**Software Take Home Challenge from Samsung Research America- TTT**

**Name: Jaykumar Maheshbhai Bhingaradia**

**Education: Master’s in Computer Engineering at Northeastern University**

**Email –** [**jaybhigradia30@gmail.com**](mailto:jaybhigradia30@gmail.com)

**Mobile no - + (857) 300 – 3551**

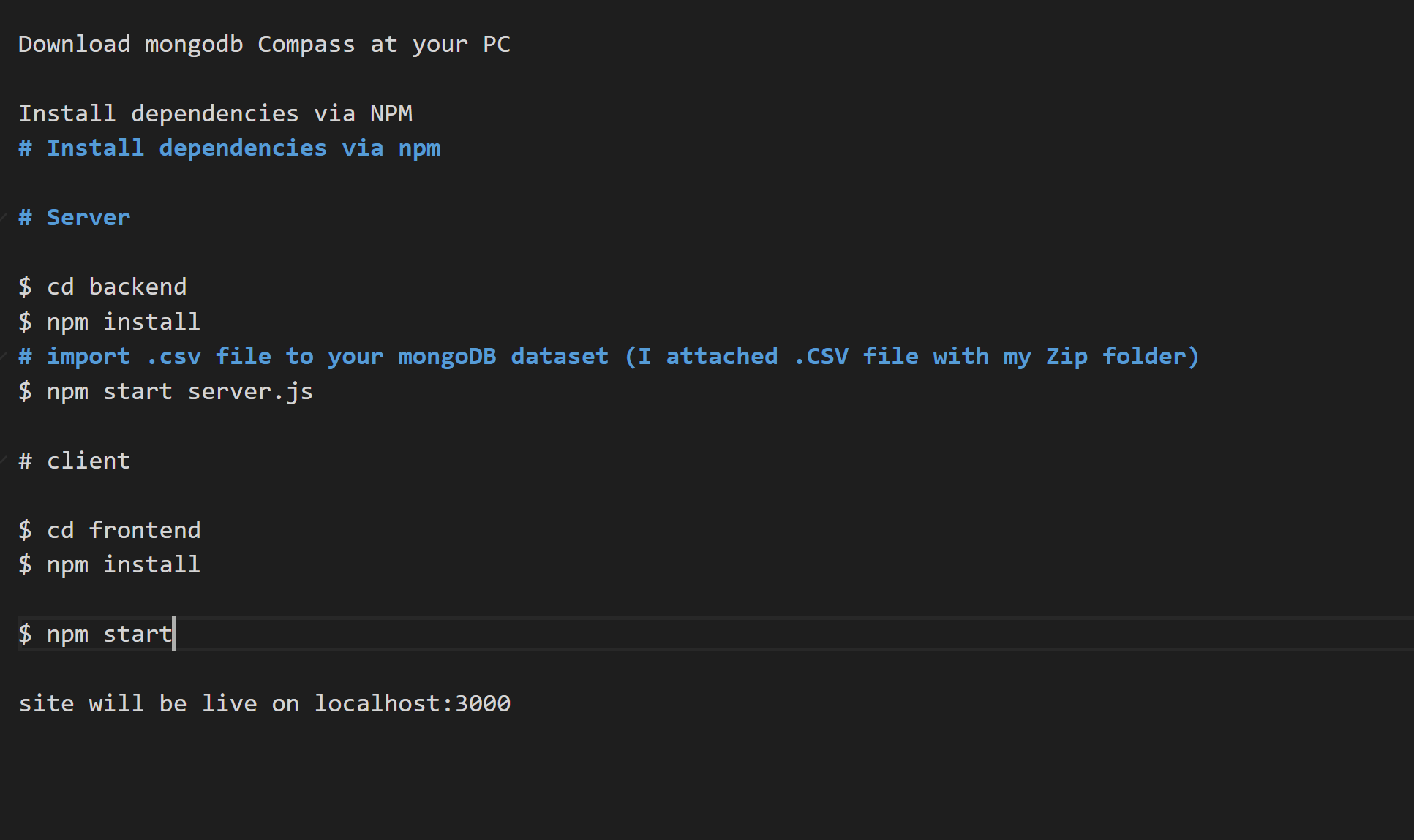
**Project – Full-Stack web application using MERN**

**(Mongo dB, React.js, Express, JavaScript)**

**User Manual**

**Implementation code:**

Readme:



**Setting React**

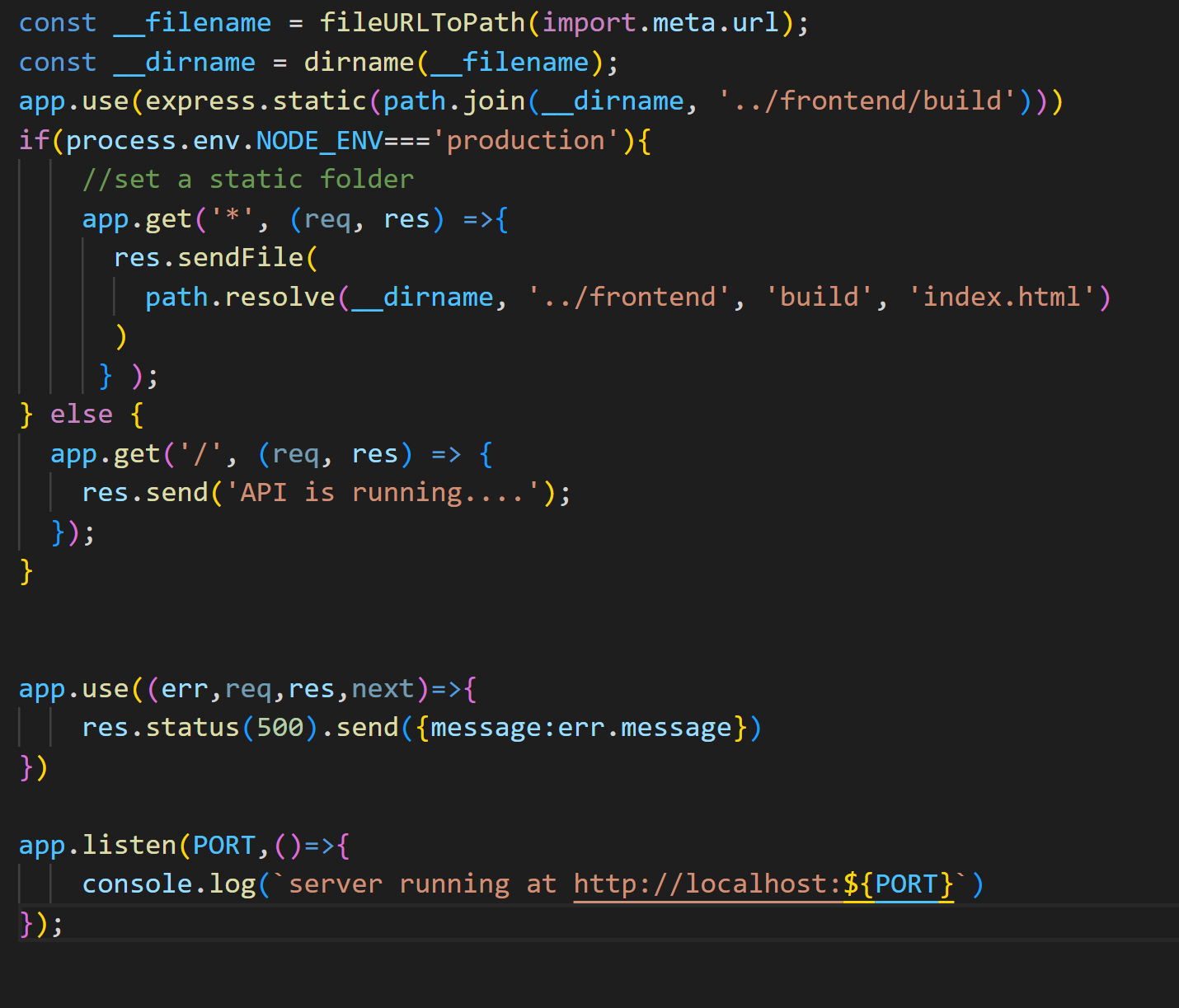
Index.js:



**Setting Node**

Server.js



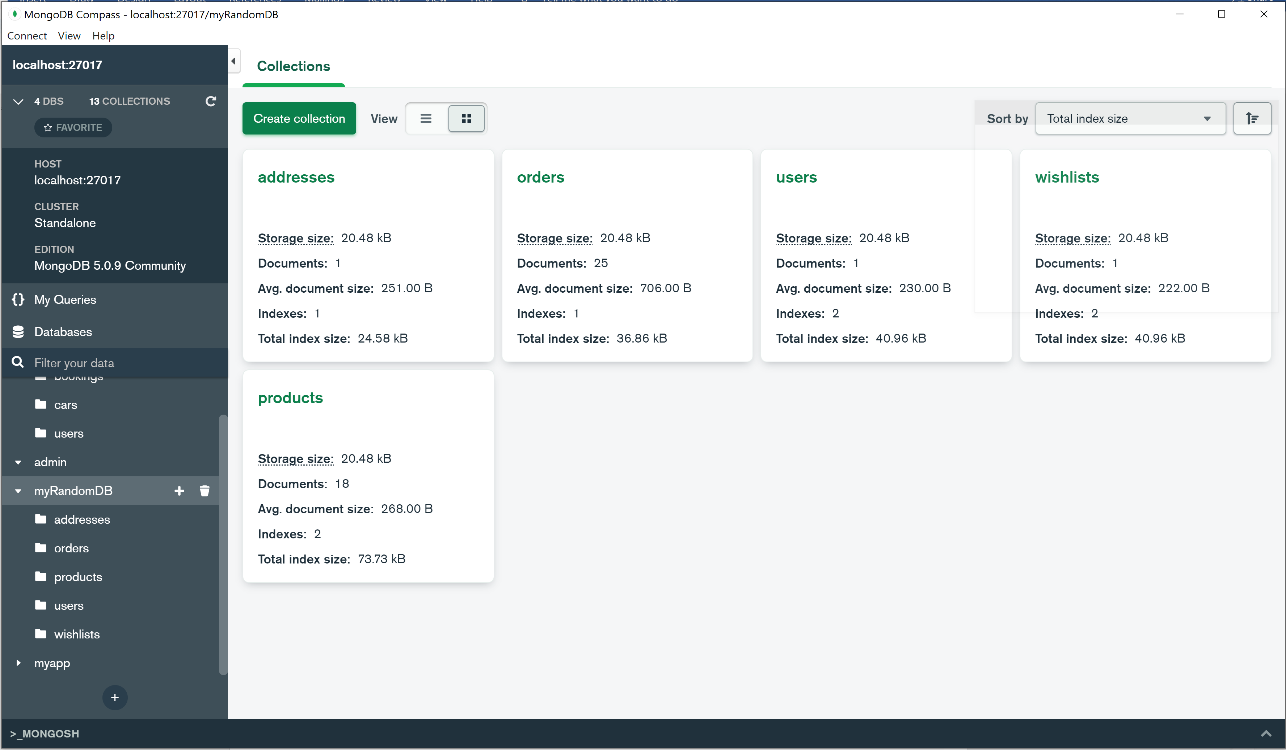


**Setting MongoDB:**

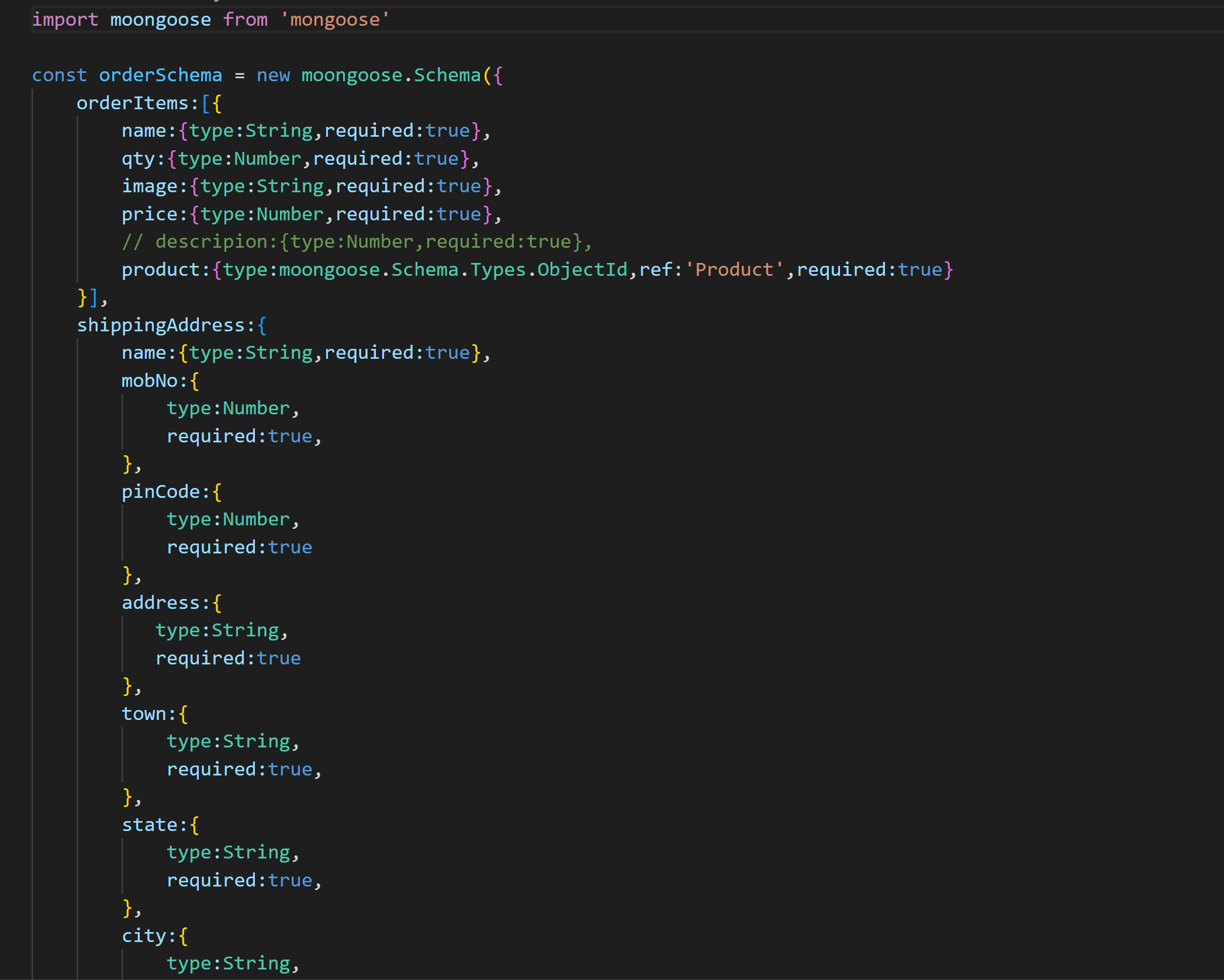
Go to MongoDB webpage and login to your account

Create new collection

Copy name and key to your server code.

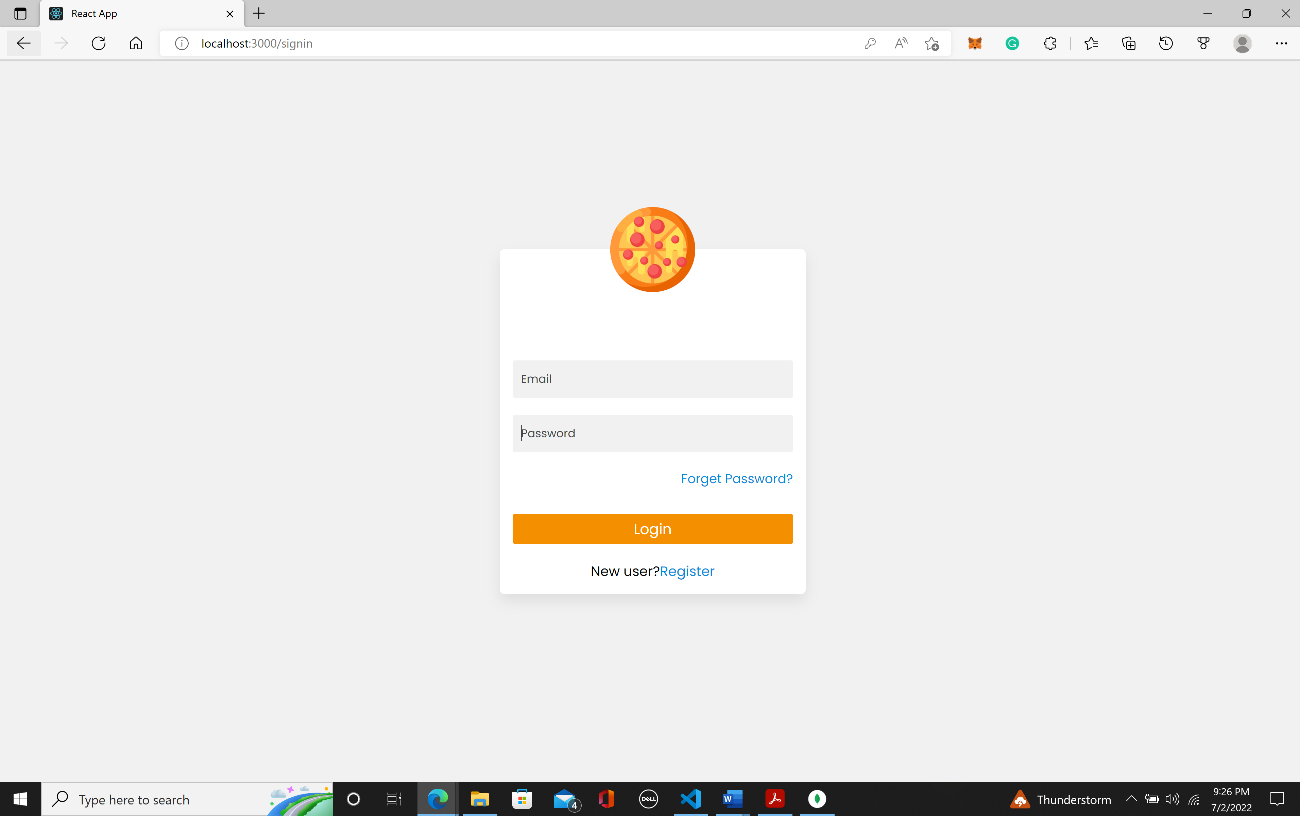


**Database Model**

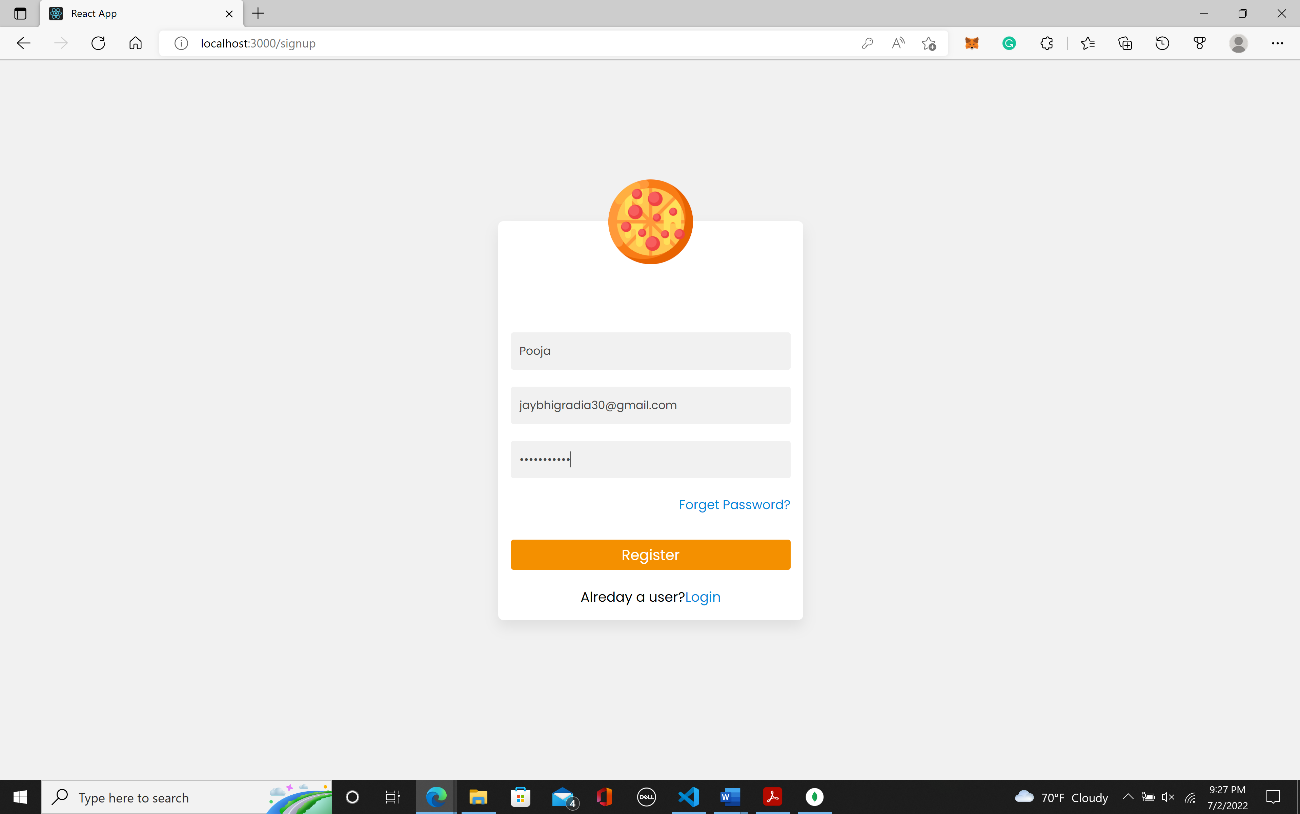


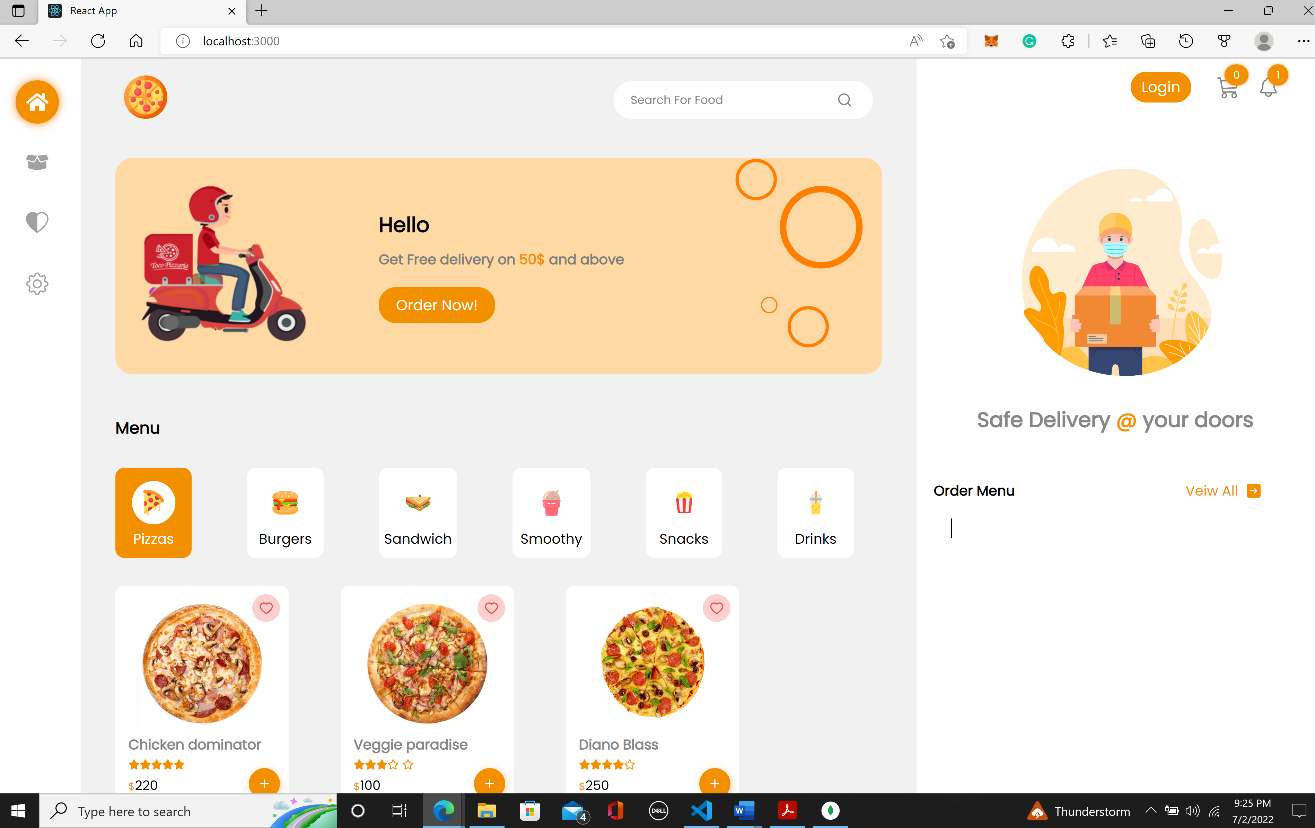
**Output:**

**Login:**

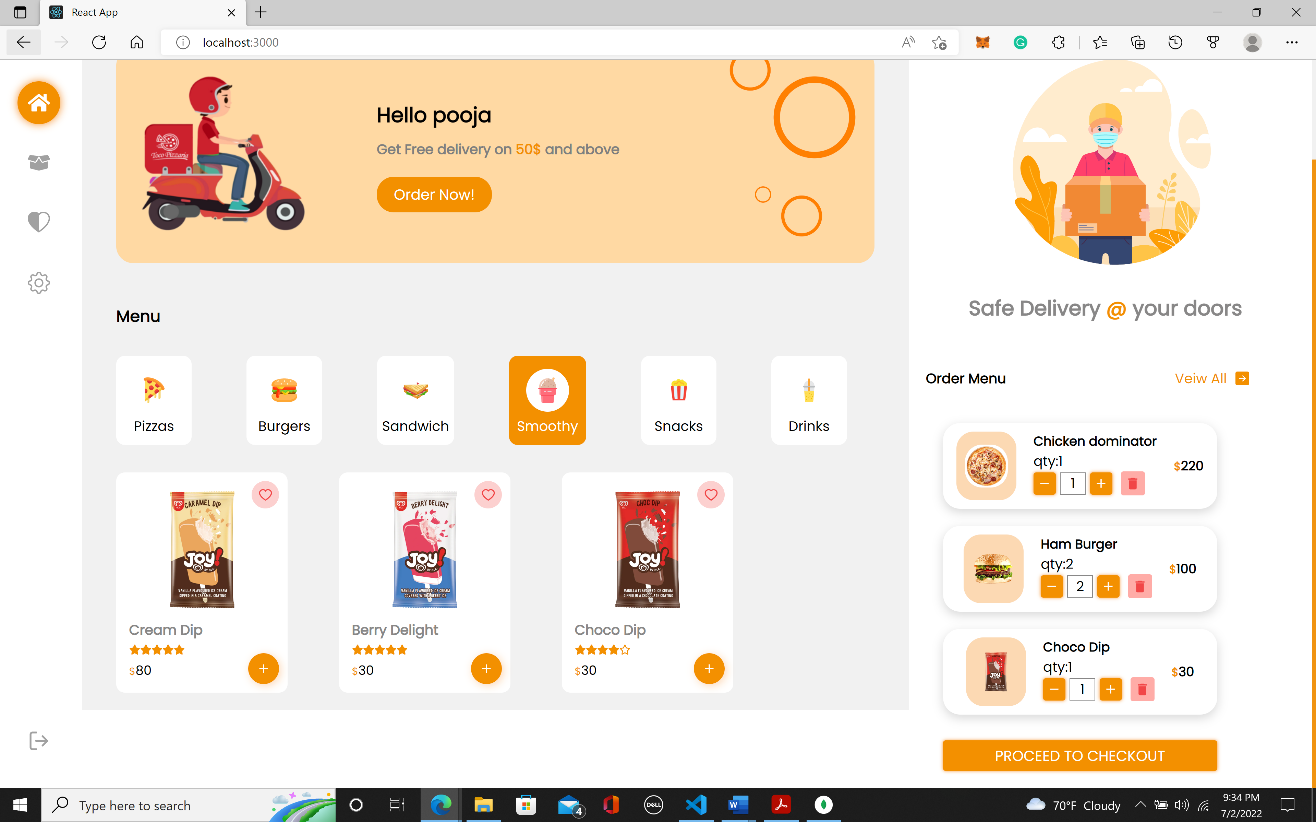


**Sign Up:**

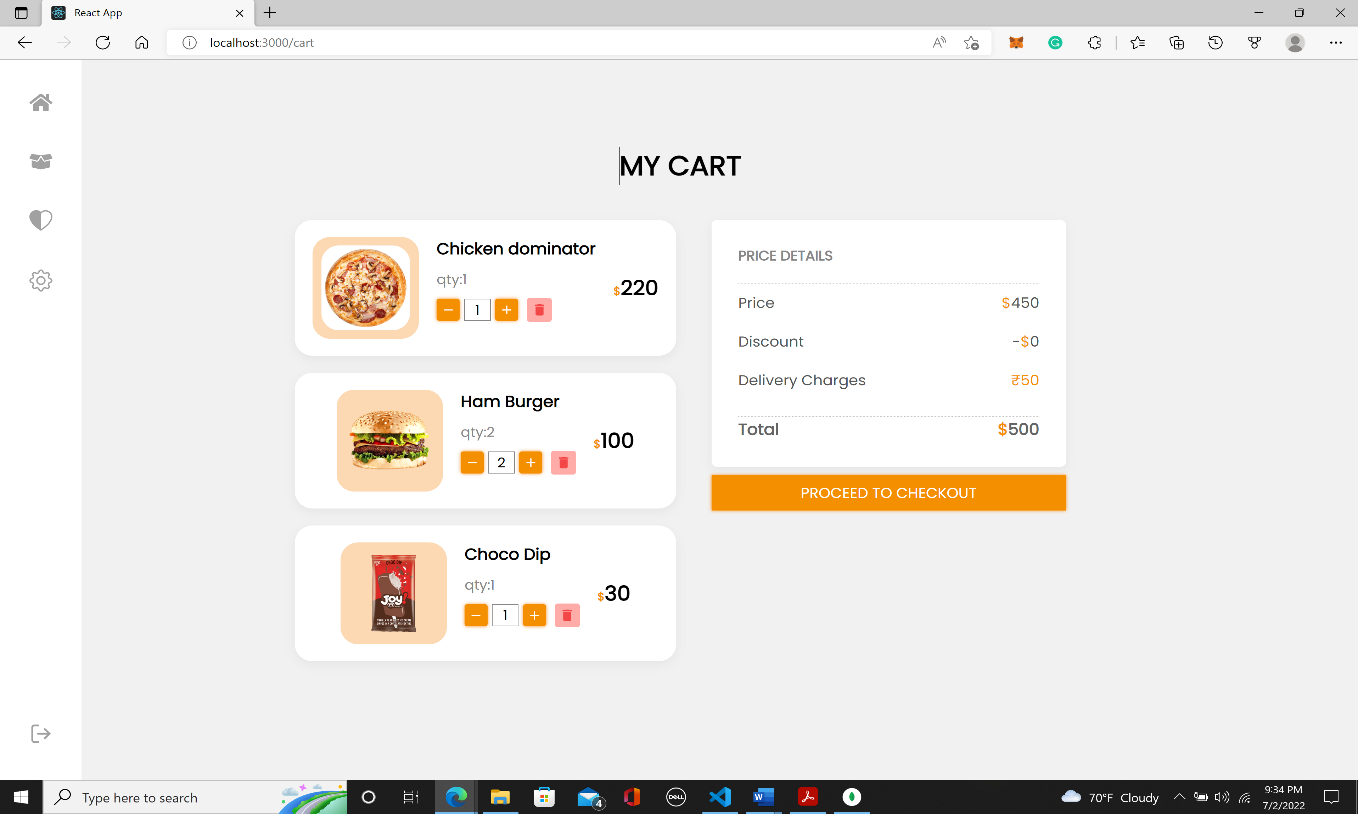
**Home page**



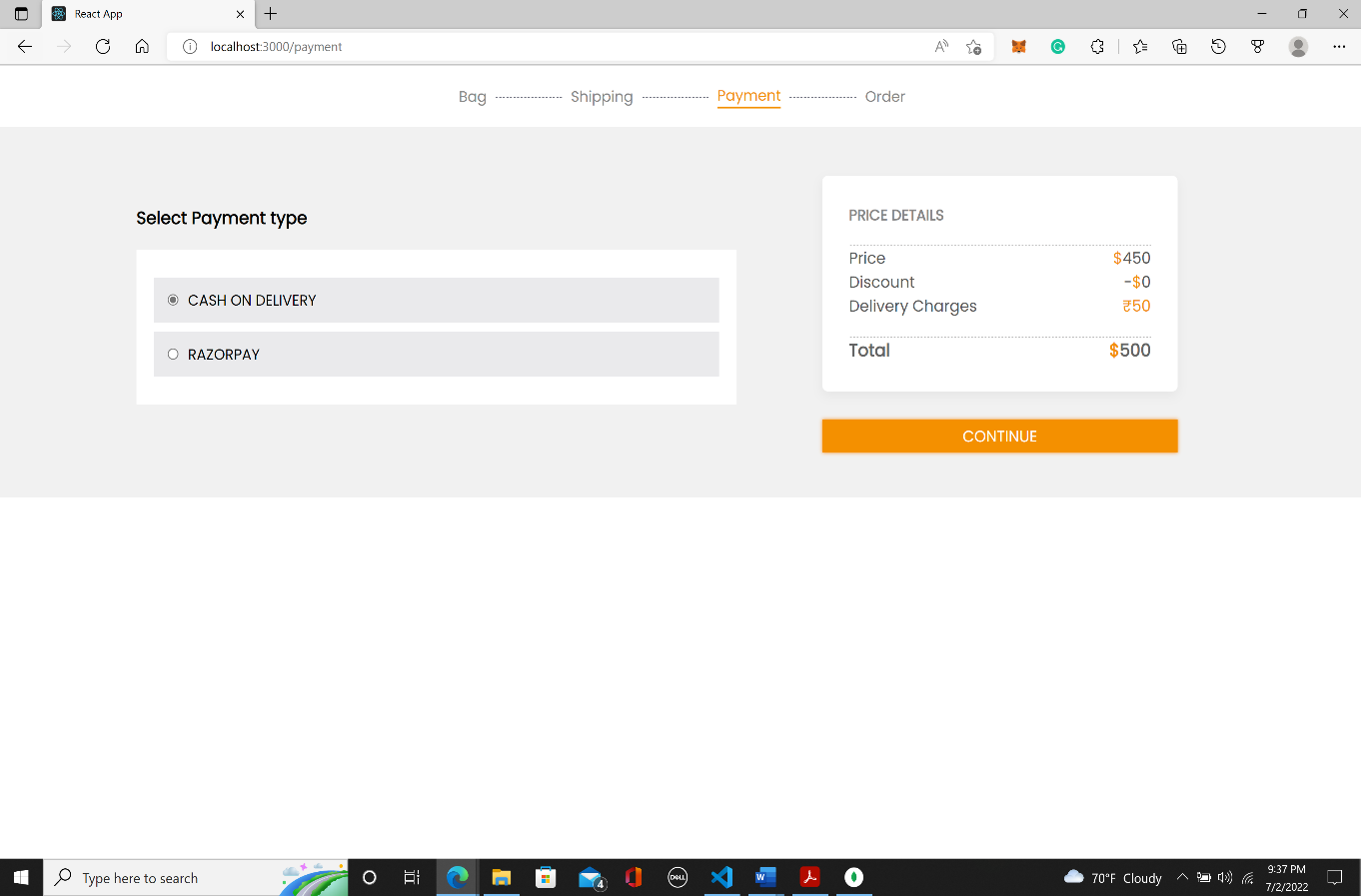
**Order Menu:**

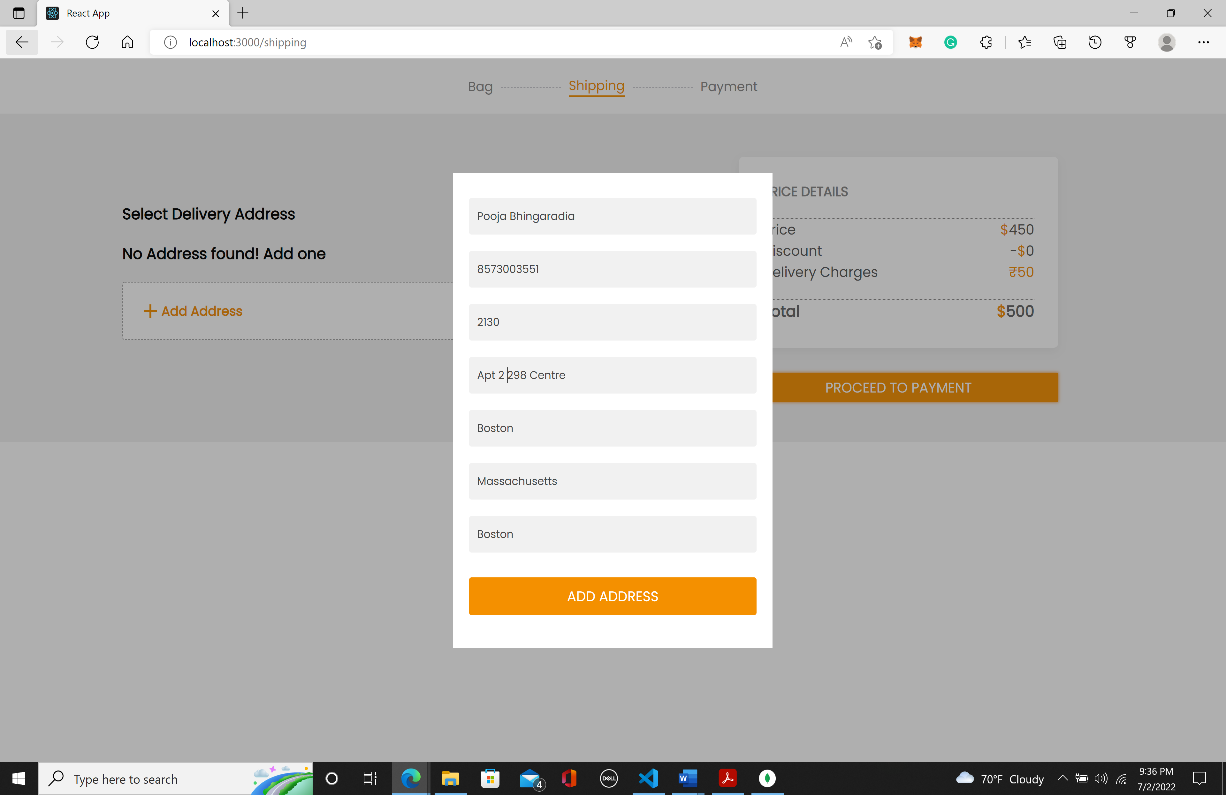


**MY CART:**

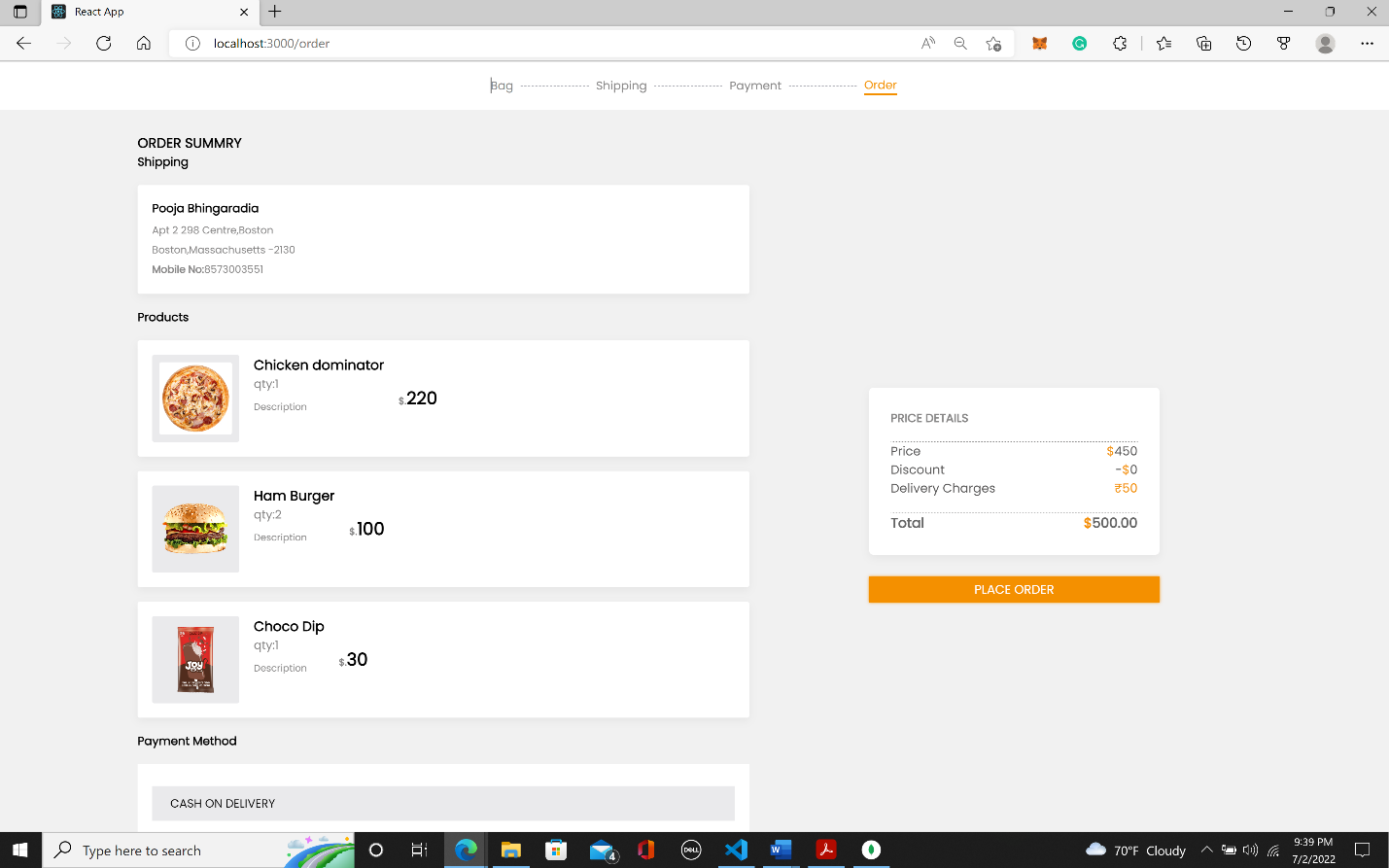


**Payment:**

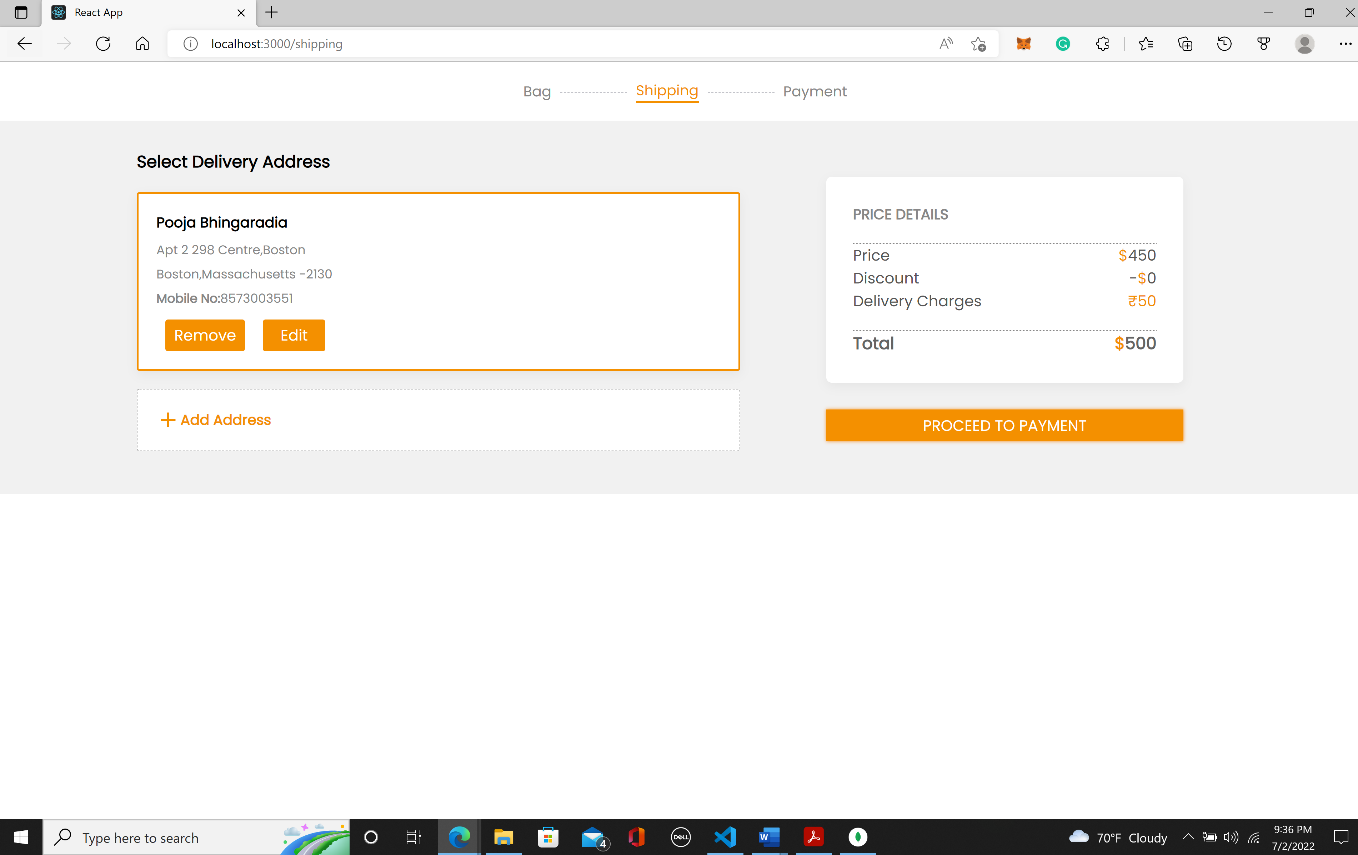
**Address:**



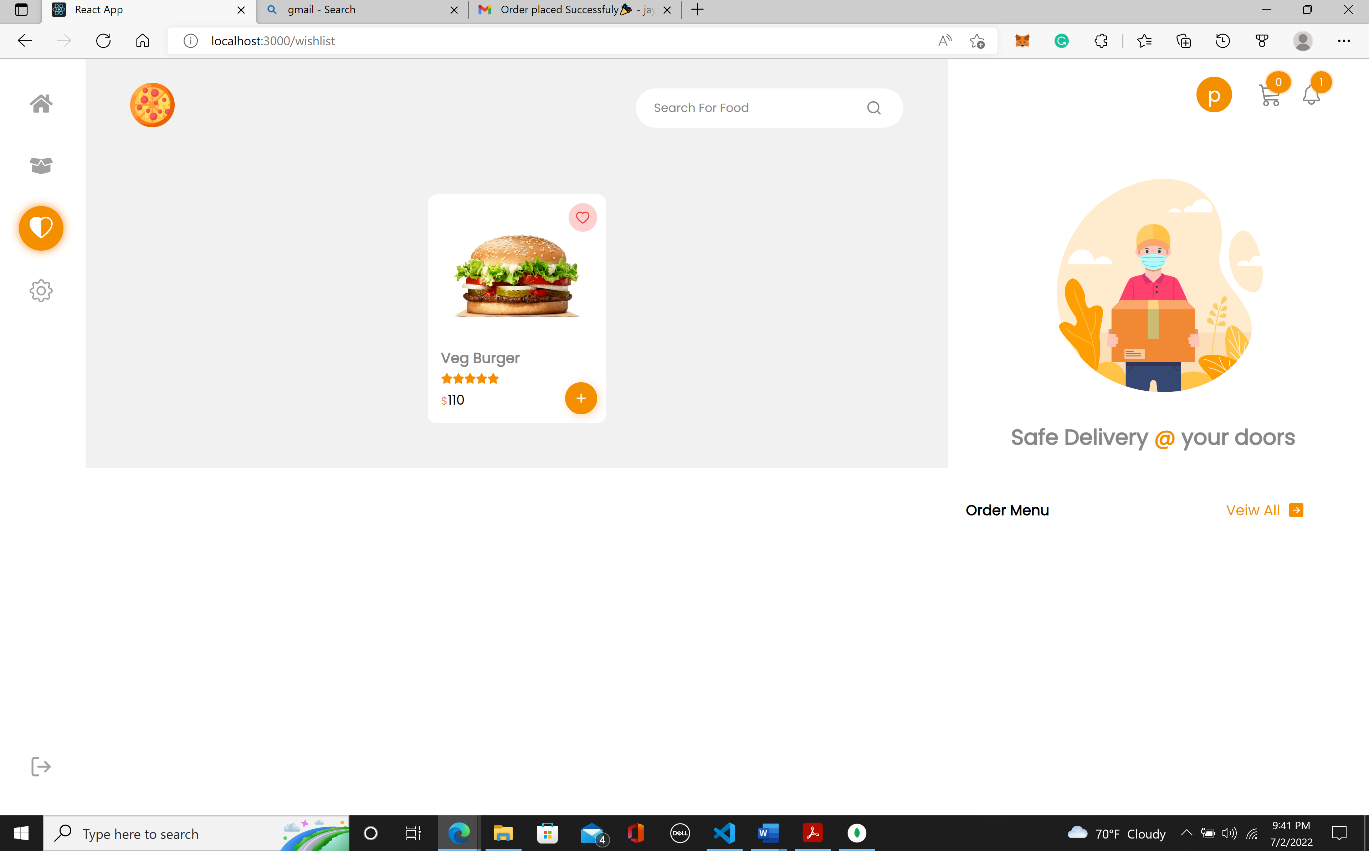
**Shipping:**

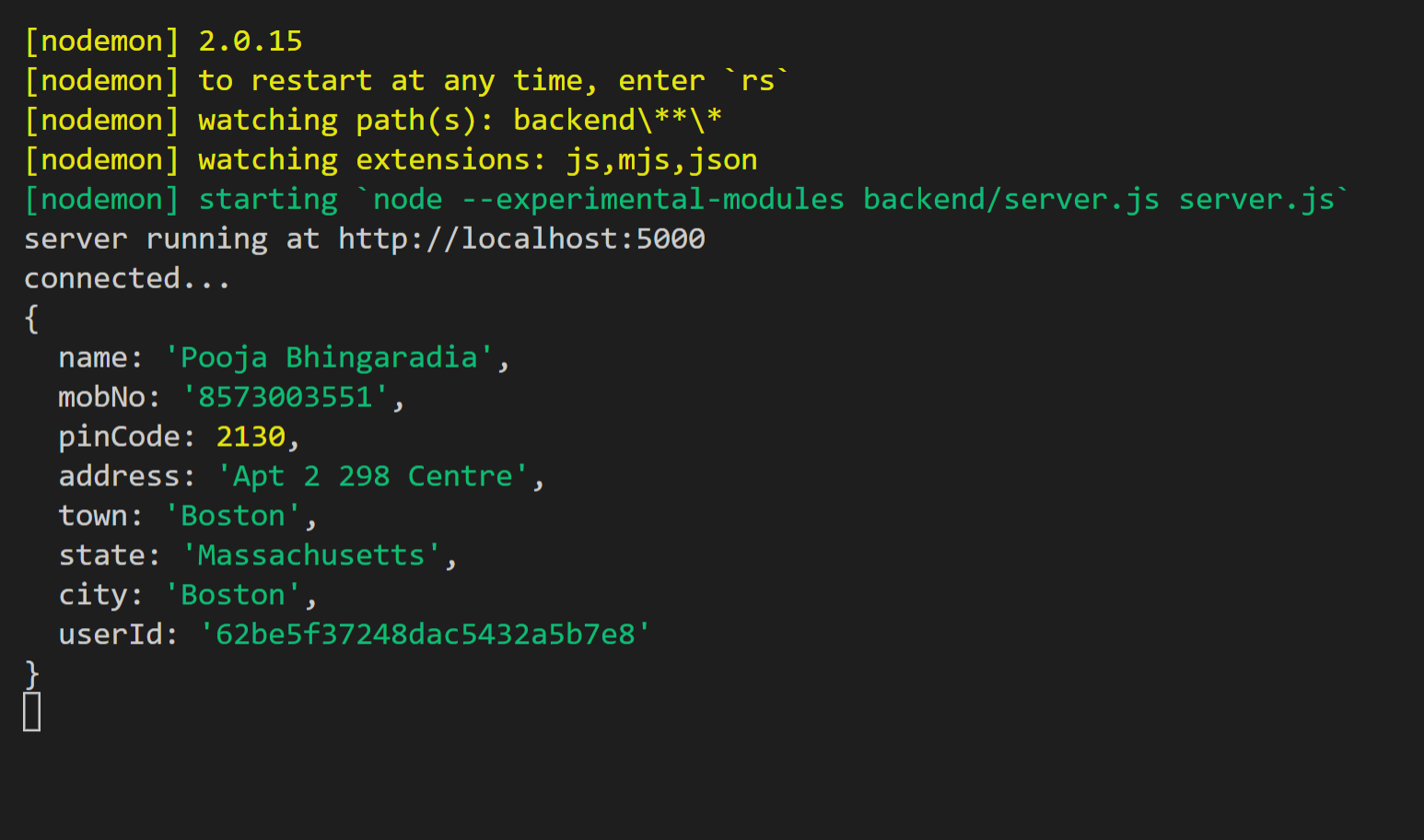


**Delivery Address:**

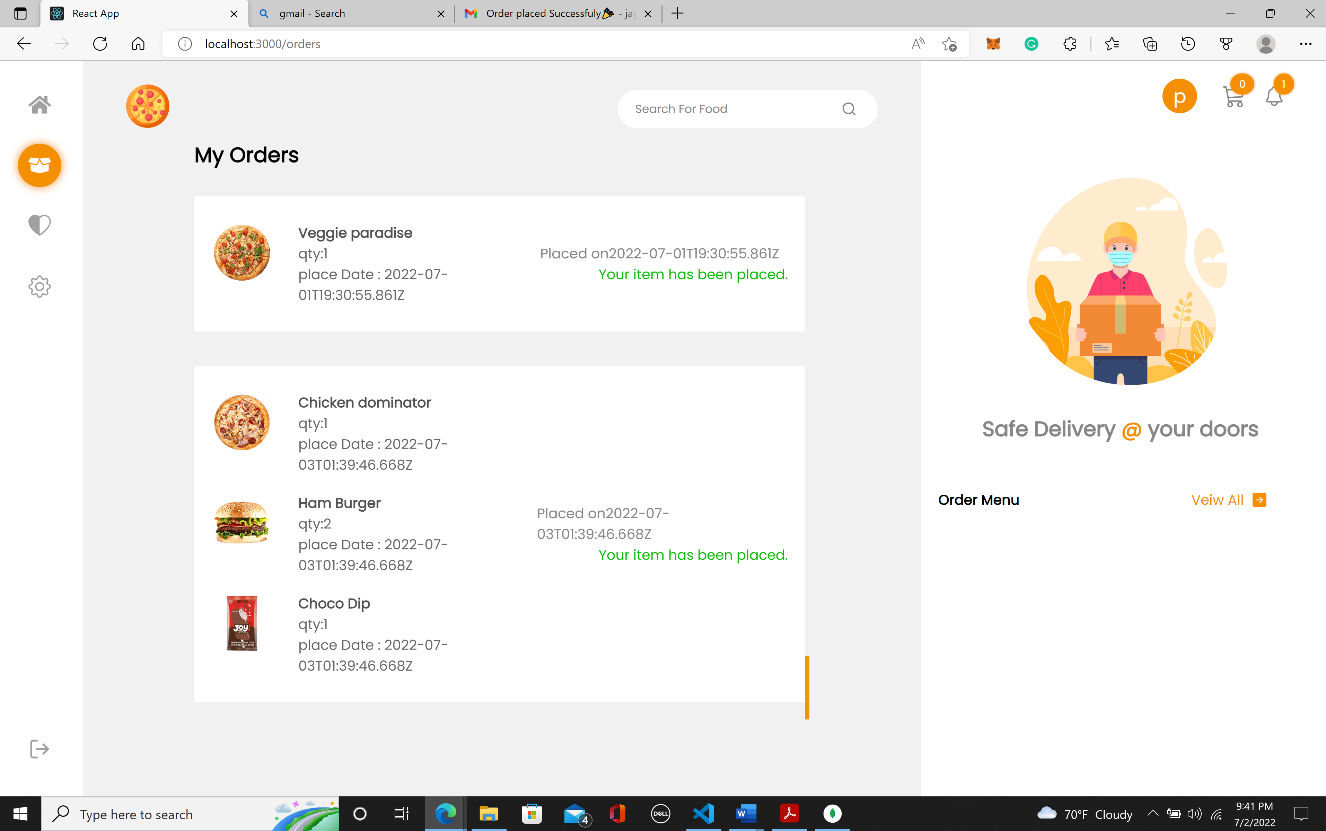


**Wishlist:**

**Conformation:**



**Order History**



**Abstract:**

A fully integrated restaurant management system that helps to place an order online with the payment by enabling small restaurants to offer their customers an online ordering option without having to spend a significant amount of time and money having custom software designed specifically for them, the online food ordering system described in this document has been created to fill a specific gap in the market. Through a very user-friendly graphical interface, the system, which is very adaptable, enables the restaurant staff to quickly modify the site content, most crucially the menu, themselves.

The website is then constructed dynamically depending on the system's current state so that any changes are reflected in real-time. The website is the only component that restaurant patrons see. When users register on the website, they are then able to simply peruse the menu, add food items to their order, and designate delivery choices with just a few clicks, substantially streamlining the ordering process. Placed orders are swiftly recovered and shown for processing back in the restaurant in an easily understandable style.

This document's goal is to give comprehensive explanations of the system's design and implementation details, as well as descriptions of all currently offered capabilities and future plans. To offer the reader a thorough understanding of the intended usual use cases for the system, user manuals and troubleshooting advice have also been given for all three components.

**Introduction**

**Project Overview**

In the era of fast food and take-out, many businesses have opted to prioritize faster order processing and delivery above providing a memorable dining experience. All these delivery orders have historically been placed over the phone, but there are a few drawbacks to this practice. First, a tangible menu that is current and available to the consumer is a need for placing an order at the restaurant. Although this expectation is realistic, it is undoubtedly uncomfortable.

Second, because orders are only submitted orally, it is far more difficult for customers to get feedback right away on the orders they have placed. This frequently causes misunderstanding and wrong order. The existing approach is also cumbersome for the restaurant itself since they must either hire a dedicated person to accept orders and answer the phone, or certain staff members must work double shifts, which takes them away from their primary duties.

My system's biggest benefit is how much easier it makes it for customers and restaurants to place orders. When a consumer accesses the ordering webpage, they are shown an interactive menu that is always up to date, includes all available options and dynamically changes costs based on the selections picked. The item is then added to the customer's order after selection, which they can examine at any moment before completing the transaction.

Except for a few small flaws that are certain to surface during more thorough testing, the system is now fully operational. Currently, a user may register, log in, and place an order on the website. The order retrieval desktop program then displays that order accurately and thoroughly. The majority of the work that needs to be done focuses on enhancing user experience rather than functionality. To do this, richer graphical interfaces for the user to interact with must be created, and the application's icons and color schemes must be changed to make them more aesthetically pleasing and easier to use.

**Purpose**

The purpose is to have a system where the payment gateway is well integrated with the orders. In the future, we can endeavor to connect the online stores with this ordering system. Additionally, by creating this application, we hope to get a user-friendly interface.

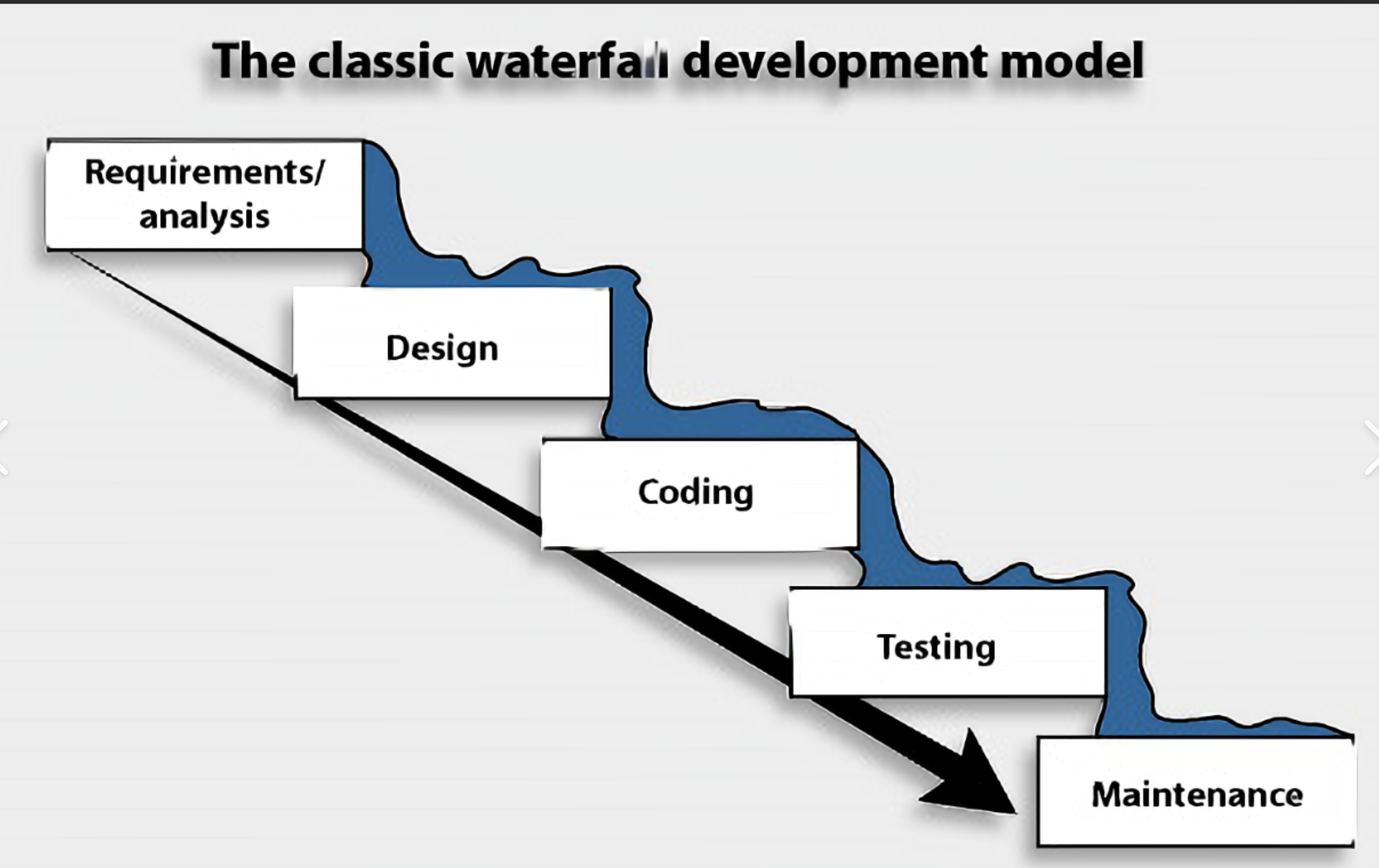
**Scope**

The structure of the online ordering system is similar to that of a typical web application; thus, it doesn't need any specialized hardware and can run on conventional software such as web and database servers. I thus anticipate completing all necessary work on time or, ideally, earlier than expected, giving me time to research a few more features that I would want to include but are not essential to the system.

**Model Selection:**

To develop the whole model, I follow the standard waterfall model to develop the whole system.

In the development of this project, we will check in the first Sprint whether our project is feasible functionally, technically, and economically. Hence, we gather all the requirements which we need to develop our system. Then, after thoroughly understanding the requirements, we will start development. Our development process divides basically into three parts.



In contrast to software life cycle models, software process models often represent a networked sequence of activities, objects, transformations, and events that embody strategies for accomplishing software evolution. Such models can be used to develop more precise and formalized descriptions of software life cycle activities. Their power emerges from their utilization of a sufficiently rich notation, syntax, or semantics, often suitable for computational Processing.

**Incremental Model:**

There is a good chance that a requirements error will be recognized as soon as the corresponding software is incorporated into the system. It is then not a big deal to correct it. The waterfall model relies on a careful review of documents to avoid errors. Once a phase has been completed, there is limited provision for stepping back. It is difficult to verify documents precisely and this is, again, a weakness of the waterfall model. An incremental model is an intuitive approach to the waterfall model. Multiple development cycles take place here, making the life cycle a “multi-waterfall” cycle. Cycles are divided up into smaller, more easily managed iterations. Each iteration passes through the requirements, design, implementation, and testing phases.

A working version of the software is produced during the first iteration, so you have working software early on during the software life cycle. Subsequent iterations build on the initial software produced during the first iteration.

**sRequirements / Analysis:**

**Hardware Requirements**

● **Processor:** Intel i5 or above

**● RAM:** 8GB or above

**● Hard Disk:** Minimum 5 GB free anytime

**Software Requirements**

**● OS:** Windows/Linux/Mac

**● Web Browser:** Chrome, Firefox

**● Mongo Db**

**● React.js**

**● Express**

**● Visual Studio**

**● Node.js**

**● npm Dependency**

**SYSTEM ANALYSIS**

**REQUIREMENTS FOR NEW SYSTEM**

**Non-Functional Requirement**

* **Usability** the UI of the Splunk App should be user-friendly so that users can navigate easily through the app.
* **Accuracy** As we were developing the application, we must make the system that is very accurate in its functions. All the data should keep working properly, keep getting perfect input, process accurately and produce the perfect output. Accuracy is the most important non-functional characteristic or requirement of the system.
* **Reliability** Error handling mechanism must be robust to avoid failure of the operation and in case of failure, the app reports it to the user without any due harm.
* **Performance and Automation** Application should work fast but here in this case automation is more important than performance. Once the application starts, the user application should complete all tasks without errors and give final visualization dashboards of the Splunk App.

**Software Requirements and Specification (SRS)**

**R.1. Authentication System**

**Description:**

The menu management system will be available only to restaurant employees and will, as the name suggests, allow them to manage the menu that is displayed to users of the web ordering system. The functions afforded by the menu management system provide users with the ability to, use a graphical interface:

* Add a new/update/delete vendor to/from the menu.
* Add a new/update/delete food category to/from the menu.
* Add a new/update/delete food item to/from the menu.
* Add a new/update/delete option for a given food item.
* Update the price for a given food item.
* Update default options for a given food item.
* Update additional information (description, photo, etc.) for a given food item.

It is anticipated that the functionality provided by this component will be one of the first things noted by the restaurant user, as they will have to go through it to configure their menu, etc. before beginning to take orders. Once everything is initially configured, however, this component will likely be the least used, as menu updates generally do not occur with great frequency.

**R.1.1 Login**

Input: Username, password

Output: Directed to home page

Processing: Find user is already registered or not then according to the database this gives permission to the user

**R.1.2. Sign Up**

Input: Username, login, email, password etc.

Output: New user register.

**R.2 Place Order:**

Input: Item Components, Payments Type, Shipping Address Etc

Output: Get invoice

**R.3 Search Food Item:**

Input: Food name

Output: Get the data of your food Item

**R.4 Payments:**

Input: Select Cash on delivery, Credit /Debit/Wallet type

Output: Go request to the Payment Gateways

**R.5 Create Wishlist:**

Input: select product which you want to add on Wishlist

Output: add item into the Wishlist

**R.6 Review:**

Input: give review for food item between (1-5 star)

Output: Register

**R.7 Update Profile:**

Input: modify detail add

Output: Updated on your profile

**R.8 Add Address:**

Input: Street No, Apartment name, City, State, Zip code, Name, Contact details

Output: Updated on portal

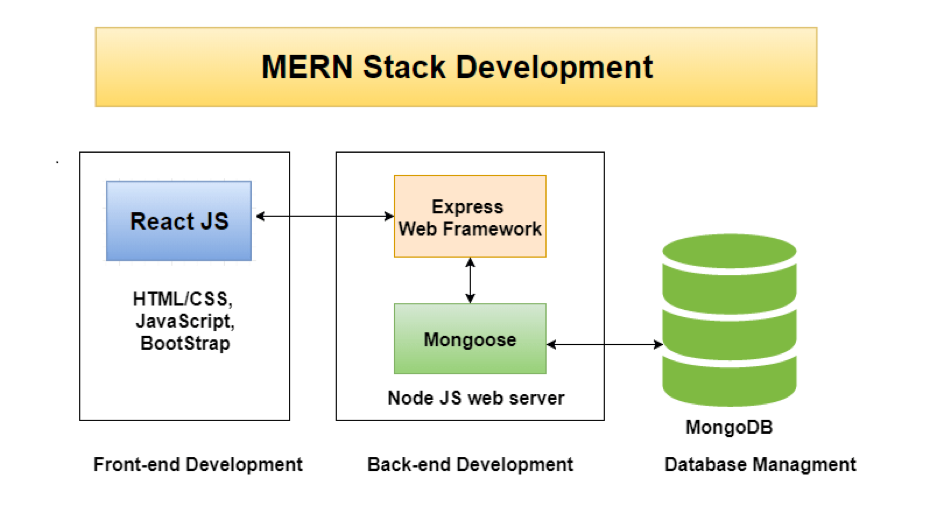
**R.9 Notification:**

Input: any new event generated in your portal you get notification

Output: according your notification you get action in portal

.

**SYSTEM ARCHITECTURE:**



MERN stands for MongoDB, Express, React, Node, after the four key technologies that make up the stack.

* MongoDB - document database
* Express(.js) - Node.js web framework
* React(.js) - a client-side JavaScript framework
* Node(.js) - the premier JavaScript web server

Express and Node make up the middle (application) tier. Express.js is a server-side web framework and Node.js is the popular and powerful JavaScript server platform. Regardless of which variant you choose, ME(RVA)N is the ideal approach to working with JavaScript and JSON, all the way through.

**React.js Front End**

The top tier of the MERN stack is React.js, the declarative JavaScript framework for creating dynamic client-side applications in HTML. React lets you build up complex interfaces through simple Components, connect them to data on your backend server, and render them as HTML.

React’s strong suit is handling stateful, data-driven interfaces with minimal code and minimal pain, and it has all the bells and whistles you’d expect from a modern web framework: great support for forms, error handling, events, lists, and more.

**Express.js and Node.js Server Tier**

The next level down is the Express.js server-side framework, running inside a Node.js server. Express.js bills itself as a “fast, unopinionated, minimalist web framework for Node.js,” and that is indeed exactly what it is. Express.js has powerful models for URL routing (matching an incoming URL with a server function), and handling HTTP requests and responses.

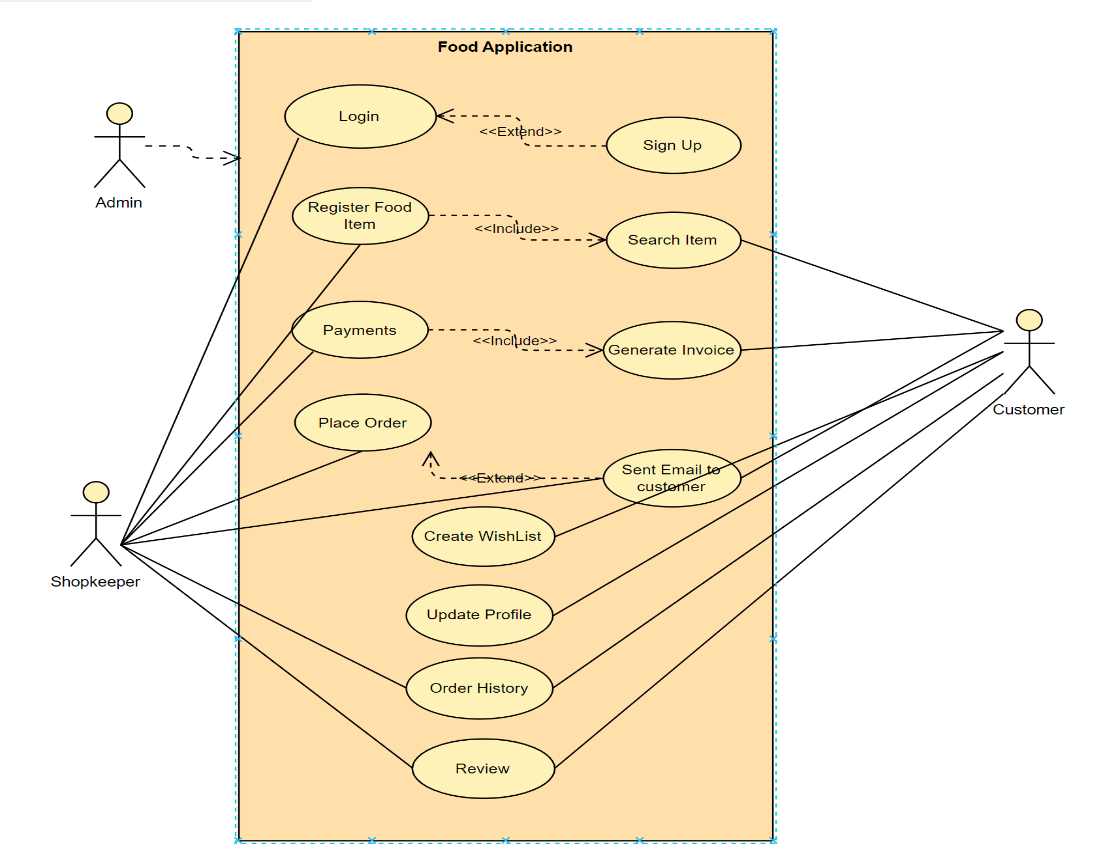
By making XML HTTP Requests (XHRs) or GETs or POSTs from your React.js front-end, you can connect to Express.js functions that power your application. Those functions in turn use MongoDB’s Node.js drivers, either via callbacks for using Promises, to access and update data in your MongoDB database.

**MongoDB Database Tier**

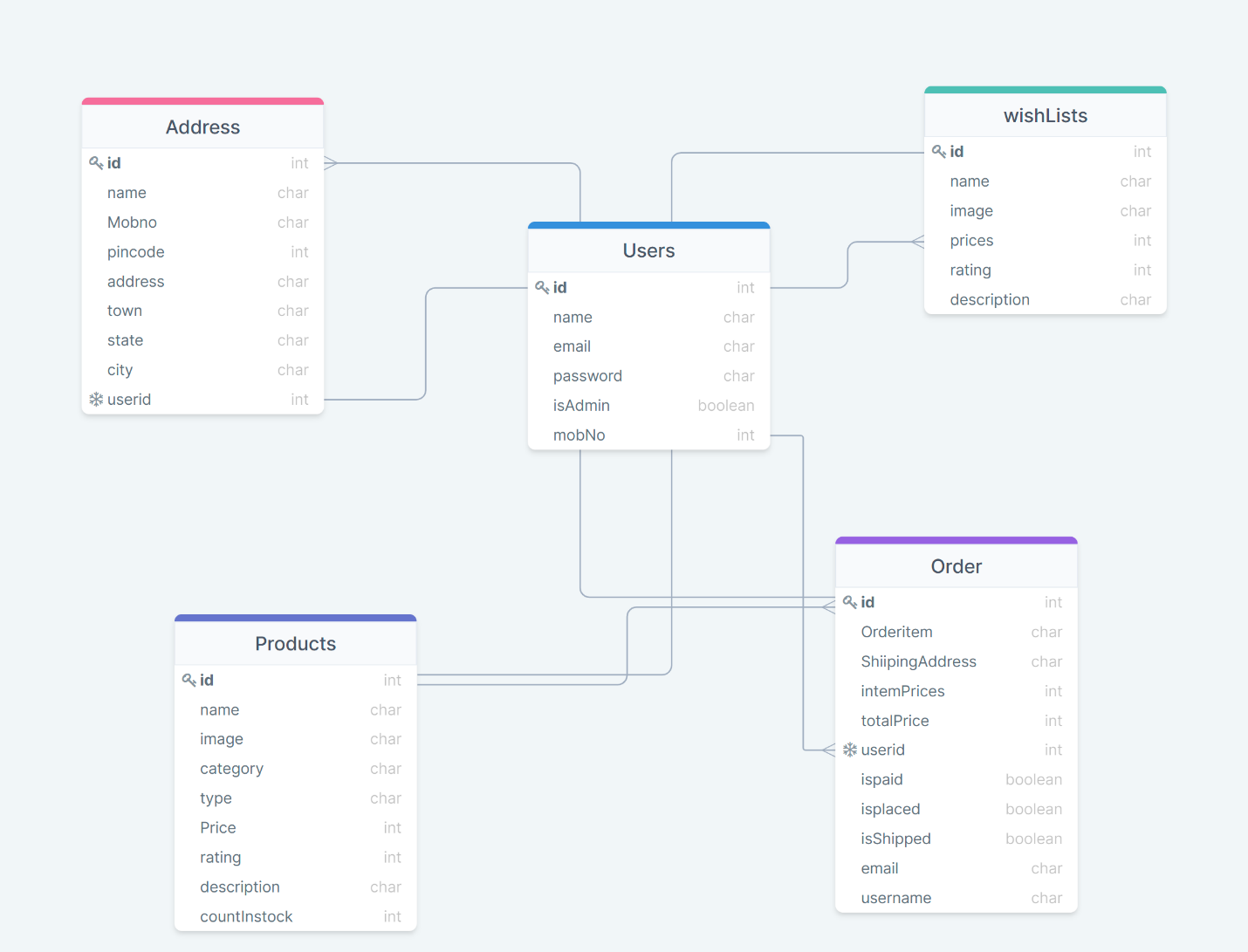
If your application stores any data (user profiles, content, comments, uploads, events, etc.), then you’re going to want a database that’s just as easy to work with as React, Express, and Node.

That’s where MongoDB comes in: JSON documents created in your React.js front end can be sent to the Express.js server, where they can be processed and (assuming they’re valid) stored directly in MongoDB for later retrieval. Again, if you’re building in the cloud, you’ll want to look at Atlas. If you’re looking to set up your own MERN stack, read on!

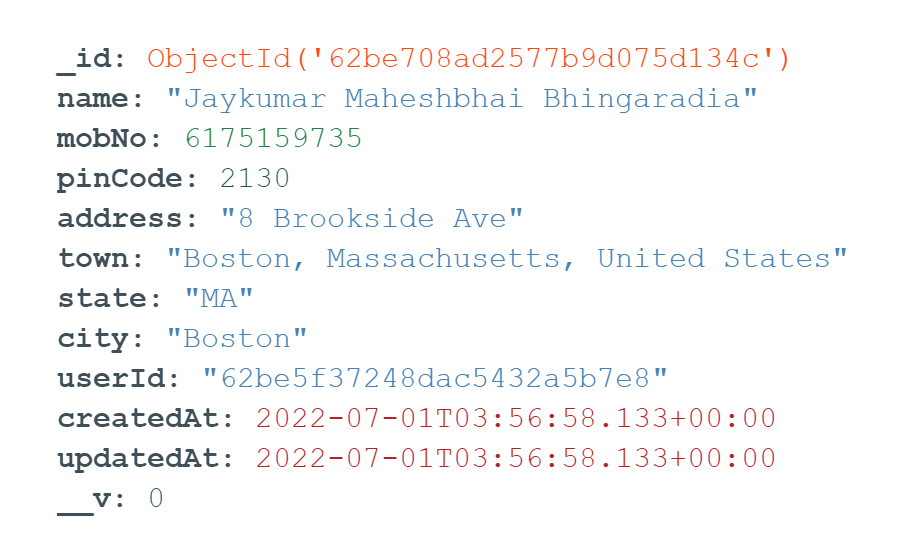
**Use case diagram**



**Schema diagram**

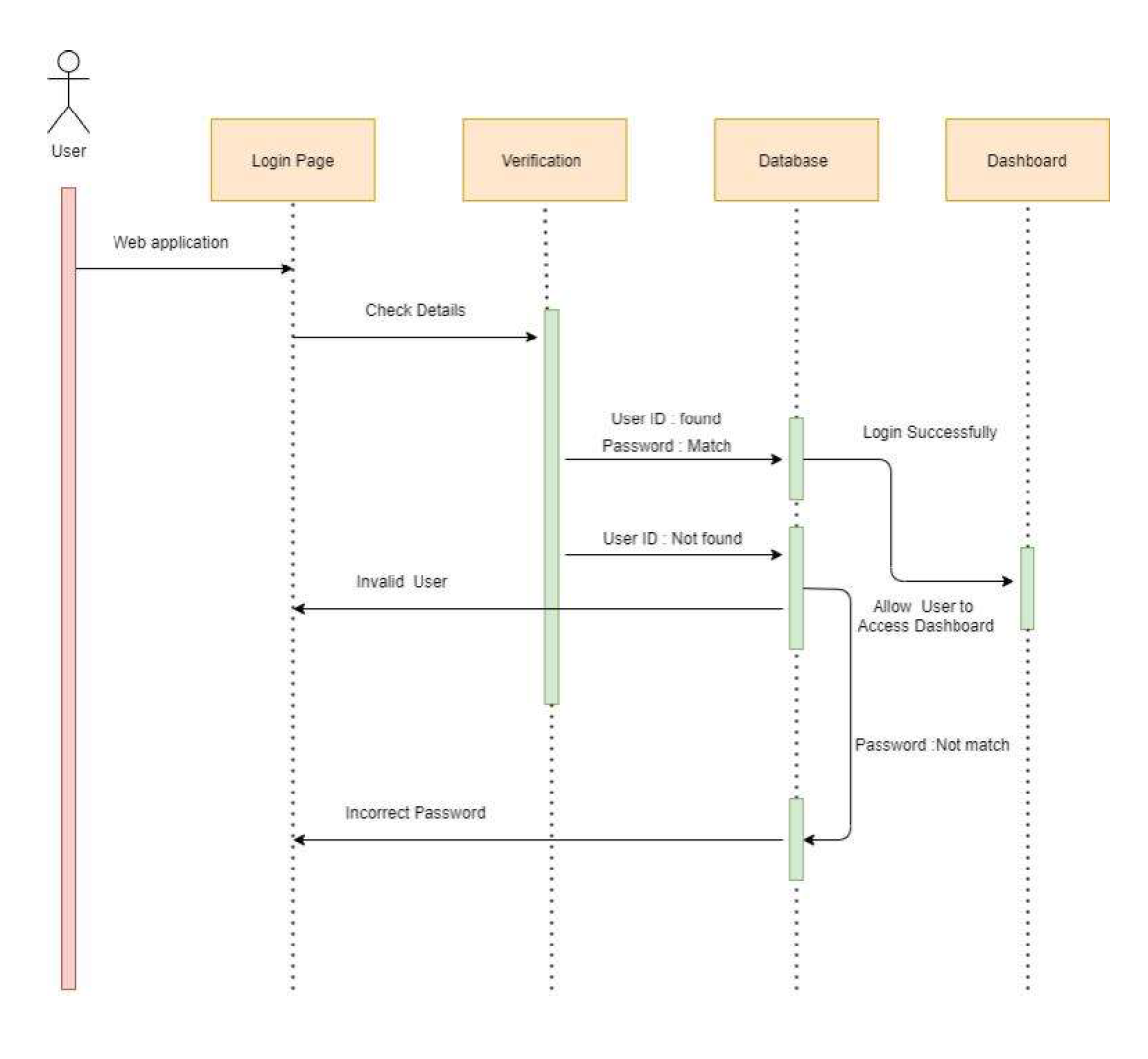


**MongoDB database Components:**

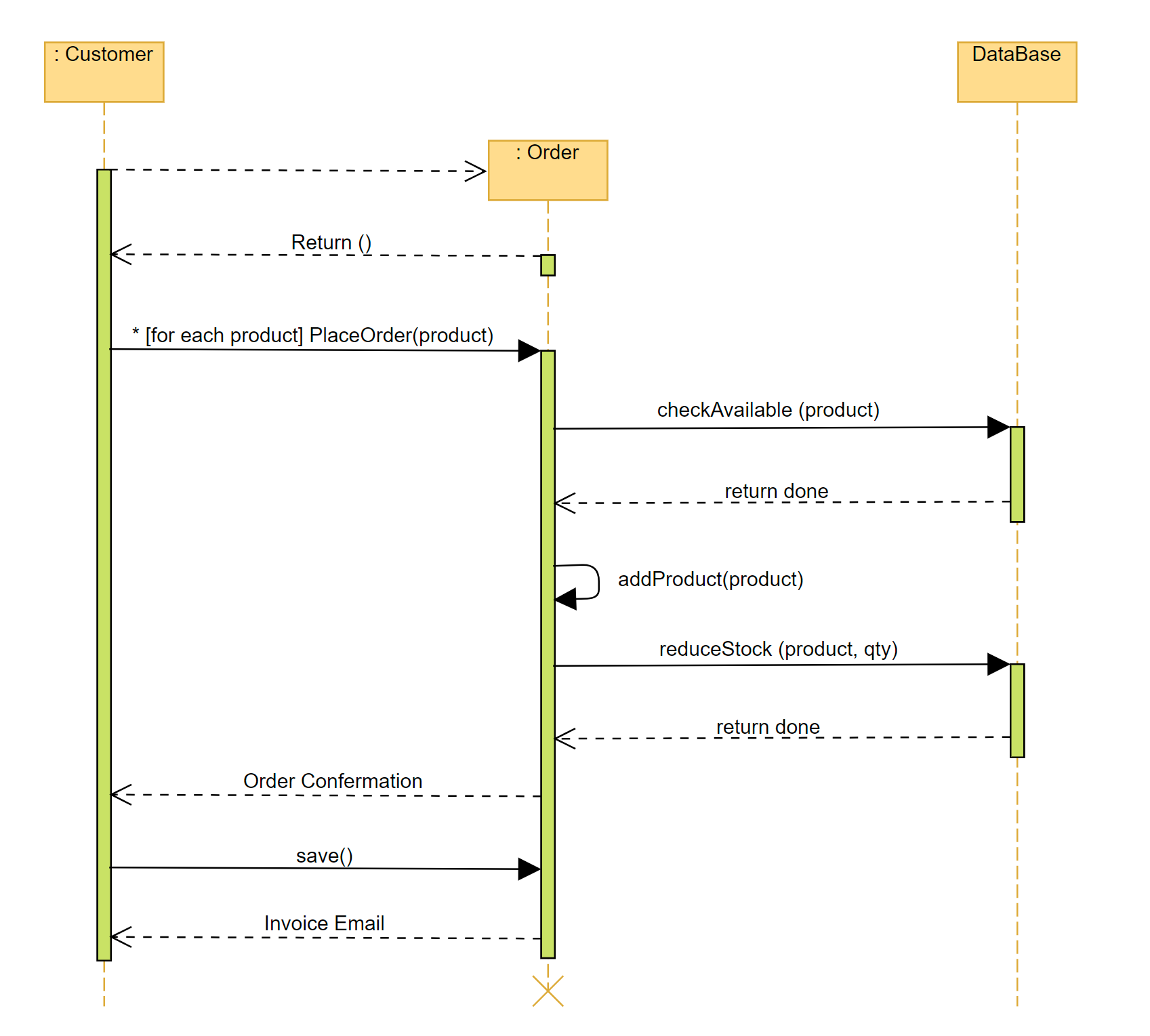


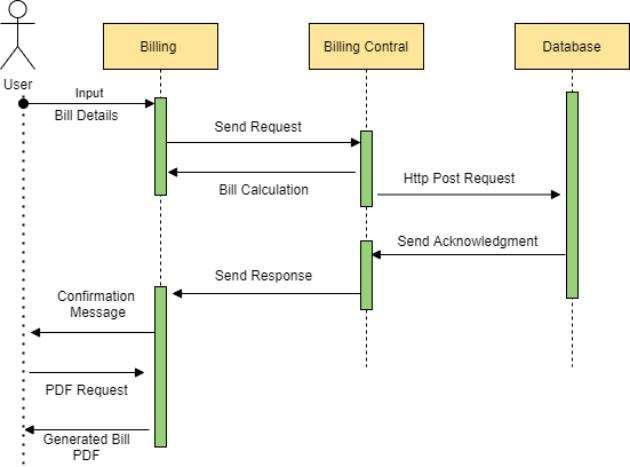
**Sequence diagram:**

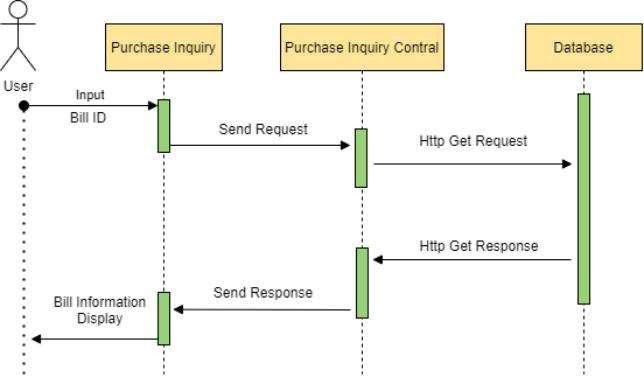
**Signup/Login Functionality**



**Order Placing Functionality:**



**Payments:**

**Invoice Generation:**

**Implementation Planning**

**Implementation Environment:**

**VS Code Installation: -**

Step 1: Download the VS Code Installer.

1. Open a browser window and navigate to the download page for windows at

<https://code.visualstudio.com>.

2. Underneath the heading at the top download VS Code for 64 bits.

Step 2: Run the Installer Once you have chosen and downloaded an installer, simply

run it by double-clicking on the downloaded file.

**Install React**

**Using the create-react-app command**

Instead of using webpack and babel you can install ReactJS more simply by installing create-react-app.

Step 1 - install create-react-app

Browse through the desktop and install the Create React App using command prompt as shown below − C:\Users\Tutorialspoint>cd C:\Users\Tutorialspoint\Desktop\

C:\Users\Tutorialspoint\Desktop>npx create-react-app my-app

This will create a folder named my-app on the desktop and installs all the required files in it.

Step 2 - Delete all the source files

Browse through the src folder in the generated my-app folder and remove all the files in it as shown below –

C:\Users\Tutorialspoint\Desktop>cd my-app/src

C:\Users\Tutorialspoint\Desktop\my-app\src>del \*

C:\Users\Tutorialspoint\Desktop\my-app\src\\*, Are you sure (Y/N)?

Step 3 - Add files

Add files with names index.css and index.js in the src folder as –

C:\Users\Tutorialspoint\Desktop\my-app\src>type nul > index.css

C:\Users\Tutorialspoint\Desktop\my-app\src>type nul > index.js

In the index.js file add the following code

import React from 'react';

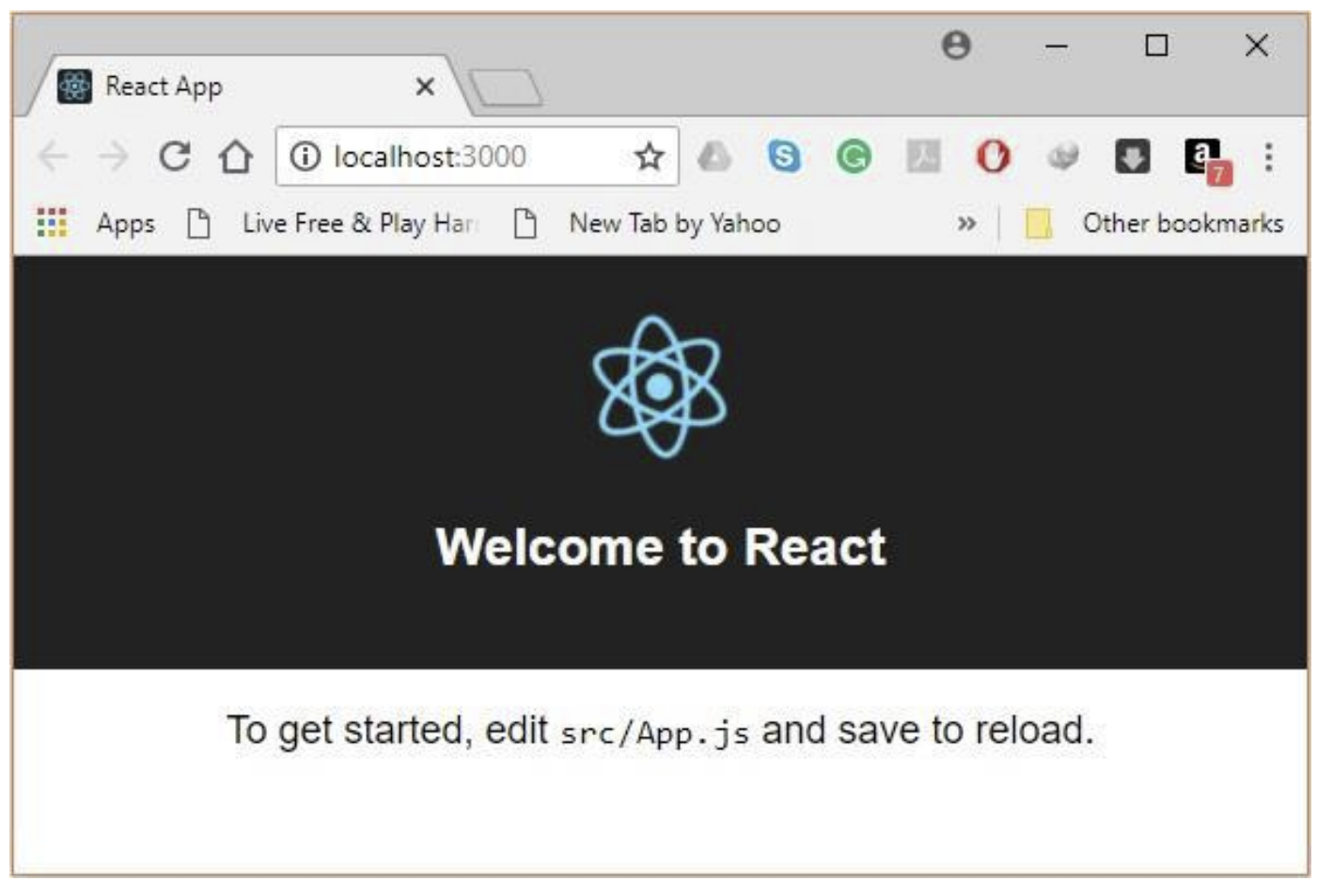
import ReactDOM from 'react-dom';

import './index.css';

Step 4 - Run the project

Finally, run the project using the start command.

npm start



**Install nodejs.**

**Step-1**: Downloading the Node.js ‘.msi’ installer.

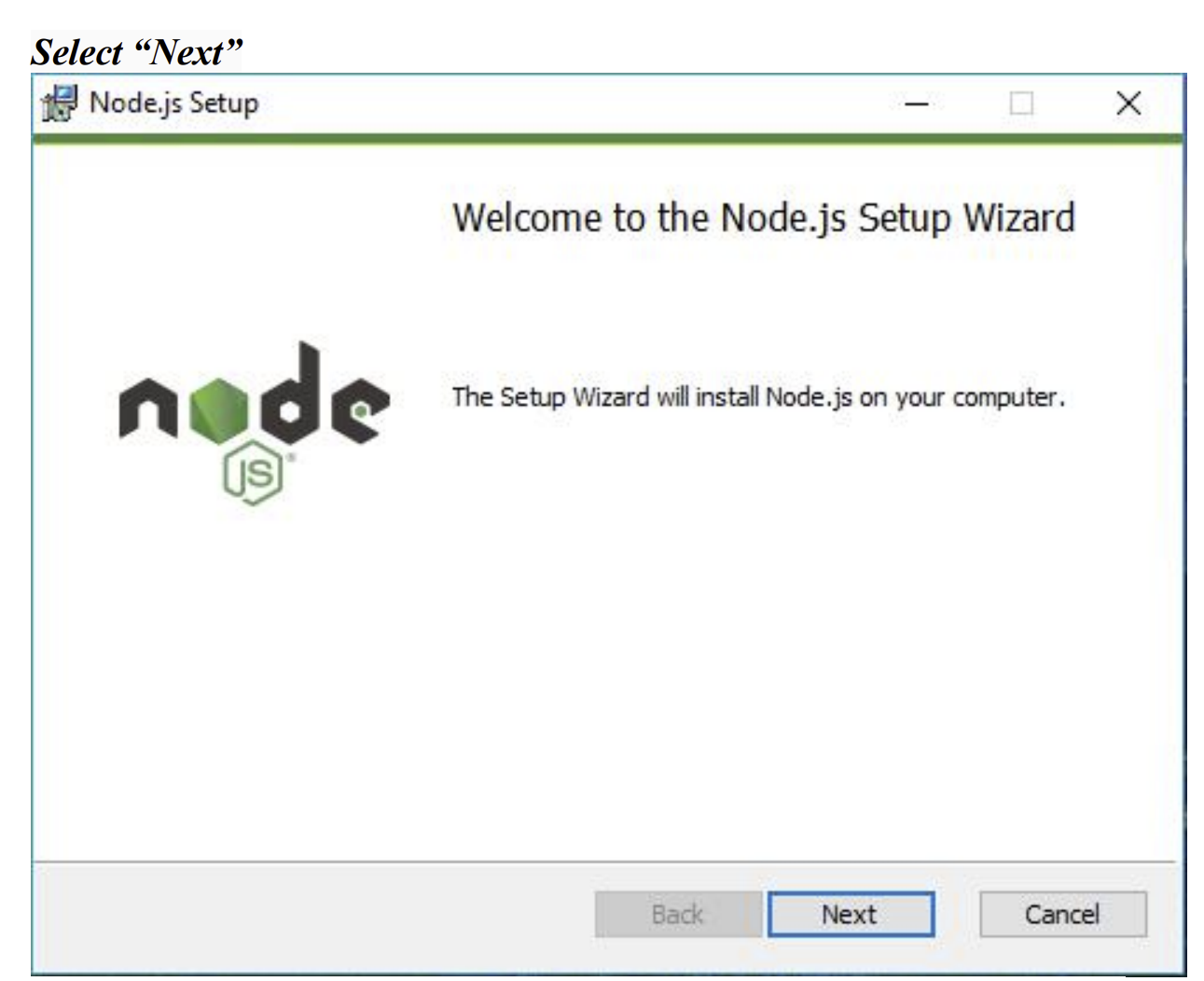
The first step to install Node.js on windows is to download the installer. Visit the official Node.js website i.e) https://nodejs.org/en/download/ and download the .msi file according to your system environment (32-bit & 64-bit). An MSI installer will be downloaded on your system.

**Step-2:** Running the Node.js installer. Now you need to install the node.js installer on your PC. You need to follow the following steps for the Node.js to be installed: -

● Double click on the .msi installer.

The Node.js Setup wizard will open.

● Welcome To Node.js Setup Wizard.



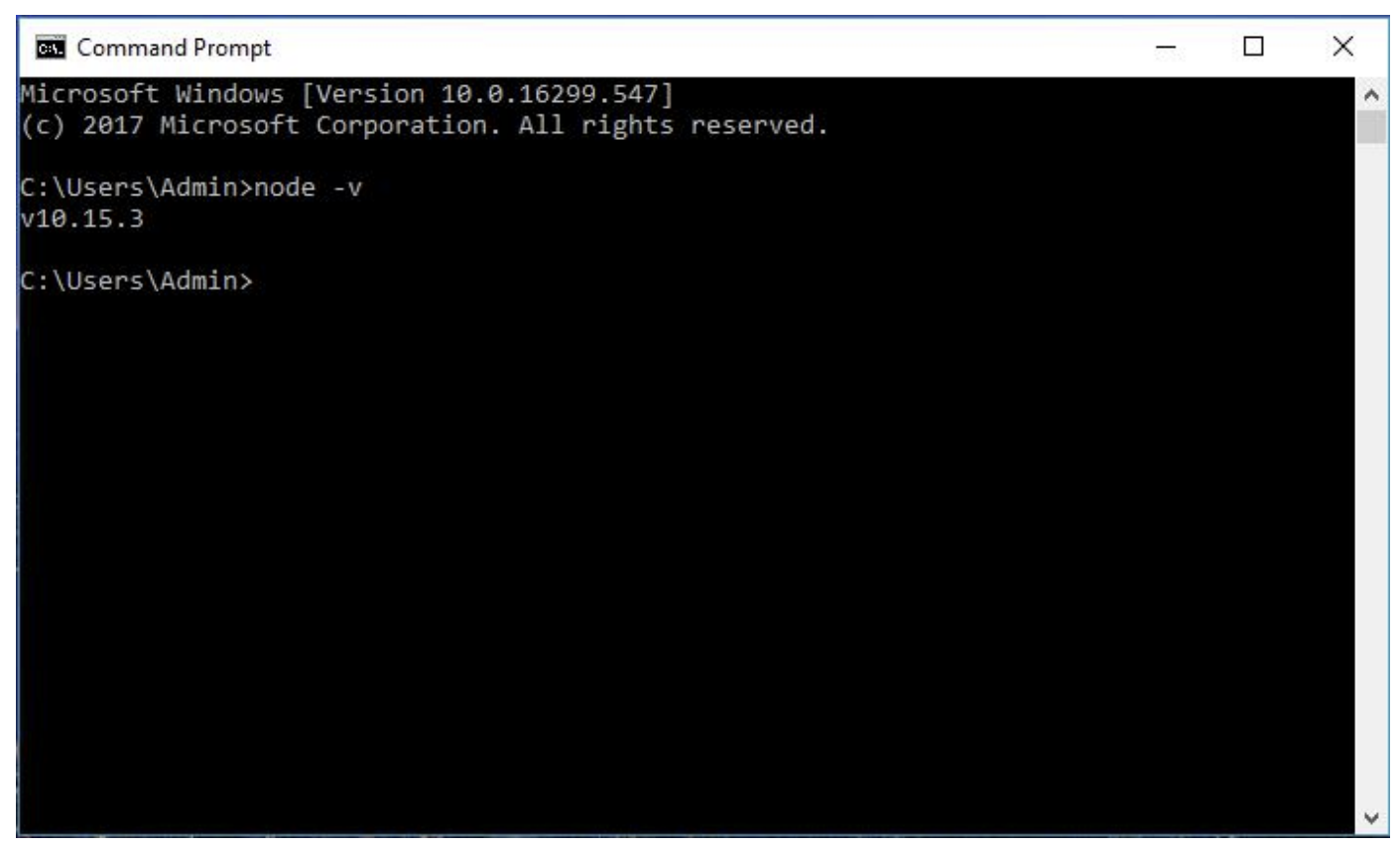
After clicking “Next”, End-User License Agreement (EULA) will open.

**● Click next, finish.**

**Step3: Verify if Node is installed or note.**

To check that node.js was completely installed on your system or not, you can run the following command in your command prompt or Windows PowerShell and test it:-

**C:\Users\Admin> node -**



If node.js was completely installed on your system, the command prompt will print the version of the node.js installed.

**Installing Express.**

Goto your current folder in CMD. and run above code.

$ npm install --save expr

**TESTING**

**TESTING PLAN**

The objective of the system testing is to ensure that all individual programs are working as expected, that the programs link together to meet the requirements specified and ensure that the computer system and the associated clerical and other procedures work together. Systems are not designed as entire systems, but they are tested as a single system. The analyst must perform both unit and system testing.

Different types of testing methods are available. We have tested our system for different aspects like Does the application meet the goals for which it has been designed. This was a very important question that stood before us as the application was designed to be implemented on such a large network.

To fulfill its goal of being able to run on different systems we went through a series of tests at different places where this is supposed to be used the most. As we need to make our system efficient enough, we need to test it thoroughly.

Finally, we tested the system with the real-time data, for which it is actually designed. We are successful in satisfying our needs as it was designed according to the client's requirements. But it is very necessary to maintain this application, so our work is not still over.

**TESTING STRATEGY**

Once source code has been generated, software must be tested to uncover as many errors as possible before delivery to the customer. Our goal is to design a series of test cases that have a high likelihood of finding errors. Software testing techniques provide systematic guidance for designing tests that

(1) Exercise the internal logic of software components

(2) Exercise the inputs and outputs domains of the program to uncover errors in program function, behavior and performance.

During early stages of testing, a software engineer performs all tests. However, as the testing process progresses, testing specialists may become involved. Reviews and other activities can and do uncover errors, but they are not sufficient. Every time the program is executed, the customer tests it! Therefore, you must execute the program before it gets to the customer with the specific intent of finding and removing all errors. In order to find the highest possible number of errors, tests must be conducted systematically, and test cases must be designed using disciplined techniques.



**Testing Objectives**

● Testing is a process of executing a program with the intention of finding an error.

● A good test case is one that has a high probability of finding an as-yet undiscovered error.

● A successful test is one that uncovers an as-yet undiscovered error.

**Unit Testing**

Unit testing is a software development process in which the smallest testable part of an application, called units, are individually scrutinized for proper operation. Unit testing is often automated but it can also be done manually. This testing mode is a component of Extreme Programming (XP), a pragmatic method of software development that takes a meticulous approach to build a product by means of continual testing and revision.

Unit testing involves only those characteristics that are vital to the performance of the unit under test. This encourages developers to modify the source code without immediate concerns about how such changes might affect the functioning of the units or the program as a whole. Once all of the units in a program have been found to be working in the most efficient and error-free manner possible, larger components of the program can be evaluated by means of integration testing.

**Subsystem Testing**

After testing each unit, we move on to larger units called sub-systems In subsystem testing we tested the whole Threat Add-on as one system and Threat App as another system. We tested each subsystem and got successful results. We found no errors or bugs after the final test.

**System Testing**

Now, it’s time for whole System testing. We have found some cosmetic bugs and minor bugs. We have fixed it and tested it again. We worked on each error and exception that we got while testing and most of them are resolved or handled programmatically.

**Recovery Testing**

It is a system test that forces the software to fail in a variety of ways and verifies that recovery is properly performed.

**Performance Testing**

It is designed to test the run-time performance of software within the context of an integrated system performance testing occurs throughout all steps in the testing process

**TESTING METHODS**

**Acceptance Testing**

Acceptance testing can be connected by the end-user, customer, or client to validate whether or not to accept the product. Acceptance testing may be performed as part of the hand-off process between any two phases of development. The acceptance test suite is run against the supplied input data or using an acceptance test script to direct the tester. Then the results obtained are compared with the expected results. If there is a correct match for every case, the test suite is said to pass.

**Alpha & beta testing**

The alpha test is conducted at the developer’s site by a customer. The software is used in a natural setting with the developer “looking over the shoulder” of the user and recording errors and usage problems. Alpha test is conducted in a controlled environment. The beta testing is conducted at one or more customer sites by the end-user of the software. Unlike alpha testing, the developer is generally not present. Therefore, the beta test is a “live” application of the software in an environment that cannot be controlled by the developer.

**Black-box testing**

Also known as functional testing. Software testing techniques whereby the internal working of the item being tested is not known by the tester. For example, in a black box test on a software design the tester only knows the inputs and what the expected outcomes should be and not how the program arrives at those outputs. The tester does not ever examine the programming code and does not need any further knowledge of the program other than its specification.

* The advantages of this type of testing include:
* The test is unbiased as the designer and the tester are independent of each other.
* The tester does not need knowledge of any specific programming languages
* The test is done from the point of view of the user, not the designer. Test cases can be designed as soon as the specifications are complete.

The disadvantages of this type of testing include:

* The test can be redundant if the software designer has already run a test case.
* The test cases are difficult to design. Testing every possible input stream is unrealistic because it would take an inordinate amount of time: hence many program paths will go untested.

**White Box Testing**

* Also known as glass box, structural, clear box, and open box testing. A software testing technique whereby explicit knowledge of the internal workings of the item being tested is used to select the test data. Unlike black-box testing, white-box testing uses specific knowledge of programming code to examine outputs. The test is accurate only if the tester knows what the program is supposed to do. He or she can then see if the program diverges from its intended goal.