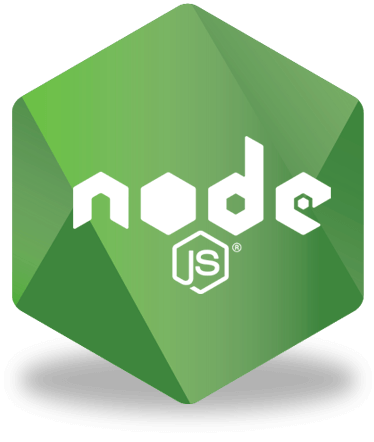


ASE\_Lab6 Report

Implementing CRUD APIusing Nodejs and Express.js







March 09, 2019

Documented by:

Dharani Muli (Class ID : 18),

Chakra Pavan Kumar (Class ID: 13)



Introduction

Objectives

Implement customer CRUD API using nodejs and expressjs

Design/Implementation

We have followed below steps to successfully complete this lab assignment:

**Step -1:** We make sure all the pre-requisites are ready before start of the project and below are the technologies/languages used:

1. WebStorm IDE
2. Node
3. Npm
4. Expressjs

**Step-2:** We analyzed already existing source code and done changes on top of that.

**Step-3:** Using below command installed and imported lodash package, yarg and fs into app.js file

1. Command: $npm i - - save lodash

Importing: **const** *\_* = require(**'lodash'**);

1. Command: $npm i - - save yarg

Imported: **const** *yargs* = require(**'yargs'**);

1. Import file system module to make use of utilities functions under that

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

**Step-4:** Created app.js, customers.js and customer.json files and using below require method we can access methods of customers.js to app.js file.

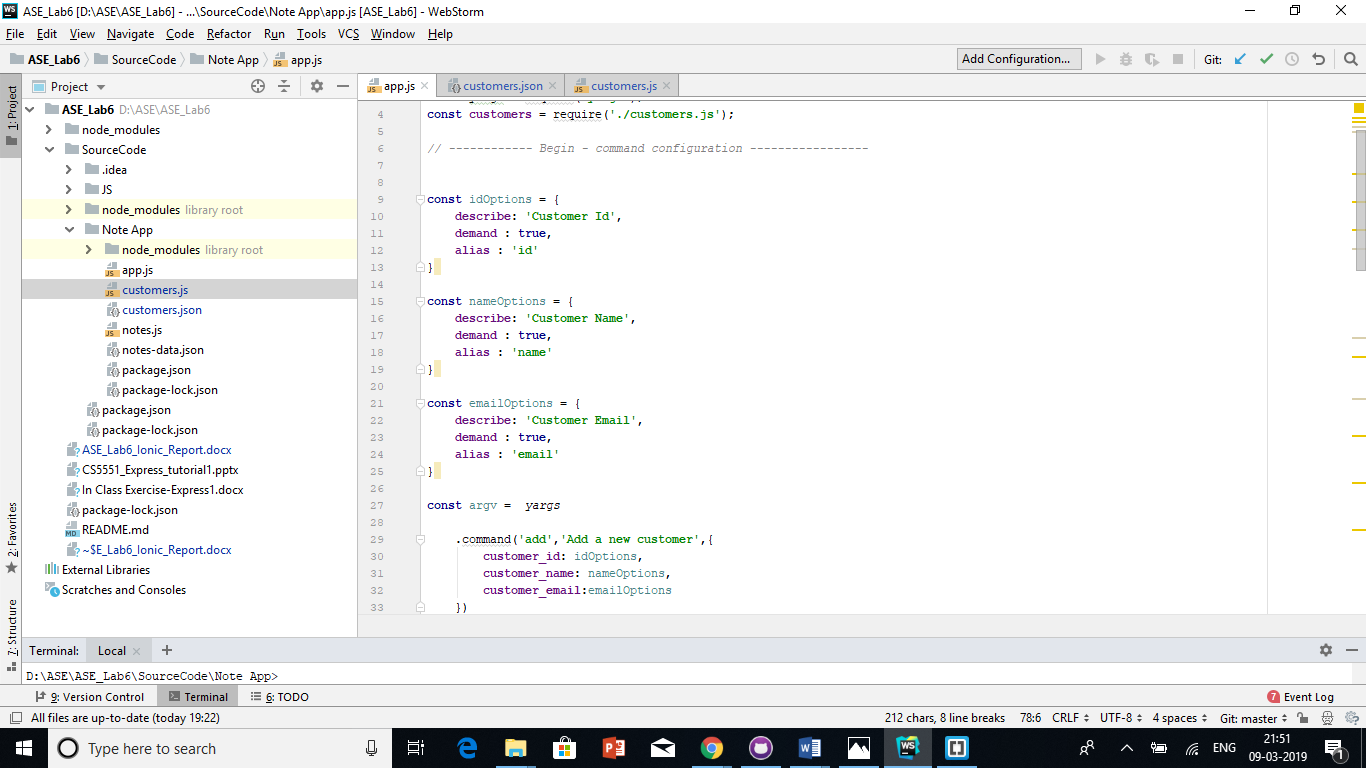
**const customers** = require(**'./customers.js'**);

should exports all the methods to app.js in order to access customer.js mentods in app.js file

module.**exports** = {  
 *addCustomer*, *getAll*, *remove*, *getCustomer*,*logCustomer*, *update*};

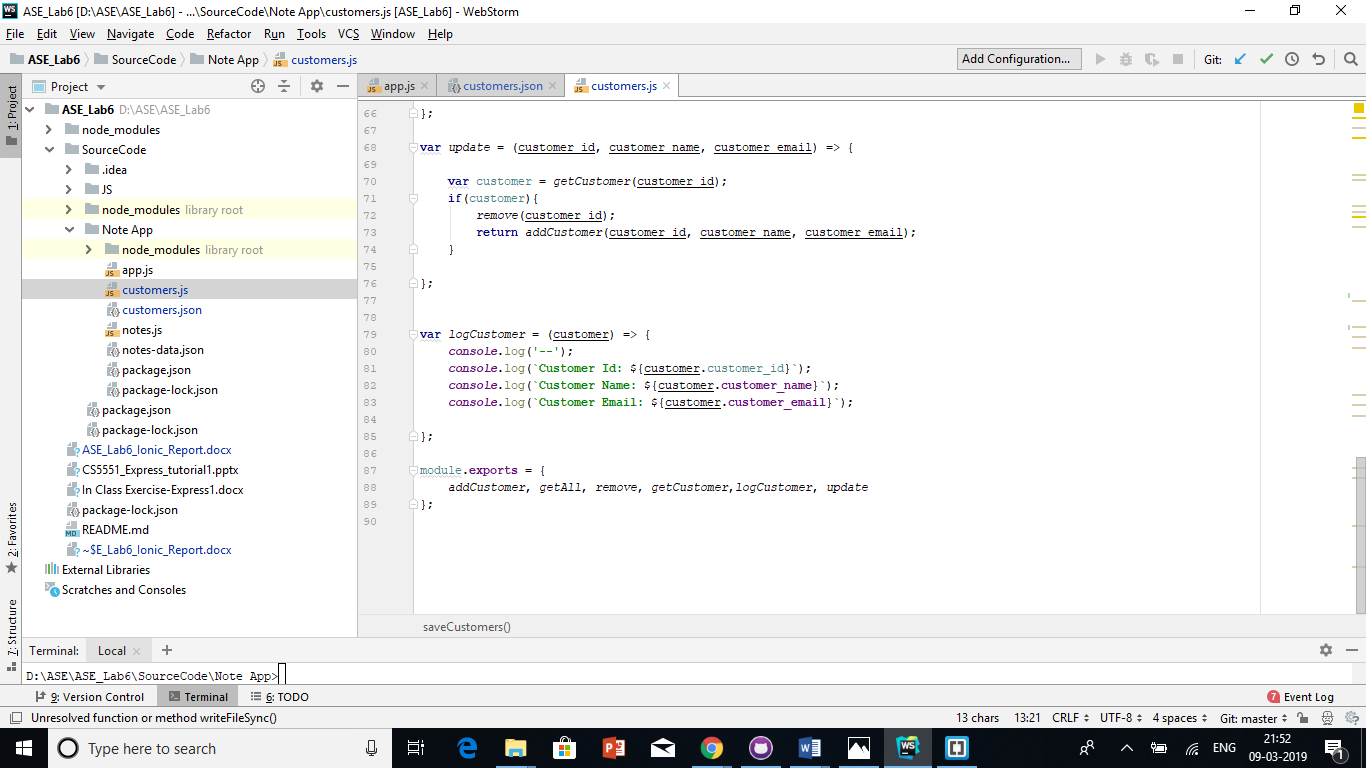
**Step-5:** Created command line options to configure customer properties

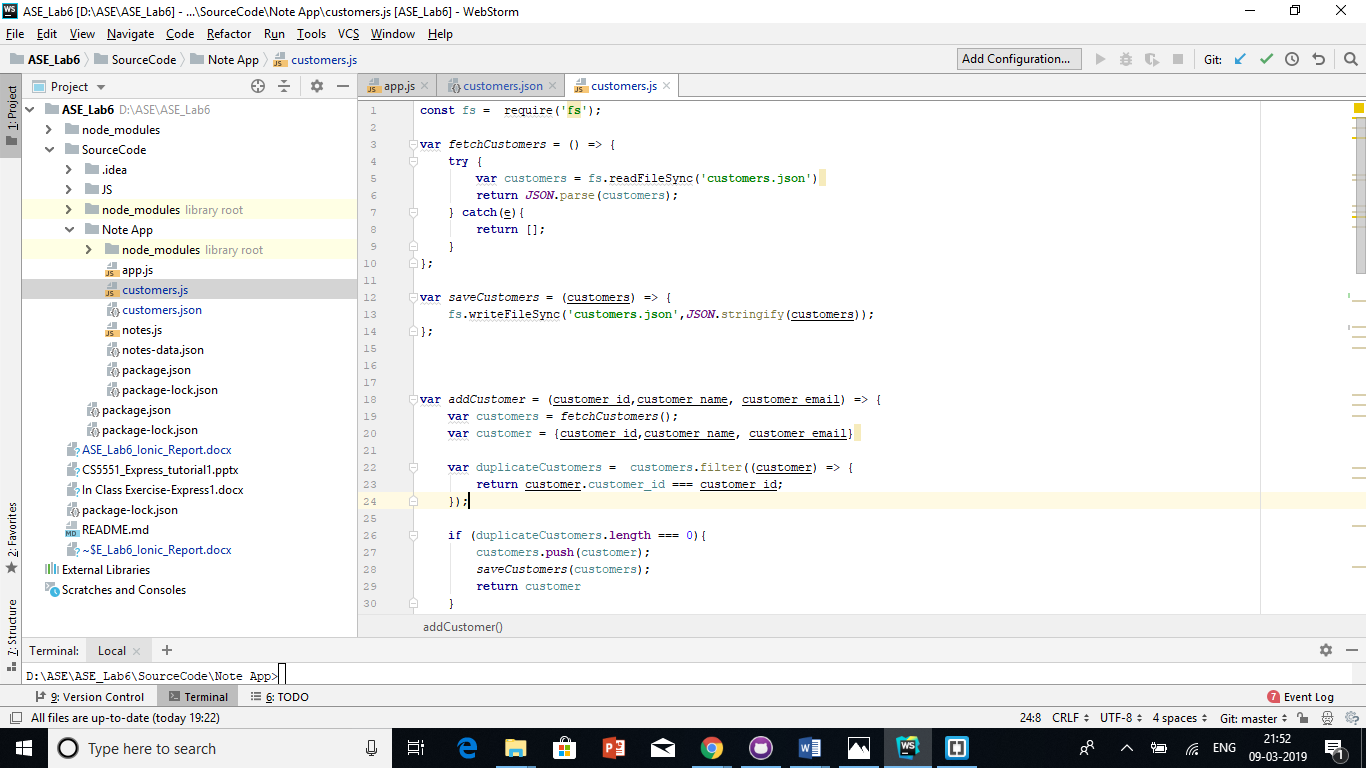
app.js



**Step-6:** Created “saveCustomers” which is used to save customers details into JSON format and by converting into string format and also created “logCustomer” method to show the output on the command line. We have used these 2 methods in multiple places to save and log customer details

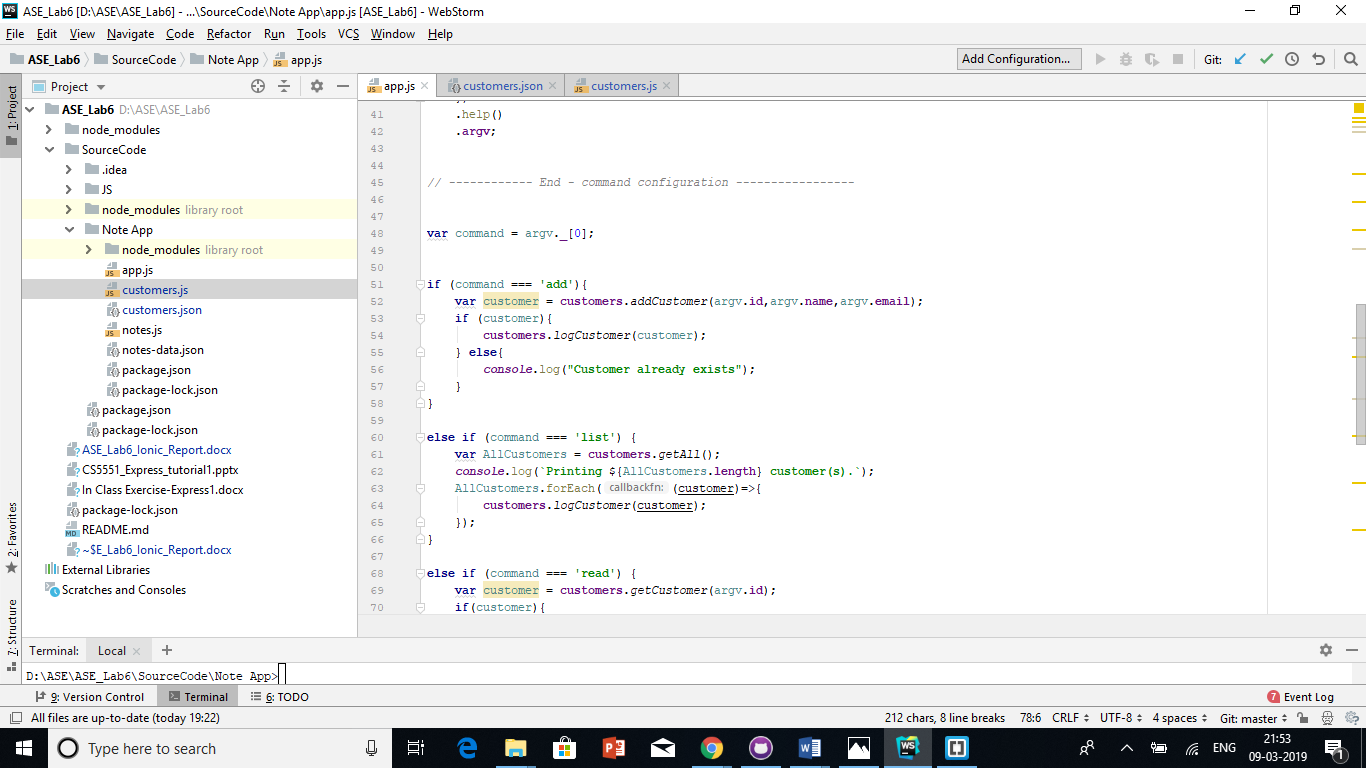
Customer.js



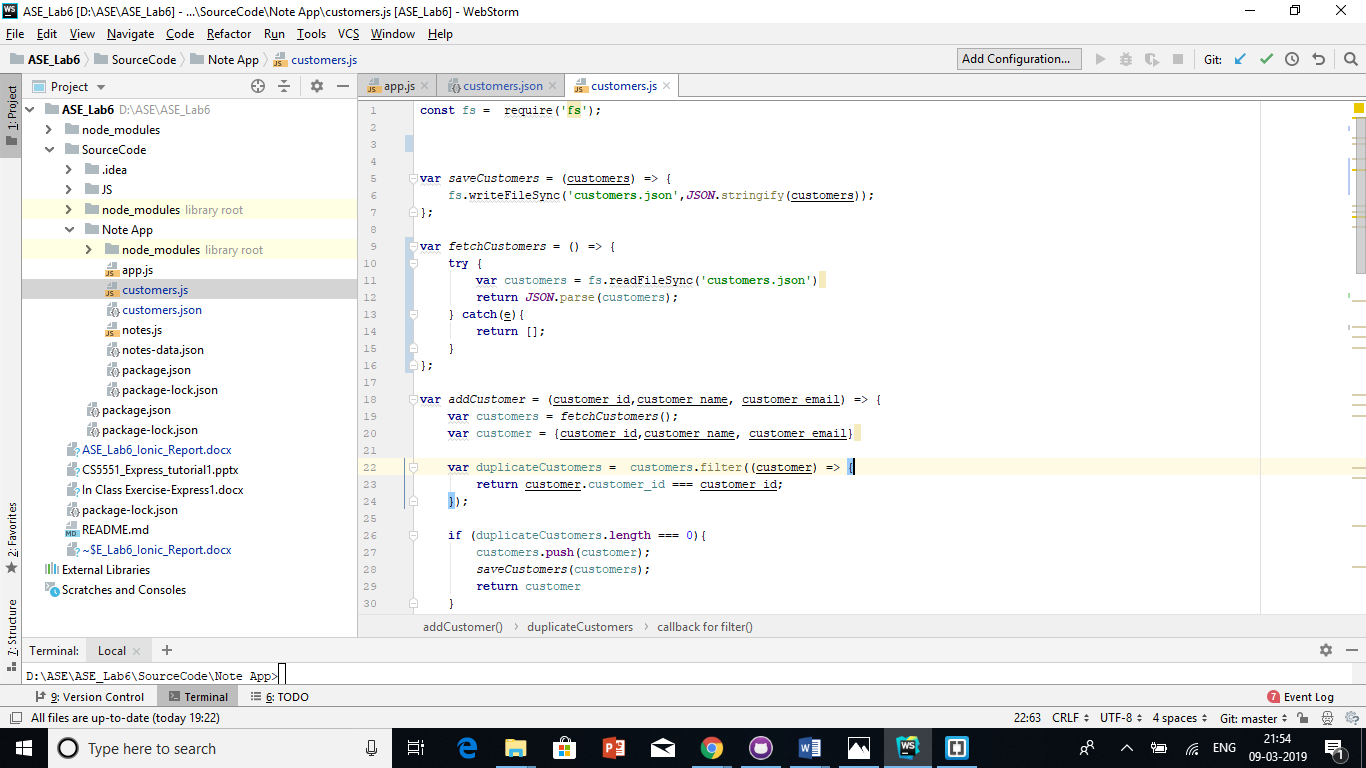


**Step-7:** Configured “add” command and Created “addCustomer” function to add customer with properties id, name and email. Before adding we are doing duplicate check if customer is already existing or not by fetching already existing customers using “fetchCustomer” method.

app.js

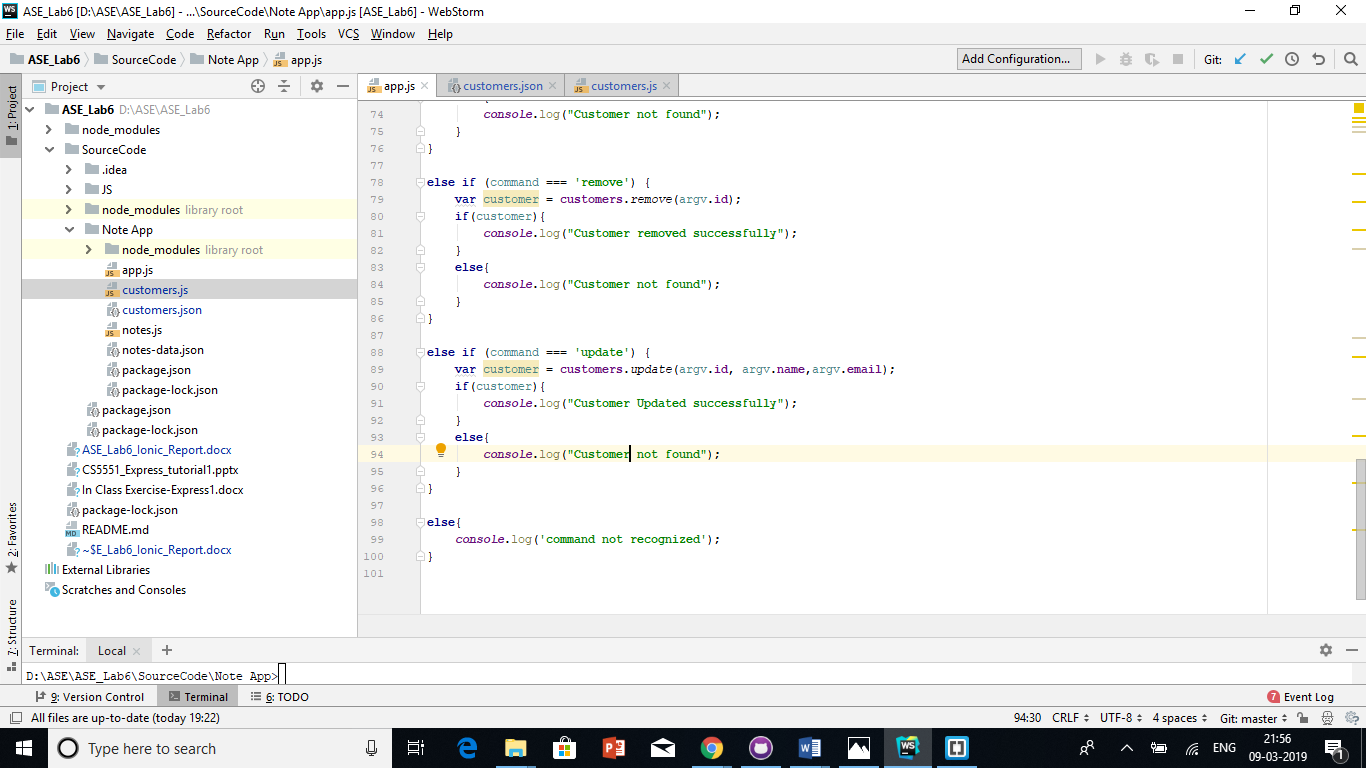


customer.js

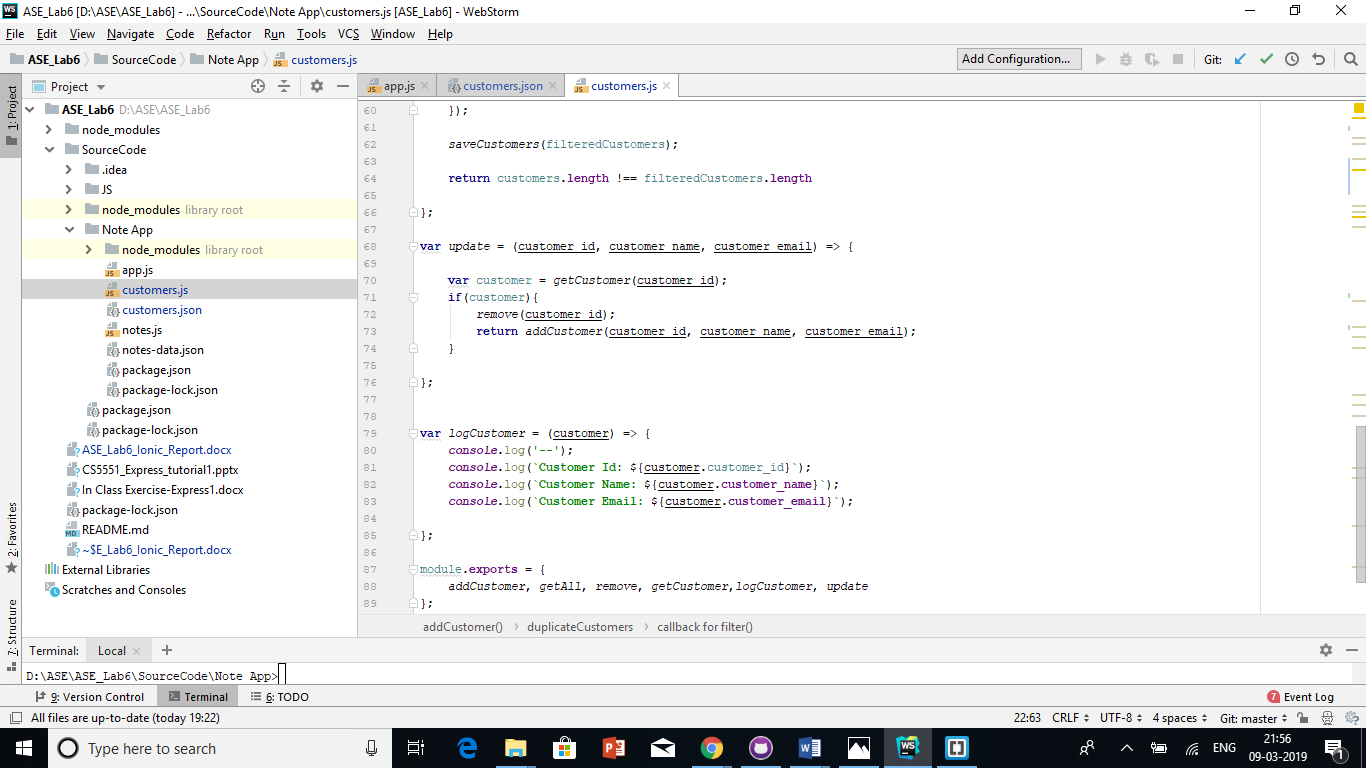


**Step-8:** Similarly, configured “update” command and created Update method to update already existing customer details

App.js

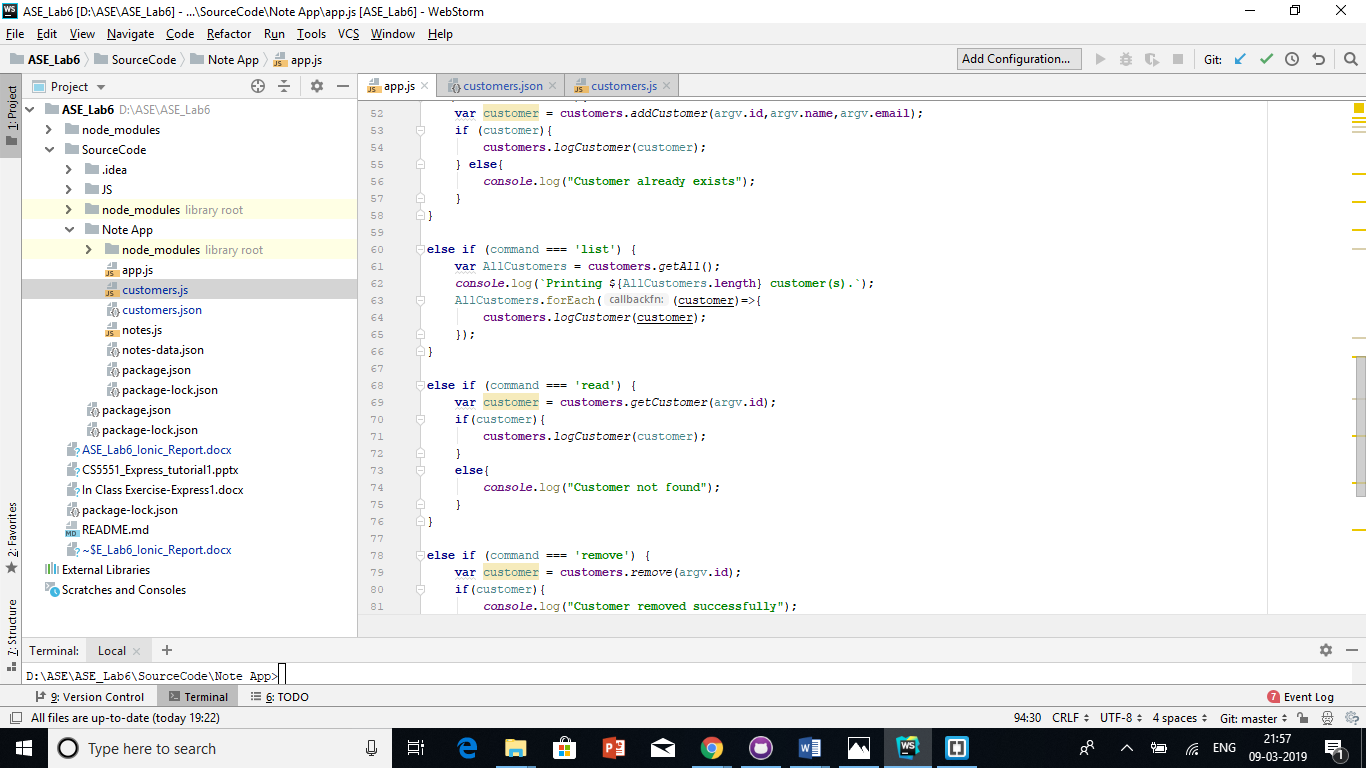


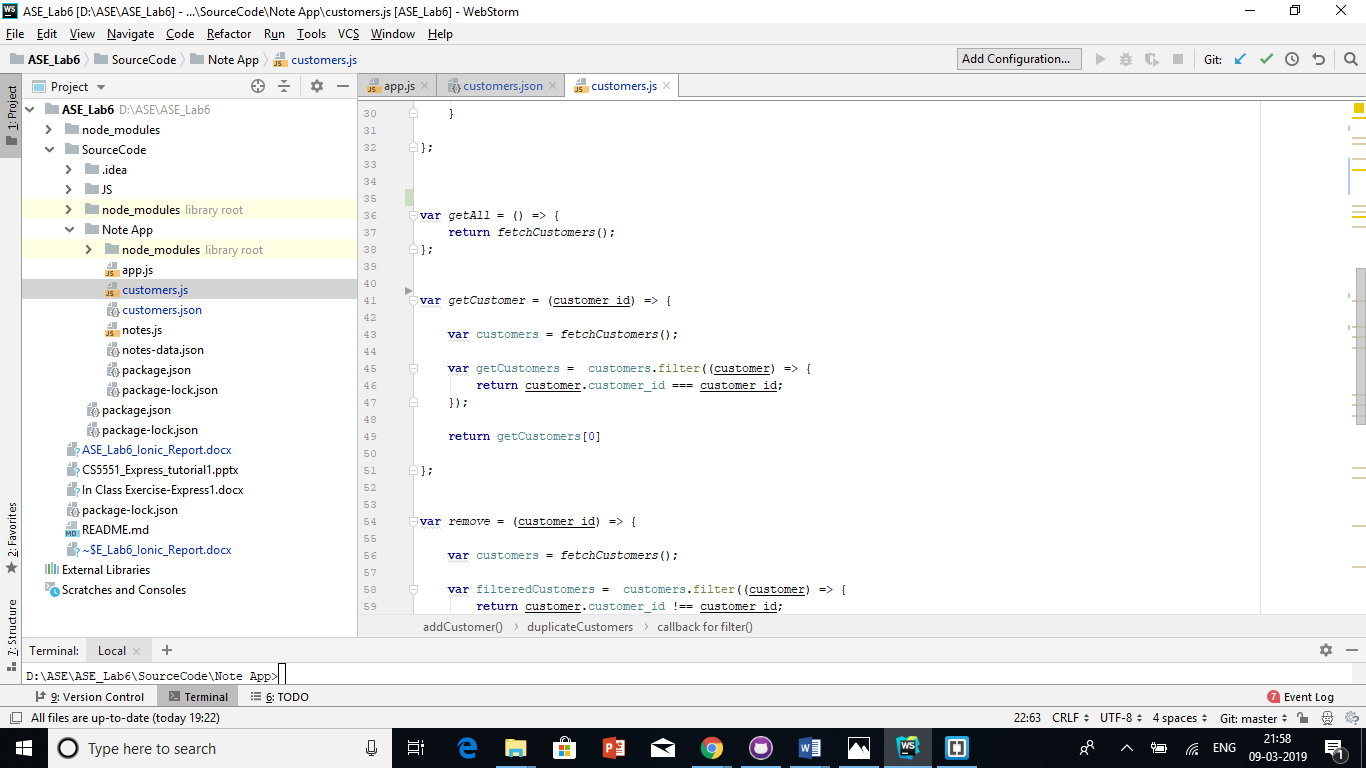
customer.js



**Step-9:** Implemented “list” command configuration and created function to read and get the single or list of customers

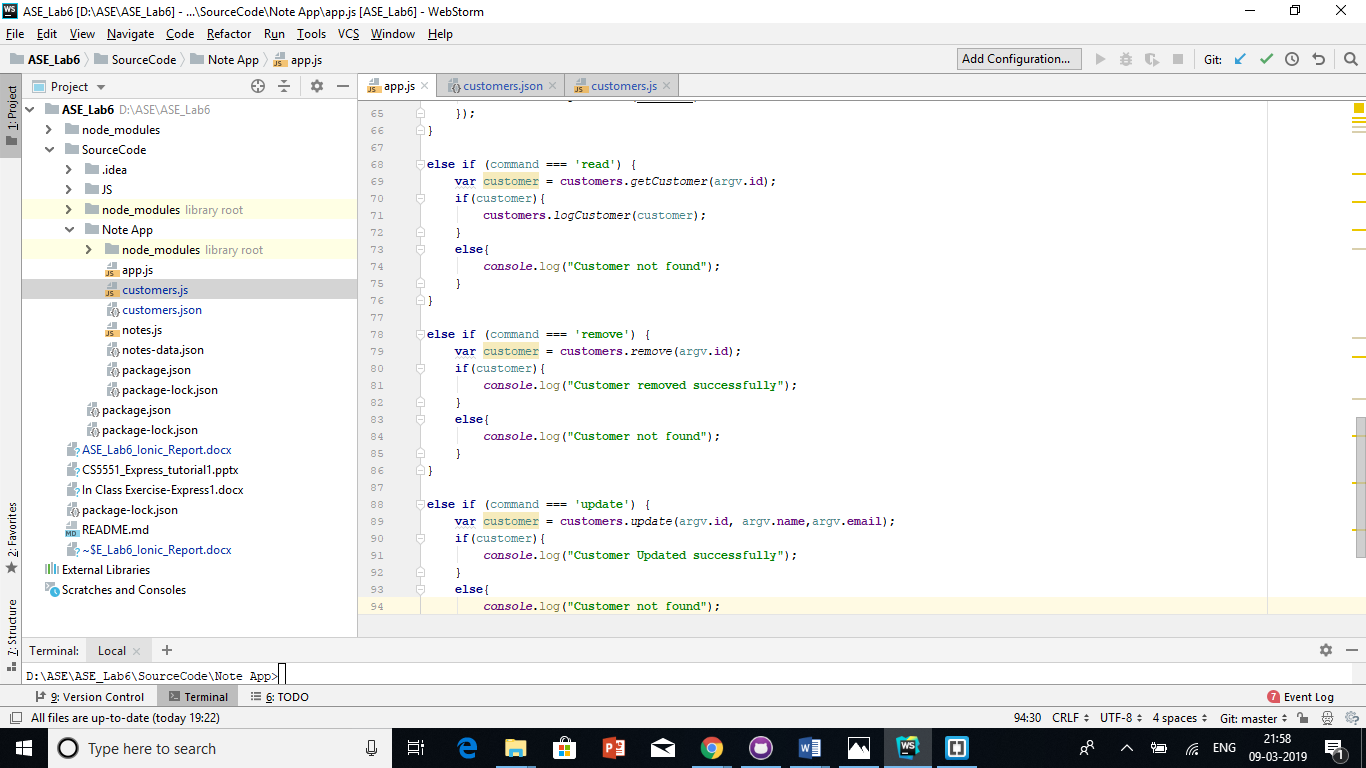
app.js



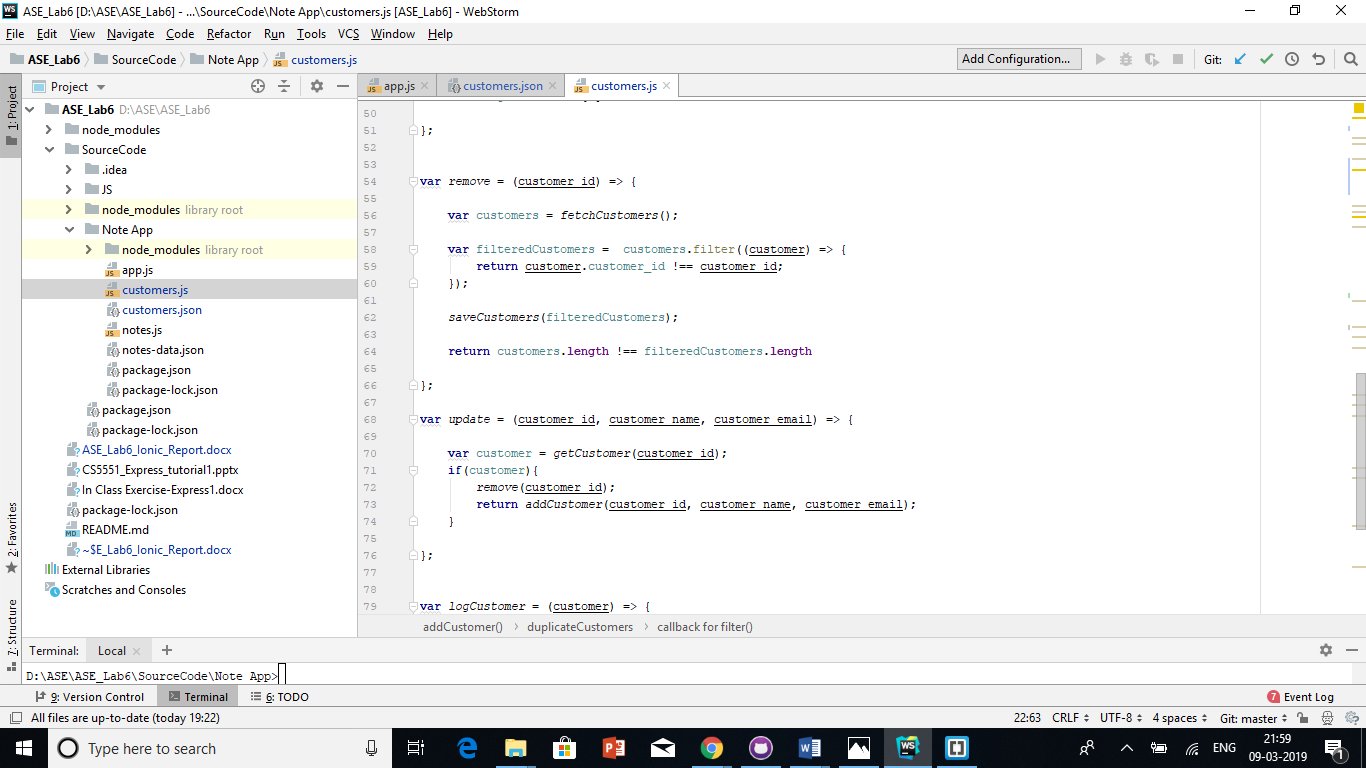
*customer.js*  


**Step-10:** Below methods are used to configured “remove” command and remove customer details from the list of the customers

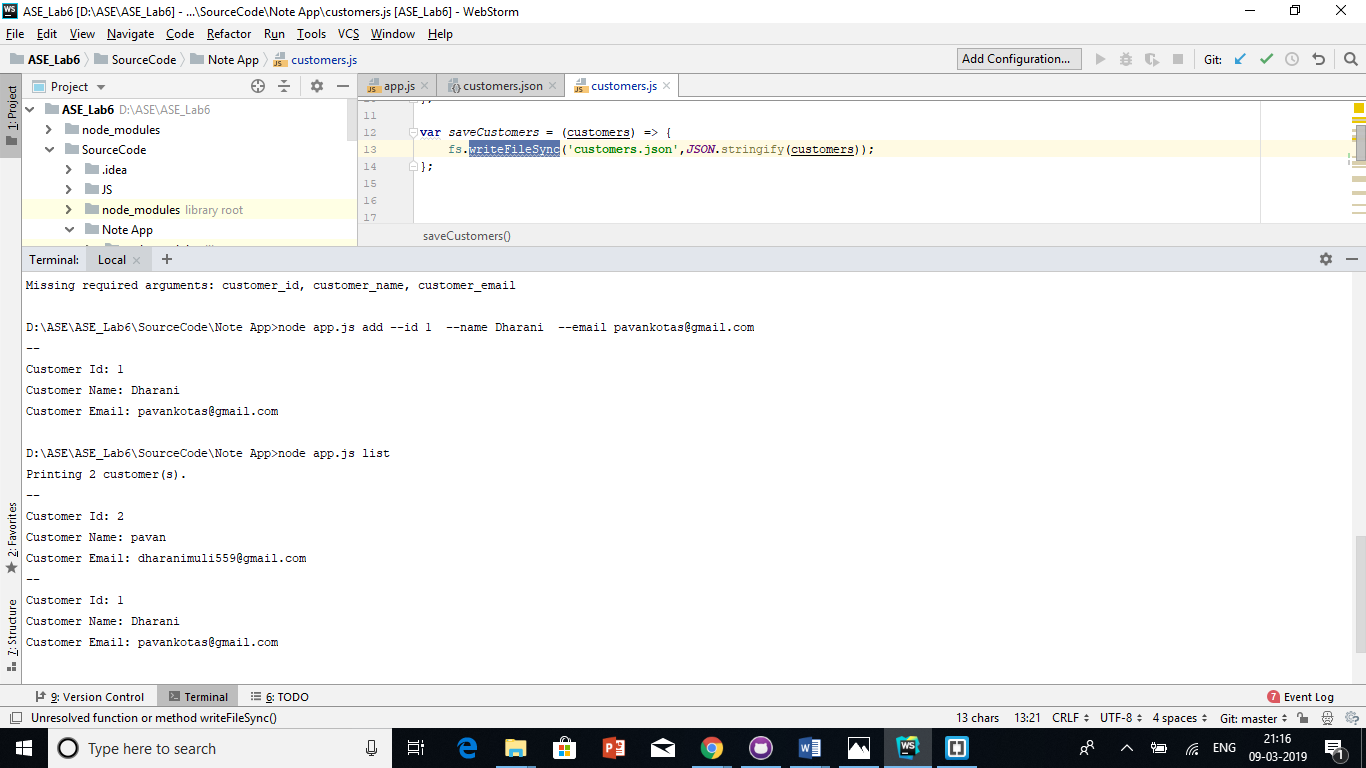
*app.js*



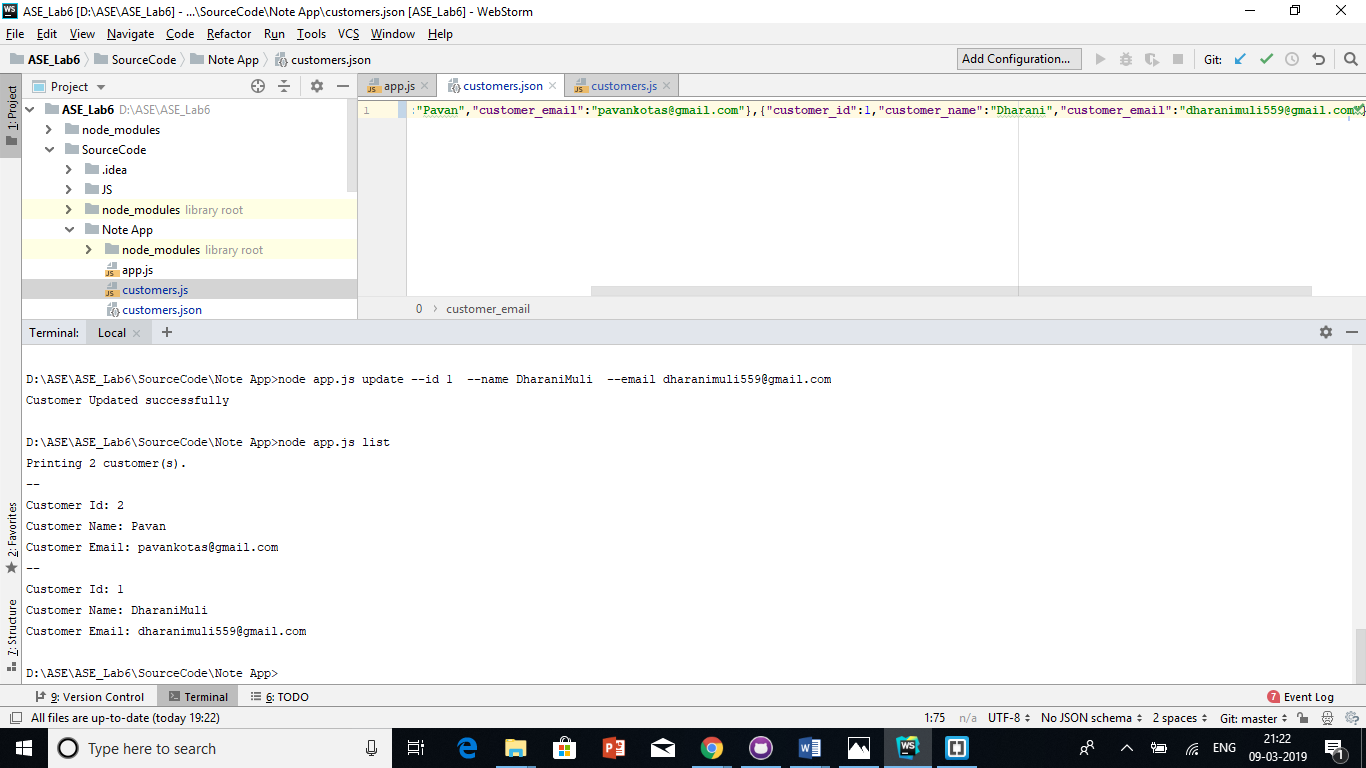
*customer.js*



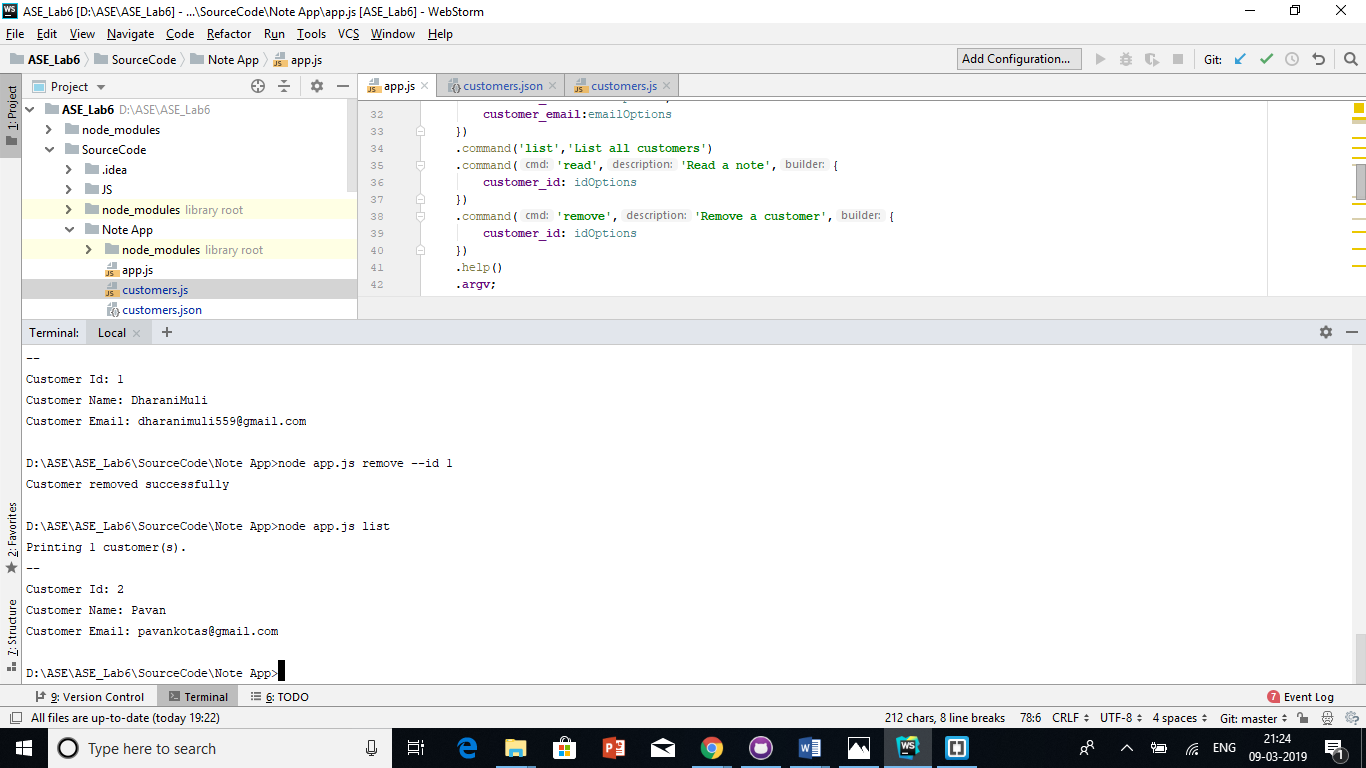
**Sample output**

1. Adding and listing of Customers

2. Updating customer



3. Removing Customer



Issues/Limitations:

None

Team Contribution

Chakra Pavan Kumar Kota: Implemented “Add” and “Update” functionalities and also created interactive commands for the same.

Dharani Muli: Implemented “List” and “Remove” functionalities and also created interactive commands for the same.

Conclusion:

With this exercise we are able to understand the way CRUD operations can be performed in Node.js. We are able to analyse the previous source code provided and reapply the skills while implementing the CRUD repository for Customers. This javascript based node.js approach to access files, update them seemed to be very exciting which is otherwise impossible in plain JavaScript world. This also helps us in incorporating the skills in the final project whose backend is Node.js.