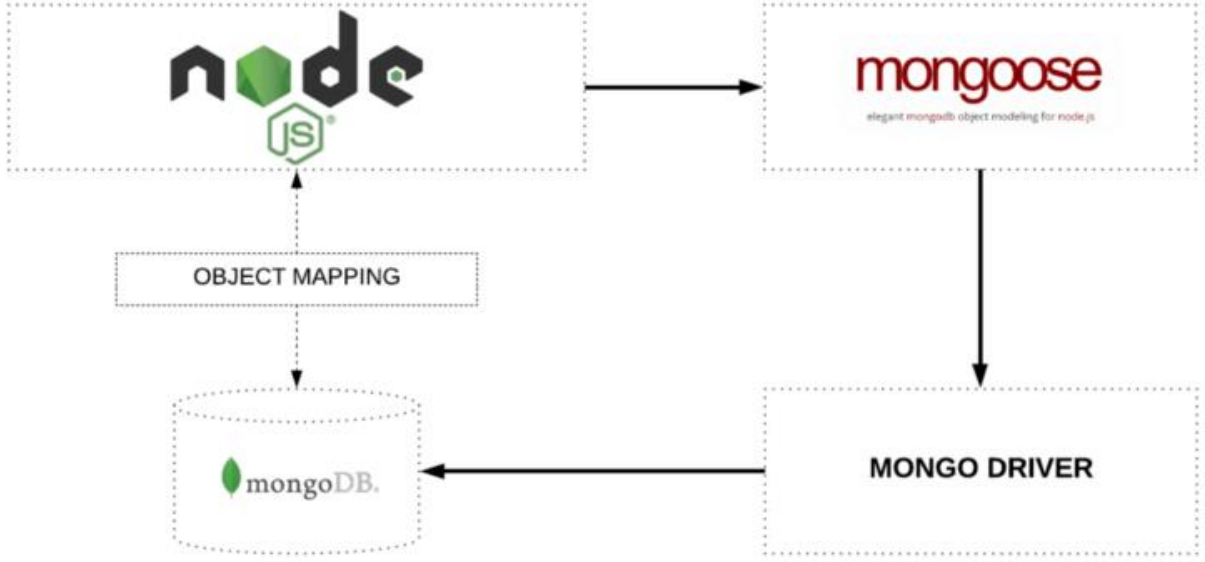
Mongoose

#Node JS Notes

Mongoose

What is Mongoose

- Mongoose is an Object Data Modeling (ODM) library for MongoDB and Node.js.
- It manages relationships between data, provides schema validation, and is used to translate between objects in code and the representation of those objects in MongoDB.



SQL

PERSON

Id	FirstName	LastName	Email
1	Ada	Lovelace	ada.lovelace@gmail.com
2	Grace	Hopper	grace.hopper@gmail.com
3	Kathy	Sierra	kathy.sierra@gmail.com

PHONE_NUMBER

PersonId	PhoneId	Phone Number	Type
1	1	+1.123.456.7890	Home
1	2	+1.111.222.3333	Work

MONGO

PEOPLE

```
{
        "Id": 1,
        "FirstName": "Ada",
        "LastName": "Lovelace",
        "Email": "ada.lovelace@gmail.com",
        "Phone": [{
                        "Home": "+1.123.456.7890"
                },
                {
                        "Work": "+1.111.222.3333"
                }
        ]
}
```

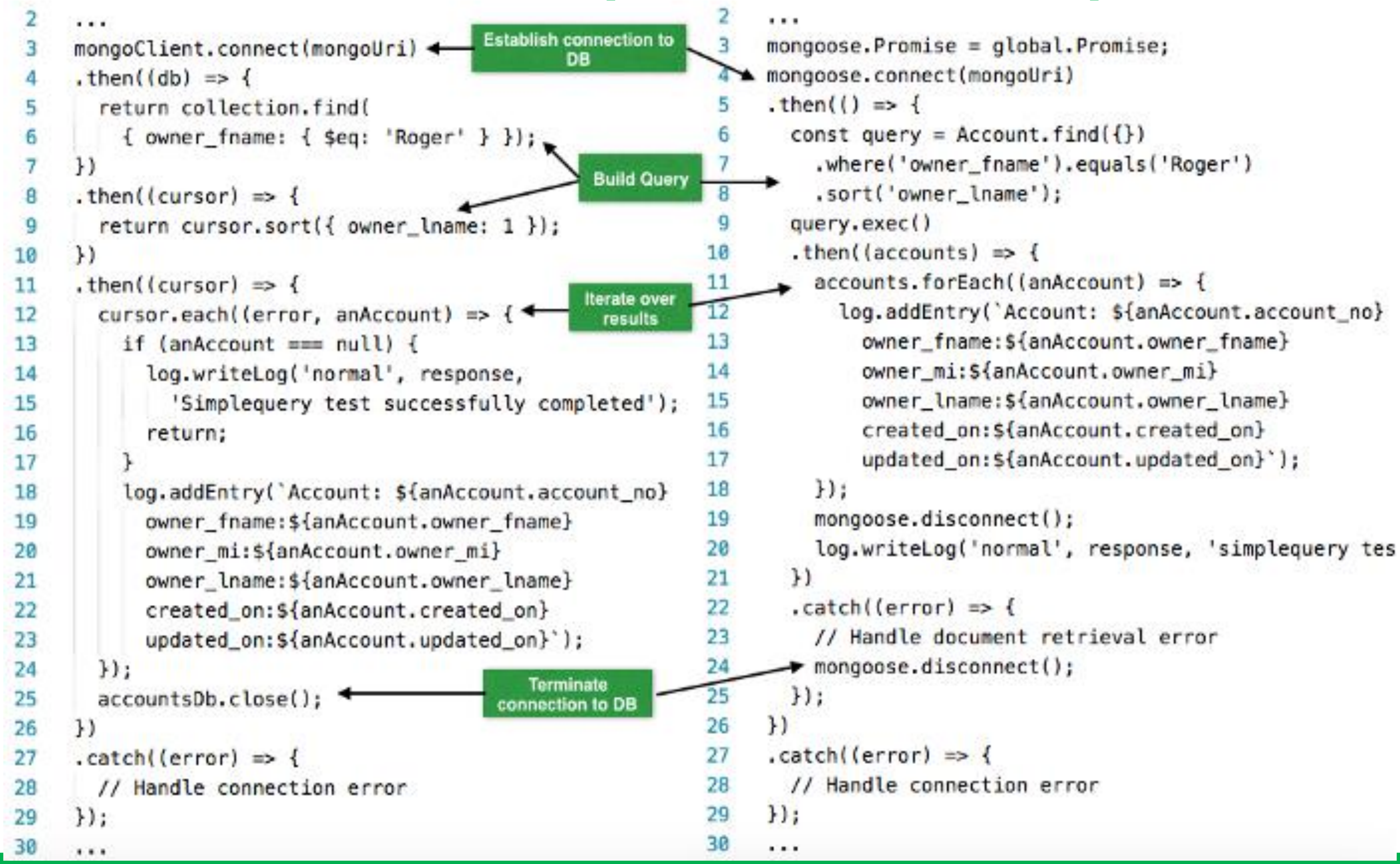
```
{
        "Id": 2,
        "FirstName": "Grace",
        "LastName": "Hopper",
        "Email": "grace.hopper@gmail.com"
}
```

```
{
        "Id": 3,
        "FirstName": "Kathy",
        "LastName": "Sierra",
        "Email": "kathy.sierra@gmail.com"
}
```

MongoDB Dictionary

Term	Description
Database	An organized collection of one or more data set.
Collection	It’s a single or multiple sets of documents.
Document	A structured set of documents in the form of Key/Value pair, same like JSON objects.
Schema	Data structure type for String, Number, Date, Buffer, Boolean, Mixed, ObjectID, Array in a MongoDB document.
Model	MongoDB constructors are fancy constructors, and it accepts a particular schema to make an instance of a MongoDB document. The model helps in retrieval and creating documents from a collection.

Native-MongoDB vs Mongoose



```
...
mongoClient.connect(mongoUri)
.then((db) => {
  return collection.find(
    { owner_fname: { $eq: 'Roger' } });
})
.then((cursor) => {
  return cursor.sort({ owner_lname: 1 });
})
.then((cursor) => {
  cursor.each((error, anAccount) => {
    if (anAccount === null) {
      log.writeLog('normal', response,
        'Simplequery test successfully completed');
      return;
    }
    log.addEntry(`Account: ${anAccount.account_no}
      owner_fname:${anAccount.owner_fname}
      owner_mi:${anAccount.owner_mi}
      owner_lname:${anAccount.owner_lname}
      created_on:${anAccount.created_on}
      updated_on:${anAccount.updated_on}`);
  });
  accountsDb.close();
})
.catch((error) => {
  // Handle connection error
});
...
```

```
...
mongoose.Promise = global.Promise;
mongoose.connect(mongoUri)
.then(() => {
  const query = Account.find({})
    .where('owner_fname').equals('Roger')
    .sort('owner_lname');
  query.exec()
  .then((accounts) => {
    accounts.forEach((anAccount) => {
      log.addEntry(`Account: ${anAccount.account_no}
        owner_fname:${anAccount.owner_fname}
        owner_mi:${anAccount.owner_mi}
        owner_lname:${anAccount.owner_lname}
        created_on:${anAccount.created_on}
        updated_on:${anAccount.updated_on}`);
    });
    mongoose.disconnect();
    log.writeLog('normal', response, 'simplequery tes
  })
  .catch((error) => {
    // Handle document retrieval error
    mongoose.disconnect();
  });
})
.catch((error) => {
  // Handle connection error
});
...
```

SchemaTypes

- Mongoose currently contains eight SchemaTypes that a property is saved as when it is persisted to MongoDB. They are:
  1. String
  2. Number
  3. Date
  4. Buffer
  5. Boolean
  6. Mixed
  7. ObjectId
  8. Array
  9. Decimal128
  10. Map

https://mongoosejs.com/docs/schematypes.html

Each data type allows you to specify:

- a default value
- a custom validation function
- indicate a field is required
- a get function that allows you to manipulate the data before it is returned as an object
- a set function that allows you to manipulate the data before it is saved to the database
- create indexes to allow data to be fetched faster

How to Install

- npm install mongoose --save

Connection

- var mongoose = require('mongoose');
- mongoose.connect('mongodb://localhost:27017/demo', {useNewUrlParser: true, useUnifiedTopology: true});

Referencing Mongoose

- let mongoose = require('mongoose')
- const Schema = mongoose.Schema;

Defining a Mongoose Schema

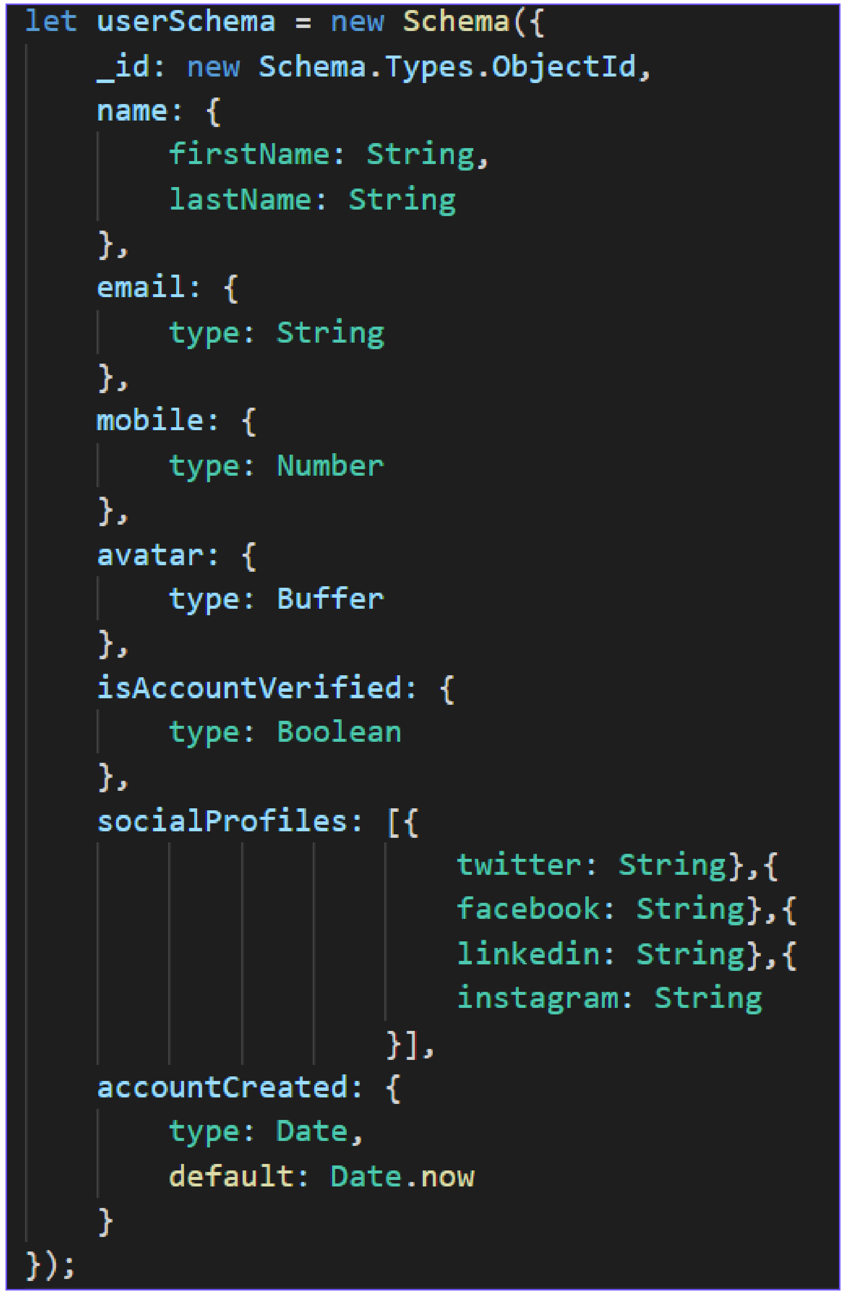
- A schema defines document properties through an object where the key name corresponds to the property name in the collection.

```
const Schema = mongoose.Schema;

var userSchema = mongoose.Schema({
   firstName: String,
   lastName: String
});
```

Creating and Saving Mongoose Models

- var User = mongoose.model('User', userSchema);



```
let userSchema = new Schema({
    _id: new Schema.Types.ObjectId,
    name: {
        firstName: String,
        lastName: String
    },
    email: {
        type: String
    },
    mobile: {
        type: Number
    },
    avatar: {
        type: Buffer
    },
    isAccountVerified: {
        type: Boolean
    },
    socialProfiles: [{
                        twitter: String},{
                        facebook: String},{
                        linkedin: String},{
                        instagram: String
                    }],
    accountCreated: {
        type: Date,
        default: Date.now
    }
});
```

```
let userSchema = new Schema({
  _id: new Schema.Types.ObjectId,
  name: {
    firstName: String,
    lastName: String
  },
  email: {
    type: String
  },
  mobile: {
    type: Number
  },
  avatar: {
    type: Buffer
  },
  isAccountVerified: {
    type: Boolean
  },
  socialProfiles: [{
            twitter: String},{
            facebook: String},{
            linkedin: String},{
            instagram: String
          }],
  accountCreated: {
    type: Date,
    default: Date.now
  }
});
```

Schema Type Options

- In addition to the type property, you can also define additional properties for a path. For example, if you want to change the case of the string, you can use lowercase and uppercase properties:

```
const schema = new Schema({
   title: String,
   permalink: {
      type: String,
      lowercase: true
   },
   uuid: {
      type: String,
      uppercase: true
   }
});
```

Validator

built-in Mongoose Schema validators

- Mongoose gives us a set of useful built-in validation rules such:
- Required validator checks if a property is not empty.
- Numbers have min and max validators.
- Strings have enum, match, minlength, and maxlength validators.

1. **Strings**: minlength, maxlength, match, enum
2. **Numbers**: min, max
3. **Dates**: min, max
4. **All types**: required

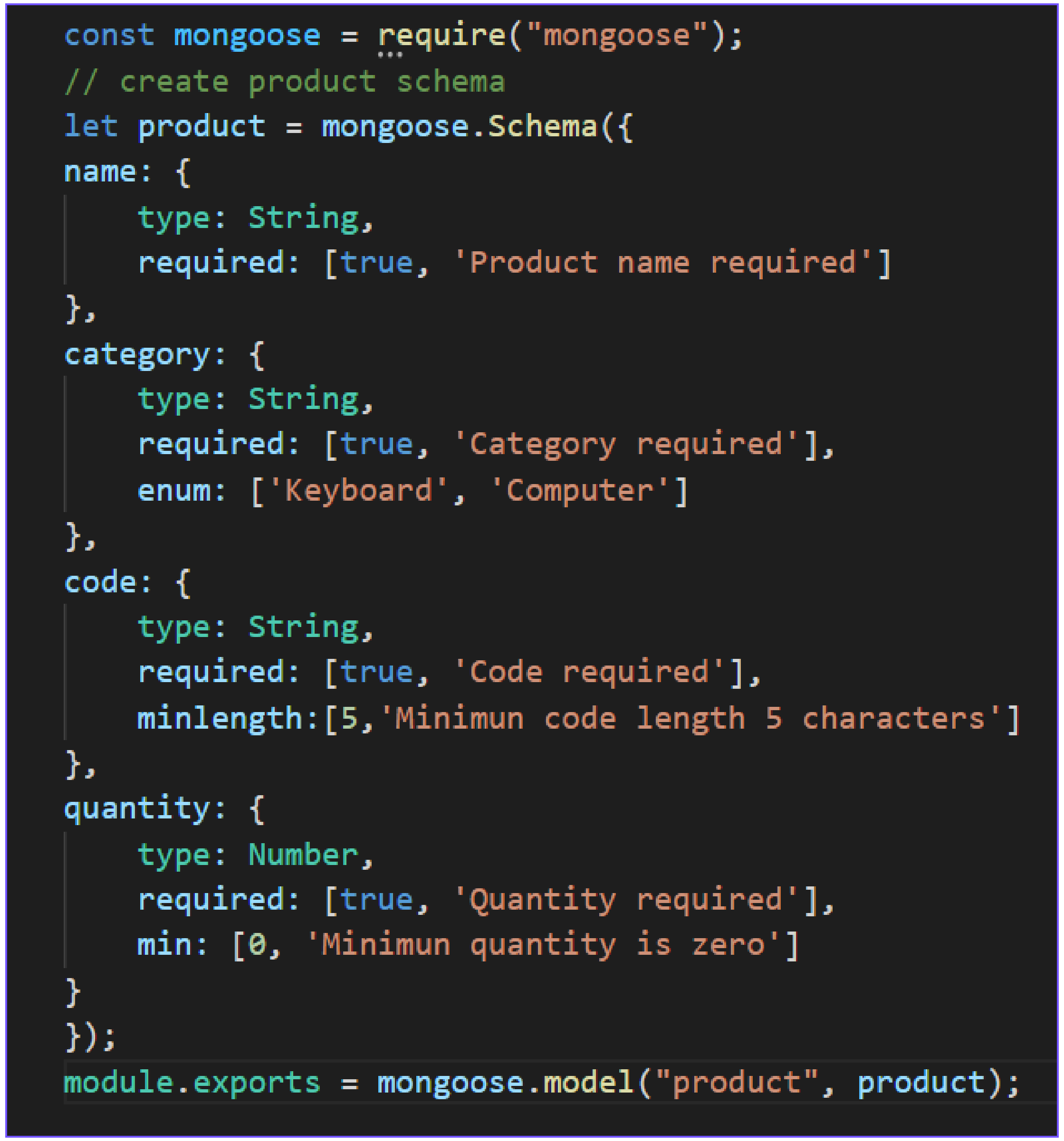
https://mongoosejs.com/docs/validation.html

- **Required**
  - Required validator takes an array with 2 items, first a boolean var and a message to return the validation if it fails.
  - required: [true, 'Product name required']
- **Enum**
  - Enum validator takes an array with items to check if the property is equal with one of the given array items.
  - enum: ['Keyboard', 'Computer']
- **Min**
  - Min validator takes an array with 2 items, first a number to set a minimum value and a message to return the validation if it fails.
  - **Max, minlength and maxlegth** works as same as min validator.
  - min: [0, 'Minimun quantity is zero']

Example

```
title: {
      type: String,
      required: true,
      minlength: 4,
      maxlength: 200
   }
```

```
tags: {
      type: [String],
      required: true,
      enum: ['sports', 'racing', 'action', 'rpg']
   },
```



```
const mongoose = require("mongoose");
// create product schema
let product = mongoose.Schema({
name: {
    type: String,
    required: [true, 'Product name required']
},
category: {
    type: String,
    required: [true, 'Category required'],
    enum: ['Keyboard', 'Computer']
},
code: {
    type: String,
    required: [true, 'Code required'],
    minlength:[5,'Minimun code length 5 characters']
},
quantity: {
    type: Number,
    required: [true, 'Quantity required'],
    min: [0, 'Minimun quantity is zero']
}
});
module.exports = mongoose.model("product", product);
```

Custom Validators

- If the built-in validators aren't enough, you can define custom validators to suit your needs.

```
 phone: {
   type: String,
   validate: {
     validator: function(v) {
       return /\d{3}-\d{3}-\d{4}/.test(v);
     },
     message: props => `${props.value} is not a valid phone
number!`
   },
   required: [true, 'User phone number required']
 }
```

Third Party Package

- Mongoose Validator simply returns Mongoose style validation objects that utilises validator.js for the data validation.
- https://www.npmjs.com/package/mongoose-validator

Query Building

- Mongoose has a very rich API that handles many complex operations supported by MongoDB.
- Consider a query where we can incrementally build query components.

Queries

- Create Methods:
  - Model.save()
- Read Methods:
  - Model.find()
  - Model.findById()
  - Model.findOne()
- Update Methods:
  - Model.updateMany()
  - Model.updateOne()
- Delete Methods
  - Model.deleteMany()
  - Model.deleteOne()

Example

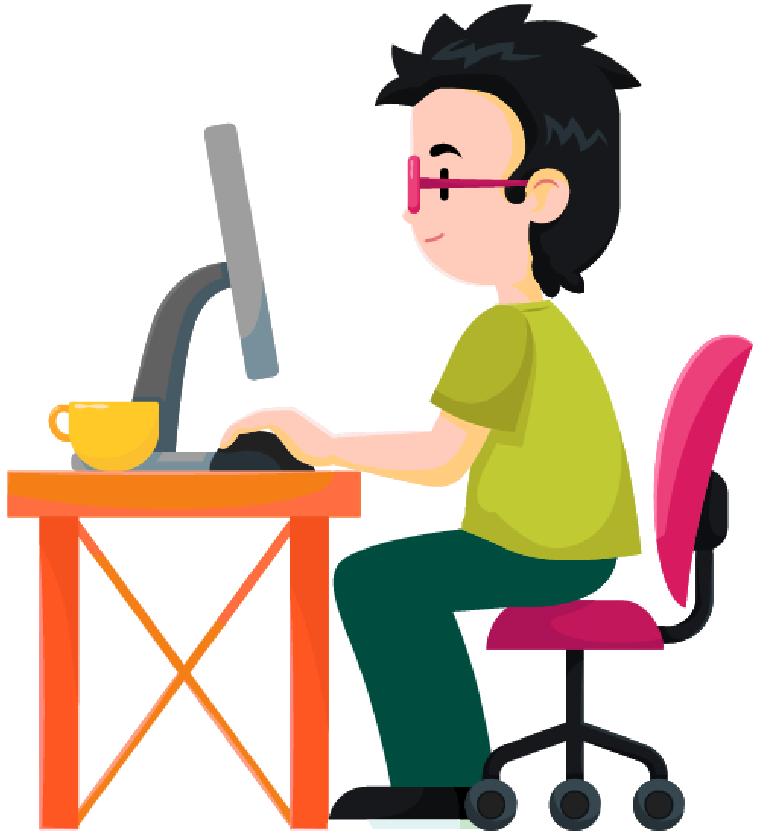
1. Find all users
2. Skip the first 100 records
3. Limit the results to 10 records
4. Sort the results by the firstName field
5. Select the firstName
6. Execute that query

```
UserModel.find()                    // find all users
    .skip(100)                      // skip the first 100 items
    .limit(10)                      // limit to 10 items
    .sort({firstName: 1}            // sort ascending by firstName
    .select({firstName: true}       // select firstName only
    .exec()                         // execute the query
    .then(docs => {
      console.log(docs)
     })
    .catch(err => {
      console.error(err)
     })
```

Get Exclusive Video Tutorials

www.aptutorials.com

https://www.youtube.com/user/Akashtips



Get More Details

www.akashsir.com

If You Liked It !

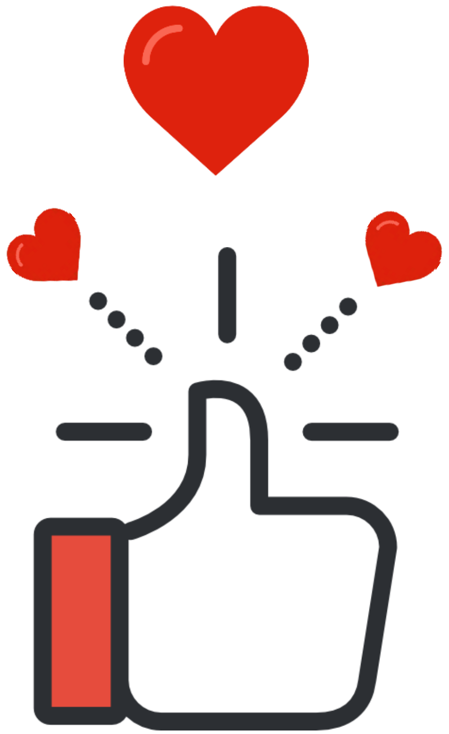
Rating Us Now

Just Dial

https://www.justdial.com/Ahmedabad/Akash-Technolabs-Navrangpura-Bus-Stop-Navrangpura/079PXX79-XX79-170615221520-S5C4_BZDET

Sulekha

https://www.sulekha.com/akash-technolabs-navrangpura-ahmedabad-contact-address/ahmedabad



Connect With Me

Akash Padhiyar
#AkashSir

www.akashsir.com
www.akashtechnolabs.com
www.akashpadhiyar.com
www.aptutorials.com

# Social Info

Akash.padhiyar
Akashpadhiyar
Akash_padhiyar
+91 99786-21654

#Akashpadhiyar
#aptutorials