Identified Use cases:

1. Customers can chat with gift-shop seller directly and can discuss about basic products details and if customization, it can be easily worked.
2. Shopper can search for available products and he/she can also add and new products with discount availability till valid dates and delete old product.

Actors:

1. **Redis Chat** – Users & Seller
2. **Search Product & Add discount on products -** Seller

Detailed Description:

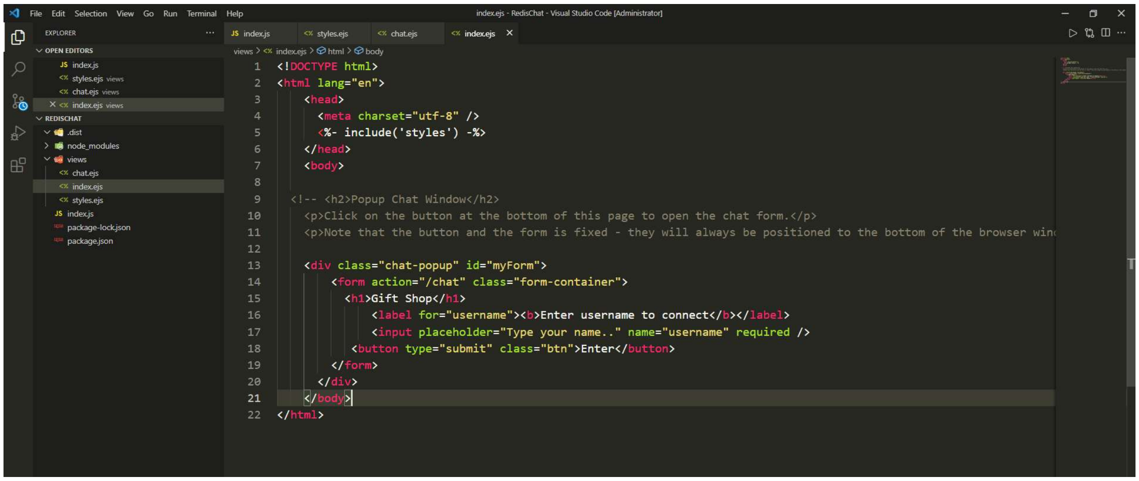
1. **Redis Chat**: A Customer needs to talk directly Seller about product he wants some customization so easily interact by real-time chat. A customers needs to enter his/her username and when seller enter by his/her username. Customer also notified that gift seller is connected and chat as real time. Both can interact so easily and discussed.

2. **Search Product & Add discount on products**: A seller can search by product id and identified about product details (Discount percentage & Valid discount till date). Seller can add new product and enter as per discount details (product name, Percentage of discount & availability for discount). Apart from that, seller can delete any product to manage status for product.

Frontend used / Command lines to reproduce execution:

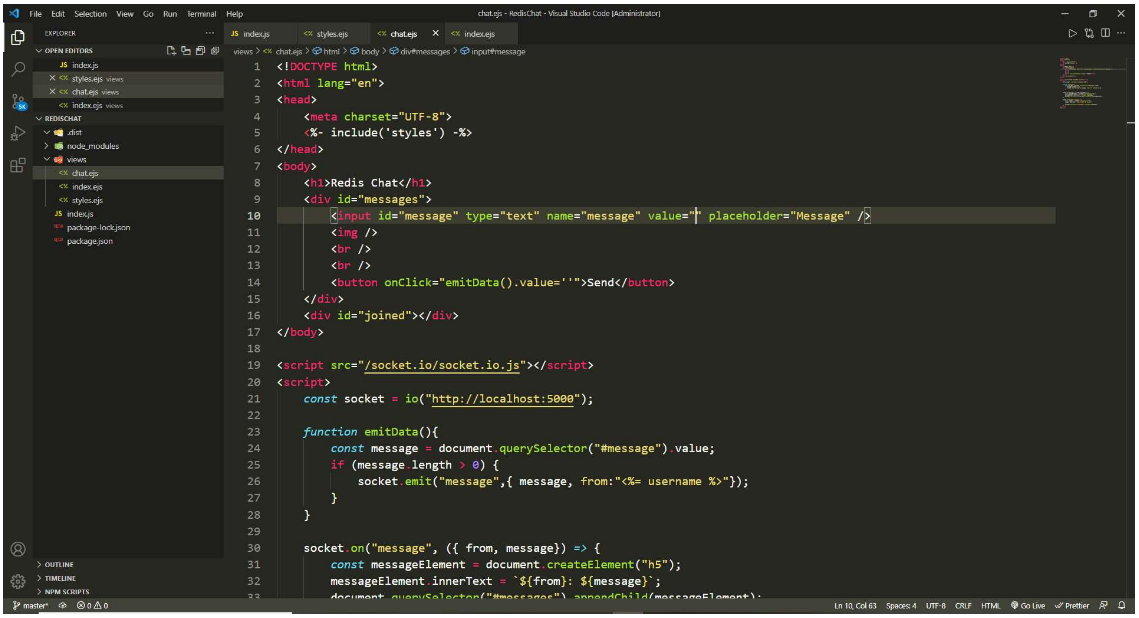
Redis Chat

- User Enter to chat by username



```
<!DOCTYPE html>
<html lang="en">
    <head>
      <meta charset="utf-8" />
      <%- include('styles') -%>
    </head>
    <body>

  <!-- <h2>Popup Chat Window</h2>
    <p>Click on the button at the bottom of this page to open the chat form.</p>
    <p>Note that the button and the form is fixed - they will always be positioned to the bottom of the browser win

    <div class="chat-popup" id="myForm">
        <form action="/chat" class="form-container">
          <h1>Gift Shop</h1>
              <label for="username"><b>Enter username to connect</b></label>
              <input placeholder="Type your name.." name="username" required />
           <button type="submit" class="btn">Enter</button>
        </form>
      </div>
    </body>
</html>
```

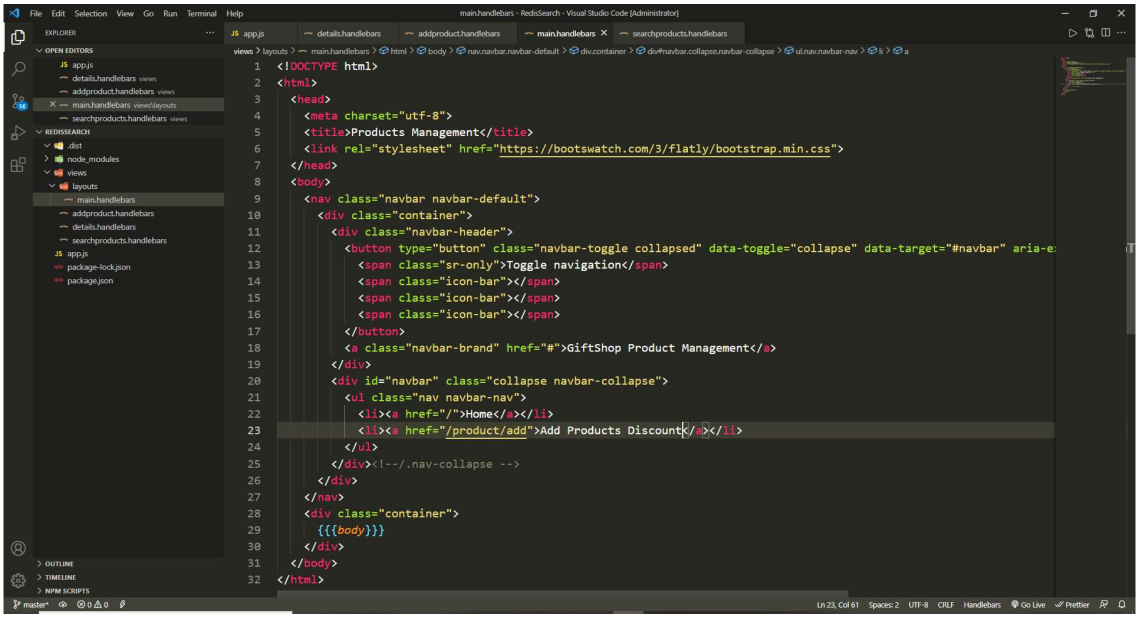
- Notified by gift-shop enters & Real time Chat



```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <%- include('styles') -%>
</head>
<body>
    <h1>Redis Chat</h1>
    <div id="messages">
        <input id="message" type="text" name="message" value="" placeholder="Message" />
        <img />
        <br />
        <br />
        <button onClick="emitData().value=''">Send</button>
    </div>
    <div id="joined"></div>
</body>

<script src="/socket.io/socket.io.js"></script>
<script>
    const socket = io("http://localhost:5000");

    function emitData(){
        const message = document.querySelector("#message").value;
        if (message.length > 0) {
            socket.emit("message",{ message, from:"<%= username %>"});
        }
    }

    socket.on("message", ({ from, message}) => {
        const messageElement = document.createElement("h5");
        messageElement.innerText = `${from}: ${message}`;
        document.querySelector("#messages").appendChild(messageElement);
```

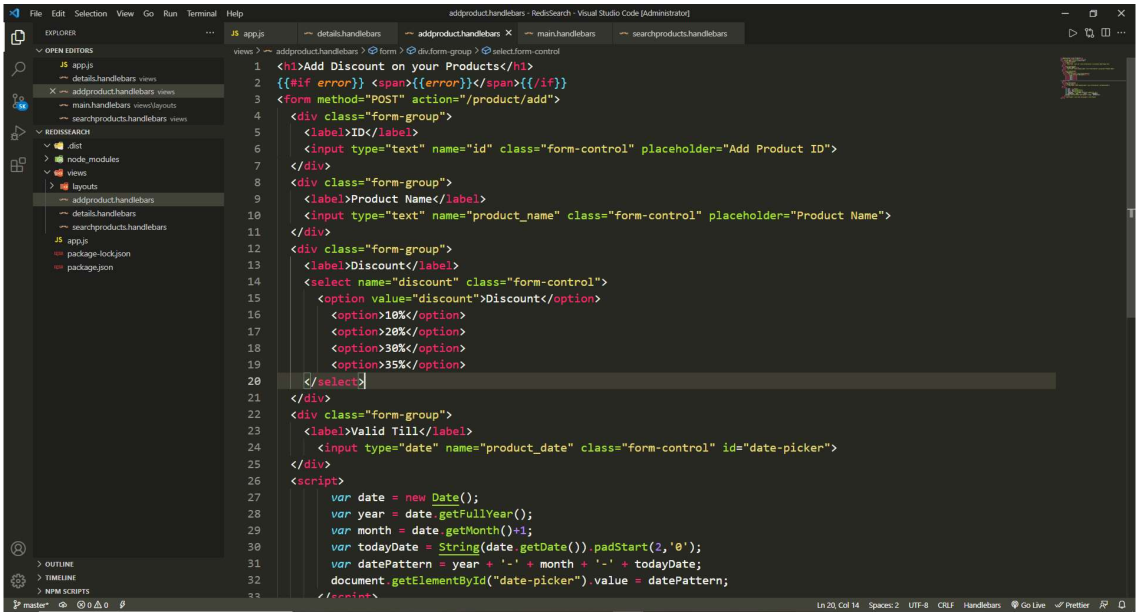
Search Product & Add discount on products:

- ❖ Home Page :



```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <title>Products Management</title>
    <link rel="stylesheet" href="https://bootswatch.com/3/flatly/bootstrap.min.css">
  </head>
  <body>
    <nav class="navbar navbar-default">
      <div class="container">
        <div class="navbar-header">
          <button type="button" class="navbar-toggle collapsed" data-toggle="collapse" data-target="#navbar" aria-e
            <span class="sr-only">Toggle navigation</span>
            <span class="icon-bar"></span>
            <span class="icon-bar"></span>
            <span class="icon-bar"></span>
          </button>
          <a class="navbar-brand" href="#">GiftShop Product Management</a>
        </div>
        <div id="navbar" class="collapse navbar-collapse">
          <ul class="nav navbar-nav">
            <li><a href="/">Home</a></li>
            <li><a href="/product/add">Add Products Discount</a></li>
          </ul>
        </div><!--/.nav-collapse -->
      </div>
    </nav>
    <div class="container">
      {{{body}}}
    </div>
  </body>
</html>
```

- ❖ Add discount Page.



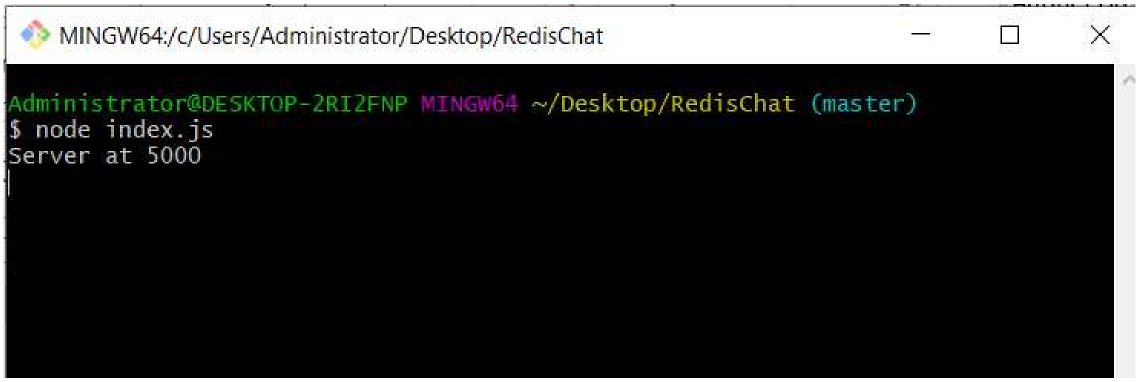
```
<h1>Add Discount on your Products</h1>
{{#if error}} <span>{{error}}</span>{{/if}}
<form method="POST" action="/product/add">
  <div class="form-group">
    <label>ID</label>
    <input type="text" name="id" class="form-control" placeholder="Add Product ID">
  </div>
  <div class="form-group">
    <label>Product Name</label>
    <input type="text" name="product_name" class="form-control" placeholder="Product Name">
  </div>
  <div class="form-group">
    <label>Discount</label>
    <select name="discount" class="form-control">
      <option value="discount">Discount</option>
        <option>10%</option>
        <option>20%</option>
        <option>30%</option>
        <option>35%</option>
    </select>
  </div>
  <div class="form-group">
    <label>Valid Till</label>
      <input type="date" name="product_date" class="form-control" id="date-picker">
  </div>
  <script>
        var date = new Date();
        var year = date.getFullYear();
        var month = date.getMonth()+1;
        var todayDate = String(date.getDate()).padStart(2,'0');
        var datePattern = year + '-' + month + '-' + todayDate;
        document.getElementById("date-picker").value = datePattern;
      </script>
```

Data Flow

Redis Chat:

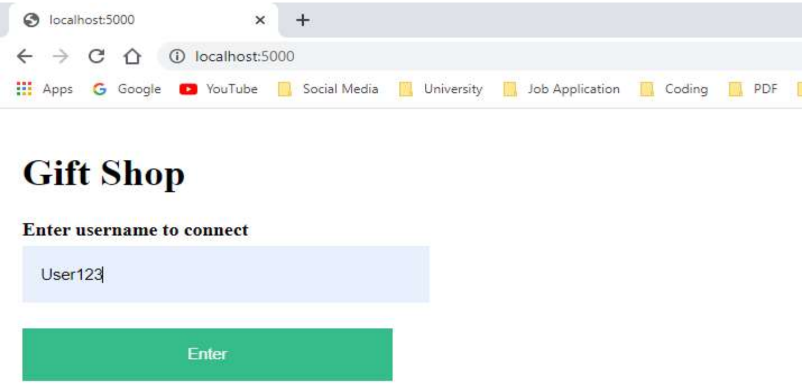
- ❖ Type code in Gitbash. (Make sure that Redis Server and Redis CLI is opened before type this code)

node index.js and will indicate to run code by localhost 5000



```
Administrator@DESKTOP-2RI2FNP MINGW64 ~/Desktop/RedisChat (master)
$ node index.js
Server at 5000
```

- ❖ User will enter by his/her username.

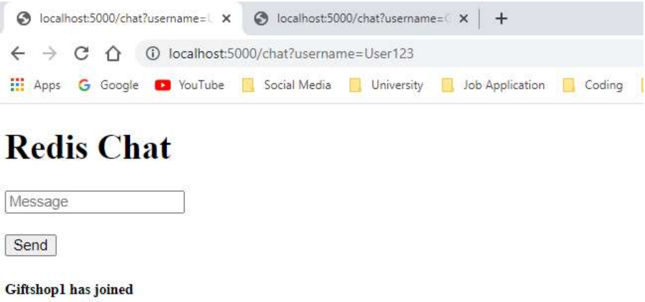


Gift Shop

Enter username to connect

User123

- ❖ User will be notified when Gift-shop seller joined to chat.

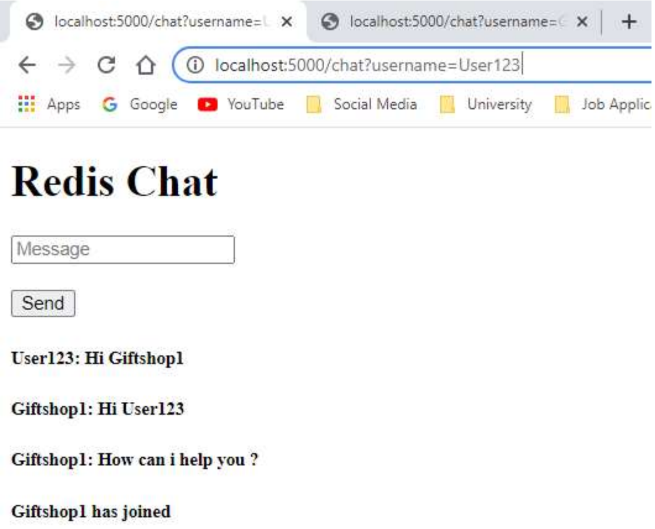

localhost:5000/chat?username=
localhost:5000/chat?username=
localhost:5000/chat?username=User123
Apps
Google
YouTube
Social Media
University
Job Application
Coding
Message
Send

Redis Chat

Giftshop1 has joined

❖ Real-time Chat messages between User and Seller

User123 Chat


localhost:5000/chat?username=
localhost:5000/chat?username=
localhost:5000/chat?username=User123
Apps
Google
YouTube
Social Media
University
Job Applic
Message
Send

Redis Chat

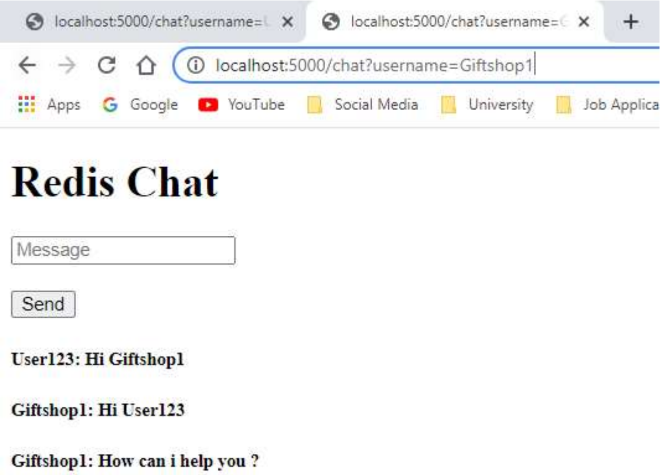
User123: Hi Giftshop1

Giftshop1: Hi User123

Giftshop1: How can i help you ?

Giftshop1 has joined

Giftshop1 Chat


localhost:5000/chat?username=
localhost:5000/chat?username=
localhost:5000/chat?username=Giftshop1
Apps
Google
YouTube
Social Media
University
Job Applica
Message
Send

Redis Chat

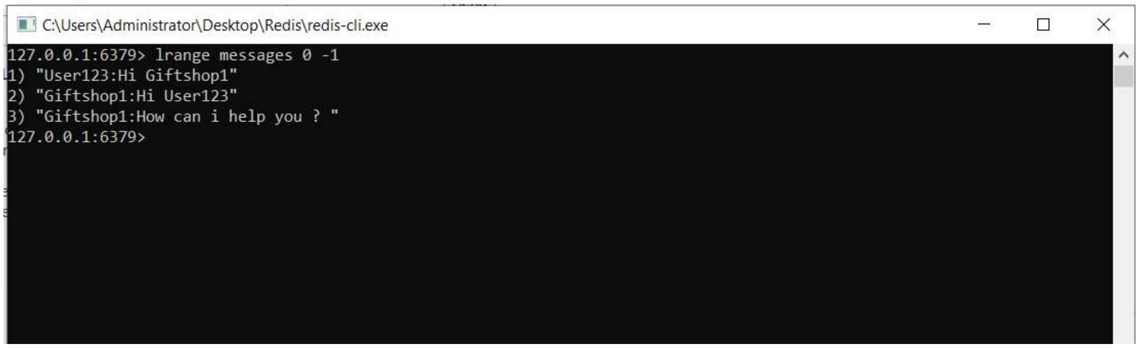
User123: Hi Giftshop1

Giftshop1: Hi User123

Giftshop1: How can i help you ?

❖ Redis Database storing and showing by Redis cli

Command: lrange messages 0 -1

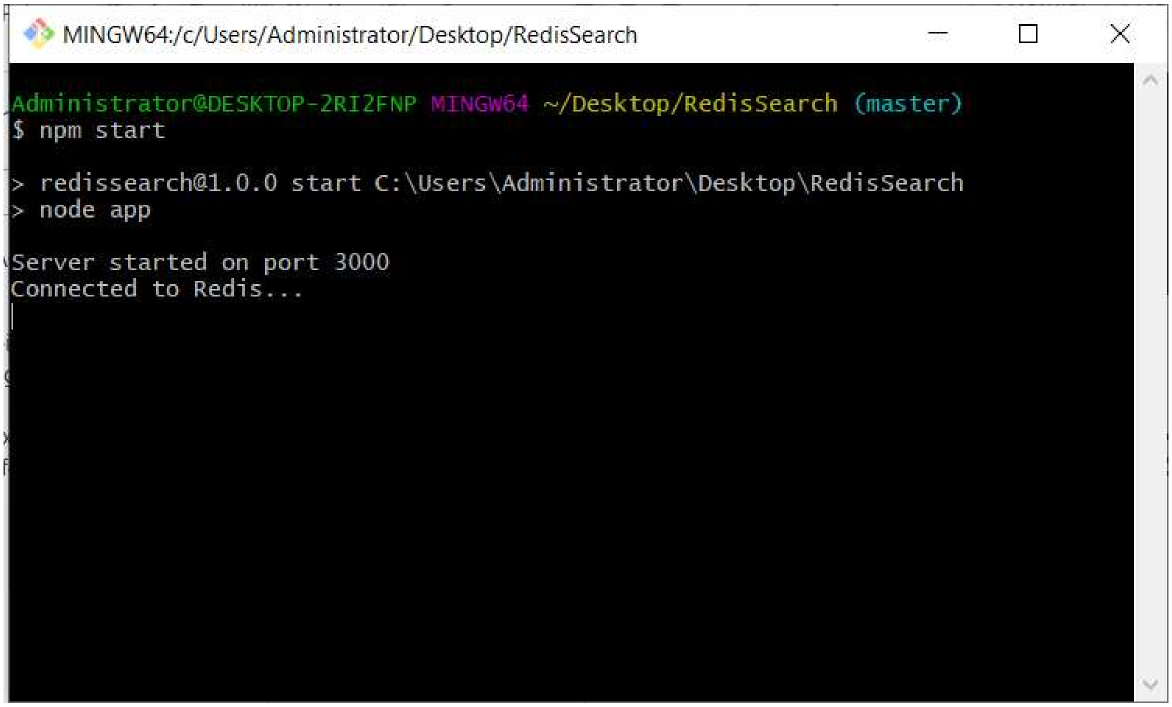


```
127.0.0.1:6379> lrange messages 0 -1
1) "User123:Hi Giftshop1"
2) "Giftshop1:Hi User123"
3) "Giftshop1:How can i help you ? "
127.0.0.1:6379>
```

Search Product & Add discount on products

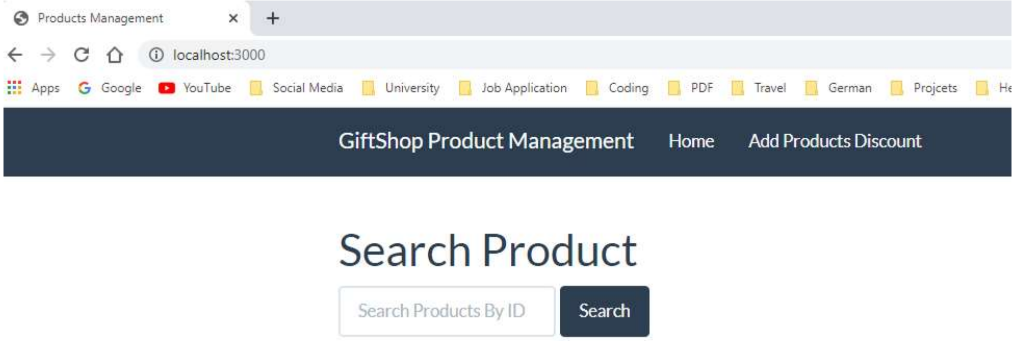
- Type code in Gitbash. (Make sure that Redis Server and Redis CLI is opened before type this code)

npm start and will indicate to run code by localhost 3000



```
Administrator@DESKTOP-2RI2FNP MINGW64 ~/Desktop/RedisSearch (master)
$ npm start

> redissearch@1.0.0 start C:\Users\Administrator\Desktop\RedisSearch
> node app

Server started on port 3000
Connected to Redis...
```

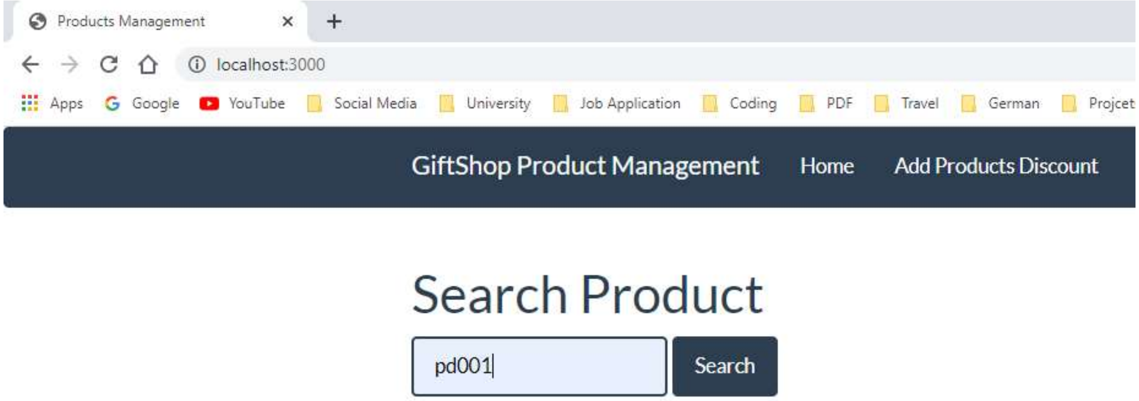
- Home Page



Search Product

Search Products By ID

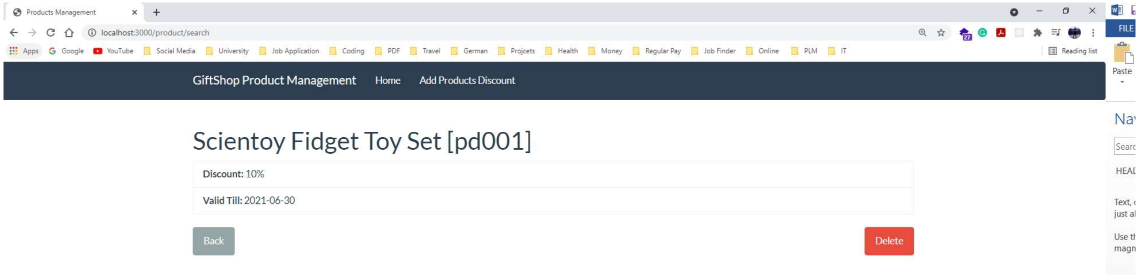
- ❖ Using Search Functionality (Type by product ID)



Search Product

pd001

- ❖ Result (Showing details of Product Name, Discount and Valid date)

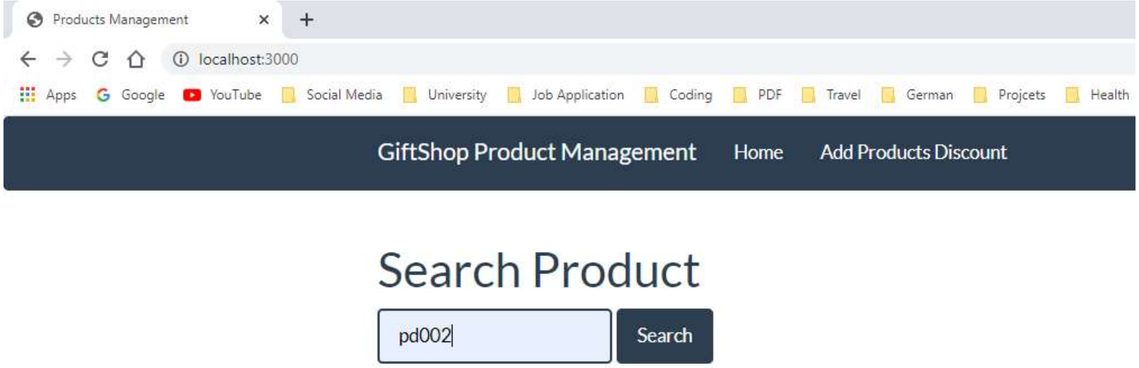


Scientoy Fidget Toy Set [pd001]

Discount: 10%

Valid Till: 2021-06-30

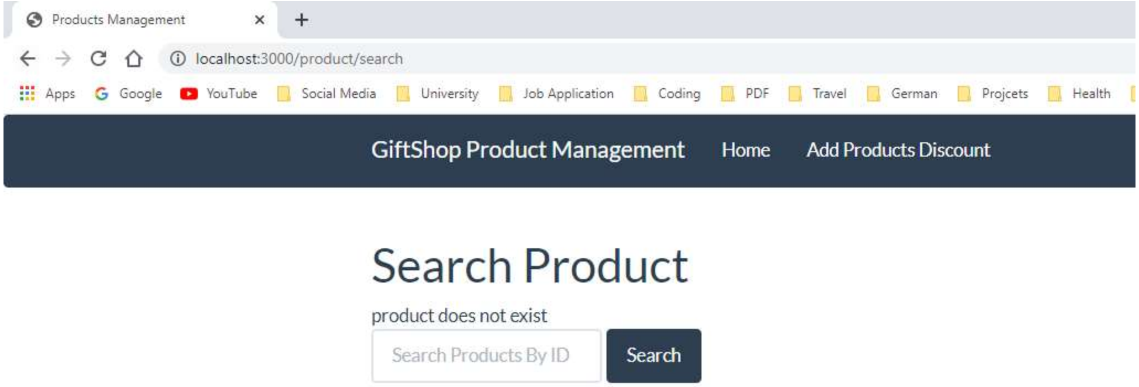
- ❖ Usage of Searching functionality (I mentioned other product id which is not available)



Search Product

pd002

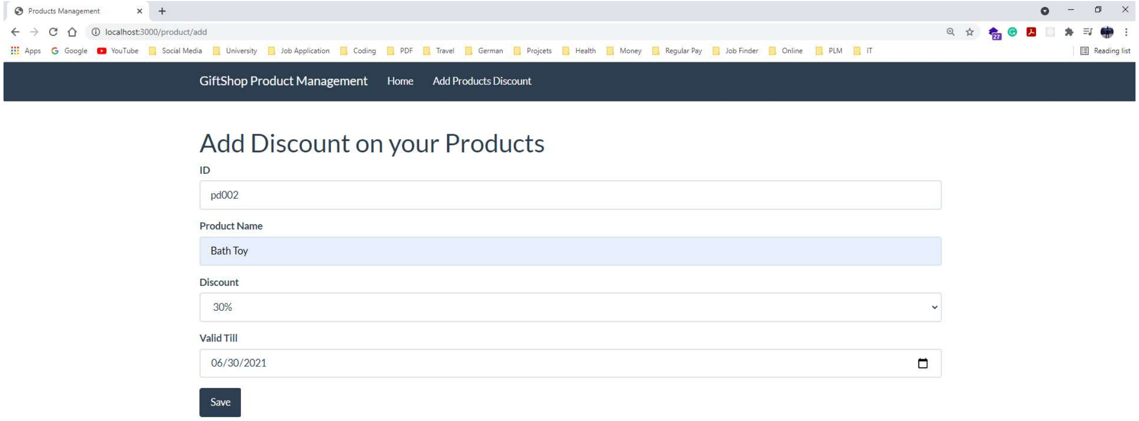
- ❖ Result (Product does not exist)



Search Product

product does not exist

- ❖ Now Using Add discount on products:



Add Discount on your Products

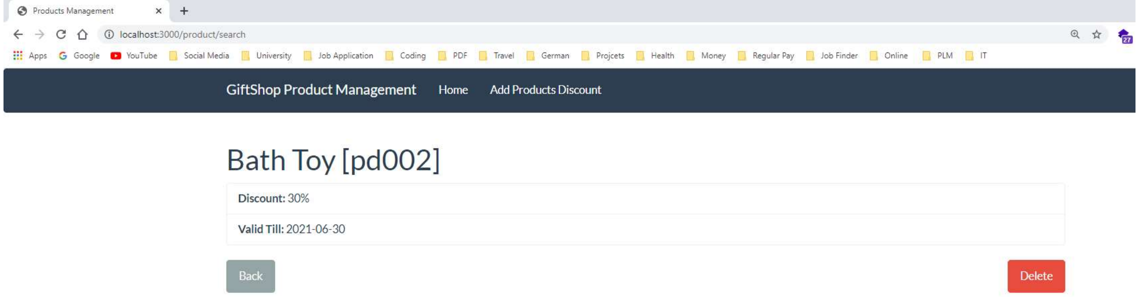
ID: pd002

Product Name: Bath Toy

Discount: 30%

Valid Till: 06/30/2021

- After addition outcomes : (pd002 ID is available with details)

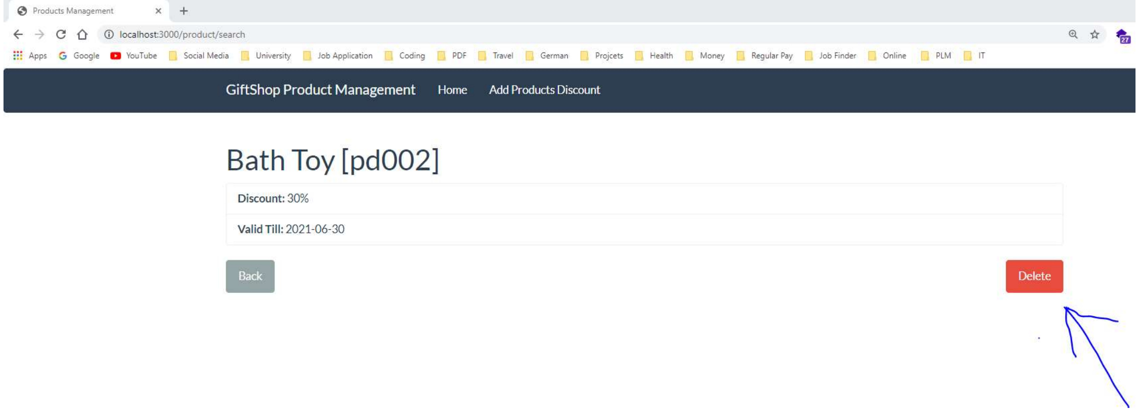


Bath Toy [pd002]

Discount: 30%

Valid Till: 2021-06-30

- Using delete button functionality

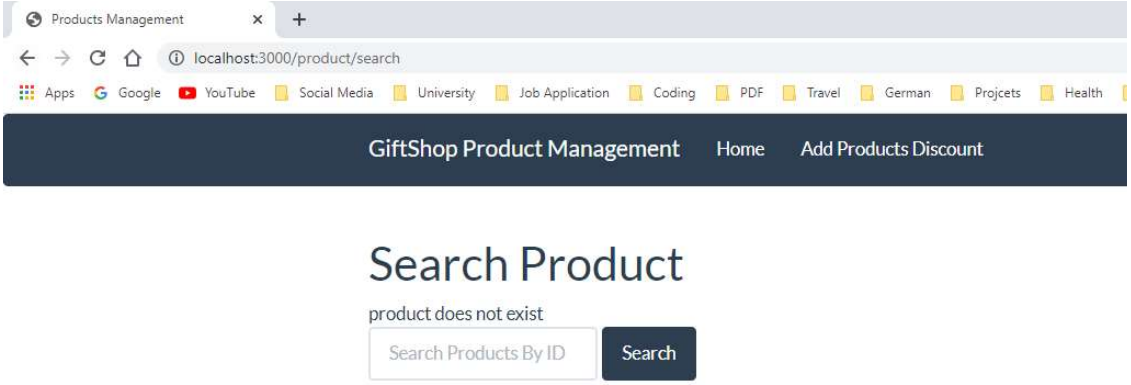


Bath Toy [pd002]

Discount: 30%

Valid Till: 2021-06-30

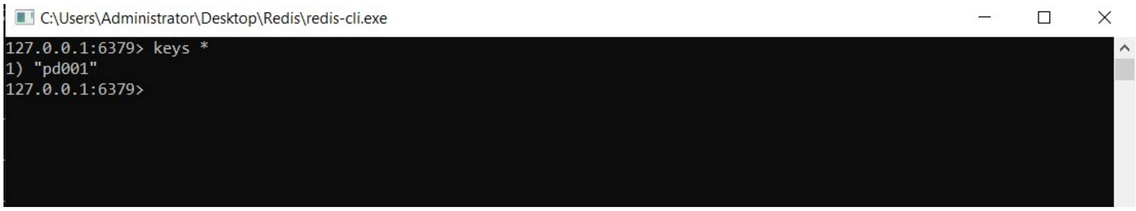
- Result - Product doesn’t exist



Search Product

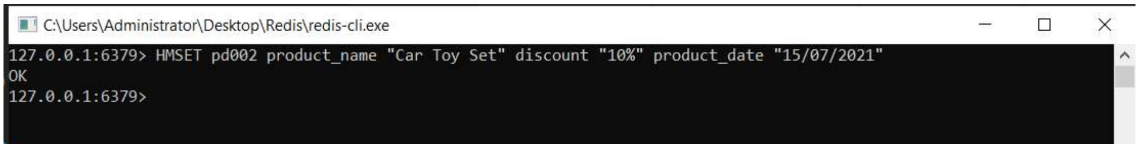
product does not exist

- Redis CLI command
  Command: Keys *



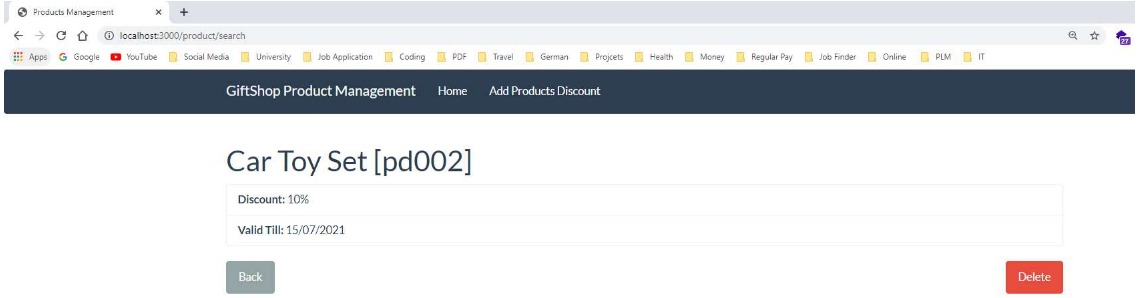
```
127.0.0.1:6379> keys *
1) "pd001"
127.0.0.1:6379>
```

- By Redis CLI we can store product
  Command: HMSET pd002 product_name "Car Toy Set" discount "10%" product_date "15/07/2021"



```
127.0.0.1:6379> HMSET pd002 product_name "Car Toy Set" discount "10%" product_date "15/07/2021"
OK
127.0.0.1:6379>
```

- Output



Car Toy Set [pd002]

Discount: 10%

Valid Till: 15/07/2021

Databases

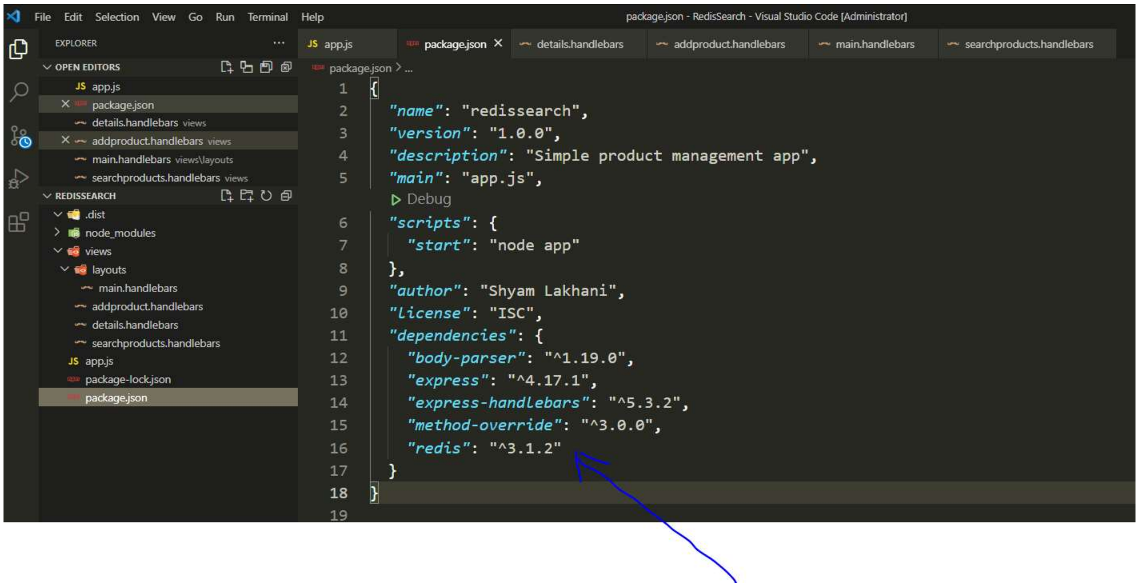
Why Redis database for Chat?

Redis is a pretty good choice as a database for a chat as it provides a couple of data structures that are not only very handy for various chat use cases but also processed in a really performant way.

Redis is a very useful data service for tying microservices together and following the 12 factor app principles. For workloads focusing on rapidly changing ephemeral data sets where privilege control is not a concern (i.e. apps that you trust enough or less sensitive data) Redis is a strong choice for database.

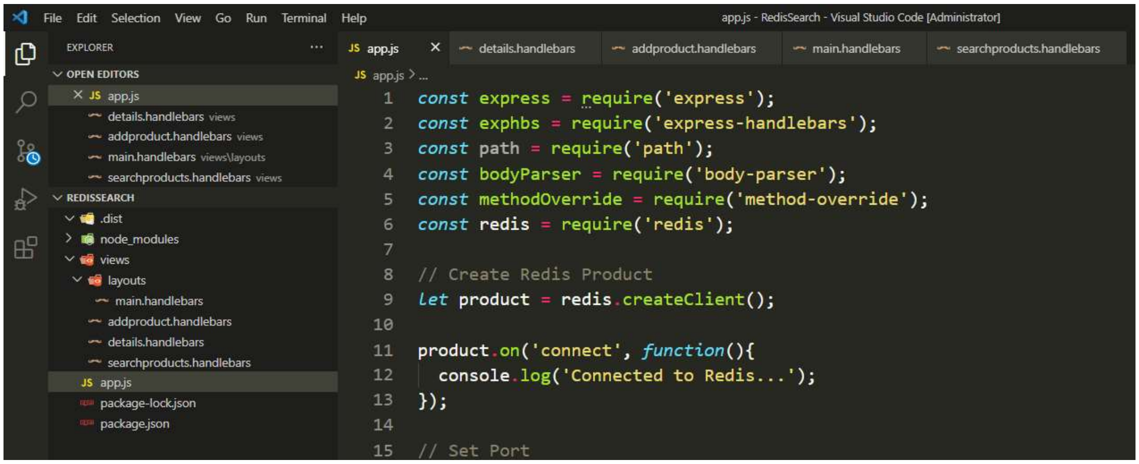
❖ Expressions used for this use case:

Before start entire coding, I install Redis and you can find redis along with version in package.json


File Edit Selection View Go Run Terminal Help
package.json - RedisSearch - Visual Studio Code [Administrator]
EXPLORER
OPEN EDITORS
app.js
package.json
details.handlebars views
addproduct.handlebars views
main.handlebars views\layouts
searchproducts.handlebars views
REDISSEARCH
.dist
node_modules
views
layouts
main.handlebars
addproduct.handlebars
details.handlebars
searchproducts.handlebars
app.js
package-lock.json
package.json
package.json > ...
▷ Debug

```
{
  "name": "redissearch",
  "version": "1.0.0",
  "description": "Simple product management app",
  "main": "app.js",
  "scripts": {
    "start": "node app"
  },
  "author": "Shyam Lakhani",
  "license": "ISC",
  "dependencies": {
    "body-parser": "^1.19.0",
    "express": "^4.17.1",
    "express-handlebars": "^5.3.2",
    "method-override": "^3.0.0",
    "redis": "^3.1.2"
  }
}
```

Here I used in node.js and also create connection to show in console.log that connected to Redis…

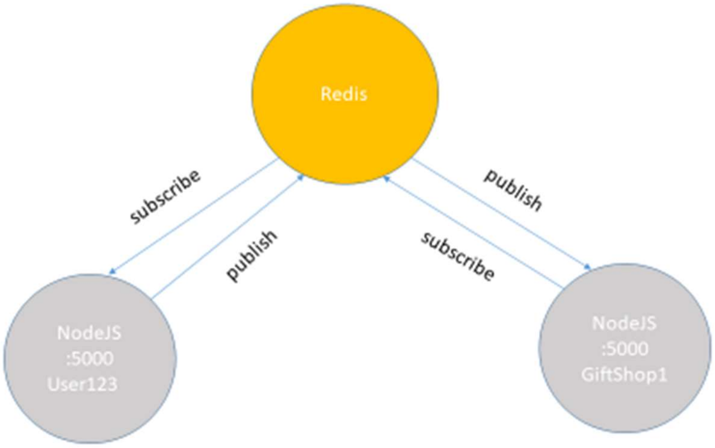

File Edit Selection View Go Run Terminal Help
app.js - RedisSearch - Visual Studio Code [Administrator]
EXPLORER
OPEN EDITORS
app.js
details.handlebars views
addproduct.handlebars views
main.handlebars views\layouts
searchproducts.handlebars views
REDISSEARCH
.dist
node_modules
views
layouts
main.handlebars
addproduct.handlebars
details.handlebars
searchproducts.handlebars
app.js
package-lock.json
package.json
app.js > ...

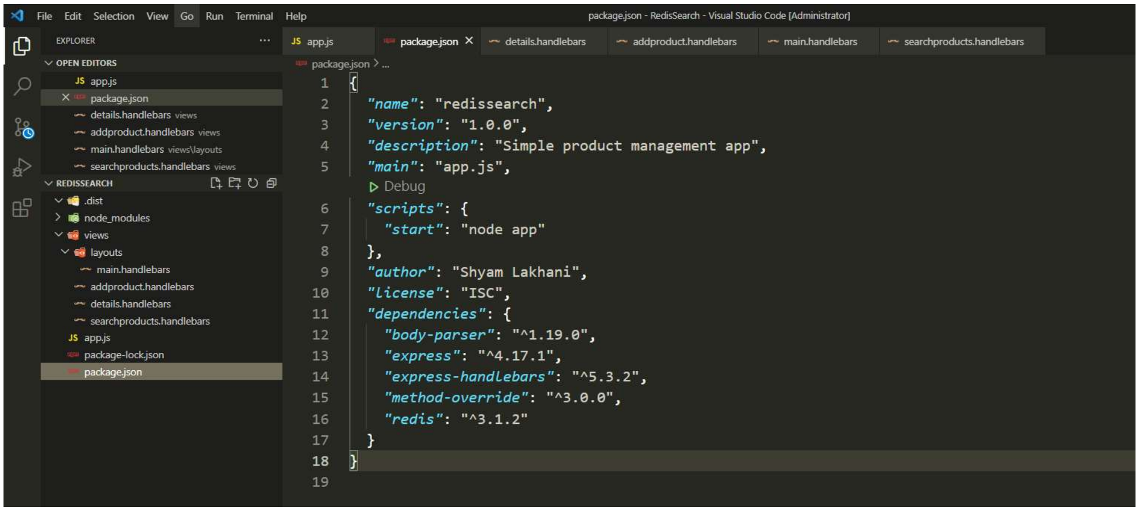
```
const express = require('express');
const exphbs = require('express-handlebars');
const path = require('path');
const bodyParser = require('body-parser');
const methodOverride = require('method-override');
const redis = require('redis');

// Create Redis Product
let product = redis.createClient();

product.on('connect', function(){
  console.log('Connected to Redis...');
});

// Set Port
```

Outcomes - What did you learn?

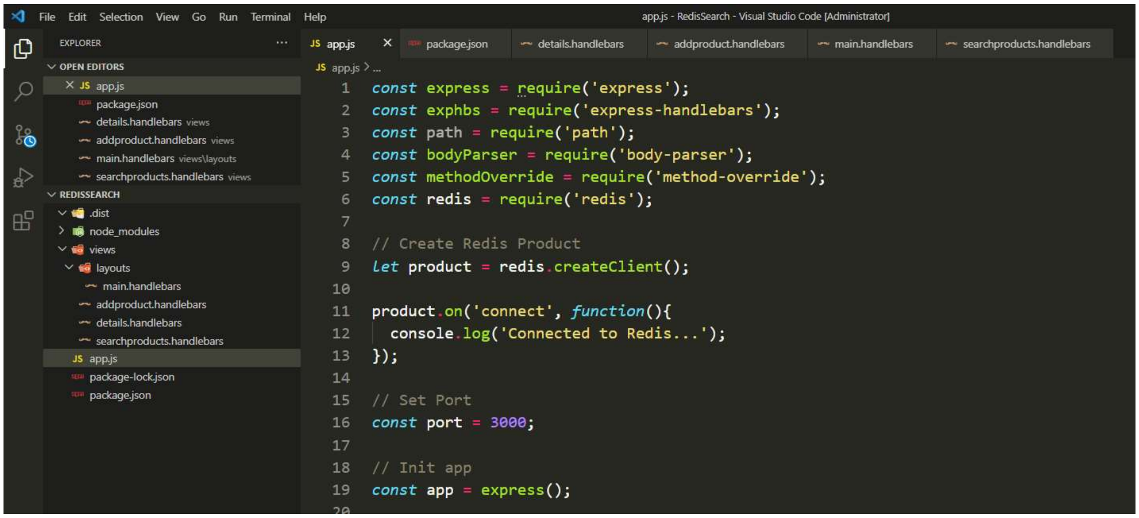
This powerful database is perfect for high performance jobs such as caching. Redis is a fast database for many different functions including as a cache or a message broker. I learned everything through Redis tutorial, which is the best place to progress from a newbie to an advanced user of Redis, the basic fundamentals of Redis such as the different data structures, various clients that work with Redis, different key-value pair commands (scan, config, commands, and client), how to persist data to disks and even the different methods of persisting data. After that, I build a functional working task how to actually work with Redis in a real-world example. I built a task manager using NodeJS and Redis. I also learnt how to incorporate Twitter Bootstrap for designing the manager.

Redis UML Keys





```
{
  "name": "redissearch",
  "version": "1.0.0",
  "description": "Simple product management app",
  "main": "app.js",
  "scripts": {
    "start": "node app"
  },
  "author": "Shyam Lakhani",
  "license": "ISC",
  "dependencies": {
    "body-parser": "^1.19.0",
    "express": "^4.17.1",
    "express-handlebars": "^5.3.2",
    "method-override": "^3.0.0",
    "redis": "^3.1.2"
  }
}
```



```javascript
const express = require('express');
const exphbs = require('express-handlebars');
const path = require('path');
const bodyParser = require('body-parser');
const methodOverride = require('method-override');
const redis = require('redis');

// Create Redis Product
let product = redis.createClient();

product.on('connect', function(){
  console.log('Connected to Redis...');
});

// Set Port
const port = 3000;

// Init app
const app = express();
```