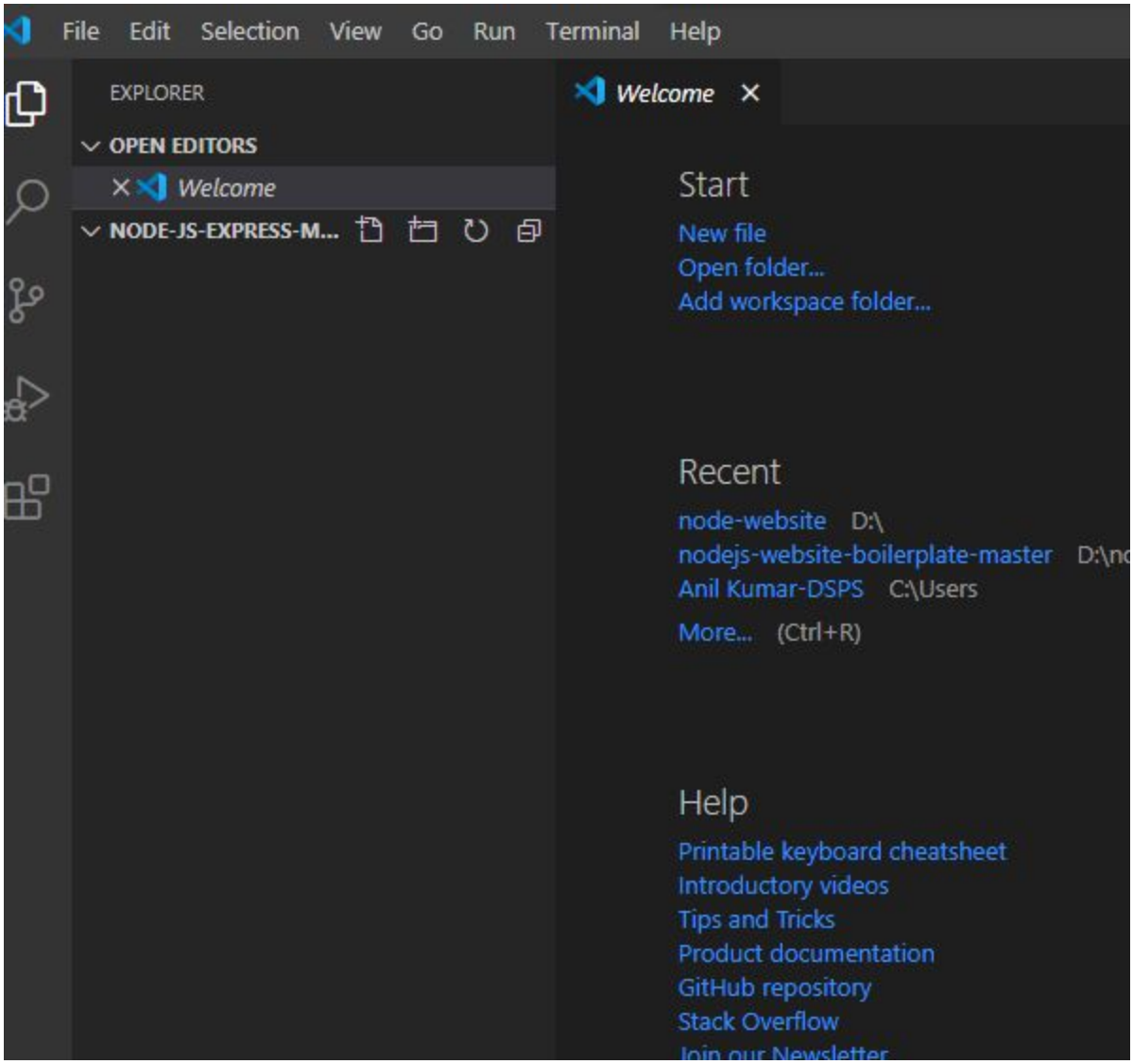
Creating Node.js Application - API

1. Create a Folder anywhere in your computer with name “nodejs-express-mysql”
2. Open the Folder in your Visual Studio



3. Open the terminal of Visual Studio

4. Initialize the Node.js application with package.json file

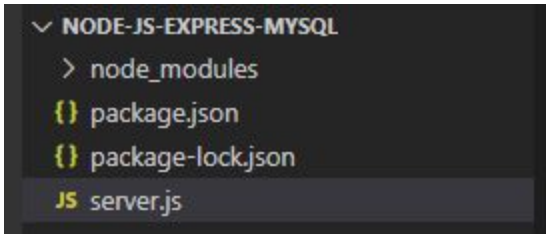
```
PS D:\node-js-express-mysql> npm init
This utility will walk you through creating a package.json file.
It only covers the most common items, and tries to guess sensible defaults.

See `npm help init` for definitive documentation on these fields
and exactly what they do.

Use `npm install <pkg>` afterwards to install a package and
save it as a dependency in the package.json file.

Press ^C at any time to quit.
package name: (js-express-mysql)
version: (1.0.0)
description:
entry point: (index.js)
test command:
git repository:
keywords:
author:
license: (ISC)
```

```
About to write to D:\node-js-express-mysql\package.json:

{
  "name": "js-express-mysql",
  "version": "1.0.0",
  "description": "",
  "main": "index.js",
  "scripts": {
    "test": "echo \"Error: no test specified\" && exit 1"
  },
  "author": "",
  "license": "ISC"
}


Is this OK? (yes) yes
```

5. I need to install modules : express, mysql and body-parser

```
PS D:\node-js-express-mysql> npm install express mysql body-parser --save
npm notice created a lockfile as package-lock.json. You should commit this file.
npm WARN js-express-mysql@1.0.0 No description
npm WARN js-express-mysql@1.0.0 No repository field.

+ body-parser@1.19.0
+ mysql@2.18.1
+ express@4.17.1
added 59 packages from 48 contributors and audited 59 packages in 3.486s
found 0 vulnerabilities
```

6. Setup Express Web server
    a. Create the file name as “server.js” in your root directory



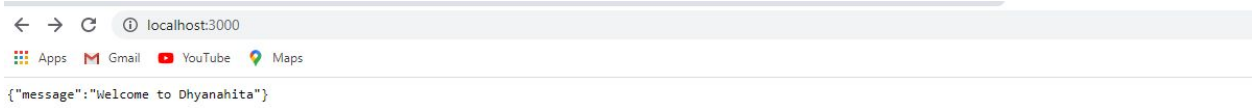
Build the server for request and response

```
JS server.js > ...
1   const express = require('express');
2   const bodyparser = require('body-parser');
3
4   const app = express();
5   //Content-type : application/json
6   app.use(bodyparser.json());
7
8   app.use(bodyparser.urlencoded({extended:true}));
9
10  app.get("/",(req,res)=>{
11      res.json({message:'Welcome to Dhyanahita'});
12  });
13
14  app.listen(3000,()=>{
15      console.log('Server is running on port 3000');
16  });
```

Run the server.,js file as node server.js

```
PS D:\node-js-express-mysql> node .\server.js
Server is running on port 3000
```

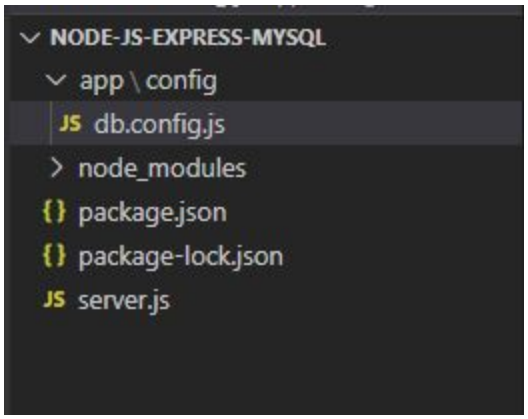
Go to browser and type localhost:3000



7. Create a Table on MySQL server

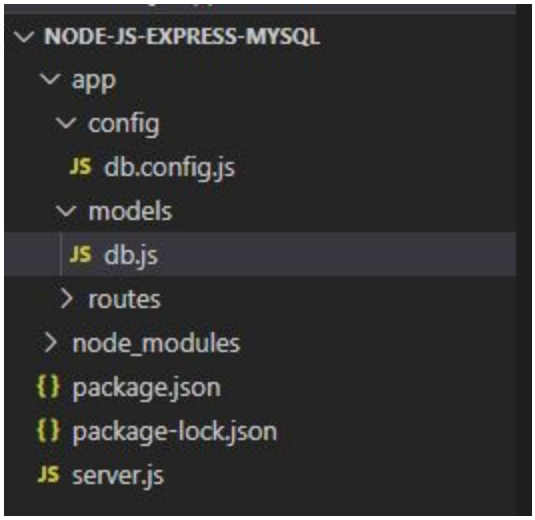
```sql
use testdb;

create table customers(
    Id int auto_increment,
    email varchar(50) not  null,
    name varchar(50) not null,
    active boolean default false,
    primary key(Id)
);

select * from customers;
```

8. Configure and connection with MySQL Database




```
app > config > JS db.config.js > ...
1   module.exports = {
2       HOST : "localhost",
3       USER : "root",
4       PASSWORD : "dsps@123",
5       DB:"testdb"
6   };
```

\



app > models > db.js > ...

```
1   const mysql = require('mysql');
2   const dbConfig = require('../config/db.config.js');
3   const connection = mysql.createConnection({
4       host:dbConfig.HOST,
5       user:dbConfig.USER,
6       password:dbConfig.PASSWORD,
7       database:dbConfig.DB
8   });
9
10  connection.connect(error =>{
11      if(error){
12          return console.error(error.message);
13      }
14      console.log('Successfully connected to MySQL DATABASE');
15  });
16
17  module.exports = connection;
```

9. Creating the Model in application
    a. Create a New user
    b. Find the customer by Id
    c. Get all customers
    d. Update a customer by Id
    e. Remove customer
    f. Remove all customers

app > models > JS customer.model.js > ...

```
const sql = require('./db.js');

const Customer = function(customer){
    this.email = customer.email;
    this.name = customer.name;
    this.active = customer.active;
};

Customer.create = (newCustomer,result) => {
    sql.query('Insert into customers set ?',newCustomer,(err,res) =>{
        if(err){
            console.log(err);
            result(err,null);
            return;
        }
        console.log("Created Customer : ",{id:res.insertedId,...newCustomer});
        return (null,{id:res.insertedId,...newCustomer});
    })
};

Customer.findById = (customerId,result) => {
    sql.query('select * from customers where Id = ${customerID}',(err,res) =>{
        if(err){
            console.log(err);
            result(err,null);
            return;
        }
        if(res.length){
            console.log('found customer:',res[0]);
            result(null,res[0]);
            return;
        }
        result({kind:'not_found'},null);
    })
};

Customer.getAll = result =>{
    sql.query('select * from customers',(err,res) =>{
        if(err){
            console.log(err);
            result(err,null);
            return;
        }
        console.log('Customers : ',res);
        result(null,res);
    })
};
```

```
Customer.updateById = (id,customer,result) =>{
    sql.query('Update customers set email = ?,name = ?,active = ? where id = ?',
    [customer.email,customer.name,customer.active,id],(err,res) =>{
        if(err){
            console.log(err);
            result(null,res);
            return;
        }
        if(res.affectedRows == 0){
            result({kind:'Not_Found'},null);
            return ;
        }
        console.log('updated customer : ',{id:id,...customer});
        result(null,{id:id,...customer});
    });
};

Customer.remove = (id,result) =>{
    sql.query('delete from customers where id = ?',id,(err,res)=>{
        if(err){
            console.log(err);
            result(null,res);
            return;
        }
        if(res.affectedRows == 0){
            result({kind:'Not_Found'},null);
            return ;
        }
        console.log('deleted customer with id ',id);
        result(null,res);
    });
};


Customer.removeAll = result =>{
    sql.query('delete from customers',(err,res) =>{
        if(err){
            console.log(err);
            result(null,res);
            return;
        }
        console.log('delete ${res.affectedRows} customers');
        result(null,res);
    });
};
```

```
94
95
96  module.exports = Customer;
```

App->Routes - > customer.routes.js

app > routes > JS customer.routes.js > ...

```
1   module.exports = app =>{
2       const customers = require('../controllers/customer.controller.js');
3
4       //create a new customer
5       app.post("/customers",customers.create);
6       //retrieve all the users
7       app.get('/customers',customers.findAll);
8       //single user
9       app.get('/customers/:customerId',customers.findOne);
10      //update the customer with customerId
11      app.get('/customers/:customerId',customers.update);
12      //delete a customer with customerID
13      app.get('customers/:customerId',customers.delete);
14      //delete all
15      app.get('/customers',customers.deleteAll);
16  };
```

JS server.js > ...

```
1   const express = require('express');
2   const bodyparser = require('body-parser');
3
4   const app = express();
5   //Content-type : application/json
6   app.use(bodyparser.json());
7
8   app.use(bodyparser.urlencoded({extended:true}));
9
10  app.get("/",(req,res)=>{
11      res.json({message:'Welcome to Dhyanahita'});
12  });
13
14  require('./app/routes/customer.routes.js')(app);
15
16
17  app.listen(3000,()=>{
18      console.log('Server is running on port 3000');
19  });
```

Download the PostMan from this URL: https://www.postman.com/downloads/
Used for test the api with input and checks the output

Please download this file from : http://tiny.cc/JSFile
And place in app>controller>