LAB-6 Express CRUD API for Customers data

Introduction:

Creating basic CRUD functions for adding, getting, updating, removing customers data using NodeJS and calling functions from command line using express framework.

Main Objective:

Task-1: Creating a new CRUD API for Customers data with fields like customer_id, customer_name, customer_email. Creating methods for adding a new customer, listing all the customers, updating a customer and deleting a customer.
Task-2: Creating command line functions using express framework.

Design and Implementation of tasks:

Task-1: Creating a new CRUD API for Customers data with fields like customer_id, customer_name, customer_email. Creating methods for adding a new customer, listing all the customers, updating a customer and deleting a customer:

1. Imported fs module to make use of utility functions for opening,

reading and writing to a file.

```
const fs = require('fs');
```

2. Created utility functions like fetchCustomers and saveCustomers which will be reused multiple times by the CRUD functions.
   fetchCustomers used to read a file and load the data in JSON format.
   saveCustomers function saves the JSON variable to the file(writing to a file)

```
var fetchCustomers = () => {
try { //if file won't exist
var customersString = fs.readFileSync('customers-data.json')
return JSON.parse(customersString);
} catch(e){
return [];
}
};

var saveCustomers = (customers) => {
fs.writeFileSync('customers-data.json',JSON.stringify(customers));
};
```

3. Implemented function to create a customer using id, name and email address.

```
var addCustomer = (id, name, email) => {
var customers = fetchCustomers();
```

```
var customer = {id, name, email}

var duplicateCustomers = customers.filter((customer) => { // to check if customer already exists
return customer.id === id;
});

if (duplicateCustomers.length === 0){
customers.push(customer);
saveCustomers(customers);
return customer;
}

};
```

4. Implemented function to get details about a customer using ID.

```
var getCustomer = (id) => {

var customers = fetchCustomers();

var getCustomers = customers.filter((customer) => { // to check if customer exists and returns the customer
return customer.id === id;
});

return getCustomers[0];

};
```

5. Implemented function to update customer details using id.

```
var updateCustomer = (id, name, email) => {

var customers = fetchCustomers();
var customer = {id, name, email};

var filteredCustomers = customers.filter((customer) =>
{ // return all customers except the matched customer
return customer.id !== id;
});

if ( customers.length !== filteredCustomers.length ) {
filteredCustomers.push(customer);
saveCustomers(filteredCustomers);
return customer;
}
};
```

6. Implemented function to remove the customer details from the list using customer ID.

```
var removeCustomer = (id) => {

var customers = fetchCustomers(); // reusable func
var filteredCustomers = customers.filter((customer) =>
{ // will return all other customers other than "note
to be removed"
return customer.id !== id;
```

```
});
saveCustomers(filteredCustomers); //save new notes
array
return customers.length !== filteredCustomers.length
};
```

Task-2: Creating command line functions using express framework:

1. Imported fs for read write operations on a file.
   imported yargs for enabling the command line interface.
   imported customers.js script file to make use of CRUD API functions.

```
const fs =  require('fs');
const _ = require('lodash');
const yargs = require('yargs');

const customers = require('./customers.js');
```

2. Created command line options configuration for customer id, customer name, customer email address.

```
const idOptions = {
    describe: 'Customer ID',
    demand : true,
    alias : 'i'
}

const nameOptions = {
    describe: 'Customer Name',
    demand : true,
    alias : 'n'
}

const emailOptions = {
    describe: 'Customer Email',
    demand : true,
    alias : 'e'
}
```

3. Configured list of commands like add, list, read, update and remove commands.

```
const argv =  yargs

    .command('add','Add a new customer details',{
      id: idOptions,
      name: nameOptions,
      email: emailOptions
    })
    .command('list','List all customer details')
    .command( cmd: 'read', description: 'Get a customer details by ID', builder: {
      id: idOptions
    })
    .command( cmd: 'update', description: "Update customer details", builder: {
        id: idOptions,
        name:nameOptions,
        email: emailOptions
    })
    .command( cmd: 'remove', description: 'Remove customer by ID', builder: {
      id:idOptions
    })
    .help()
    .argv;
```

4. Implemented the add command execution which takes all 3 options and adds the customer details in JSON format.

```
var command = argv._[0];


if (command === 'add'){
    var customer = customers.addCustomer(argv.id, argv.name, argv.email);
    if (customer){
      customers.logCustomer(customer)  // print added Customer
    } else{
      console.log("Customer already exists");
    }
}
```

5. Implemented list command execution which outputs the list of all customer details.

```
else if (command === 'read') {
   var customer = customers.getCustomer(argv.id);
   if(customer){
    customers.logCustomer(customer);                              //print customer details
         }
   else{
    console.log("Customer not found");
   }
}
```

6. Implemented read, update and delete commands:
   read command takes input as ID and returns the details of the customer matching to the ID.
   update command takes input as ID, name, email and returns the details of the updated customer (based on the ID)
   remove command takes the input as ID and removed the customer details from the list of customers.

```
else if (command === 'read') {
    var customer = customers.getCustomer(argv.id);
    if(customer){
     customers.logCustomer(customer);                                    //print customer details
           }
    else{
     console.log("Customer not found");
    }
}

else if (command === 'update') {
    var customer = customers.updateCustomer(argv.id, argv.name, argv.email);
    if(customer){
        customers.logCustomer(customer);
    }
    else{
        console.log("Customer not found");
    }
}

else if (command === 'remove'){
    var customer = customers.removeCustomer(argv.id);
    if(customer){
        console.log("Removed the customer details");
    }
    else{
        console.log("Customer not found");
    }
}
```

7. Sample output on running the javascript files.

   command to execute: `node app.js`

```
F:\Downloads\CS5551_Express_Tutorial1\SourceCode\Note App>node app.js list
Printing 1 customer(s).
--
Customer ID: 1
Customer Name: sindhu
Customer Email address: sindhu1@gmail.com

F:\Downloads\CS5551_Express_Tutorial1\SourceCode\Note App>node app.js add -i 2 -n s2 -e s2@gmail.com
--
Customer ID: 2
Customer Name: s2
Customer Email address: s2@gmail.com

F:\Downloads\CS5551_Express_Tutorial1\SourceCode\Note App>node app.js add -i 2 -n s2 -e s2@gmail.com
Customer already exists

F:\Downloads\CS5551_Express_Tutorial1\SourceCode\Note App>node app.js read -i 1
--
Customer ID: 1
Customer Name: sindhu
Customer Email address: sindhu1@gmail.com

F:\Downloads\CS5551_Express_Tutorial1\SourceCode\Note App>node app.js update -i 2 -n s3 -e s3@g.com
--
```

```
F:\Downloads\CS5551_Express_Tutorial1\SourceCode\Note App>node app.js update -i 2 -n s3 -e s3@g.com
--
Customer ID: 2
Customer Name: s3
Customer Email address: s3@g.com

F:\Downloads\CS5551_Express_Tutorial1\SourceCode\Note App>node app.js update -i 3
app.js update

Options:
  --help       Show help                                           [boolean]
  --id, -i     Customer ID                                        [required]
  --name, -n   Customer Name                                      [required]
  --email, -e  Customer Email                                     [required]

Missing required arguments: name, email

F:\Downloads\CS5551_Express_Tutorial1\SourceCode\Note App>node app.js remove -i 2
Removed the customer details
```

Task Responsibilities:

1. Sindhusha Tiyyagura: Created the basic CRUD API functions (customers.js) javascript file.
2. Pradeepika kolluru: Created interactive command line js script using express framework.