{

      type:AuthorType,

      args:{id:{type:GraphQLID}},

      resolve(parent,args){

*// return \_.find(authors,{id:args.id});*

      }

    },

*// books:{*

*//   type:new GraphQLList(BookType),*

*//   resolve(parent,args){*

*//     // return books*

*//   }*

*// }*

*// ,*

*// authors:{*

*//   type:new GraphQLList(AuthorType),*

*//   resolve(parent,args){*

*//     // return authors*

*//   }*

*// }*

  },

});

const Mutation=new GraphQLObjectType(

  {

    name:'Mutation',

    fields:{

      addAuthor:{

      type:AuthorType,

      args:{

        name:{type:GraphQLString},

        age:{type:GraphQLInt}},

        resolve(parent,args){

          let author=new Author({

            name:args.name,

            age:args.age

          });

*return* author.save();

        }

      },

      addBook:{

        type:BookType,

        args:{

          name:{type:GraphQLString},

          genre:{type:GraphQLString},

          authorid:{type:GraphQLID}

        },

        resolve(parent,args){

           let book=new Book(

             {

               name:args.name,

               genre:args.genre,

               authorid:args.authorid

             }

           );

*return* book.save();

        }

      }

    }}

)

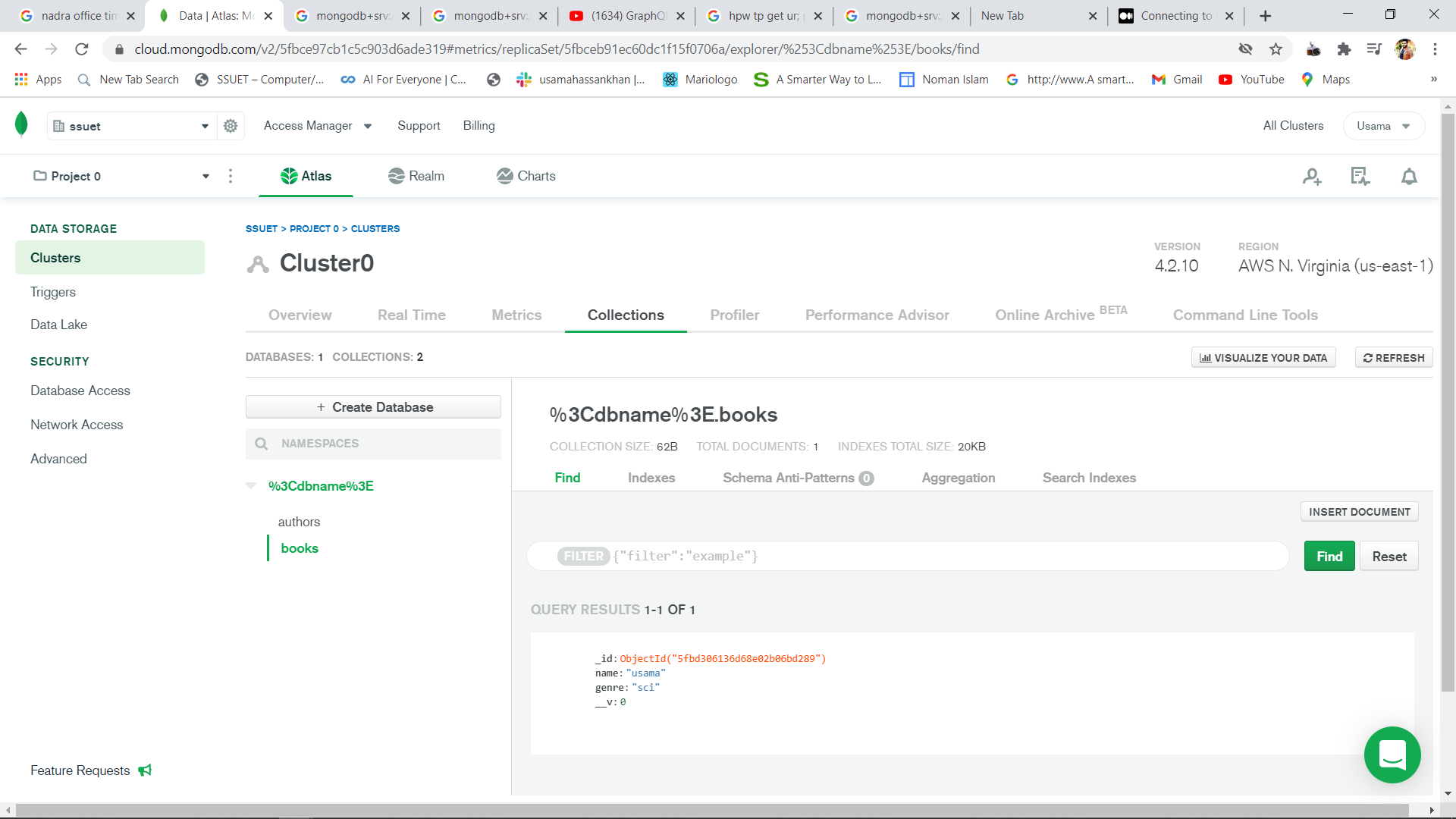
module.exports = new GraphQLSchema({

  query: RootQuery,

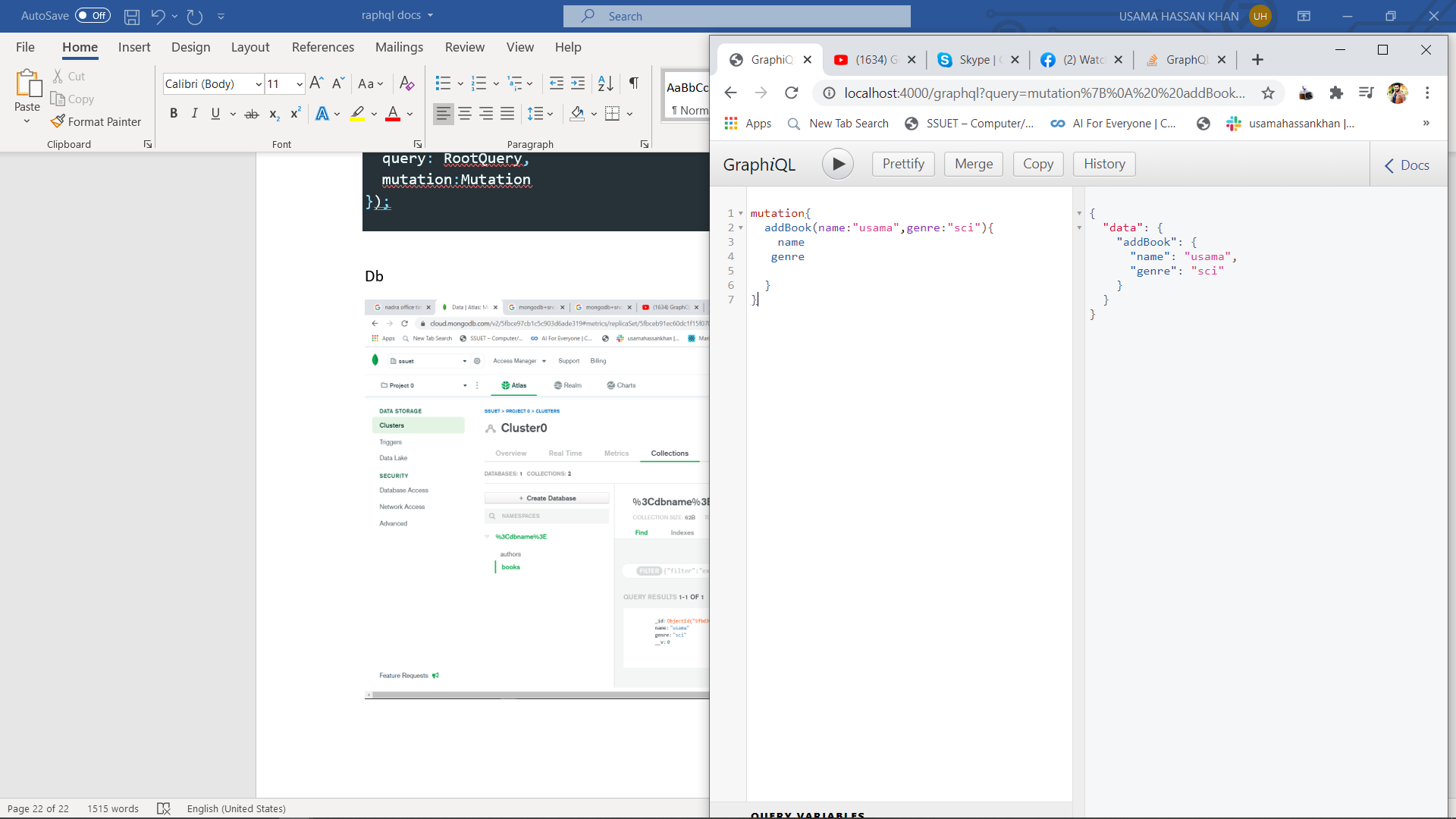
  mutation:Mutation

});

Db



Graphqli



Completed work now from data base

const graphql =require("graphql");

const\_=require("lodash");

const Book=require('../models/book');

const Author=require('../models/author');

const { GraphQLObjectType,GraphQLString,GraphQLSchema ,GraphQLID,GraphQLInt,GraphQLList}=graphql;

*// var books=[*

*//   {name:'name of the wind',genre:'fantsy',id :'1',authorid:'1'},*

*//   {name:'name of the empire',genre:'sci fic',id :'2',authorid:'2'},*

*//   {name:'name of the shore',genre:'novel',id :'3',authorid:'3'},*

*//   {name:'name of the fore',genre:'fantasy',id :'3',authorid:'3'},*

*// ];*

*// var authors=[*

*//   {name:'usama',age:'19',id :'1'},*

*//   {name:'usamah',age:'11',id :'2'},*

*//   {name:'usamahasankhan',age:'66',id :'3'},*

*// ];*

var \_ = require('underscore');

const BookType = new GraphQLObjectType({

  name: "Book",

  fields:()=>({

    id: { type:GraphQLID},

    name: { type:GraphQLString},

    genre: { type:GraphQLString},

    author:{

      type:AuthorType,

      resolve(parent,args){

*// return \_.find(authors,{id:parent.authorid});*

*return* Author.findById(parent.authorid);

      }

    }

  }),

});

const AuthorType = new GraphQLObjectType({

  name: "Author",

  fields:()=>({

    id: { type:GraphQLID},

    name: { type:GraphQLString},

    age: { type:GraphQLInt},

    books:{

      type:new GraphQLList(BookType),

      resolve(parent,args){

*// return \_.filter(books,{authorid:parent.id});*

*return* Book.find({authorid:parent.id});

      }

    }

  }),

});

const RootQuery = new GraphQLObjectType({

  name: "RootQueryType",

  fields: {

    book: {

      type: BookType,

      args: { id: { type: GraphQLID } },

      resolve(parent, args) {

*//code to get data from db*

*// return \_.find(books,{id:args.id});*

*// console.log(typeof(args.id));*

*return* Book.findById(args.id);

      },

    },

    author:{

      type:AuthorType,

      args:{id:{type:GraphQLID}},

      resolve(parent,args){

*// return \_.find(authors,{id:args.id});*

*return* Author.findById(args.id)

      }

    },

    books:{

      type:new GraphQLList(BookType),

      resolve(parent,args){

*// return books*

*return* Book.find({})

      }

    }

    ,

    authors:{

      type:new GraphQLList(AuthorType),

      resolve(parent,args){

*// return authors*

*return*  Author.find({});

      }

    }

  },

});

const Mutation=new GraphQLObjectType(

  {

    name:'Mutation',

    fields:{

      addAuthor:{

      type:AuthorType,

      args:{

        name:{type:GraphQLString},

        age:{type:GraphQLInt}},

        resolve(parent,args){

          let author=new Author({

            name:args.name,

            age:args.age

          });

*return* author.save();

        }

      },

      addBook:{

        type:BookType,

        args:{

          name:{type:GraphQLString},

          genre:{type:GraphQLString},

          authorid:{type:GraphQLID}

        },

        resolve(parent,args){

           let book=new Book(

             {

               name:args.name,

               genre:args.genre,

               authorid:args.authorid

             }

           );

*return* book.save();

        }

      }

    }}

)

module.exports = new GraphQLSchema({

  query: RootQuery,

  mutation:Mutation

});

