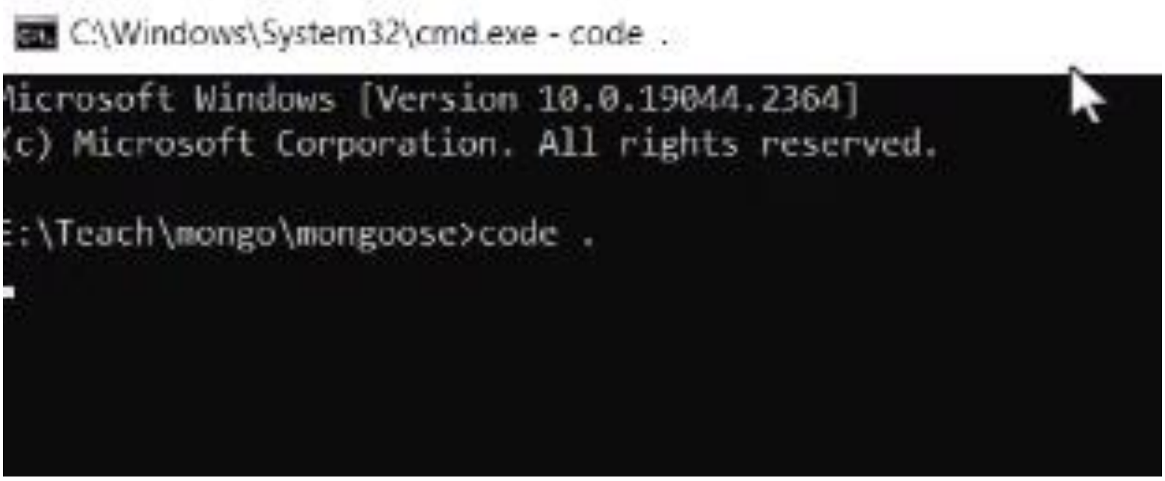
Create folder, type cmd in address bar, code .

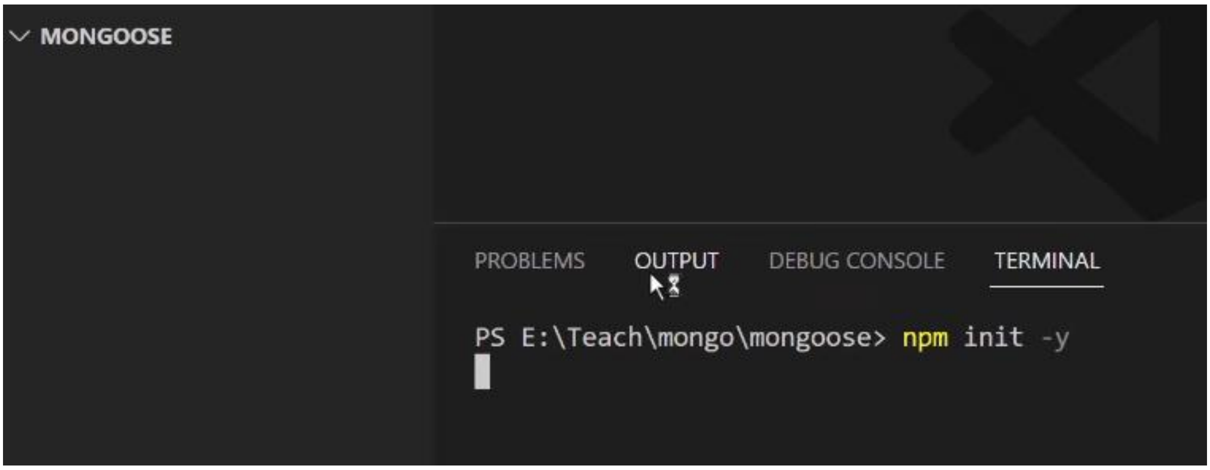


C:\Windows\System32\cmd.exe - code .

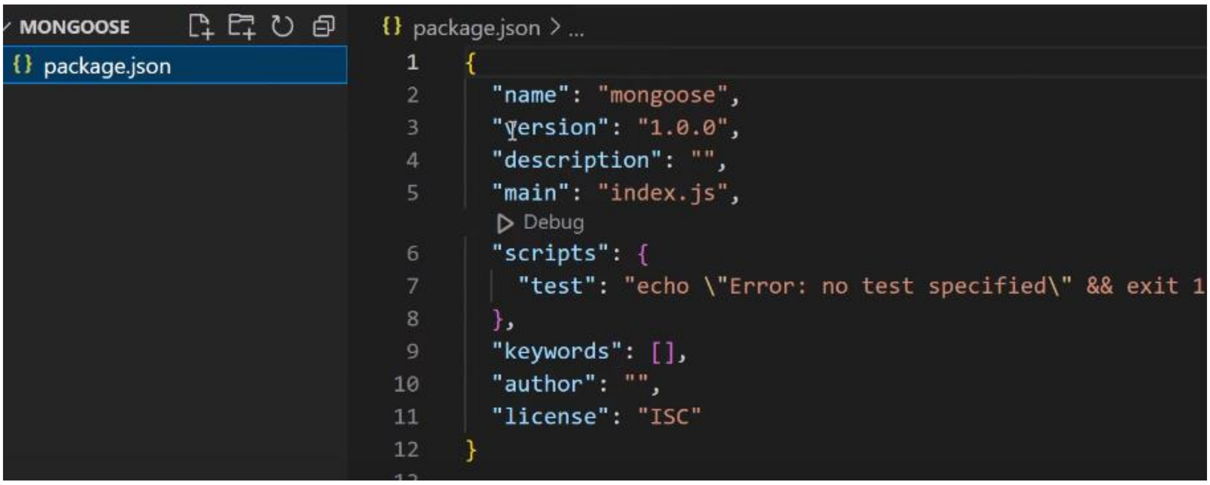
```
Microsoft Windows [Version 10.0.19044.2364]
(c) Microsoft Corporation. All rights reserved.

E:\Teach\mongo\mongoose>code .
```

npm init : Initialize a new project and create a package. json file. npm install <package-name> : Install a package and add it to the dependencies object in your package. json file.

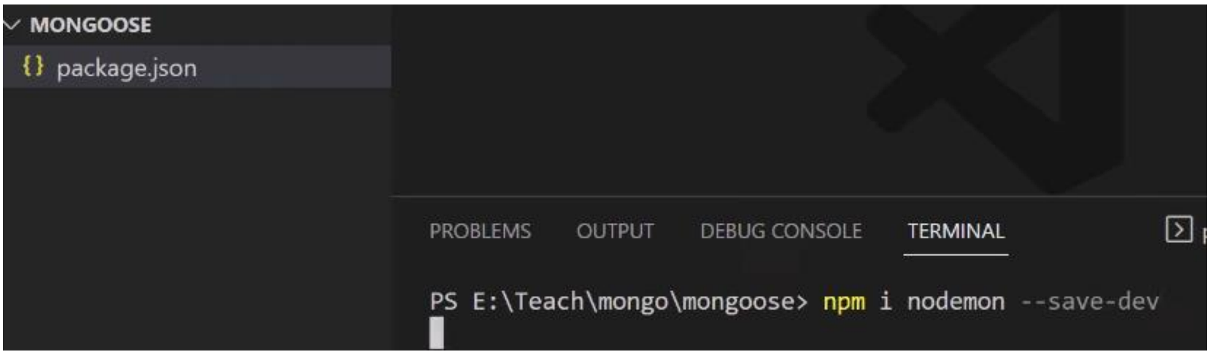
Y meaning : Yes



```
PS E:\Teach\mongo\mongoose> npm init -y
```



```
{
  "name": "mongoose",
  "version": "1.0.0",
  "description": "",
  "main": "index.js",
  "scripts": {
    "test": "echo \"Error: no test specified\" && exit 1
  },
  "keywords": [],
  "author": "",
  "license": "ISC"
}
```



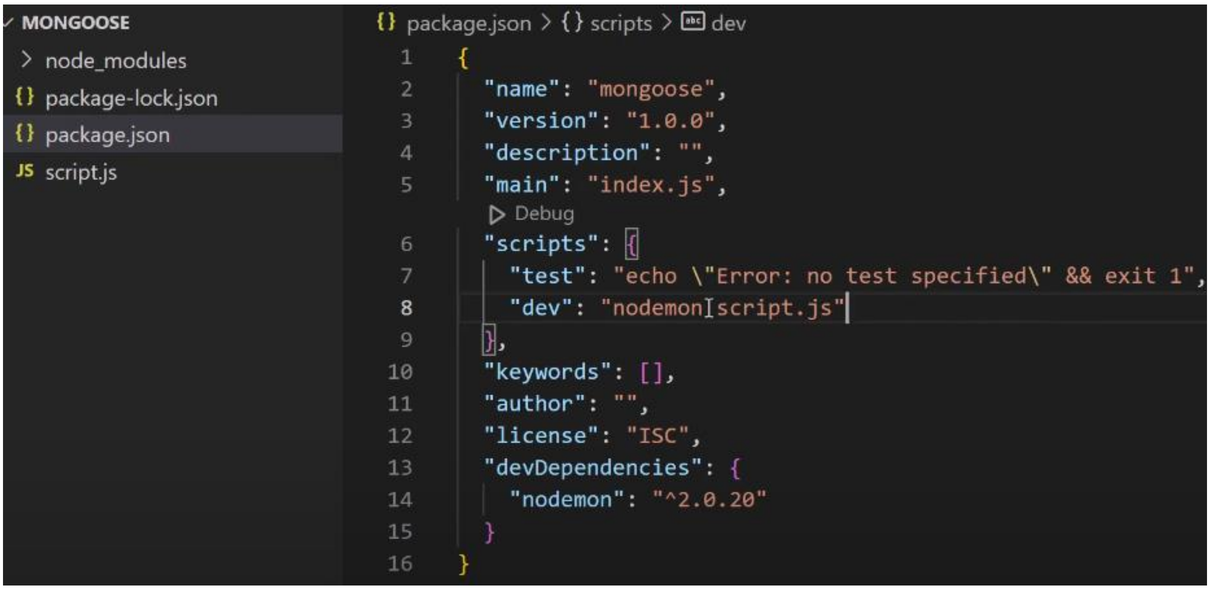
```
PS E:\Teach\mongo\mongoose> npm i nodemon --save-dev
```

--save is default use (npm install, it will work after clone in github)

--save-dev is for development

When you use the --save-dev flag, the package is added to your devDependencies object.

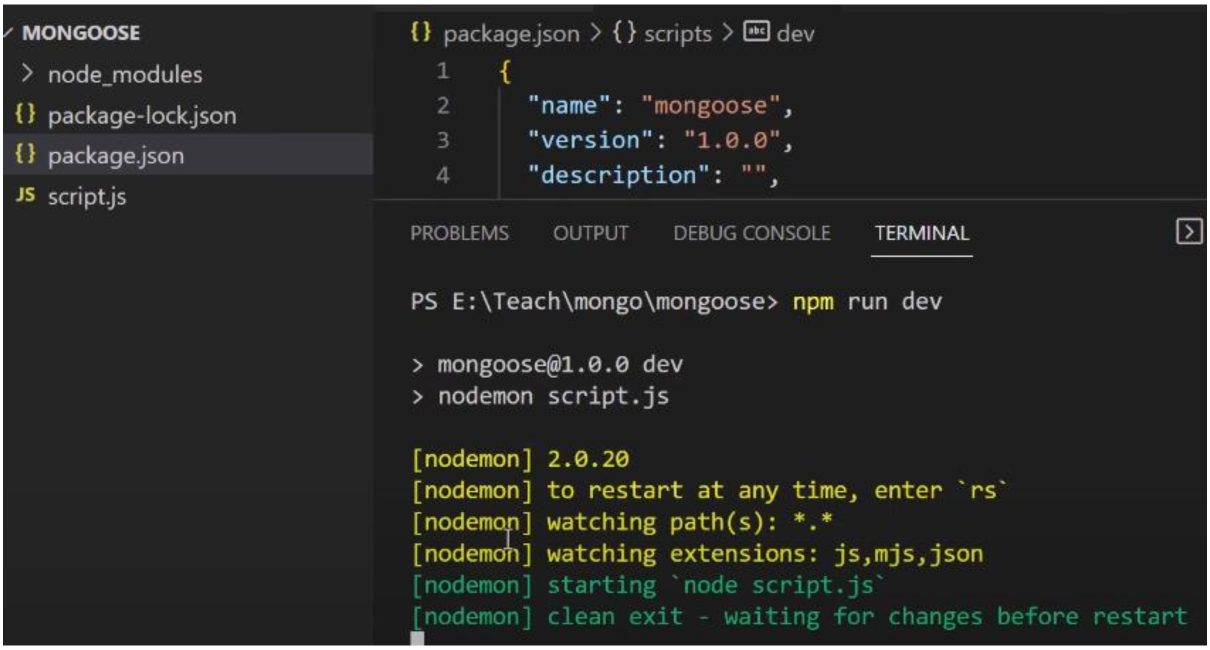
if you download a repo from github and type npm install, the devDependencies are ignored



```json
{
  "name": "mongoose",
  "version": "1.0.0",
  "description": "",
  "main": "index.js",
  "scripts": {
    "test": "echo \"Error: no test specified\" && exit 1",
    "dev": "nodemon script.js"
  },
  "keywords": [],
  "author": "",
  "license": "ISC",
  "devDependencies": {
    "nodemon": "^2.0.20"
  }
}
```

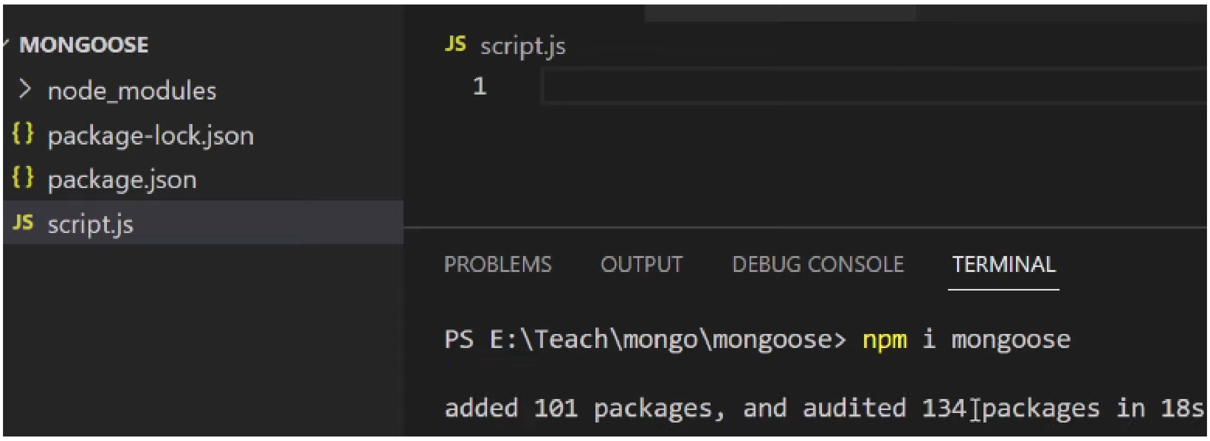
Don’t run every time: node script.js

Run only one time : npm run dev (it will take automatically refresh when saved)



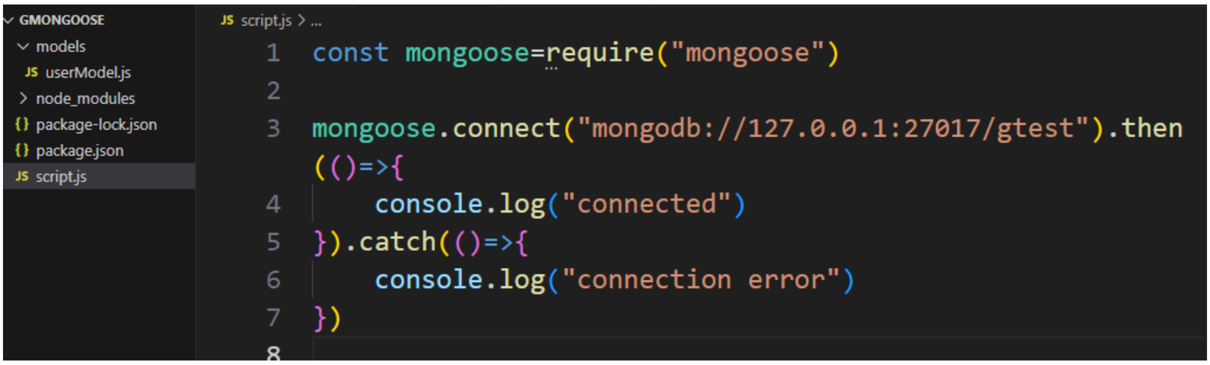
```
{
  "name": "mongoose",
  "version": "1.0.0",
  "description": "",
```

```
PS E:\Teach\mongo\mongoose> npm run dev

> mongoose@1.0.0 dev
> nodemon script.js

[nodemon] 2.0.20
[nodemon] to restart at any time, enter `rs`
[nodemon] watching path(s): *.*
[nodemon] watching extensions: js,mjs,json
[nodemon] starting `node script.js`
[nodemon] clean exit - waiting for changes before restart
```



```
PS E:\Teach\mongo\mongoose> npm i mongoose

added 101 packages, and audited 134 packages in 18s
```

Terminal hide/show shortcut :ctrl +~



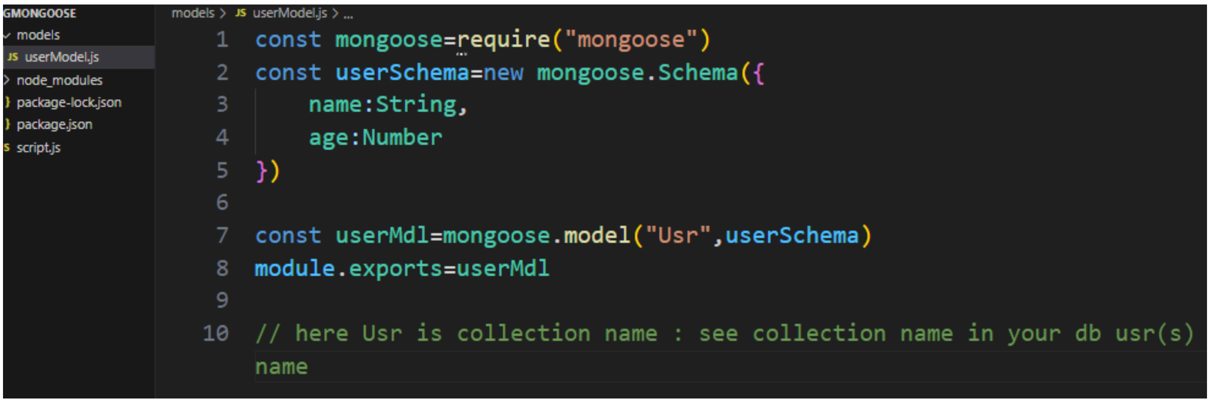
```
const mongoose=require("mongoose")

mongoose.connect("mongodb://127.0.0.1:27017/gtest").then
(()=>{
    console.log("connected")
}).catch(()=>{
    console.log("connection error")
})
```

Mongoose gowthaman kalvi sahas karur 98940-83890

Schema : inbuilt class of mongoose

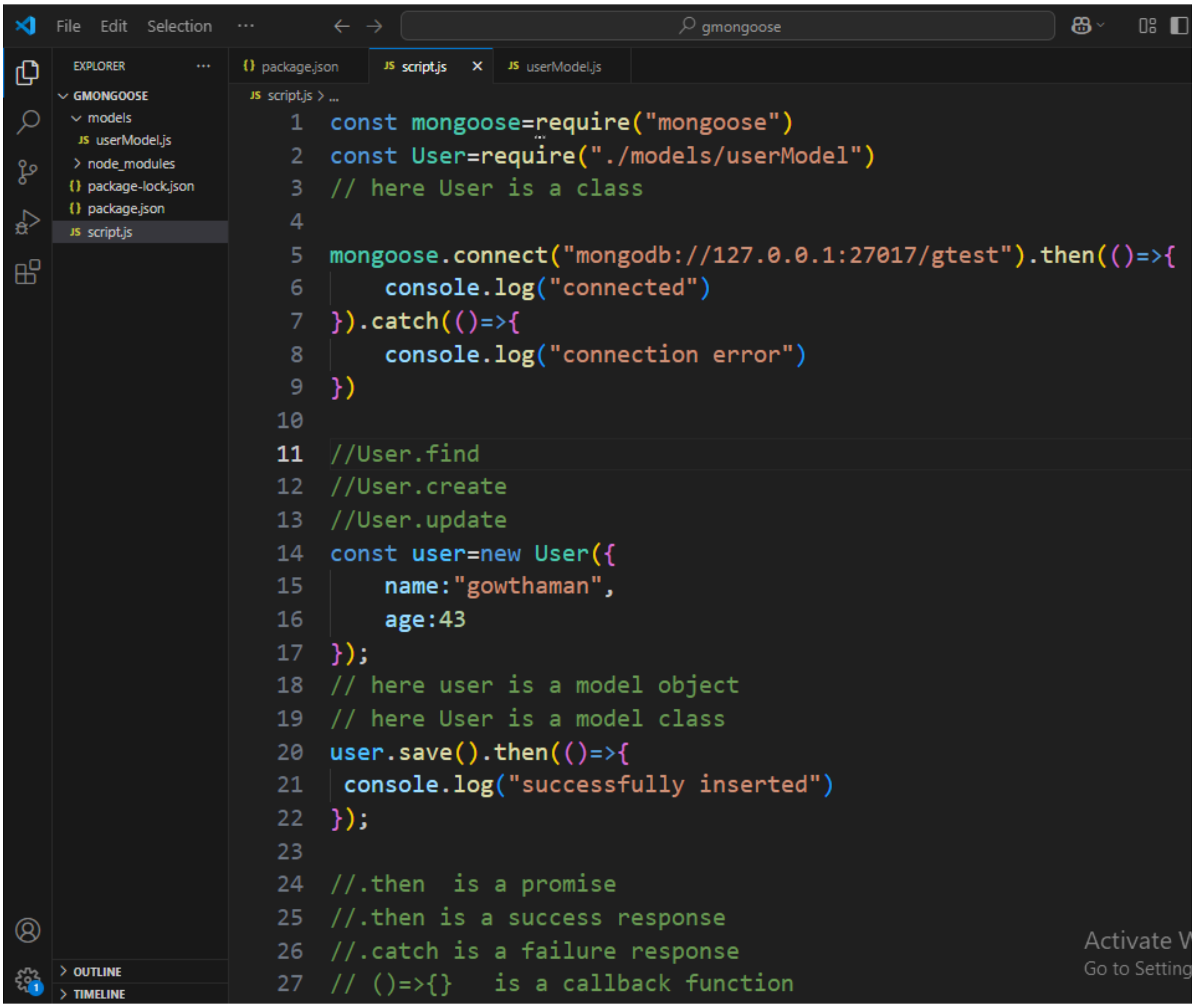
Use: only allow to store schema defined keys and datatype.

userSchemea: userDefined object.



```javascript
const mongoose=require("mongoose")
const userSchema=new mongoose.Schema({
    name:String,
    age:Number
})

const userMdl=mongoose.model("Usr",userSchema)
module.exports=userMdl

// here Usr is collection name : see collection name in your db usr(s) name
```

Insert method 1: (with out using async await)



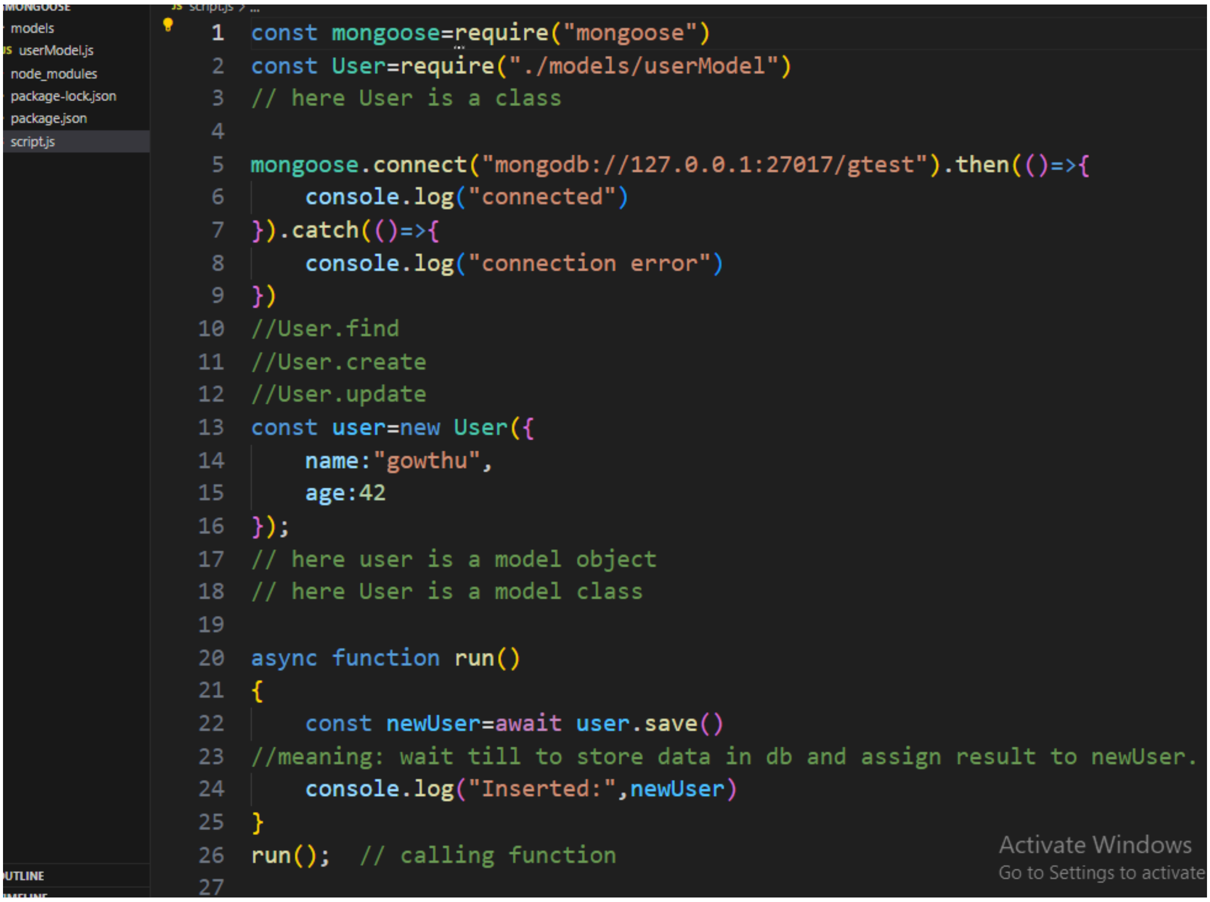
```javascript
const mongoose=require("mongoose")
const User=require("./models/userModel")
// here User is a class

mongoose.connect("mongodb://127.0.0.1:27017/gtest").then(()=>{
    console.log("connected")
}).catch(()=>{
    console.log("connection error")
})

//User.find
//User.create
//User.update
const user=new User({
    name:"gowthaman",
    age:43
});
// here user is a model object
// here User is a model class
user.save().then(()=>{
 console.log("successfully inserted")
});

//.then  is a promise
//.then is a success response
//.catch is a failure response
// ()=>{}   is a callback function
```

Do not change name & age key. because schema defined.

Do not add extra field. Only name and age accept

Same insert method using async function because promise return (.then)

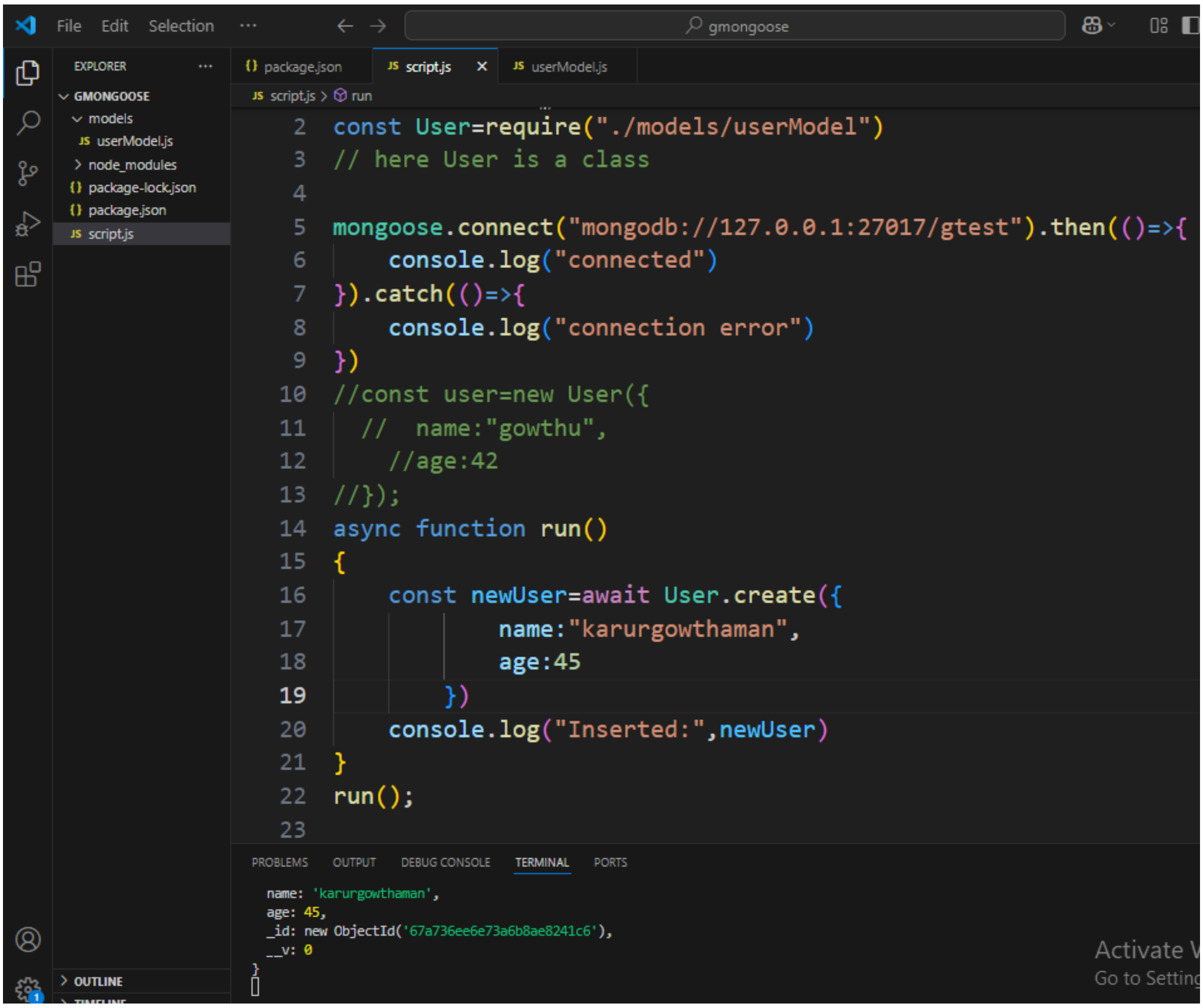
Method 2: insertion using save with async function



```
const mongoose=require("mongoose")
const User=require("./models/userModel")
// here User is a class

mongoose.connect("mongodb://127.0.0.1:27017/gtest").then(()=>{
    console.log("connected")
}).catch(()=>{
    console.log("connection error")
})
//User.find
//User.create
//User.update
const user=new User({
    name:"gowthu",
    age:42
});
// here user is a model object
// here User is a model class

async function run()
{
    const newUser=await user.save()
//meaning: wait till to store data in db and assign result to newUser.
    console.log("Inserted:",newUser)
}
run();  // calling function
```

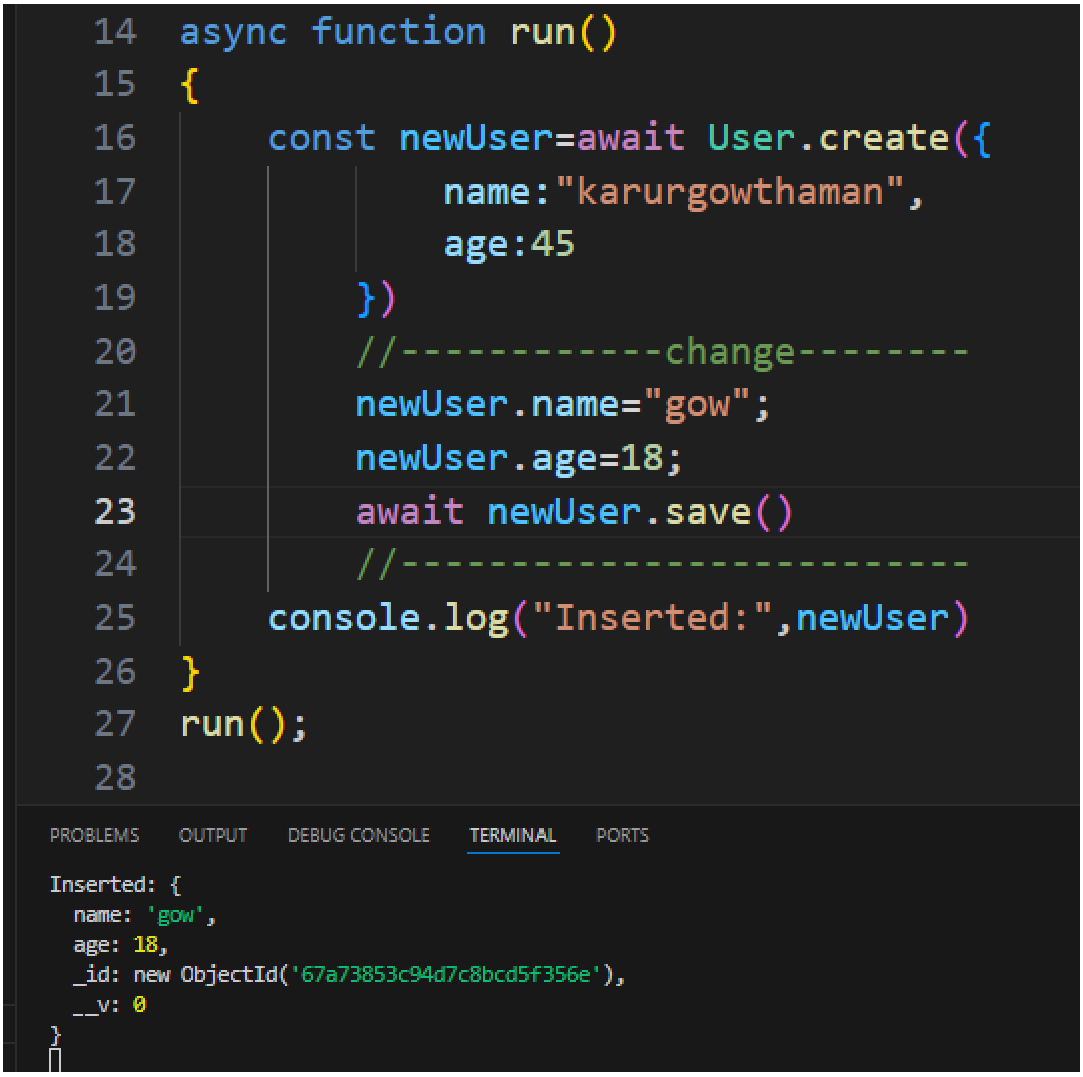
Method:3 insertion (without using save method)

Only using create



```
const User=require("./models/userModel")
// here User is a class

mongoose.connect("mongodb://127.0.0.1:27017/gtest").then(()=>{
    console.log("connected")
}).catch(()=>{
    console.log("connection error")
})
//const user=new User({
  //  name:"gowthu",
    //age:42
//});
async function run()
{
    const newUser=await User.create({
            name:"karurgowthaman",
            age:45
        })
    console.log("Inserted:",newUser)
}
run();

```

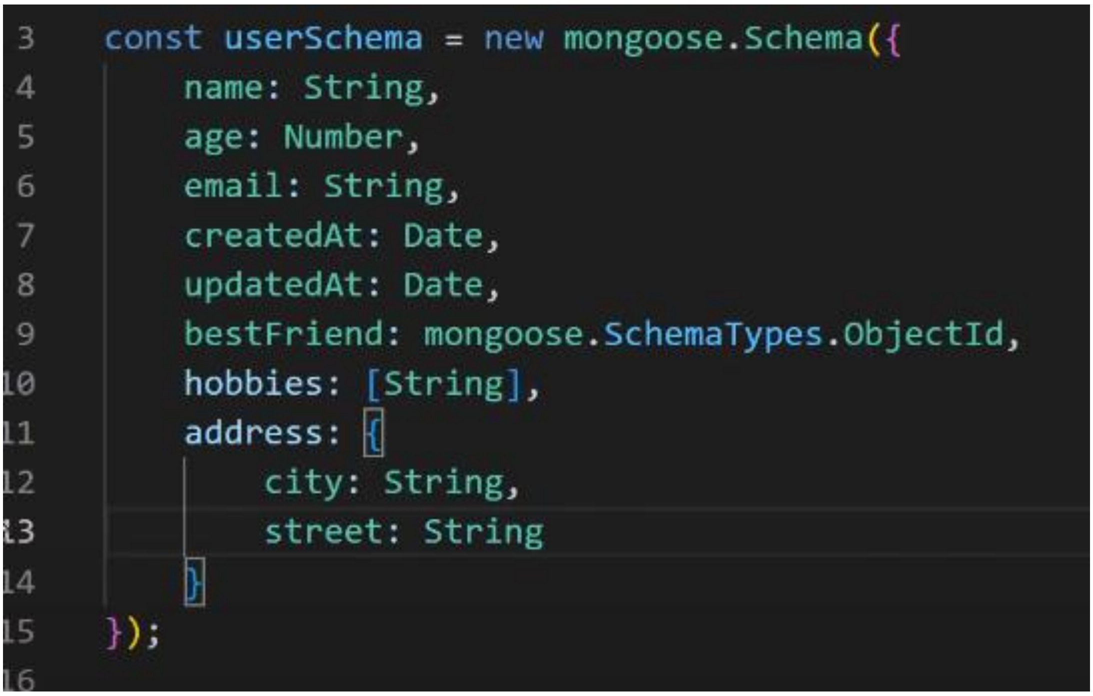
```
  name: 'karurgowthaman',
  age: 45,
  _id: new ObjectId('67a736ee6e73a6b8ae8241c6'),
  __v: 0
}
```



```
async function run()
{
    const newUser=await User.create({
            name:"karurgowthaman",
            age:45
        })
        //-----------change--------
        newUser.name="gow";
        newUser.age=18;
        await newUser.save()
        //--------------------------
    console.log("Inserted:",newUser)
}
run();
```

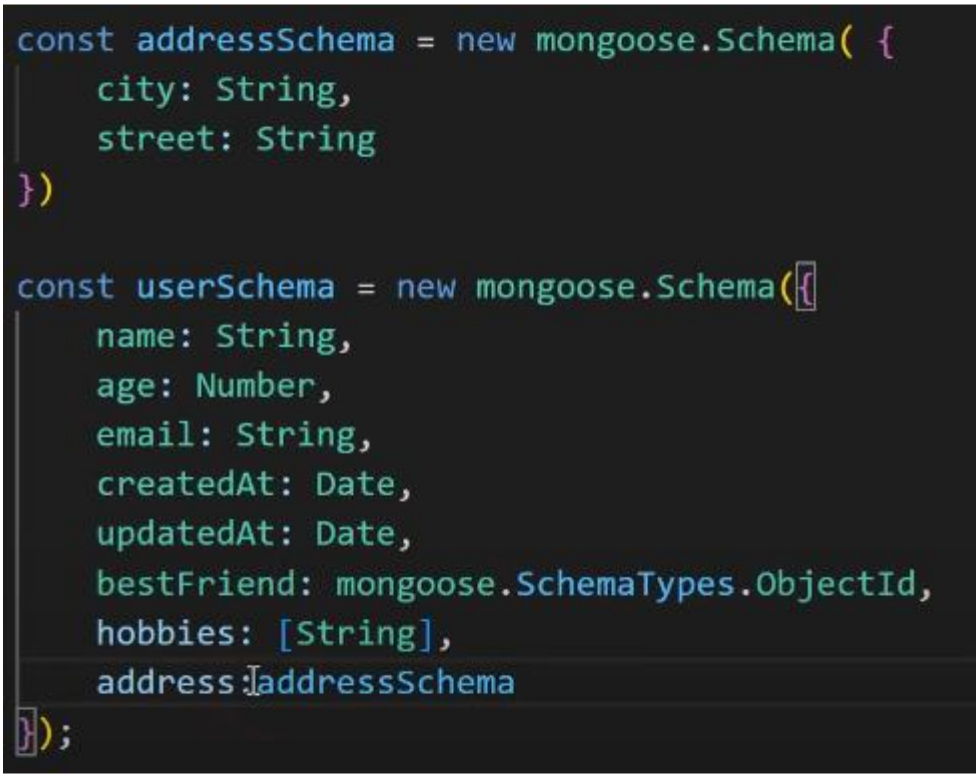
```
Inserted: {
  name: 'gow',
  age: 18,
  _id: new ObjectId('67a73853c94d7c8bcd5f356e'),
  __v: 0
}
```

Schema types:



```
const userSchema = new mongoose.Schema({
    name: String,
    age: Number,
    email: String,
    createdAt: Date,
    updatedAt: Date,
    bestFriend: mongoose.SchemaTypes.ObjectId,
    hobbies: [String],
    address: {
        city: String,
        street: String
    }
});
```

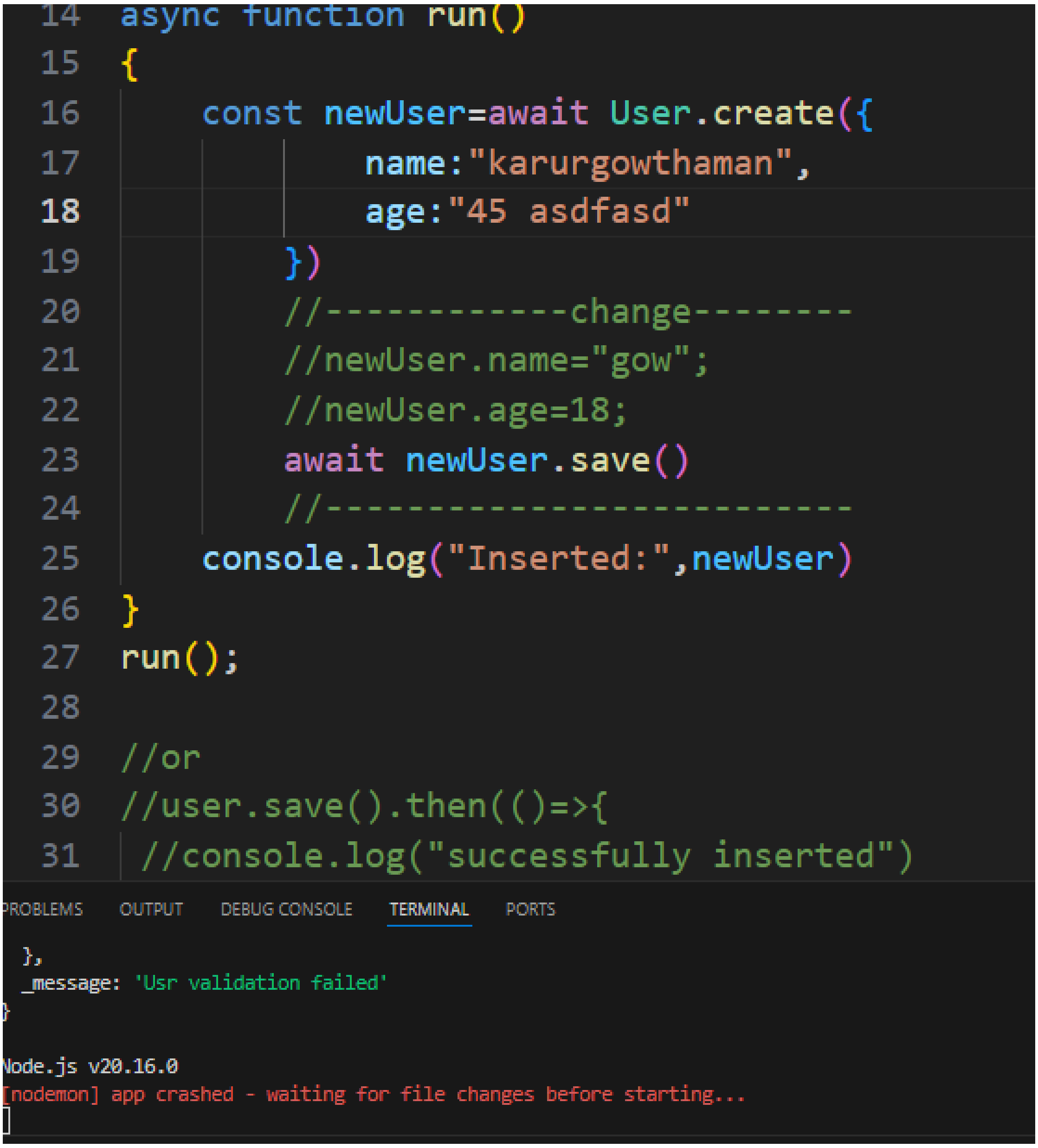
Nested schema:



```
const addressSchema = new mongoose.Schema( {
    city: String,
    street: String
})

const userSchema = new mongoose.Schema({
    name: String,
    age: Number,
    email: String,
    createdAt: Date,
    updatedAt: Date,
    bestFriend: mongoose.SchemaTypes.ObjectId,
    hobbies: [String],
    address:addressSchema
});
```

Before error handle :

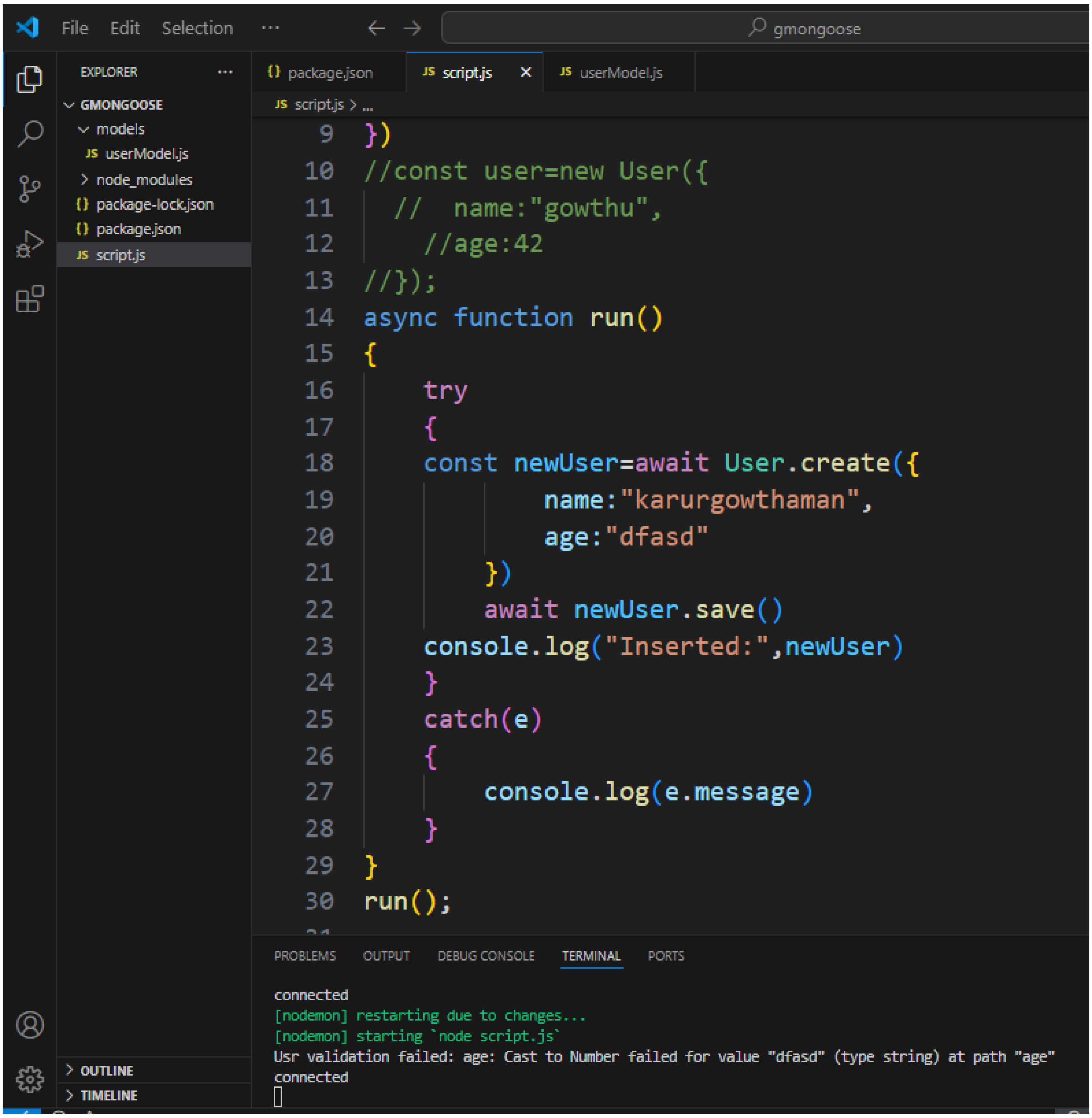
Age only accept integer types



```
async function run()
{
    const newUser=await User.create({
            name:"karurgowthaman",
            age:"45 asdfasd"
        })
        //------------change--------
        //newUser.name="gow";
        //newUser.age=18;
        await newUser.save()
        //------------------------
    console.log("Inserted:",newUser)
}
run();

//or
//user.save().then(()=>{
 //console.log("successfully inserted")
```

```
  },
  _message: 'Usr validation failed'
}

Node.js v20.16.0
[nodemon] app crashed - waiting for file changes before starting...
```

Error handle using try catch block.

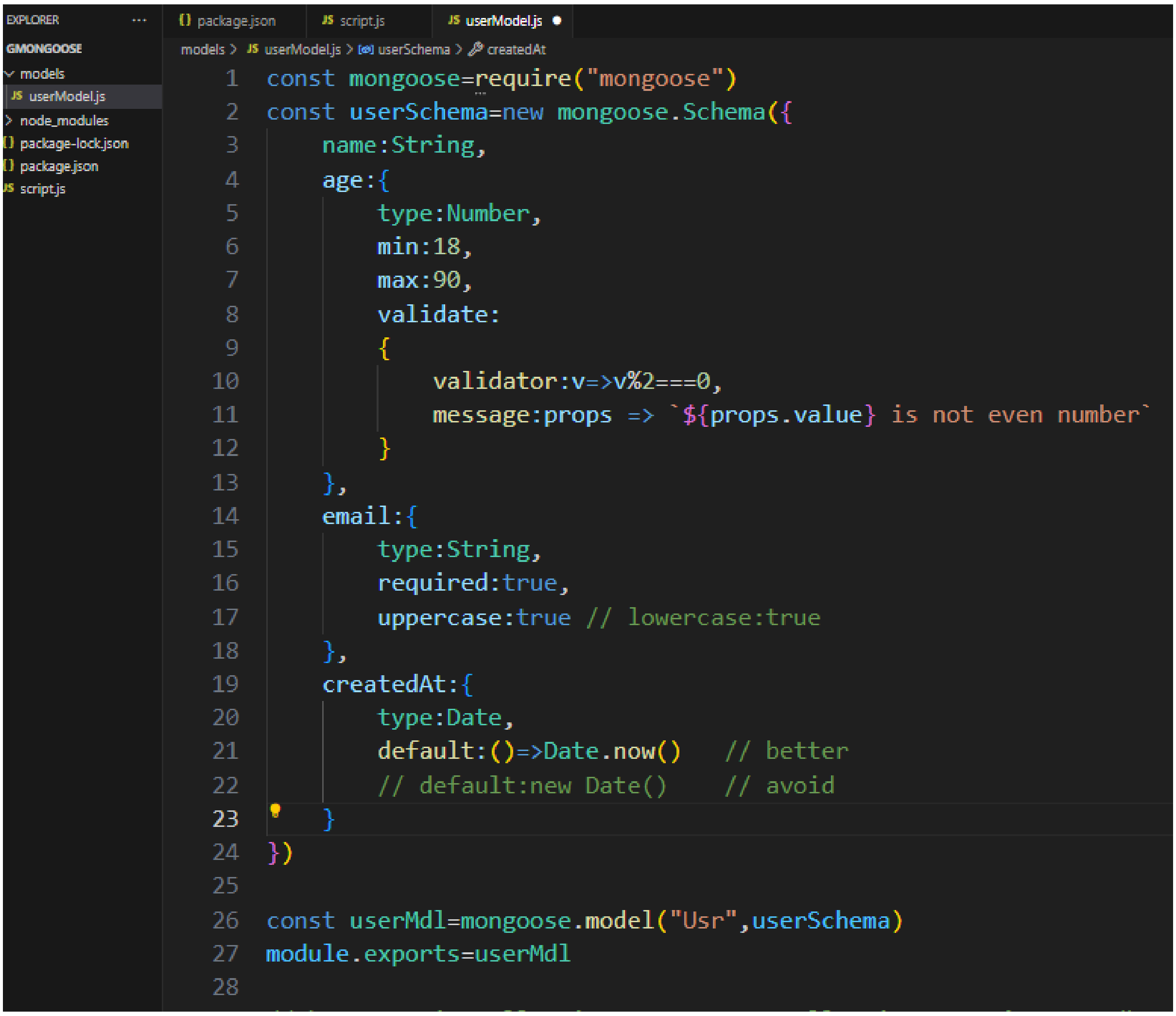


```javascript
})
//const user=new User({
  //  name:"gowthu",
    //age:42
//});
async function run()
{
    try
    {
    const newUser=await User.create({
            name:"karurgowthaman",
            age:"dfasd"
        })
        await newUser.save()
    console.log("Inserted:",newUser)
    }
    catch(e)
    {
        console.log(e.message)
    }
}
run();
```

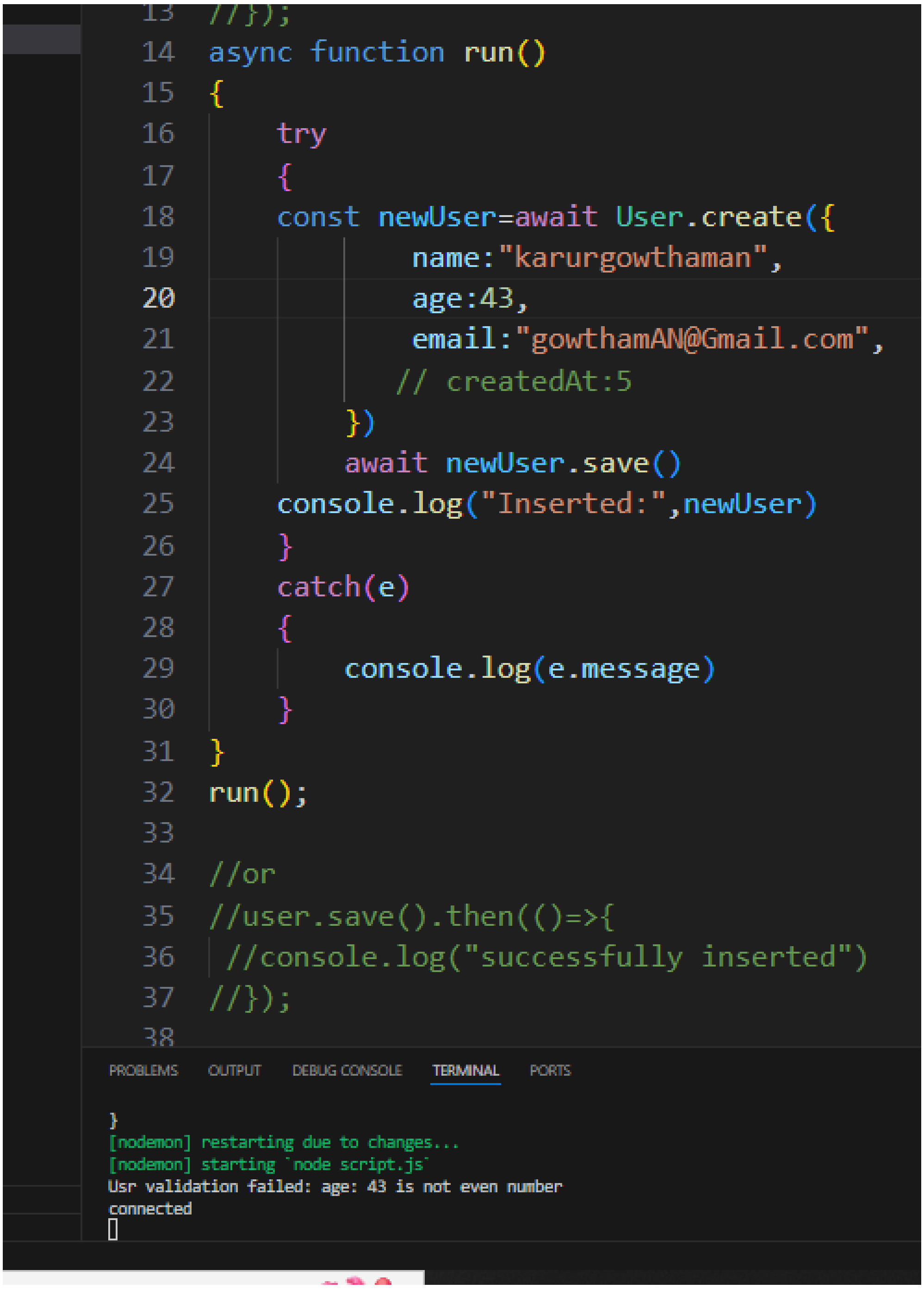
```
connected
[nodemon] restarting due to changes...
[nodemon] starting `node script.js`
Usr validation failed: age: Cast to Number failed for value "dfasd" (type string) at path "age"
connected
```

Use: console.log(e.errors)

SCHEMA VALIDATAION:



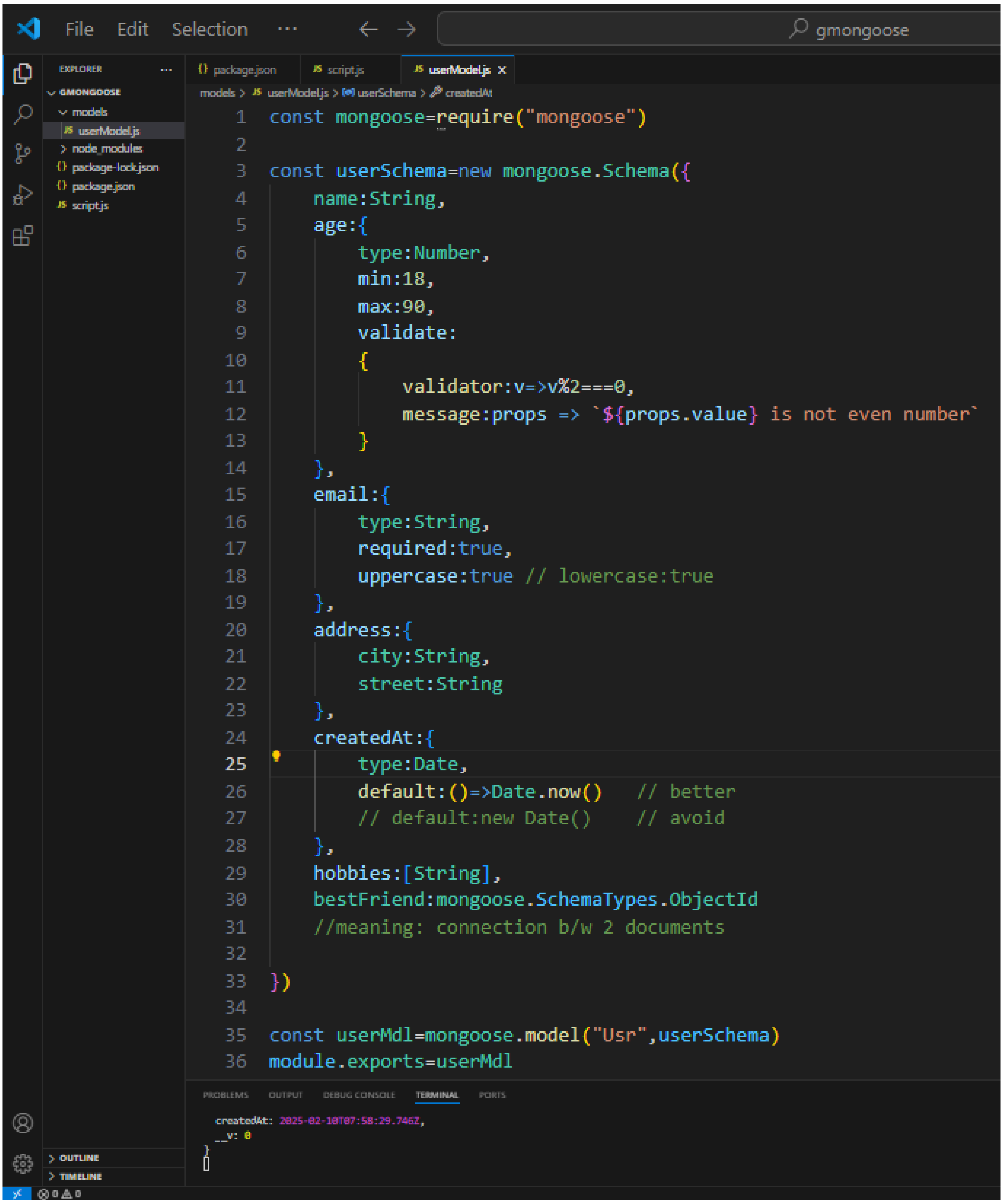
```
const mongoose=require("mongoose")
const userSchema=new mongoose.Schema({
    name:String,
    age:{
        type:Number,
        min:18,
        max:90,
        validate:
        {
            validator:v=>v%2===0,
            message:props => `${props.value} is not even number`
        }
    },
    email:{
        type:String,
        required:true,
        uppercase:true // lowercase:true
    },
    createdAt:{
        type:Date,
        default:()=>Date.now()   // better
        // default:new Date()    // avoid
    }
})

const userMdl=mongoose.model("Usr",userSchema)
module.exports=userMdl
```



```
//});
async function run()
{
    try
    {
    const newUser=await User.create({
            name:"karurgowthaman",
            age:43,
            email:"gowthamAN@Gmail.com",
           // createdAt:5
        })
        await newUser.save()
    console.log("Inserted:",newUser)
    }
    catch(e)
    {
        console.log(e.message)
    }
}
run();

//or
//user.save().then(()=>{
 //console.log("successfully inserted")
//});

```

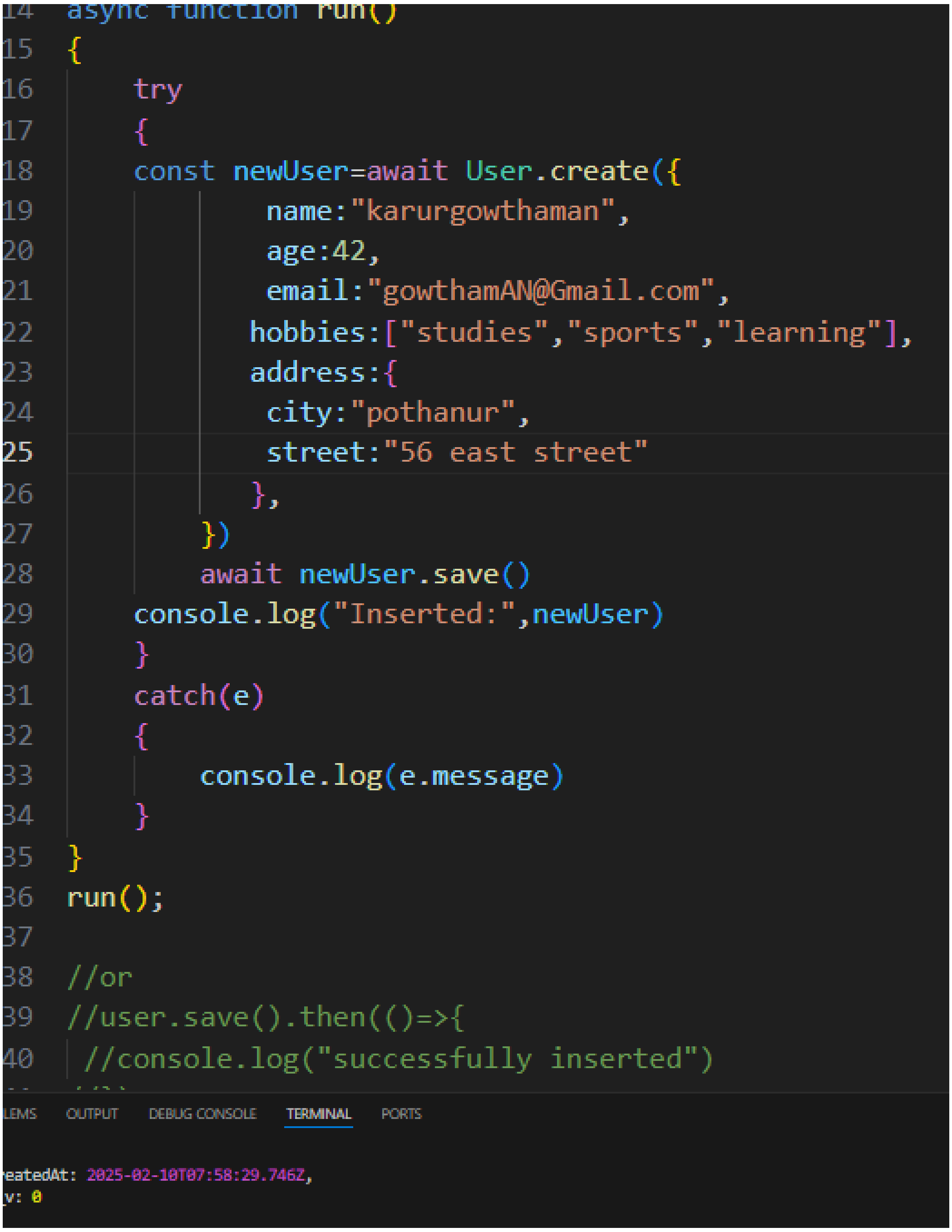
```
}
[nodemon] restarting due to changes...
[nodemon] starting `node script.js`
Usr validation failed: age: 43 is not even number
connected
```

Note: age only accept even numbers (validation)



```javascript
const mongoose=require("mongoose")

const userSchema=new mongoose.Schema({
    name:String,
    age:{
        type:Number,
        min:18,
        max:90,
        validate:
        {
            validator:v=>v%2===0,
            message:props => `${props.value} is not even number`
        }
    },
    email:{
        type:String,
        required:true,
        uppercase:true // lowercase:true
    },
    address:{
        city:String,
        street:String
    },
    createdAt:{
        type:Date,
        default:()=>Date.now()    // better
        // default:new Date()     // avoid
    },
    hobbies:[String],
    bestFriend:mongoose.SchemaTypes.ObjectId
    //meaning: connection b/w 2 documents

})

const userMdl=mongoose.model("Usr",userSchema)
module.exports=userMdl
```

```
  createdAt: 2025-02-10T07:58:29.746Z,
  __v: 0
}
```

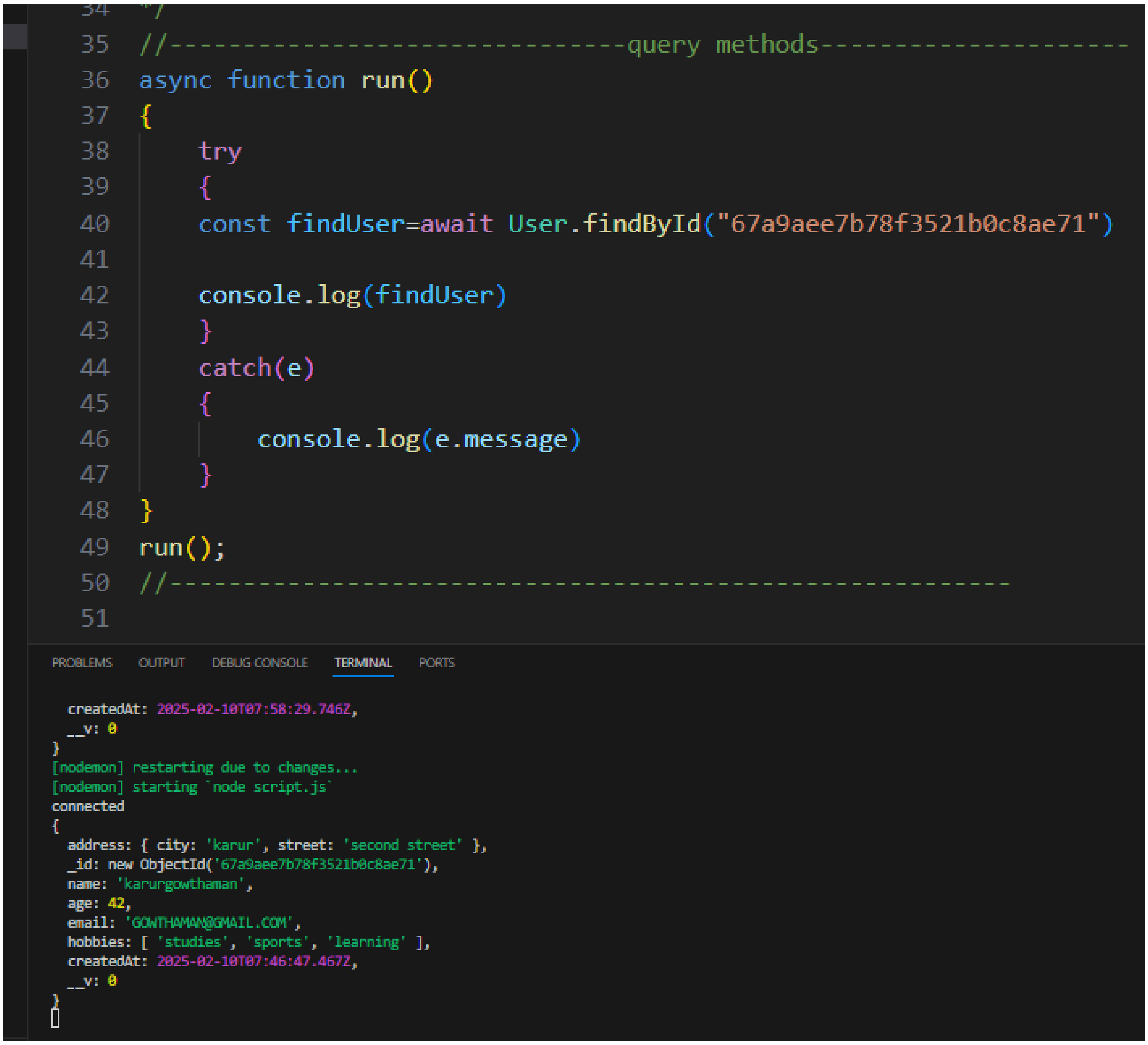


```
async function run()
{
    try
    {
    const newUser=await User.create({
            name:"karurgowthaman",
            age:42,
            email:"gowthamAN@Gmail.com",
           hobbies:["studies","sports","learning"],
           address:{
            city:"pothanur",
            street:"56 east street"
           },
        })
        await newUser.save()
    console.log("Inserted:",newUser)
    }
    catch(e)
    {
        console.log(e.message)
    }
}
run();

//or
//user.save().then(()=>{
 //console.log("successfully inserted")
```

```
eatedAt: 2025-02-10T07:58:29.746Z,
v: 0
```

Query methods:

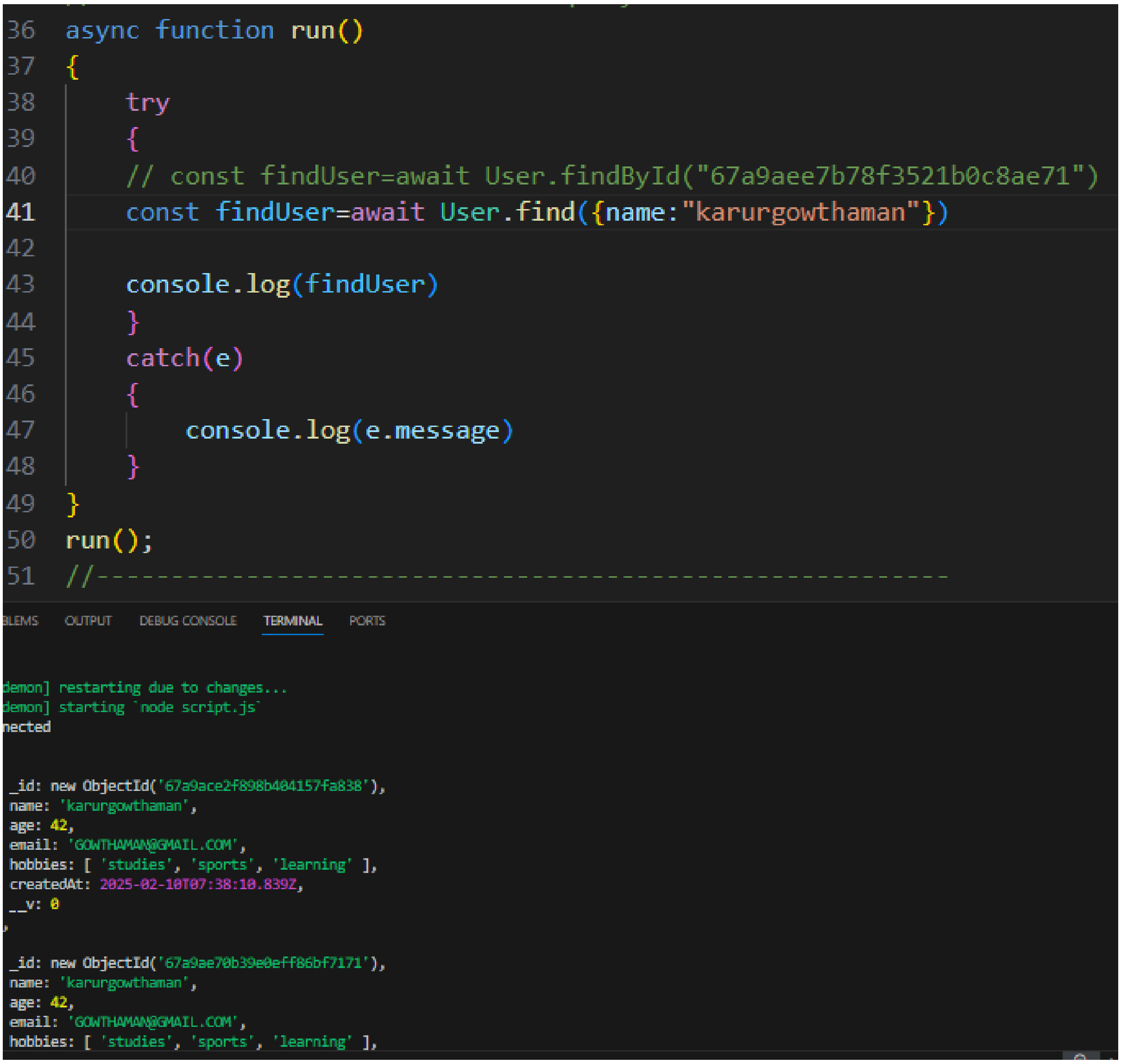
findById



```
*/
//-------------------------------query methods---------------------
async function run()
{
    try
    {
    const findUser=await User.findById("67a9aee7b78f3521b0c8ae71")

    console.log(findUser)
    }
    catch(e)
    {
        console.log(e.message)
    }
}
run();
//-----------------------------------------------------------
```

```
  createdAt: 2025-02-10T07:58:29.746Z,
  __v: 0
}
[nodemon] restarting due to changes...
[nodemon] starting `node script.js`
connected
{
  address: { city: 'karur', street: 'second street' },
  _id: new ObjectId('67a9aee7b78f3521b0c8ae71'),
  name: 'karurgowthaman',
  age: 42,
  email: 'GOWTHAMAN@GMAIL.COM',
  hobbies: [ 'studies', 'sports', 'learning' ],
  createdAt: 2025-02-10T07:46:47.467Z,
  __v: 0
}
```

find query methods:



```
async function run()
{
    try
    {
    // const findUser=await User.findById("67a9aee7b78f3521b0c8ae71")
    const findUser=await User.find({name:"karurgowthaman"})

    console.log(findUser)
    }
    catch(e)
    {
        console.log(e.message)
    }
}
run();
//----------------------------------------------------
```

```
demon] restarting due to changes...
demon] starting `node script.js`
nected


 _id: new ObjectId('67a9ace2f898b404157fa838'),
 name: 'karurgowthaman',
 age: 42,
 email: 'GOWTHAMAN@GMAIL.COM',
 hobbies: [ 'studies', 'sports', 'learning' ],
 createdAt: 2025-02-10T07:38:10.839Z,
 __v: 0
,

 _id: new ObjectId('67a9ae70b39e0eff86bf7171'),
 name: 'karurgowthaman',
 age: 42,
 email: 'GOWTHAMAN@GMAIL.COM',
 hobbies: [ 'studies', 'sports', 'learning' ],
```

task: findOne query methods (first one only appear)

use: exists query methods only show object id

ex:

//const findUser=await User.findOne({name:"karurgowthaman"})

const findUser=await User.exists({name:"karurgowthaman"})

<u>where function query methods:</u>

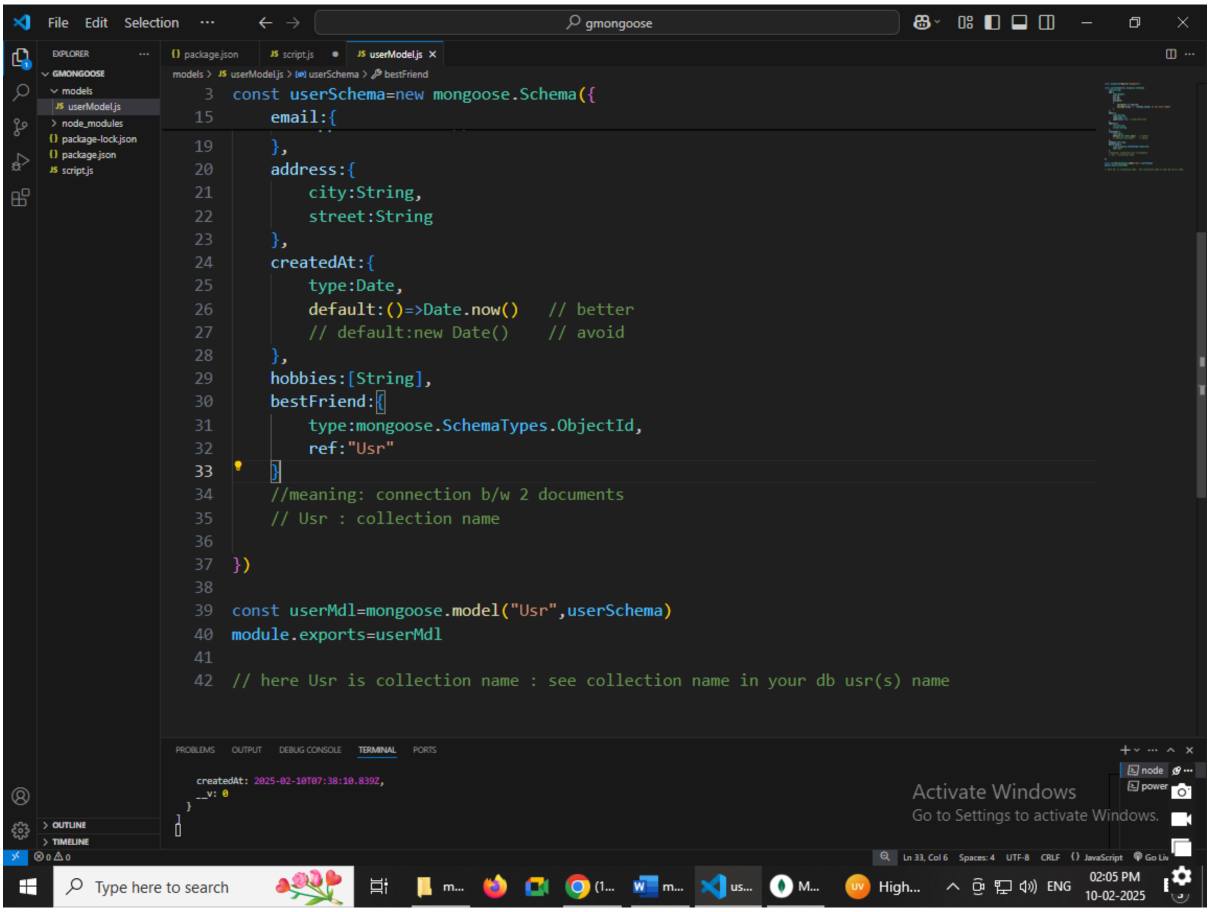
use: avoid deleteOne,deleteMany,updateOne,updateMany

const findAllUsers=await User.where('name').equals("karurgowthaman")

const findUser=await User.where('age').gt("43")

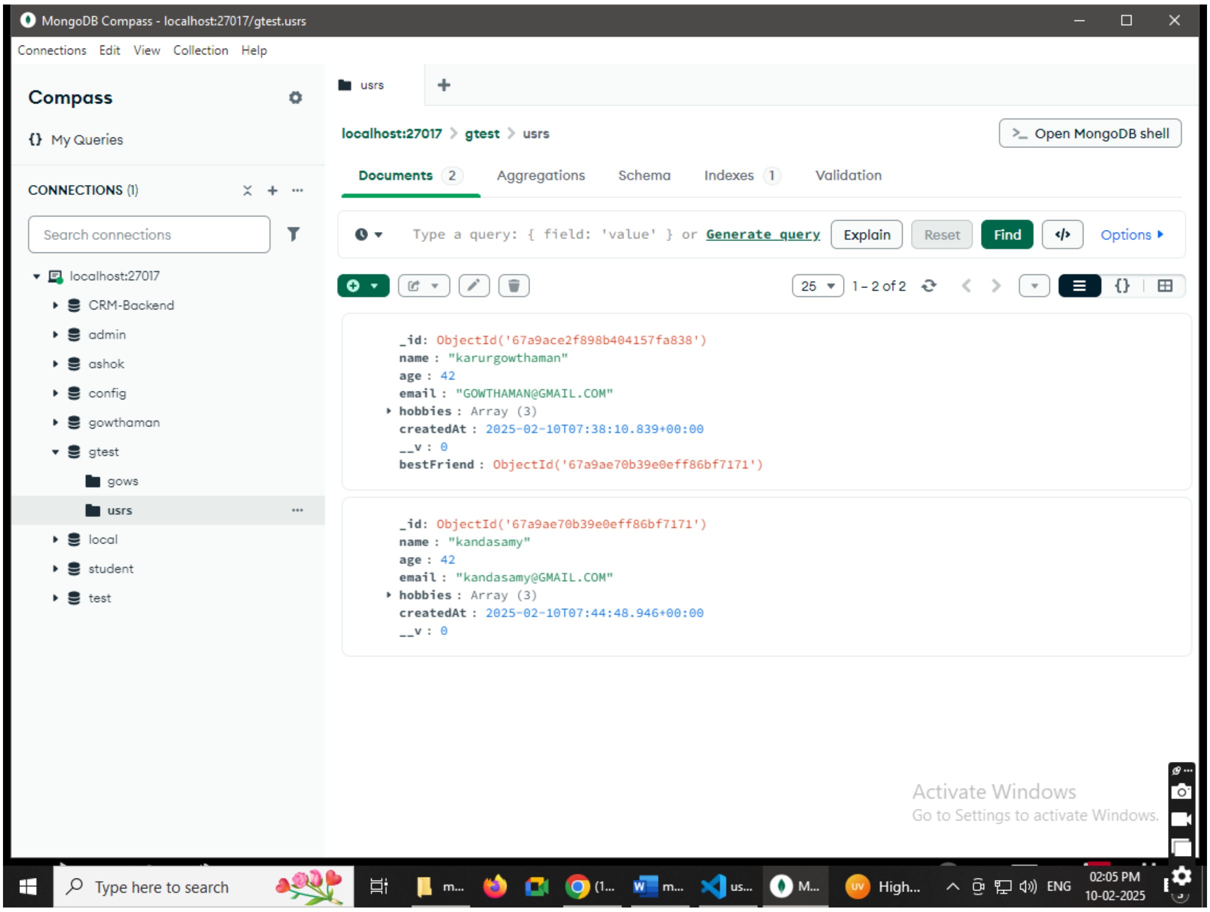
const findUser=await User.where('age').gt("40").lt('50')

const findUser=await User.where('age').gt("40").lt('50').limit(1)



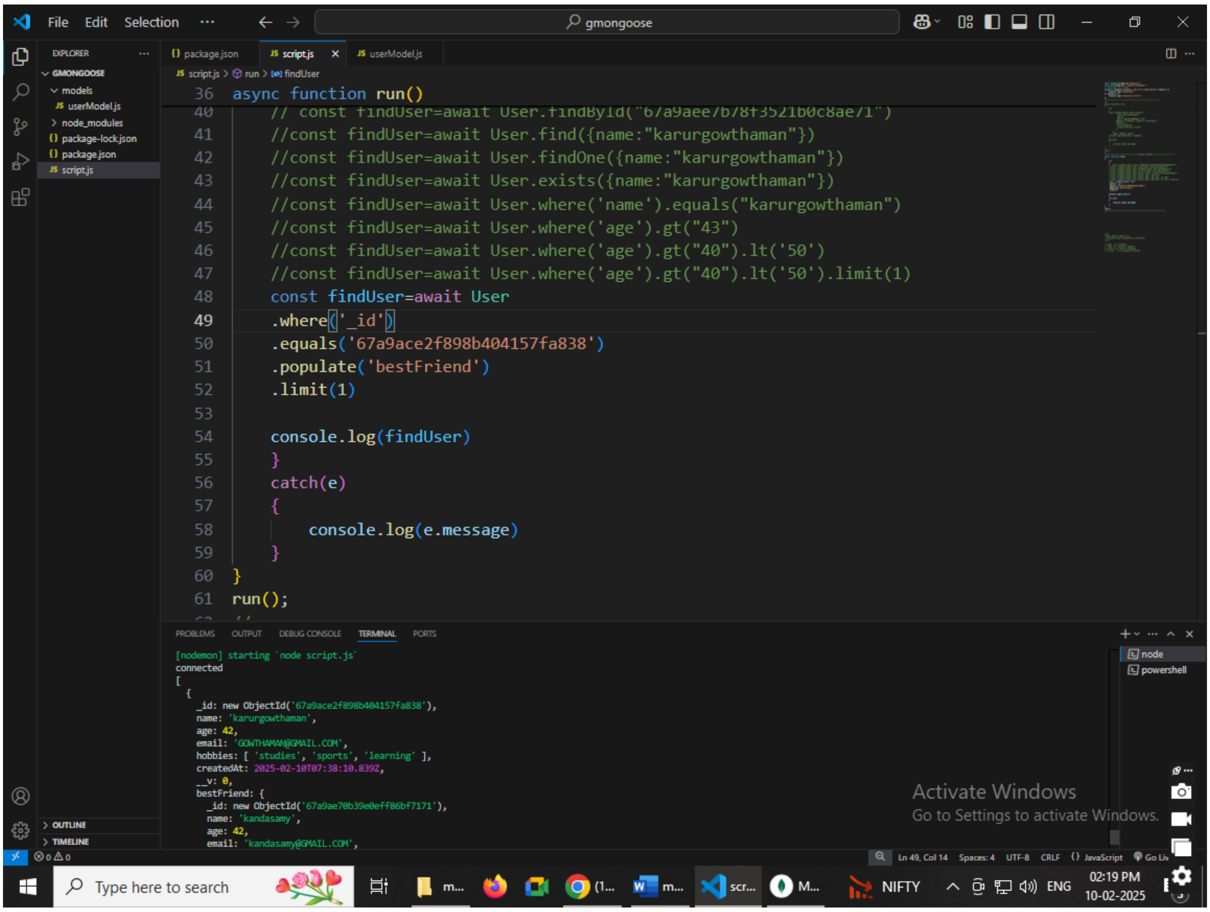
```
const userSchema=new mongoose.Schema({
    email:{
    },
    address:{
        city:String,
        street:String
    },
    createdAt:{
        type:Date,
        default:()=>Date.now()   // better
        // default:new Date()    // avoid
    },
    hobbies:[String],
    bestFriend:{
        type:mongoose.SchemaTypes.ObjectId,
        ref:"Usr"
    }
    //meaning: connection b/w 2 documents
    // Usr : collection name

})

const userMdl=mongoose.model("Usr",userSchema)
module.exports=userMdl

// here Usr is collection name : see collection name in your db usr(s) name
```

```
    createdAt: 2025-02-10T07:38:10.839Z,
    __v: 0
  }
]
```



```
_id: ObjectId('67a9ace2f898b404157fa838')
name : "karurgowthaman"
age : 42
email : "GOWTHAMAN@GMAIL.COM"
hobbies : Array (3)
createdAt : 2025-02-10T07:38:10.839+00:00
__v : 0
bestFriend : ObjectId('67a9ae70b39e0eff86bf7171')
```

```
_id: ObjectId('67a9ae70b39e0eff86bf7171')
name : "kandasamy"
age : 42
email : "kandasamy@GMAIL.COM"
hobbies : Array (3)
createdAt : 2025-02-10T07:44:48.946+00:00
__v : 0
```



```
async function run()
    // const findUser=await User.findById("67a9aee7b78f3521b0c8ae71")
    //const findUser=await User.find({name:"karurgowthaman"})
    //const findUser=await User.findOne({name:"karurgowthaman"})
    //const findUser=await User.exists({name:"karurgowthaman"})
    //const findUser=await User.where('name').equals("karurgowthaman")
    //const findUser=await User.where('age').gt("43")
    //const findUser=await User.where('age').gt("40").lt('50')
    //const findUser=await User.where('age').gt("40").lt('50').limit(1)
    const findUser=await User
    .where('_id')
    .equals('67a9ace2f898b404157fa838')
    .populate('bestFriend')
    .limit(1)

    console.log(findUser)
    }
    catch(e)
    {
        console.log(e.message)
    }
}
run();
```

```
[nodemon] starting `node script.js`
connected
[
  {
    _id: new ObjectId('67a9ace2f898b404157fa838'),
    name: 'karurgowthaman',
    age: 42,
    email: 'GOWTHAMAN@GMAIL.COM',
    hobbies: [ 'studies', 'sports', 'learning' ],
    createdAt: 2025-02-10T07:38:10.839Z,
    __v: 0,
    bestFriend: {
      _id: new ObjectId('67a9ae70b39e0eff86bf7171'),
      name: 'kandasamy',
      age: 42,
      email: 'kandasamy@GMAIL.COM',
```

Schema Methods: